aws

사용 설명서

Amazon Relational Database Service



Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Relational Database Service: 사용 설명서

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon의 상표 및 브랜드 디자인은 Amazon 외 제품 또는 서비스와 함께, Amazon 브랜드 이미지를 떨어뜨리거나 고객에게 혼동을 일으킬 수 있는 방식으로 사용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 해당 상표의 소유자의 자산이며, 해당 상표의 소유자가 Amazon의 계열사이거나 Amazon과 제휴 관계에 있거나 Amazon의 후원을 받는 업체일 수 있습니다.

Table of Contents

Amazon RDS란 무엇입니까? 1
Amazon RDS의 장점 1
책임 비교 2
Amazon RDS 공동 책임 모델 3
DB 인스턴스 3
Amazon RDS 애플리케이션 아키텍처: 사례 4
DB 엔진 6
DB 인스턴스 클래스 7
DB 인스턴스 스토리지 7
Amazon Virtual Private Cloud(VPC)의 DB 인스턴스 8
AWS 리전 및 가용 영역 9
가용 영역 9
다중 AZ 배포 10
보안 그룹을 통한 액세스 제어 12
Amazon RDS 모니터링 13
Amazon RDS에 대한 사용자 인터페이스 14
AWS Management Console 15
명령줄 인터페이스 15
Amazon RDS API 15
Amazon RDS에 대한 요금이 부과되는 방법 16
Amazon RDS의 AWS 프리 티어 16
다음 단계 17
시작하기 17
데이터베이스 엔진과 관련된 주제 17
DB 인스턴스 18
DB 인스턴스 클래스 21
DB 인스턴스 클래스 유형 21
지원되는 DB 엔진 29
AWS 리전에서 DB 인스턴스 클래스 지원 확인 78
DB 인스턴스 클래스 변경 84
RDS for Oracle용 프로세서 구성 84
하드웨어 사양 106
DB 인스턴스 스토리지 141
스토리지 유형 141

프로비저닝된 IOPS 스토리지 142
범용 스토리지 146
SSD 스토리지 유형 비교 150
마그네틱 스토리지(레거시, 권장하지 않음) 154
전용 로그 볼륨(DLV) 154
스토리지 성능 모니터링 155
스토리지 성능에 영향을 끼치는 요인 156
리전, 가용 영역 및 Local Zones 159
AWS 리전 160
가용 영역 164
Local Zones 165
리전 및 엔진별 지원되는 Amazon RDS 기능 167
테이블 규칙 167
기능 빠른 참조 168
블루/그린 배포 170
교차 리전 자동 백업 171
리전 간 읽기 전용 복제본 172
데이터베이스 활동 스트림 175
듀얼 스택 모드 182
S3로 스냅샷 내보내기 200
IAM 데이터베이스 인증 211
Kerberos 인증 215
다중 AZ DB 클러스터 228
성능 개선 도우미 236
RDS Custom 237
Amazon RDS Proxy 250
Secrets Manager 통합 263
제로 ETL 통합 264
엔진 네이티브 기능 270
Amazon RDS에 대한 DB 인스턴스 결제 271
온디맨드 DB 인스턴스 273
예약 DB 인스턴스 275
설정 288
AWS 계정에 등록 288
관리자 액세스 권한이 있는 사용자 생성 289
프로그래밍 방식 액세스 권한 부여 290

요구 사항 결정 291
DB 인스턴스에 대한 액세스 권한 제공 293
시작하기 296
MariaDB DB 인스턴스 생성 및 해당 인스턴스에 연결 297
사전 조건 298
EC2 인스턴스 생성 298
MariaDB DB 인스턴스 생성 304
(선택 사항) AWS CloudFormation를 사용하여 VPC, EC2 인스턴스 및 MariaDB 인스턴스 생성 308
MariaDB DB 인스턴스에 연결 310
EC2 인스턴스 및 DB 인스턴스 삭제 314
(선택 사항) CloudFormation으로 생성한 EC2 인스턴스 및 DB 인스턴스 삭제 315
(선택 사항) DB 인스턴스를 Lambda 함수에 연결 315
Microsoft SQL Server DB 인스턴스 생성 및 해당 인스턴스에 연결 316
사전 조건 317
EC2 인스턴스 생성 317
SQL Server DB 인스턴스 생성 322
(선택 사항) AWS CloudFormation를 사용하여 VPC, EC2 인스턴스 및 SQL Server 인스턴스 생성 327
SQL Server DB 인스턴스에 연결 330
샘플 DB 인스턴스 탐색 332
EC2 인스턴스 및 DB 인스턴스 삭제 334
(선택 사항) CloudFormation으로 생성한 EC2 인스턴스 및 DB 인스턴스 삭제 335
(선택 사항) DB 인스턴스를 Lambda 함수에 연결 335
MySQL DB 인스턴스 생성 및 해당 인스턴스에 연결 336
사전 조건 337
EC2 인스턴스 생성 337
MySQL DB 인스턴스 생성 343
(선택 사항) AWS CloudFormation를 사용하여 VPC, EC2 인스턴스 및 MySQL 인스턴스 생성 348
MySQL DB 인스턴스에 연결 350
EC2 인스턴스 및 DB 인스턴스 삭제 353
(선택 사항) CloudFormation으로 생성한 EC2 인스턴스 및 DB 인스턴스 삭제 354
(선택 사항) DB 인스턴스를 Lambda 함수에 연결 354
Oracle DB 인스턴스 생성 및 해당 인스턴스에 연결 355
사전 조건 356

1단계: EC2 인스턴스 생성 356
2단계: Oracle DB 인스턴스 생성 362
(선택 사항) AWS CloudFormation를 사용하여 VPC, EC2 인스턴스 및 Oracle DB 인스턴스 생성 366
3단계: SQL 클라이언트를 Oracle DB 인스턴스에 연결 369
4단계: EC2 인스턴스 및 DB 인스턴스 삭제 372
(선택 사항) CloudFormation으로 생성한 EC2 인스턴스 및 DB 인스턴스 삭제 373
(선택 사항) DB 인스턴스를 Lambda 함수에 연결 373
PostgreSQL DB 인스턴스 생성 및 해당 인스턴스에 연결 374
사전 조건 375
EC2 인스턴스 생성 375
PostgreSQL DB 인스턴스 생성 381
(선택 사항) AWS CloudFormation를 사용하여 VPC, EC2 인스턴스 및 PostgreSQL 인스턴스 생성 385
PostgreSQL DB 인스턴스에 연결 388
EC2 인스턴스 및 DB 인스턴스 삭제 391
(선택 사항) CloudFormation으로 생성한 EC2 인스턴스 및 DB 인스턴스 삭제 392
(선택 사항) DB 인스턴스를 Lambda 함수에 연결 392
자습서: 웹 서버 및 Amazon RDS DB 인스턴스 생성 393
EC2 인스턴스를 시작하여 DB 인스턴스에 연결 394
DB 인스턴스 생성 400
웹 서버 설치 421
자습서: Amazon RDS DB 인스턴스에 액세스하기 위해 Lambda 함수 생성 433
사전 조건 434
Amazon RDS DB 인스턴스 생성 434
Lambda 함수 및 프록시 생성 435
함수 실행 역할 생성 436
Lambda 배포 패키지 생성 438
Lambda 함수 업데이트 440
콘솔에서 Lambda 함수 테스트 442
Amazon SQS 대기열 생성 443
이벤트 소스 매핑을 생성하여 Lambda 함수 호출 444
설정 테스트 및 모니터링 444
리소스 정리 445
자습서 및 샘플 코드 447
이 안내서의 자습서 447

다른 AWS 안내서의 자습서 448
Amazon RDS PostgreSQL에 대한 AWS 워크숍 및 랩 콘텐츠 포털 449
Amazon RDS MySQL에 대한 AWS 워크숍 및 랩 콘텐츠 포털 449
GitHub의 자습서 및 샘플 코드 450
AWS SDK 작업 450
Amazon RDS의 모범 사례 452
Amazon RDS 기본 운영 지침 452
DB 인스턴스 RAM 권장 사항 453
데이터베이스 엔진 버전을 최신 상태로 유지 454
AWS 데이터베이스 드라이버 454
Enhanced Monitoring 사용 455
지표 사용 455
성능 지표 보기 455
성능 지표 평가 458
쿼리 튜닝 460
MySQL 작업 모범 사례 461
테이블 크기 461
테이블 수 461
스토리지 엔진 462
MariaDB 작업 모범 사례 463
테이블 크기 463
테이블 수 463
스토리지 엔진 464
Oracle 작업의 모범 사례 464
PostgreSQL로 작업하기 위한 모범 사례 465
PostgreSQL DB 인스턴스에 데이터 로드 465
PostgreSQL Autovacuum 기능 사용 466
Amazon RDS for PostgreSQL 모범 사례 동영상 467
SQL Server로 작업하기 위한 모범 사례 467
Amazon RDS for SQL Server 모범 사례 동영상 468
DB 파라미터 그룹 작업 468
DB 인스턴스 만들기 자동화 468
새로운 기능 동영상 469
Amazon RDS에 대한 프로그래밍 방식 액세스 470
콘솔 투 코드 470
DB 인스턴스 구성 472

DB 인스턴스 생성 473
사전 조건 473
DB 인스턴스 생성 479
사용 가능한 설정 486
을 사용하여 리소스 생성AWS CloudFormation 518
RDS과AWS CloudFormation템플릿 518
AWS CloudFormation에 대해 자세히 알아보기 518
DB 인스턴스에 연결 519
연결 정보 찾기 519
DB 인스턴스에 액세스하는 시나리오 523
AWS 드라이버를 사용하여 DB 인스턴스에 연결 524
특정 DB 엔진을 실행하는 DB 인스턴스에 연결 525
RDS Proxy와의 연결 관리 525
데이터베이스 인증 옵션 525
암호화된 연결 526
옵션 그룹 작업 527
옵션 그룹 개요 527
옵션 그룹 생성 529
옵션 그룹 생성 531
옵션 그룹에 옵션 추가 533
옵션 그룹의 옵션 및 옵션 설정 표시하기 539
옵션 설정 수정 540
옵션 그룹에서 옵션 제거 543
옵션 그룹 삭제 545
파라미터 그룹 548
파라미터 그룹 개요 548
DB 파라미터 그룹 552
DB 클러스터 파라미터 그룹 568
DB 파라미터 그룹 비교 582
DB 파라미터 지정 582
Amazon RDS에서 ElastiCache 캐시 생성 590
RDS DB 인스턴스 설정을 사용하는 ElastiCache 캐시 생성 개요 590
RDS DB 인스턴스의 설정을 사용하여 ElastiCache 캐시 생성 591
EC2 데이터베이스 자동 마이그레이션 594
개요 594
사전 조건 595

제한 사항 596
IAM 리소스 생성 597
데이터 마이그레이션 설정 604
마이그레이션 관리 606
모니터링 608
자습서: 사용자 지정 파라미터 및 새로운 옵션 그룹을 사용하여 MySQL DB 인스턴스 만들기 610
사전 조건 611
사용자 지정 DB 파라미터 그룹 만들기 611
파라미터 값 수정 612
옵션 그룹 만들기 612
옵션 추가 613
DB 인스턴스의 DB 만들기 613
DB 인스턴스 관리 615
DB 인스턴스 중지 616
사용 사례 617
지원되는 DB 엔진, 클래스 및 리전 617
다중 AZ 지원 618
작동 방법 618
제한 사항 619
옵션 및 파리미터 그룹 620
퍼블릭 IP 주소 620
DB 인스턴스 중지 620
DB 인스턴스 시작 622
DB 인스턴스 재부팅 624
DB 인스턴스 재부팅 사용 사례 624
재부팅하는 방법 625
다중 AZ에서 재부팅 625
고려 사항 626
사전 조건 626
DB 인스턴스 재부팅: 기본 단계 627
EC2 인스턴스 연결 629
개요 630
EC2 인스턴스 연결 634
연결된 컴퓨팅 리소스 보기 637
특정 DB 엔진을 실행하는 DB 인스턴스에 연결 638
Lambda 함수 연결 639

개요 ... 641
Lambda 함수 연결 ... 650
연결된 컴퓨팅 리소스 보기 ... 652
DB 인스턴스 수정 ... 654
수정 예약 ... 655
사용 가능한 설정 ... 656
DB 인스턴스 유지 관리 ... 689
DB 인스턴스 유지 관리 업데이트 개요 ... 689
보류 중인 유지 관리 보기 ... 691
업데이트 적용 ... 693
다중 AZ 배포 유지 ... 695
유지 관리 기간 ... 696
DB 인스턴스의 유지 관리 기간 조정 ... 699
운영 체제 업데이트 ... 701
엔진 버전 업그레이드 ... 705
엔진 버전 수동 업그레이드 ... 706
마이너 엔진 버전 자동 업그레이드 ... 708
DB 인스턴스 이름 변경 ... 713
기존 DB 인스턴스 교체를 위한 이름 바꾸기 ... 714
DB 인스턴스 읽기 전용 복제본 작업 ... 717
개요 ... 719
DB 엔진 간의 차이점 ... 723
읽기 전용 복제본 생성 ... 726
읽기 전용 복제본 승격 ... 729
읽기 전용 복제본 모니터링 ... 734
리전 간 읽기 전용 복제본 ... 737
RDS 리소스에 태그 지정 ... 751
RDS 태그를 사용하는 이유는 무엇인가요? ... 752
RDS 태그 작동 방식 ... 753
모범 사례 ... 755
DB 스냅샷에 태그 복사 ... 756
Amazon RDS에서 태그 추가 및 삭제 ... 757
자습서: 태그를 사용하여 중지할 DB 인스턴스를 지정합니다. ... 761
Amazon RDS의 ARN ... 765
ARN 생성 ... 765
기존 ARN 가져오기 ... 772

스토리지 작업 776
DB 인스턴스 스토리지 용량 증가 776
스토리지 Autoscaling을 사용한 용량 자동 관리 778
스토리지 파일 시스템 업그레이드 786
프로비저닝된 IOPS 설정 수정 788
I/O 집약적 스토리지 수정 790
범용(gp3) 설정 수정 791
전용 로그 볼륨(DLV) 사용 793
DB 인스턴스 삭제 800
DB 인스턴스를 삭제하기 위한 사전 조건 800
DB 인스턴스 삭제 시 고려 사항 800
DB 인스턴스 삭제 802
자습서: MySQL DB 인스턴스 관리 805
소개 805
사전 조건 805
DB 인스턴스에 태그 추가 806
DB 스토리지 늘리기 806
읽기 전용 복제본 생성 807
태그 업데이트 808
다중 AZ 배포 구성 및 관리 811
다중 AZ DB 인스턴스 배포 813
DB 인스턴스를 다중 AZ 배포로 변환 815
다중 AZ DB 인스턴스 장애 조치 817
다중 AZ DB 클러스터 배포 821
다중 AZ DB 클러스터에서 사용 가능한 인스턴스 클래스 822
다중 AZ DB 클러스터 아키텍처 822
다중 AZ DB 클러스터용 파라미터 그룹 824
RDS 프록시를 다중 AZ DB 클러스터와 함께 사용 824
복제본 지연 시간 및 다중 AZ DB 클러스터 824
다중 AZ DB 클러스터 스냅샷 827
다중 AZ DB 클러스터 생성 828
다중 AZ DB 클러스터에 연결 853
AWS 컴퓨팅 리소스와 다중 AZ DB 클러스터 연결 859
다중 AZ DB 클러스터 수정 883
다중 AZ DB 클러스터 업그레이드 900
다중 AZ DB 클러스터 이름 바꾸기 903

다중 AZ DB 클러스터 재부팅 906
다중 AZ DB 클러스터 장애 조치 908
다중 AZ DB 클러스터의 PostgreSQL 논리적 복제 912
다중 AZ DB 클러스터 읽기 전용 복제본 사용 917
다중 AZ DB 클러스터에서 외부 복제 설정 928
다중 AZ DB 클러스터 삭제 929
다중 AZ DB 클러스터의 제한 사항 931
RDS 추가 지원 932
RDS 추가 지원 개요 933
RDS 추가 지원 요금 934
RDS 추가 지원에 대한 비용 방지 934
RDS 확장 지원이 포함된 버전 934
RDS 추가 지원 버전 명명 규칙 935
RDS 추가 지원 관련 책임 936
Amazon RDS 책임 936
귀하의 책임 936
DB 인스턴스 또는 다중 AZ DB 클러스터 생성 937
RDS 추가 지원 동작 937
RDS 추가 지원 고려 사항 938
RDS 추가 지원이 포함된 DB 인스턴스 또는 다중 AZ DB 클러스터 생성 938
RDS 추가 지원 등록 보기 939
지원 날짜 보기 943
DB 인스턴스 또는 다중 AZ DB 클러스터 복원 945
RDS 추가 지원 동작 946
RDS 추가 지원 고려 사항 946
RDS 확장 지원이 적용되는 DB 인스턴스 또는 다중 AZ DB 클러스터 복원 947
데이터베이스 업데이트에 블루/그린 배포 사용 949
Amazon RDS 블루/그린 배포 개요 950
리전 및 버전 사용 가능 여부 951
이점 951
워크플로 951
액세스 권한 부여 956
제한 사항 및 고려 사항 957
모범 사례 963
블루/그린 배포 생성 967
블루/그린 배포 준비 968

변경 사항 지정 ... 970
지연 로딩 및 스토리지 초기화 ... 972
블루/그린 배포 생성 ... 974
사용 가능한 설정 ... 976
블루/그린 배포 보기 ... 981
블루/그린 배포 전환 ... 985
전환 제한 시간 ... 986
전환 가드레일 ... 986
전환 작업 ... 987
전환 모범 사례 ... 988
전환 전 CloudWatch 지표 확인 ... 989
전환 전 복제 지연 모니터링 ... 989
블루/그린 배포로의 전환 ... 990
전환 후 ... 992
블루/그린 배포 삭제 ... 994
데이터 백업, 복원 및 내보내기 ... 998
백업 소개 ... 999
백업 스토리지 ... 999
자동 백업 관리 ... 1001
백업 기간 ... 1001
백업 보관 기간 ... 1004
자동 백업 활성화 ... 1004
자동 백업 보존 ... 1007
보관된 자동 백업 삭제 ... 1009
지원되지 않는 MySQL 스토리지 엔진 ... 1012
지원되지 않는 MariaDB 스토리지 엔진 ... 1013
교차 리전 자동 백업 ... 1015
수동 백업 관리 ... 1031
단일 AZ DB 인스턴스용 DB 스냅샷 생성 ... 1032
다중 AZ DB 클러스터의 스냅샷 생성 ... 1035
DB 스냅샷 삭제 ... 1037
DB 인스턴스 복원 ... 1039
스냅샷에서 복원 ... 1040
고려 사항 ... 1041
시점 복구 ... 1046
다중 AZ DB 클러스터를 특정 시점으로 복원 ... 1051

스냅샷에서 다중 AZ DB 클러스터로 복원 1054
다중 AZ DB 클러스터 스냅샷에서 DB 인스턴스로 복원 1057
자습서: DB 스냅샷에서 DB 인스턴스 복원 1060
DB 스냅샷 복사 1064
DB 스냅샷 복사 1064
제한 사항 1073
고려 사항 1074
DB 스냅샷 공유 1083
스냅샷 공유 1084
퍼블릭 스냅샷 공유 1088
암호화된 스냅샷 공유 1090
스냅샷 공유 중지 1094
Amazon S3로 DB 스냅샷 데이터 내보내기 1096
스냅샷 데이터 내보내기 개요 1097
S3 버킷에 대한 액세스 권한 설정 1098
DB 스냅샷 내보내기 1103
리전 및 버전 사용 가능 여부 1107
제한 사항 1107
스냅샷 내보내기 모니터링 1108
스냅샷 내보내기 취소 1111
오류 메시지 1112
PostgreSQL 권한 오류 문제 해결 1114
파일 이름 지정 규칙 1114
데이터 변환 1116
사용 AWS Backup 1125
DB 인스턴스에서 지표 모니터링 1126
모니터링 계획 1126
성능 기준 1127
성능 지침 1127
모니터링 도구 1129
자동 모니터링 도구 1129
수동 모니터링 도구 1131
인스턴스 상태 조회 1132
Amazon RDS DB 인스턴스 상태 보기 1133
Amazon RDS의 권장 사항 1139
권장 사항 보기 1141

권장 사항 적용 1149
권장 사항 무시 1155
무시된 권장 사항을 활성으로 수정 1157
권장 사항 참조 1158
Amazon RDS 콘솔에서 지표 보기 1178
성능 개선 도우미 대시보드 보기 1179
모니터링 탭에서 새 모니터링 보기 선택 1180
성능 개선 도우미 페이지에서 새 모니터링 보기 선택 1181
사용자 지정 대시보드 생성 1183
사전 구성된 대시보드 선택 1186
CloudWatch를 사용하여 RDS 모니터링 1188
Amazon RDS 및 Amazon CloudWatch 개요 1189
CloudWatch 지표 보기 1190
CloudWatch에 성능 개선 도우미 지표 내보내기 1196
CloudWatch 경보 생성 1202
자습서: DB 클러스터 복제본 지연에 대한 CloudWatch 경보 생성 1202
Database Insights를 사용한 모니터링 1210
요금 1210
엔진, 리전, 인스턴스 클래스 지원 1210
고급 모드 켜기 1216
표준 모드 켜기 1220
느린 쿼리 모니터링 1225
고려 사항 1227
Performance Insights로 DB 로드 모니터링 1228
성능 개선 도우미 개요 1229
성능 개선 도우미 설정 및 해제 1241
MariaDB 또는 MySQL용 성능 스키마 1246
Performance Insights 정책 1250
성능 개선 도우미 대시보드를 사용한 지표 분석 1263
성능 개선 도우미 사전 권장 사항 보기 1312
성능 개선 도우미 API를 사용하여 지표 검색 1315
AWS CloudTrail을 사용하여 Performance Insights 호출 로깅 1339
VPC 엔드포인트(AWS PrivateLink) 1341
DevOps Guru for RDS로 성능 분석 1345
DevOps Guru for RDS의 이점 1345
DevOps Guru for RDS 작동 방식 1346

DevOps Guru for RDS 설정 1348
향상된 모니터링을 사용하여 OS 모니터링 1356
Enhanced Monitoring 개요 1356
Enhanced Monitoring 설정 및 활성화 1358
RDS 콘솔에서 OS 지표 보기 1363
CloudWatch Logs을 사용하여 OS 지표 보기 1367
RDS 지표 참조 1369
RDS에 대한 CloudWatch 지표 1369
RDS에 대한 CloudWatch 측정기준 1388
Performance Insights 위한 CloudWatch 지표 1388
성능 개선 도우미에 대한 카운터 지표. 1391
성능 개선 도우미에 대한 SQL 통계 1420
향상된 모니터링의 OS 지표 1437
데이터베이스 이벤트, 로그 및 데이터베이스 활동 스트림 모니터링 1450
Amazon RDS 콘솔에서 로그, 이벤트 및 스트림 보기 1451
RDS 이벤트 모니터링 1454
Amazon RDS에 대한 이벤트 개요 1454
Amazon RDS 이벤트 보기 1456
Amazon RDS 이벤트 알림 작업 1460
Amazon RDS 이벤트에서 트리거되는 규칙 생성 1486
Amazon RDS 이벤트 카테고리 및 이벤트 메시지 1491
RDS 로그 모니터링 1529
데이터베이스 로그 파일 보기 및 나열 1529
데이터베이스 로그 파일 다운로드 1530
데이터베이스 로그 파일 조사 1532
CloudWatch Logs에 게시 1533
REST를 사용하여 로그 파일 내용 읽기 1536
Db2 데이터베이스 로그 파일 1538
MariaDB 데이터베이스 로그 파일 1543
Microsoft SQL Server 데이터베이스 로그 파일 1557
MySQL 데이터베이스 로그 파일 1562
Oracle 데이터베이스 로그 파일 1578
PostgreSQL 데이터베이스 로그 파일 1588
CloudTrail에서 RDS API 호출 모니터링 1601
CloudTrail를 Amazon RDS와 통합 1601
Amazon RDS 로그 파일 항목 1602

데이터베이스 활동 스트림을 사용하여 RDS 모니터링 1606
개요 1606
Oracle 통합 감사 구성 1613
SQL Server 감사 구성 1614
데이터베이스 활동 스트림 시작 1615
데이터베이스 활동 스트림 수정 1618
활동 스트림 상태 가져오기 1621
데이터베이스 활동 스트림 중지 1622
활동 스트림 모니터링 1623
활동 스트림의 IAM 정책 예제 1666
Amazon GuardDuty RDS Protection을 이용한 위협 모니터링 1669
Amazon RDS Custom 1670
데이터베이스 커스터마이징 문제 1670
RDS Custom을 위한 관리 모델 및 이점 1672
RDS Custom의 공동 책임 모델 1672
RDS Custom에서 지원 범위 및 지원하지 않는 구성 1674
RDS Custom의 주요 이점 1674
RDS Custom 아키텍처 1675
VPC 1676
RDS Custom 자동화 및 모니터링 1677
Amazon S3 1681
AWS CloudTrail 1682
RDS Custom 보안 1683
RDS Custom이 사용자를 대신하여 작업을 안전하게 관리하는 방법 1683
SSL 인증서 1684
Amazon S3 버킷 보안으로 혼동된 대리자 문제 방지 1684
RDS Custom for Oracle 자격 증명 교체 1685
RDS Custom for Oracle 작업 1690
RDS Custom for Oracle 워크플로 1690
Amazon RDS Custom for Oracle 데이터베이스 아키텍처 1695
RDS Custom for Oracle에 대한 기능 가용성 및 지원 1697
RDS Custom for Oracle 요구 사항 및 제한 1700
RDS Custom for Oracle 환경 설정 1704
CEV 버전의 RDS Custom for Oracle으로 작업 1724
RDS for Oracle DB 인스턴스 구성 1755
RDS Custom for Oracle DB 인스턴스 관리 1774

RDS Custom for Oracle에 대한 다중 AZ 배포 관리 1791
RDS Custom for Oracle 복제본으로 작업 1809
RDS Custom for Oracle DB 인스턴스 백업 및 복원 1819
RDS Custom for Oracle에서 옵션 그룹을 사용한 작업 1829
RDS Custom for Oracle로 마이그레이션 1838
RDS Custom for Oracle DB 인스턴스 업그레이드 1839
RDS Custom for Oracle 문제 해결 1852
RDS Custom for Oracle에 대해 알려진 문제 1875
RDS Custom for SQL Server 작업 1879
RDS for SQL Server 워크플로 1879
RDS Custom for SQL Server 요구 사항 및 제한 1882
RDS Custom for SQL Server 환경 설정 1952
RDS Custom for SQL Server에서 기존 보유 미디어 사용(BYOM) 1976
RDS Custom for SQL Server CEV 작업 1978
RDS Custom for SQL Server DB 인스턴스 생성 및 연결 2000
RDS Custom for SQL Server DB 인스턴스 관리 2012
RDS Custom for SQL Server를 사용하여 Microsoft Active Directory 작업 2026
RDS Custom for SQL Server에 대한 다중 AZ 배포 구성 및 관리 2051
RDS Custom for SQL Server DB 인스턴스 백업 및 복원 2066
RDS Custom for SQL Server DB 스냅샷 복사 2083
온프레미스 데이터베이스를 SQL Server용 RDS Custom으로 마이그레이션 2094
RDS Custom for SQL Server DB 인스턴스 업그레이드 2097
Amazon RDS Custom for SQL Server의 문제 해결 2101
AWS Outposts 기반 Amazon RDS 2136
사전 조건 2137
Amazon RDS 기능 지원 2138
지원되는 DB 인스턴스 클래스 2144
고객 소유 IP 주소 2146
COIP 사용 2146
제한 사항 2148
다중 AZ 배포 2149
공동 책임 모델 작업 2149
가용성 개선 2149
사전 조건 2150
Amazon EC2 권한에 대한 API 작업 수행 2151
Outposts 기반 RDS DB 인스턴스 생성 2153

Amazon RDS on Outposts 읽기 전용 복제본 생성 2162
DB 인스턴스 복원에 대한 고려 사항 2165
Amazon RDS Proxy 2166
리전 및 버전 사용 가능 여부 2167
제한 사항 및 할당량 2167
RDS for MariaDB 제한 사항 2168
RDS for SQL Server 제한 사항 2169
MySQL 제한 사항 2170
PostgreSQL 제한 사항 2171
RDS Proxy 사용 대상 계획 2172
RDS Proxy 개념 및 용어 2173
RDS Proxy 개념 개요 2174
연결 풀링 2175
보안 2175
장애 조치 2177
트랜잭션 2178
RDS 프록시 시작하기 2179
프록시 네트워크 설정 2179
데이터베이스 자격 증명 설정 2182
IAM 정책 설정 2186
프록시 만들기 2188
프록시 보기 2197
RDS Proxy를 통해 연결 2198
RDS 프록시 관리 2202
RDS 프록시 수정 2202
데이터베이스 사용자 추가 2209
RDS 프록시 연결 고려 사항 2210
RDS 프록시 고정 방지 2213
RDS 프록시 삭제 2219
RDS 프록시 엔드포인트 작업 2220
프록시 엔드포인트 개요 2221
프록시 엔드포인트에 대한 제한 사항 2221
다중 AZ DB 클러스터의 프록시 엔드포인트 2222
VPC 간에 RDS 데이터베이스 액세스 2223
프록시 엔드포인트 생성 2224
프록시 엔드포인트 보기 2227

프록시 엔드포인트 수정 2228
프록시 엔드포인트 삭제 2229
CloudWatch를 사용하여 RDS 프록시 모니터링 2231
RDS 프록시 이벤트 작업 2237
RDS 프록시 이벤트 2238
RDS 프록시 문제 해결 2240
프록시에 대한 연결 확인 2241
일반적인 문제 및 해결 방법 2242
RDS for MySQL 문제 해결 2243
RDS for PostgreSQL의 문제 해결 2245
RDS Proxy를 AWS CloudFormation에서 사용 2250
제로 ETL 통합 2252
이점 2254
주요 개념 2254
제한 사항 2255
일반 제한 사항 2255
RDS for MySQL 제한 사항 2256
Amazon Redshift 제한 사항 2256
Amazon SageMaker 레이크하우스 제한 사항 2256
할당량 2257
지원되는 리전 2257
제로 ETL 통합 시작하기 2257
1단계: 사용자 지정 DB 파라미터 그룹 생성 2258
2단계: 소스 데이터베이스 선택 또는 생성 2258
3단계: 대상 Amazon Redshift 데이터 웨어하우스 생성 2259
다음 단계 2260
Amazon Redshift와 제로 ETL 통합 생성 2260
사전 조건 2261
필수 권한 2261
제로 ETL 통합 생성 2264
통합 암호화 2268
다음 단계 2270
Amazon SageMaker 레이크하우스와 제로 ETL 통합 생성 2270
사전 조건 2270
필수 권한 2271
Amazon SageMaker 레이크하우스와 제로 ETL 통합 생성 2273

통합 암호화 2278
다음 단계 2280
제로 ETL 통합의 데이터 필터링 2280
데이터 필터의 형식 2281
"필터 로직" 2283
필터 우선순위 2283
예제 2284
데이터 필터 추가 2285
데이터 필터 제거 2287
데이터 추가 및 쿼리 2287
Amazon Redshift에서 대상 데이터베이스 생성 2288
소스 데이터베이스에 데이터 추가 2288
Amazon Redshift에서 Amazon RDS 데이터 쿼리 2289
데이터 형식 차이 2290
제로 ETL 통합 조회 및 모니터링 2293
통합 보기 2294
시스템 테이블을 사용한 모니터링 2295
EventBridge로 모니터링 2296
제로 ETL 통합 수정 2296
제로 ETL 통합 삭제 2297
제로 ETL 통합 문제 해결 2299
제로 ETL 통합을 생성할 수 없습니다 2299
내 통합이 Syncing 상태에서 멈췄습니다. 2300
내 테이블이 Amazon Redshift에 복제되지 않는 경우 2300
Amazon Redshift 테이블 중 하나 이상을 재동기화해야 합니다 2300
Amazon RDS의 Db2 2303
Db2 개요 2304
Db2 기능 2305
Db2 버전 2308
Db2 라이선스 2312
Db2 인스턴스 클래스 2328
Db2 기본 역할 2331
Db2 파라미터 2331
EBCDIC 데이터 정렬 2337
Db2 현지 시간대 2338
DB 인스턴스 사전 조건 2341

관리자 계정 2341
추가 고려 사항 2341
여러 Db2 데이터베이스 2343
DB 인스턴스에 연결 2345
엔드포인트 찾기 2345
IBM Db2 CLP 2347
IBM CLPPlus 2351
DBeaver 2354
IBM Db2 Data Management Console 2358
보안 그룹 고려 사항 2366
Db2 연결 보안 2368
SSL/TLS를 사용하여 암호화 2368
Kerberos 인증 사용 2374
RDS for Db2 DB 인스턴스 관리 2388
시스템 작업 2390
데이터베이스 작업 2402
S3과 통합 2425
IAM 정책 생성 2425
IAM 역할 생성 및 IAM 정책 연결 2428
DB 인스턴스에 IAM 역할 추가 2430
RDS for Db2로 데이터 마이그레이션 2433
AWS 서비스를 사용하여 데이터 마이그레이션 2433
네이티브 Db2 도구를 사용한 데이터 마이그레이션 2444
연동 2457
동종 페더레이션 2457
이기종 페더레이션 2462
Db2 복제본 작업 2466
대기 복제본 2466
데이터베이스 활성화 2466
HADR 구성 2466
아카이브 로그 보존 2467
Db2 복제 중 중단 2467
Db2 복제본에 대한 요구 사항 및 고려 사항 2467
Db2 복제본 생성 준비 2469
대기 Db2 복제본 생성 2470
Db2 복제본 백업 작업 2472

Db2 복제 문제 해결 2474
RDS for Db2 DB 인스턴스 옵션 2476
Db2 감사 로깅 2477
외부 저장 프로시저 2492
Java 기반 외부 저장 프로시저 2492
알려진 문제 및 제한 2501
인증 제한 2501
울타리가 없는 루틴 2501
마이그레이션 중 자동이 아닌 스토리지 테이블스페이스 2501
db2_compatibility_vector 파라미터 설정 2502
INVALID 패키지가 포함된 데이터베이스 마이그레이션 2502
RDS for Db2 저장 프로시저 2504
저장 프로시저 고려 사항 2511
권한 부여 및 취소 2512
감사 정책 2525
버퍼 풀 2530
데이터베이스 관리 2536
스토리지 액세스 2558
테이블스페이스 2561
Amazon RDS for Db2 사용자 정의 함수 2572
rdsadmin.get_task_status 2572
rdsadmin.list_databases 2577
문제 해결 2579
데이터베이스 연결 오류 2579
파일 I/O 오류 2579
저장된 프로시저 오류 2583
Amazon RDS MariaDB 2594
MariaDB 기능 지원 2596
MariaDB 메이저 버전 2596
지원되는 스토리지 엔진 2604
캐시 워밍 2605
지원되지 않는 기능 2607
MariaDB 버전 2609
지원되는 MariaDB 마이너 버전 2609
지원되는 MariaDB 메이저 버전 2612
데이터베이스 미리 보기 환경 2612

데이터베이스 미리 보기 환경의 MariaDB 버전 11.4 2616
더 이상 사용되지 않는 MariaDB 버전 2616
MariaDB를 실행하는 DB 인스턴스에 연결 2617
연결 정보 찾기 2618
명령줄 클라이언트에서 연결 2621
AWS 드라이버를 사용하여 연결 2621
문제 해결 2622
MariaDB 연결 보안 2624
MariaDB 보안 2624
암호 확인 플러그인 2626
SSL/TLS를 사용하여 암호화 2626
새 SSL/TLS 인증서 사용 2631
RDS Optimized Reads를 통한 쿼리 성능 개선 2636
개요 2636
사용 사례 2637
모범 사례 2637
사용 2638
모니터링 2639
제한 사항 2639
RDS Optimized Writes for MariaDB를 통한 쓰기 성능 개선 2641
개요 2641
새 데이터베이스와 함께 사용 2642
기존 데이터베이스에서 활성화 2647
제한 사항 2648
MariaDB DB 엔진 업그레이드 2649
고려 사항 2650
유효한 업그레이드 대상 찾기 2651
MariaDB 버전 번호 2652
RDS 버전 번호 2654
메이저 버전 업그레이드 2654
MariaDB DB 인스턴스 업그레이드 2655
마이너 버전 자동 업그레이드 2655
가동 중지 시간을 단축하여 업그레이드 2658
MariaDB DB 스냅샷 엔진 버전 업그레이드 2662
지원되지 않는 엔진 버전에 대한 업그레이드 옵션 2664
RDS for MariaDB DB 인스턴스로 데이터 가져오기 2666

데이터 가져오기 고려 사항 2670
외부 데이터베이스에서 데이터 가져오기 2675
가동 중지 시간을 단축하여 데이터 가져오기 2679
모든 소스에서 데이터 가져오기 2698
MariaDB 복제 2705
MariaDB 읽기 전용 복제본 2706
외부 소스 인스턴스를 사용하여 GTID 기반 복제 구성 2720
외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성 2724
MariaDB를 위한 옵션 2729
MariaDB 감사 플러그인 지원 2729
MariaDB에 대한 파라미터 2734
MariaDB 파라미터 보기 2734
사용할 수 없는 MySQL 파라미터입니다. 2736
MySQL DB 스냅샷에서 MariaDB DB 인스턴스로 데이터 마이그레이션 2738
마이그레이션 수행 2738
MariaDB와 MySQL 간의 호환성 관련 문제 2740
Amazon RDS SQL의 MariaDB 참조 2742
mysql.rds_replica_status 2742
mysql.rds_set_external_master_gtid 2744
mysql.rds_kill_query_id 2747
현지 시간대 2749
MariaDB에 대해 알려진 문제 및 제한 사항 2751
파일 크기 제한 2751
InnoDB 예약어 2753
사용자 지정 포트 2753
성능 개선 도우미 2753
Amazon RDS의 Microsoft SQL Server 2754
공통 관리 작업 2756
제한 사항 2758
DB 인스턴스 클래스 지원 2761
보안 2765
규정 준수 프로그램 2766
HIPAA 2766
SSL 지원 2767
버전 지원 2768
버전 관리 2770

데이터베이스 엔진 패치 및 버전 2770
사용 중단 일정 2770
기능 지원 2772
SQL Server 2022 기능 2772
SQL Server 2019 기능 2772
SQL Server 2017 기능 2773
SQL Server 2016 기능 2774
SQL Server 2014 기능 2774
Amazon RDS에서 Microsoft SQL Server 2012 지원 종료 2774
Amazon RDS에서 SQL Server 2008 R2 지원 종료 2775
CDC 지원 2775
지원되지 않는 기능과 지원이 제한된 기능 2775
다중 AZ 배포 2777
TDE 사용 2778
함수 및 저장 프로시저 2778
현지 시간대 2784
지원되는 시간대 2784
Amazon RDS의 SQL Server 라이선싱 2796
라이선스가 종료된 DB 인스턴스 복원 2796
SQL Server Developer 버전 2797
SQL Server를 실행하는 DB 인스턴스에 연결 2798
연결하기 전에 2798
DB 인스턴스 엔드포인트 및 포트 번호 찾기 2799
SSMS로 DB 인스턴스에 연결 2800
SQL Workbench/J로 DB 인스턴스에 연결 2803
보안 그룹 고려 사항 2805
문제 해결 2805
RDS for SQL Server를 사용하여 Active Directory 작업 2808
SQL Server DB 인스턴스를 사용하여 자체 관리형 Active Directory 작업 2809
RDS for SQL Server를 사용하여 AWS 관리형 Active Directory 작업 2828
새 SSL/TLS 인증서에 대한 애플리케이션 업데이트 2844
애플리케이션에서 SSL을 사용해 Microsoft SQL Server DB 인스턴스에 연결하는지 여부 확인 2845
클라이언트에서 연결을 위해 인증서 확인이 필요한지 여부 확인 2845
애플리케이션 트러스트 스토어 업데이트 2847
SQL Server DB 엔진 업그레이드 2849

메이저 버전 업그레이드 2850
업그레이드 고려 사항 2852
업그레이드 테스트 2855
SQL Server DB 인스턴스 업그레이드 2856
지원 종료 전에 사용되지 않는 DB 인스턴스 업그레이드 2856
SQL Server 데이터베이스 가져오기 및 내보내기 2858
제한 및 권장 사항 2861
설정 2862
기본 백업 및 복원 사용 2867
백업 파일 압축 2883
문제 해결 2884
다른 방법으로 SQL Server 데이터 가져오기 및 내보내기 2887
Linux에서 BCP 사용 2900
SQL Server 읽기 전용 복제본 2906
SQL Server용 읽기 전용 복제본 구성 2906
SQL Server의 읽기 전용 복제본 제한 사항 2907
옵션 고려 사항 2908
데이터베이스 사용자 및 객체 동기화 2909
문제 해결 2911
RDS for SQL Server의 다중 AZ 2912
다중 AZ를 SQL Server DB 인스턴스에 추가 2913
Microsoft SQL Server DB 인스턴스에서 다중 AZ 제거 2914
제한, 참고 및 권장 사항 2914
보조의 위치 확인 2917
상시 작동 AG로 마이그레이션 2918
SQL Server 추가 기능 2919
SQL Server DB 인스턴스와 함께 암호 정책 사용 2920
Amazon S3 통합 2926
Database Mail 사용 2947
tempdb에 대한 인스턴스 스토어 지원 2963
확장 이벤트 사용 2966
트랜잭션 로그 백업에 액세스 2970
SQL Server용 옵션 3002
SQL Server 버전 및 에디션에 사용할 수 있는 옵션 나열 3004
Oracle OLEDB 포함 연결된 서버 3006
Teradata ODBC와 연결된 서버 3018

기본 백업 및 복원 3027
투명한 데이터 암호화 3032
SQL Server Audit 3044
SQL Server Analysis Services 3054
SQL Server Integration Services 3083
SQL Server Reporting Services 3105
Microsoft Distributed Transaction Coordinator 3125
공통 DBA 작업 3142
tempdb 데이터베이스에 액세스 3144
데이터베이스 엔진 튜닝 관리자를 사용하여 데이터베이스 워크로드 분석 3148
데이터베이스의 rdsa 계정으로 db_owner 변경 3152
데이터 정렬 및 문자 세트 관리 3153
데이터베이스 사용자 생성 3159
복구 모델 결정 3160
마지막 장애 조치 시간 확인 3161
LSN 격차로 인한 PITR 장애 문제 해결 3162
데이터베이스 이름 보기 거부 또는 허용 3163
빠른 삽입 비활성화 3164
SQL Server 데이터베이스 삭제 3164
다중 AZ 데이터베이스의 이름 변경 3165
마스터 사용자의 db_owner 역할 멤버십 재설정 3166
라이선스가 종료된 DB 인스턴스 복원 3166
오프라인에서 온라인으로 데이터베이스 전환 3167
CDC 사용 3167
SQL Server 에이전트의 사용 3170
SQL Server 로그 작업 3178
추적 및 덤프 파일 작업 3179
Amazon RDS의 MySQL 3181
MySQL 기능 지원 3184
MySQL 메이저 버전 3184
지원되는 스토리지 엔진 3186
memcached 및 기타 옵션 사용 3187
InnoDB 캐시 워밍 3187
MySQL 8.4의 포용적인 언어 변경 사항 3188
지원되지 않는 기능 3191
MySQL 버전 3193

지원되는 MySQL 마이너 버전 3193
지원되는 MySQL 메이저 버전 3197
RDS for MySQL에 대한 Amazon RDS 추가 지원 버전 3197
데이터베이스 미리 보기 환경 3199
데이터베이스 미리 보기 환경의 MySQL 버전 8.4 3203
데이터베이스 미리 보기 환경의 MySQL 버전 8.3 3203
데이터베이스 미리 보기 환경의 MySQL 버전 8.2 3203
데이터베이스 미리 보기 환경의 MySQL 버전 8.1 3204
더 이상 사용되지 않는 MySQL 버전 3204
MySQL 기반 DB 인스턴스에 연결 3205
연결 정보 찾기 3206
명령줄 클라이언트 설치 3210
명령줄 클라이언트에서 연결 3210
MySQL Workbench에서 연결 3211
AWS 드라이버를 사용하여 연결 3213
문제 해결 3214
MySQL 연결 보안 3215
암호 검증 3217
SSL/TLS를 사용하여 암호화 3218
새 SSL/TLS 인증서 사용 3222
MySQL에 Kerberos 인증 사용 3228
RDS Optimized Reads를 통한 쿼리 성능 개선 3241
개요 3241
사용 사례 3242
모범 사례 3242
사용 3243
모니터링 3244
제한 사항 3244
RDS Optimized Writes for MySQL을 통한 쓰기 성능 개선 3246
개요 2641
새 데이터베이스와 함께 사용 3247
기존 데이터베이스에서 활성화 3252
제한 사항 3252
MySQL DB 엔진 업그레이드 3254
고려 사항 3255
유효한 업그레이드 대상 찾기 3256

MySQL 버전 번호 3257
RDS 버전 번호 3259
메이저 버전 업그레이드 3259
업그레이드 테스트 3265
MySQL DB 인스턴스 업그레이드 3266
마이너 버전 자동 업그레이드 3266
가동 중지 시간을 단축하여 업그레이드 3269
MySQL DB 스냅샷 엔진 버전 업그레이드 3274
지원되지 않는 엔진 버전에 대한 업그레이드 옵션 3276
RDS for MySQL DB 인스턴스로 데이터 가져오기 3279
데이터 가져오기 고려 사항 3283
MySQL DB 인스턴스로 백업 복원 3288
외부 데이터베이스에서 데이터 가져오기 3300
가동 중지 시간을 단축하여 데이터 가져오기 3304
모든 소스에서 데이터 가져오기 3323
MySQL 복제 3330
MySQL 읽기 전용 복제본 3331
GTID 기반 복제 3347
외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성 3355
다중 소스 복제 구성 3359
액티브-액티브 클러스터 구성 3367
사용 사례 3367
액티브-액티브 클러스터의 제한 및 고려 사항 3368
교차 VPC 액티브-액티브 클러스터 준비 3371
액티브-액티브 클러스터의 필수 파라미터 설정 3373
DB 인스턴스를 액티브-액티브 클러스터로 변환 3374
새 액티브-액티브 클러스터 설정 3381
액티브-액티브 클러스터에 DB 인스턴스 추가 3387
액티브-액티브 클러스터 모니터링 3390
액티브-액티브 클러스터에서 그룹 복제 중지 3390
액티브-액티브 클러스터의 DB 인스턴스 이름 변경 3391
액티브-액티브 클러스터에서 DB 인스턴스 제거 3392
MySQL DB 인스턴스에서 데이터 내보내기 3393
외부 MySQL 데이터베이스 준비 3393
원본 MySQL DB 인스턴스 준비 3394
데이터베이스 복사 3395

내보내기 완료 3397
MySQL을 위한 옵션 3399
MariaDB 감사 플러그 3400
memcached 3407
MySQL 파라미터 3412
MySQL에 대한 공통 DBA 작업 3414
사전 정의된 사용자 이해 3414
역할 기반 권한 모델 3415
동적 권한 3418
세션 또는 쿼리 종료 3422
현재 복제 오류 넘어가기 3423
충돌 복구 시간 개선 3424
전역적 상태 이력 관리 3427
버퍼 풀 크기 및 다시 실행 로그 용량 구성 3429
현지 시간대 3431
알려진 문제 및 제한 3433
InnoDB 예약어 3433
스토리지가 가득 찬 동작 3433
일관되지 않은 InnoDB 버퍼 풀 크기 3434
인덱스 병합 최적화가 잘못된 결과를 반환 3435
Amazon RDS DB 인스턴스에 대한 MySQL 파라미터 예외 3436
Amazon RDS의 MySQL 파일 크기 제한 3436
MySQL Keyring Plugin 지원 안 됨 3439
사용자 지정 포트 3439
MySQL 저장 프로시저 제한 사항 3439
외부 소스 인스턴스를 사용하여 GTID 기반 복제 3439
MySQL 기본 인증 플러그인 3440
innodb_buffer_pool_size 재정의 3440
MySQL 5.7에서 MySQL 8.4로 업그레이드 3440
InnoDB 페이지 압축 3441
RDS for MySQL 저장 프로시저 3442
글로벌 상태 기록 수집 및 유지 3443
이진수 로그(binlog) 복제 구성, 시작 및 중지 3446
세션 또는 쿼리 종료 3486
활성-활성 클러스터 3488
다중 소스 복제 관리 3493

GTID를 사용한 트랜잭션 복제 3514
쿼리 로그 교체 3517
이진수 로그 구성 설정 및 표시 3519
InnoDB 캐시 워밍 3523
Amazon RDS의 Oracle 3525
Oracle 개요 3526
Oracle 기능 3527
Oracle 버전 3531
Oracle 라이선싱 3536
Oracle 사용자 및 권한 3540
Oracle 인스턴스 클래스 3541
Oracle 데이터베이스 아키텍처 3550
Oracle 파라미터 3551
Oracle 문자 집합 3553
Oracle 제한 사항 3557
Oracle DB 인스턴스에 연결 3560
엔드포인트 찾기 3560
SQL Developer 3562
SQL*Plus 3565
보안 그룹 고려 사항 3566
전용 및 공유 서버 프로세스 3566
문제 해결 3567
Oracle sqlnet.ora 파라미터 수정 3568
Oracle 연결 보안 3573
SSL을 사용하여 암호화 3573
새 SSL/TLS 인증서 사용 3574
NNE를 사용하여 암호화 3578
Kerberos 인증 구성 3578
UTL_HTTP 액세스 구성 3597
CDB 작업 3609
CDB 개요 3609
CDB 구성 3615
CDB 백업 및 복원 3621
비CDB를 CDB로 변환 3622
단일 테넌트 구성을 다중 테넌트로 변환 3625
CDB 인스턴스에 RDS for Oracle 테넌트 데이터베이스 추가 3627

RDS for Oracle 테넌트 데이터베이스 수정 ... 3630
CDB에서 RDS for Oracle 테넌트 데이터베이스 삭제 ... 3633
테넌트 데이터베이스 세부 정보 보기 ... 3635
CDB 업그레이드 ... 3640
Oracle DB 인스턴스 관리 ... 3641
시스템 작업 ... 3655
데이터베이스 작업 ... 3680
로그 작업 ... 3709
RMAN 작업 ... 3722
Oracle Scheduler 작업 ... 3754
문제 진단 ... 3763
기타 작업 ... 3773
RDS for Oracle 고급 기능 구성 ... 3788
인스턴스 스토어 구성 ... 3788
HugePages 활성화 ... 3799
확장 데이터 유형 활성화 ... 3802
Oracle로 데이터 가져오기 ... 3805
Oracle SQL Developer를 사용한 가져오기 ... 3806
Oracle 전송 가능한 테이블스페이스를 사용한 마이그레이션 ... 3806
Oracle Data Pump를 사용한 가져오기 ... 3822
Oracle 내보내기/가져오기를 통해 가져오기 ... 3839
Oracle SQL*Loader를 사용하여 가져오기 ... 3840
Oracle 구체화된 보기로 마이그레이션 ... 3841
Oracle 복제본 작업 ... 3844
Oracle 복제본 개요 ... 3844
Oracle 복제본에 대한 요구 사항 및 고려 사항 ... 3846
Oracle 복제본 생성 준비 ... 3850
탑재된 Oracle 복제본 생성 ... 3851
복제본 모드 수정 ... 3853
Oracle 복제본 백업 작업 ... 3854
Oracle Data Guard 전환 수행 ... 3857
Oracle 복제본 문제 해결 ... 3864
Oracle 옵션 ... 3866
Oracle DB 옵션 개요 ... 3866
Amazon S3 통합 ... 3869
Oracle Application Express(APEX) ... 3894

Amazon EFS 통합 3917
Java 가상 머신(JVM) 3933
Enterprise Manager 3937
레이블 보안 3960
로케이터 3963
기본 네트워크 암호화(NNE) 3967
OLAP 3980
Secure Sockets Layer(SSL) 3983
공간 3994
SQLT 3998
Statspack 4007
시간대 4011
시간대 파일 자동 업그레이드 4016
TDE(Transparent Data Encryption) 4026
UTL_MAIL 4030
XML DB 4034
Oracle DB 엔진 업그레이드 4035
Oracle 업그레이드 개요 4035
메이저 버전 업그레이드 4042
마이너 버전 업그레이드 4043
업그레이드 고려 사항 4049
업그레이드 테스트 4052
RDS for Oracle DB 인스턴스 업그레이드 4053
Oracle DB 스냅샷 업그레이드 4056
Oracle용 도구 및 타사 소프트웨어 4058
Oracle GoldenGate 사용 4059
Oracle Repository Creation Utility 사용 4077
CMAN 구성 4084
Amazon RDS에서 Oracle에 Siebel Database 설치 4087
Oracle 데이터베이스 엔진 릴리스 4092
Amazon RDS의 PostgreSQL 4093
공통 관리 작업 4095
데이터베이스 미리 보기 환경 작업 4099
데이터베이스 미리 보기 환경에서 지원하지 않는 기능 4100
데이터베이스 미리 보기 환경의 PostgreSQL 버전 17 4100
데이터베이스 미리 보기 환경에서 새 DB 인스턴스 생성 4100

PostgreSQL 버전 4103
지원 중단되는 PostgreSQL 버전 4103
RDS for PostgreSQL 릴리스 프로세스 4104
RDS for PostgreSQL 증분 릴리스 프로세스의 이점 4104
릴리스 업데이트 관리 4104
PostgreSQL 확장 버전 4106
PostgreSQL 확장의 설치 제한 4106
PostgreSQL 신뢰할 수 있는 확장 4107
PostgreSQL 기능 4109
사용자 지정 데이터 유형 및 열거 4109
이벤트 트리거 4110
대형 페이지 4111
논리적 복제 수행 4112
stats_temp_directory에 대한 RAM 디스크 4119
테이블스페이스 4119
EBCDIC 및 기타 메인프레임 마이그레이션을 위한 데이터 정렬 4120
논리적 슬롯 동기화 관리 4125
PostgreSQL 인스턴스에 연결 4127
psql 클라이언트 설치 4127
연결 정보 찾기 4128
pgAdmin을 사용하여 RDS for PostgreSQL DB 인스턴스에 연결 4130
psql을 사용하여 RDS for PostgreSQL DB 인스턴스에 연결 4132
AWS JDBC 드라이버를 사용하여 RDS for PostgreSQL에 연결 4134
AWS Python 드라이버를 사용하여 RDS for PostgreSQL에 연결 4134
RDS for PostgreSQL 인스턴스에 대한 연결 문제 해결 4134
SSL/TLS를 사용한 연결 보안 4137
PostgreSQL DB 인스턴스와 함께 SSL 사용 4137
새 SSL/TLS 인증서를 사용하도록 애플리케이션 업데이트 4144
Kerberos 인증 사용 4149
리전 및 버전 사용 가능 여부 4150
Kerberos 인증 개요 4150
설정 4151
Active Directory 도메인에서 RDS for PostgreSQL DB 인스턴스 관리 4165
Kerberos 인증을 사용하여 연결 4166
아웃바운드 네트워크 액세스에 사용자 지정 DNS 서버 사용 4169
사용자 지정 DNS 해결 활성화 4169

사용자 지정 DNS 해결 비활성화 4169
사용자 지정 DNS 서버 설정 4169
PostgreSQL DB 엔진 업그레이드 4171
고려 사항 4173
유효한 업그레이드 대상 찾기 4174
PostgreSQL 버전 번호 4175
RDS 버전 번호 4175
메이저 버전 업그레이드 선택 4176
메이저 버전 업그레이드를 수행하는 방법 4191
마이너 버전 자동 업그레이드 4198
PostgreSQL 확장 버전 업그레이드 4201
PostgreSQL DB 스냅샷 엔진 버전 업그레이드 4202
RDS for PostgreSQL의 읽기 전용 복제본 작업 4205
읽기 복제본의 논리적 디코딩 4205
PostgreSQL을 사용한 읽기 전용 복제본 제한 4208
PostgreSQL을 사용한 읽기 전용 복제본 구성 4209
계단식 읽기 전용 복제본 사용 4212
교차 리전 계단식 읽기 복제본 생성 4213
서로 다른 RDS for PostgreSQL 버전에서 복제가 작동하는 방식 4215
복제 프로세스 모니터링 및 튜닝 4219
RDS for PostgreSQL의 읽기 전용 복제본 문제 해결 4221
RDS Optimized Reads를 통한 쿼리 성능 개선 4223
PostgreSQL의 RDS Optimized Reads 개요 4223
사용 사례 4224
모범 사례 4224
사용 4225
모니터링 4225
제한 사항 4226
PostgreSQL로 데이터 가져오기 4227
Amazon EC2 인스턴스에서 PostgreSQL 데이터베이스 가져오기 4229
\copy 명령을 사용하여 PostgreSQL DB 인스턴스의 테이블로 데이터 가져오기 4231
RDS for PostgreSQL로 Amazon S3 데이터 가져오기 4232
DB 인스턴스 간에 PostgreSQL 데이터베이스 전송 4251
PostgreSQL 데이터를 Amazon S3로 내보내기 4260
확장 설치 4261
S3로 내보내기 개요 4262

내보낼 Amazon S3 파일 경로 지정 4263
Amazon S3 버킷에 대한 액세스 권한 설정 4264
aws_s3.query_export_to_s3 함수를 사용하여 쿼리 데이터 내보내기 4269
함수 참조 4271
Amazon S3 액세스 문제 해결 4275
RDS for PostgreSQL에서 Lambda 함수 호출 4277
1단계: 아웃바운드 연결 구성 4278
2단계: 인스턴스 및 Lambda에 대한 IAM 구성 4279
3단계: Lambda 확장 설치 4280
4단계: Lambda 도우미 함수 사용 4281
5단계: Lambda 함수 호출 4282
6단계: 사용자에게 권한 부여 4283
예제: Lambda 함수 호출 4284
Lambda 함수 오류 메시지 4286
Lambda 함수 및 파라미터 참조 4287
RDS for PostgreSQL의 일반적인 DBA 태스크 4293
RDS for PostgreSQL에서 지원되는 데이터 정렬 4294
PostgreSQL 역할 및 권한 이해 4294
PostgreSQL에서 연결 중단 처리 4310
PostgreSQL Autovacuum 사용 4312
로깅 메커니즘 4357
PostgreSQL을 사용한 임시 파일 관리 4358
pgBadger를 사용한 PostgreSQL의 로그 분석 4364
PGSnapper를 사용하여 PostgreSQL 모니터링 4364
파라미터 작업 4364
RDS for PostgreSQL의 대기 이벤트를 사용한 튜닝 4383
RDS for PostgreSQL 튜닝을 위한 필수 개념 4384
RDS for PostgreSQL 대기 이벤트 4388
Client:ClientRead 4390
Client:ClientWrite 4394
CPU 4396
IO:BufFileRead 및 IO:BufFileWrite 4402
IO:DataFileRead 4410
IO:WALWrite 4418
IPC:parallel 대기 이벤트 4421
Lock:advisory 4426

Lock:extend 4429
Lock:Relation 4432
Lock:transactionid 4435
Lock:tuple 4437
LWLock:BufferMapping (LWLock:buffer_mapping) 4441
LWLock:BufferIO(IPC:BufferIO) 4444
LWLock:buffer_content (BufferContent) 4446
LWLock:lock_manager (LWLock:lockmanager) 4448
LWLock:pg_stat_statements 4453
Timeout:PgSleep 4457
Timeout:VacuumDelay 4458
Amazon DevOps Guru의 사전 예방 인사이트를 활용하여 RDS for PostgreSQL 튜닝 4461
데이터베이스가 트랜잭션 연결 시 오랫동안 유휴 상태로 실행됨 4461
PostgreSQL 확장 사용 4465
orafce 함수 사용 4466
PostgreSQL에 대한 Amazon RDS 위임 확장 지원 사용 4468
pg_partman 확장자를 사용하여 파티션 관리하기 4481
pgAudit를 사용하여 데이터베이스 활동 로깅 4487
pg_cron 확장을 사용하여 유지 관리 예약 4500
pglogical을 사용하여 데이터 동기화 4510
pgactive를 사용하여 액티브-액티브 복제 생성 4523
pg_repack 확장을 사용하여 부풀림을 줄이기 4584
PLV8 업그레이드 및 사용 4589
PL/Rust를 사용하여 Rust 언어로 함수 작성 4591
PostGIS에서 공간 데이터 관리 4596
Amazon RDS for PostgreSQL에서 지원되는 외부 데이터 래퍼 4605
log_fdw 확장 사용 4605
postgres_fdw를 사용하여 외부 데이터 액세스 4607
MySQL 데이터베이스 작업 4608
Oracle 데이터베이스 작업 4612
SQL Server 데이터베이스 작업 4616
PostgreSQL용 신뢰할 수 있는 언어 확장 작업 4620
용어 4621
신뢰할 수 있는 언어 확장을 사용하기 위한 요구 사항 4622
신뢰할 수 있는 언어 확장 설정 4625
신뢰할 수 있는 언어 확장 개요 4628

TLE 확장 생성 4630
데이터베이스에서 TLE 확장 삭제 4635
신뢰할 수 있는 언어 확장 제거 4636
TLE 확장과 함께 PostgreSQL 후크 사용 4637
신뢰할 수 있는 언어 확장에서 사용자 지정 데이터 유형 사용 4643
신뢰할 수 있는 언어 확장에 대한 함수 참조 4643
신뢰할 수 있는 언어 확장에 대한 후크 참조 4657
코드 예시 4660
작업 4671
CreateDBInstance 4672
CreateDBParameterGroup 4687
CreateDBSnapshot 4693
DeleteDBInstance 4702
DeleteDBParameterGroup 4710
DescribeAccountAttributes 4716
DescribeDBEngineVersions 4720
DescribeDBInstances 4728
DescribeDBParameterGroups 4738
DescribeDBParameters 4746
DescribeDBSnapshots 4755
DescribeOrderableDBInstanceOptions 4762
GenerateRDSAuthToken 4770
ModifyDBInstance 4772
ModifyDBParameterGroup 4777
RebootDBInstance 4783
시나리오 4786
DB 인스턴스 시작하기 4787
서버리스 예제 4884
Lambda 함수를 사용하여 Amazon RDS 데이터베이스에 연결 4884
교차 서비스 예시 4901
Aurora 서버리스 작업 항목 트래커 만들기 4901
보안 4906
데이터베이스 인증 4908
암호 인증 4908
IAM 데이터베이스 인증 4909
Kerberos 인증 4909

RDS 및 Secrets Manager를 통한 암호 관리 4911
제한 사항 4911
개요 4912
이점 4912
권한 4913
RDS 관리 적용 4914
DB 인스턴스의 암호 관리 4915
테넌트 데이터베이스의 암호 관리 4919
다중 AZ DB 클러스터의 암호 관리 4922
DB 인스턴스의 보안 암호 교체 4926
다중 AZ DB 클러스터의 보안 암호 교체 4928
보안 암호 세부 정보 보기 4930
다중 AZ DB 클러스터의 보안 암호 세부 정보 보기 4933
테넌트 데이터베이스의 보안 암호 세부 정보 보기 4936
리전 및 버전 사용 가능 여부 4939
데이터 보호 4939
데이터 암호화 4940
인터네트워크 트래픽 개인 정보 4970
자격 증명 및 액세스 관리 4971
고객 4971
자격 증명을 통한 인증 4972
정책을 사용하여 액세스 관리 4975
Amazon RDS에서 IAM을 사용하는 방법 4977
자격 증명 기반 정책 예제 4984
AWS 관리형 정책 5005
정책 업데이트 5012
교차 서비스 혼동된 대리자 방지 5028
IAM 데이터베이스 인증 5030
문제 해결 5080
로깅 및 모니터링 5082
규정 준수 확인 5084
복원성 5085
백업 및 복원 5085
복제 5085
Failover 5086
인프라 보안 5087

보안 그룹 5087
퍼블릭 액세스 가능성 5087
VPC 엔드포인트(AWS PrivateLink) 5089
고려 사항 1341
가용성 1342
인터페이스 VPC 엔드포인트 생성 1342
VPC 엔드포인트 정책 생성 1342
보안 모범 사례 5092
보안 그룹을 통한 액세스 제어 5093
VPC 보안 그룹 개요 5093
보안 그룹 시나리오 5094
VPC 보안 그룹 생성 5096
DB 인스턴스 연결 5096
마스터 사용자 계정 권한 5096
서비스 연결 역할 5101
Amazon RDS에 대한 서비스 연결 역할 권한 5101
Amazon RDS Custom에 대한 서비스 연결 역할 권한 5104
Amazon RDS 베타에 대한 서비스 연결 역할 권한 5105
Amazon RDS 미리 보기에 대한 서비스 연결 역할 5106
Amazon RDS와 Amazon VPC 사용 5108
VPC에서 DB 인스턴스를 사용한 작업 5108
DB 인스턴스에 대한 VPC 업데이트 5124
VPC에서 DB 인스턴스에 액세스하는 시나리오 5125
자습서: DB 인스턴스에 사용할 Amazon VPC 생성(IPv4 전용) 5131
자습서: DB 인스턴스(듀얼 스택 모드)에 사용할 VPC 생성 5138
DB 인스턴스를 VPC로 이동 5148
할당량 및 제약 조건 5150
Amazon RDS의 할당량 5150
Amazon RDS의 명명 제약 조건 5155
최대 데이터베이스 연결 수 5156
Amazon RDS의 파일 크기 제한 5159
문제 해결 5160
DB 인스턴스에 연결할 수 없음 5160
DB 인스턴스 연결 테스트 5162
연결 인증 문제 해결 5163
보안 문제 5164

오류 메시지 "계정 속성을 불러오지 못했습니다. 일부 콘솔 기능이 손상되었을 수 있습니다." ... 5164
호환되지 않는 네트워크 상태 문제 해결 ... 5164
원인 ... 5164
해결 방법 ... 5165
DB 인스턴스 소유자 암호 재설정 ... 5166
DB 인스턴스 중단 또는 재부팅 ... 5166
파라미터 변경 사항이 적용 안 됨 ... 5167
DB 인스턴스 스토리지 부족 ... 5168
사용 가능한 DB 인스턴스 부족 ... 5170
RDS 여유 메모리 부족 문제 ... 5170
MySQL 및 MariaDB 문제 ... 5171
최대 MySQL 및 MariaDB 연결 수 ... 5171
메모리 제한에 대한 호환되지 않는 파라미터 상태 진단 및 해결 ... 5172
읽기 전용 복제본 간 지연 문제 진단 및 해결 ... 5174
MySQL 또는 MariaDB 읽기 복제 오류 진단 및 해결 ... 5176
이진 로깅이 활성화된 상태에서 트리거를 생성하려면 SUPER 권한 필요 ... 5177
특정 시점으로 복원 오류 진단 및 해결 ... 5179
복제 중지 오류 ... 5180
읽기 전용 복제본 만들기 실패 또는 치명적 오류 1236으로 복제 중단 ... 5180
읽기 전용 복제본 복제가 메타데이터 구조를 초기화하지 못함 ... 5181
백업 보존 기간을 0으로 설정할 수 없음 ... 5181
Amazon RDS API 참조 ... 5182
쿼리 API 사용 ... 5182
쿼리 파라미터 ... 5182
쿼리 요청 인증 ... 5183
애플리케이션 문제 해결 ... 5183
오류 검색 ... 5183
문제 해결 팁 ... 5183
문서 기록 ... 5185
이전 업데이트 ... 5322
AWS 용어집 ... 5351

Amazon Relational Database Service(Amazon RDS)란 무엇입니까?

Amazon Relational Database Service(Amazon RDS)는 AWS 클라우드에서 관계형 데이터베이스를 더 쉽게 설치, 운영 및 확장할 수 있는 웹 서비스입니다. 이 서비스는 산업 표준 관계형 데이터베이스를 위한 경제적이고 크기 조절이 가능한 용량을 제공하고 공통 데이터베이스 관리 작업을 관리합니다.

> **Note**
>
> 이 지침은 Amazon Aurora 이외의 Amazon RDS 데이터베이스 엔진에도 적용됩니다. Amazon Aurora 사용에 대한 자세한 내용은 Amazon Aurora 사용 설명서를 참조하세요.

AWS 제품 및 서비스를 처음 사용하는 경우 자세한 내용은 다음 자료를 참조하세요.

- 모든 AWS 제품의 개요는 클라우드 컴퓨팅이란 무엇입니까?를 참조하세요.
- Amazon Web Services는 수많은 데이터베이스 서비스를 제공합니다. AWS에서 사용할 수 있는 다양한 데이터베이스 옵션에 대해 자세히 알아보려면 AWS 데이터베이스 서비스 선택 및 AWS에서 데이터베이스 실행을 참조하세요.

Amazon RDS의 장점

Amazon RDS는 관리형 데이터베이스 서비스로, 대부분의 관리 작업을 담당합니다. Amazon RDS를 사용하면 번거로운 수동 프로세스를 처리할 필요가 없어 애플리케이션과 사용자에게 집중할 수 있습니다.

Amazon RDS는 완전관리형이 아닌 데이터베이스 배포와 비교해서 다음과 같은 주요 이점을 제공합니다.

- 이미 익숙하게 사용하는 IBM Db2, MariaDB, Microsoft SQL Server, MySQL, Oracle Database, PostgreSQL 등의 데이터베이스 엔진을 이용할 수 있습니다.
- Amazon RDS는 백업, 소프트웨어 패치, 자동 장애 감지 및 복구를 관리합니다.
- 자동화된 백업을 설정하거나 고유한 백업 스냅샷을 수동으로 생성할 수 있습니다. 이러한 백업을 사용하여 데이터베이스를 복원할 수 있습니다. Amazon RDS 복원 프로세스는 안정적이고 효율적입니다.

- 기본 DB 인스턴스 및 문제 발생 시 장애 조치를 수행할 수 있는 동기식 보조 DB 인스턴스에서 가용성을 높일 수 있습니다. 읽기 전용 복제본을 사용하여 읽기 조정을 높일 수도 있습니다.
- 데이터베이스 패키지의 보안 외에도 AWS Identity and Access Management(IAM)을 사용하여 사용자 및 권한을 정의하도록 액세스를 제어할 수 있습니다. 데이터베이스를 Virtual Private Cloud(VPC)에 배치하여 데이터베이스를 보호할 수도 있습니다.

Amazon EC2 및 온프레미스 배포와의 책임 비교

대부분의 관계형 데이터베이스 배포에 대해 Amazon RDS를 기본적으로 선택하는 것이 좋습니다. 다음과 같은 대안을 사용하면 소프트웨어와 하드웨어를 관리하는 데 더 많은 시간이 들 수 있다는 단점이 있습니다.

온프레미스 배포

온프레미스 서버를 구매하면 CPU, 메모리, 스토리지 및 IOPS가 함께 번들로 제공됩니다. 사용자가 서버, 운영 체제 및 데이터베이스 소프트웨어를 모두 관리해야 합니다.

Amazon EC2

Amazon Elastic Compute Cloud(Amazon EC2)는 AWS 클라우드에서 확장 가능한 컴퓨팅 용량을 제공합니다. 온프레미스 서버와 달리 CPU, 메모리, 스토리지, IOPS가 분리되어 독립적으로 규모를 조정할 수 있습니다. AWS에서는 하드웨어 계층을 관리하여 온프레미스 데이터베이스 서버를 관리해야 하는 부담을 일정 부분 덜어줍니다.

Amazon EC2에서 데이터베이스를 실행하면 사용자 오류가 발생하기 쉽다는 단점이 있습니다. 예를 들어, 운영 체제 또는 데이터베이스 소프트웨어를 수동으로 업데이트하면 원치 않는 애플리케이션 다운타임을 초래할 수 있습니다. 모든 변경 사항을 확인하여 문제를 파악하고 수정하려면 몇 시간이 걸리기도 합니다.

다음 표에서는 온프레미스 데이터베이스, Amazon EC2 및 Amazon RDS의 관리 모델을 비교합니다.

Feature	온프레미스 관리	Amazon EC2 관리	Amazon RDS 관리
애플리케이션 최적화	고객	고객	고객
확장성	고객	고객	AWS
높은 가용성	고객	고객	AWS

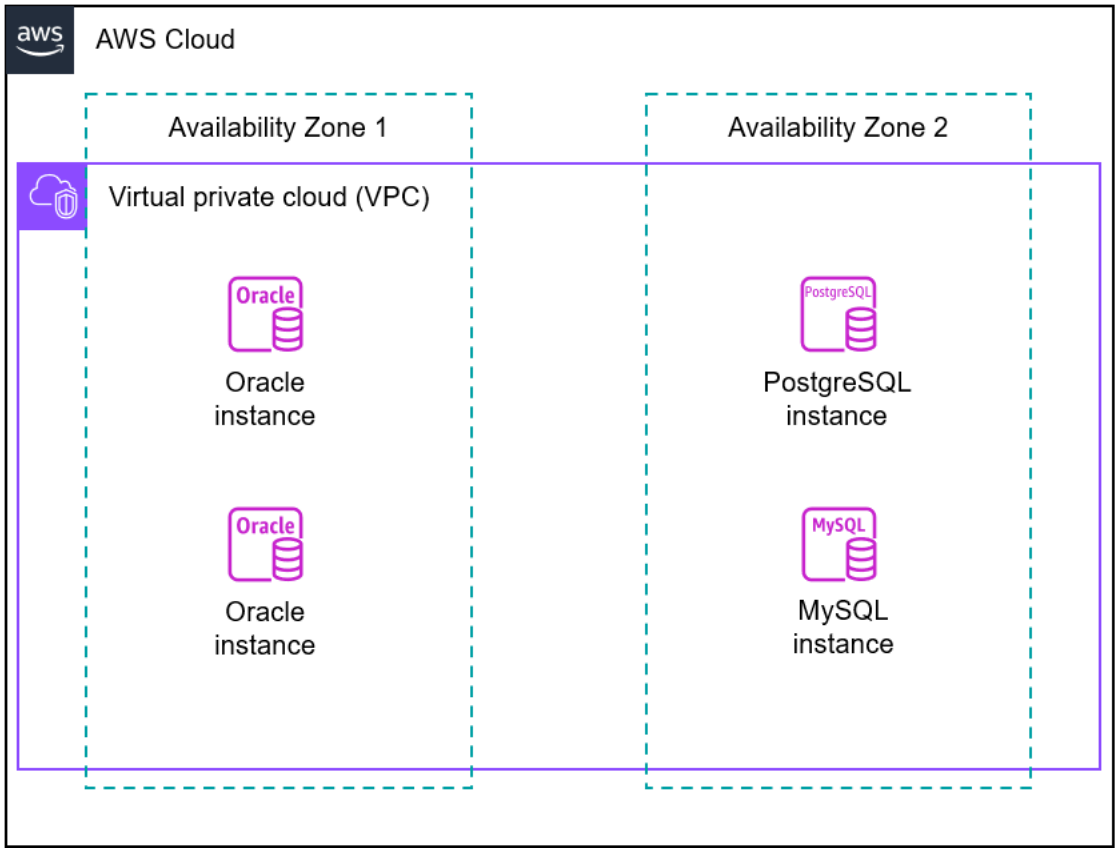
Feature	온프레미스 관리	Amazon EC2 관리	Amazon RDS 관리
데이터베이스 백업	고객	고객	AWS
데이터베이스 소프트웨어 패치	고객	고객	AWS
데이터베이스 소프트웨어 설치	고객	고객	AWS
운영 체제(OS) 패치	고객	고객	AWS
OS 설치	고객	고객	AWS
서버 유지 관리	고객	AWS	AWS
하드웨어 수명	고객	AWS	AWS
전력, 네트워크 및 냉각	Customer	AWS	AWS

Amazon RDS 공동 책임 모델

Amazon RDS는 DB 인스턴스 및 DB 클러스터의 소프트웨어 구성 요소와 인프라 호스팅을 담당합니다. 사용자는 성능을 개선하기 위해 SQL 쿼리를 조정하는 프로세스인 쿼리 튜닝을 담당합니다. 쿼리 성능은 데이터베이스 디자인, 데이터 크기, 데이터 배포, 애플리케이션 워크로드 및 쿼리 패턴에 따라 크게 달라질 수 있습니다. 모니터링 및 튜닝은 RDS 데이터베이스에 대해 사용자가 소유하는 매우 개별화된 프로세스입니다. Amazon RDS 성능 개선 도우미를 비롯한 도구를 사용하여 문제가 있는 쿼리를 식별할 수 있습니다.

Amazon RDS DB 인스턴스

DB 인스턴스는 AWS 클라우드에 있는 격리된 데이터베이스 환경입니다. Amazon RDS의 기본 빌딩 블록은 DB 인스턴스입니다. DB 인스턴스에 사용자가 만든 데이터베이스가 하나 이상 포함될 수 있습니다. 다음 다이어그램은 2개의 가용 영역을 포함하고 각 AZ에 2개의 DB 인스턴스가 포함되어 있는 Virtual Private Cloud(VPC)를 보여줍니다.



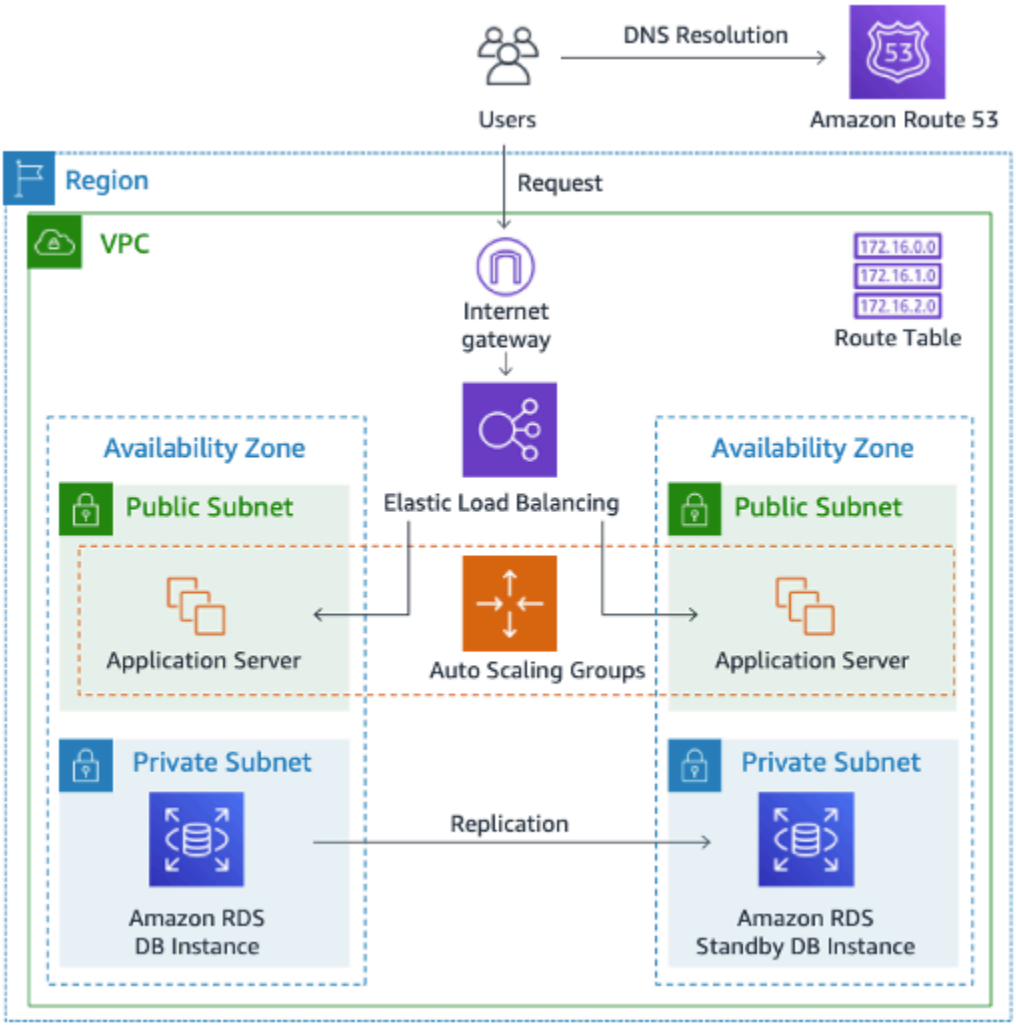
독립 실행형 데이터베이스 인스턴스와 함께 사용하는 것과 동일한 도구 및 애플리케이션을 사용하여 DB 인스턴스에 액세스할 수 있습니다. AWS Command Line Interface(AWS CLI), Amazon RDS API 또는 AWS Management Console을 사용하여 DB 인스턴스를 생성하고 수정할 수 있습니다.

주제

- Amazon RDS 애플리케이션 아키텍처: 사례
- DB 엔진
- DB 인스턴스 클래스
- DB 인스턴스 스토리지
- Amazon Virtual Private Cloud(VPC)의 DB 인스턴스

Amazon RDS 애플리케이션 아키텍처: 사례

다음 이미지는 데이터베이스 스토리지로 Amazon RDS DB 인스턴스를 사용하는 동적 웹 사이트의 일반적인 사용 사례를 보여줍니다.



이전 아키텍처의 주요 구성 요소는 다음과 같습니다.

Elastic Load Balancing

AWS는 Elastic Load Balancing을 통해 사용자 트래픽을 라우팅합니다. 로드 밸런서는 워크로드를 가상 서버와 같은 다수의 컴퓨팅 리소스로 분산합니다. 이 샘플 사용 사례에서 Elastic Load Balancer는 클라이언트 요청을 애플리케이션 서버로 전달합니다.

애플리케이션 서버

애플리케이션 서버는 RDS DB 인스턴스와 상호 작용합니다. AWS의 애플리케이션 서버는 일반적으로 확장 가능한 컴퓨팅 용량을 제공하는 EC2 인스턴스에 호스팅됩니다. 애플리케이션 서버는 동일한 가상 프라이빗 클라우드(VPC) 내에서 서로 다른 가용 영역(AZ)을 사용하는 퍼블릭 서브넷에 있습니다.

RDS DB 인스턴스

EC2 애플리케이션 서버는 RDS DB 인스턴스와 상호 작용합니다. DB 인스턴스는 동일한 Virtual Private Cloud(VPC) 내 서로 다른 가용 영역의 프라이빗 서브넷에 있습니다. 서브넷은 프라이빗이므로, 인터넷에서의 요청은 허용되지 않습니다.

기본 DB 인스턴스는 읽기 복제본이라고 하는 다른 DB 인스턴스로 복제됩니다. 두 DB 인스턴스는 모두 VPC 내의 프라이빗 서브넷에 있으므로 인터넷 사용자가 직접 액세스할 수 없습니다.

DB 엔진

DB 엔진은 DB 인스턴스에서 실행되는 특정 관계형 데이터베이스 소프트웨어입니다. Amazon RDS는 다음 데이터베이스 엔진을 지원합니다.

- IBM Db2

  자세한 내용은 Amazon RDS for Db2 섹션을 참조하세요.
- MariaDB

  자세한 내용은 Amazon RDS for MariaDB 섹션을 참조하세요.
- Microsoft SQL Server

  자세한 내용은 Amazon RDS for Microsoft SQL Server 섹션을 참조하세요.
- MySQL

  자세한 내용은 Amazon RDS for MySQL 섹션을 참조하세요.
- Oracle Database

  자세한 내용은 Amazon RDS for Oracle 섹션을 참조하세요.
- PostgreSQL

  자세한 내용은 Amazon RDS for PostgreSQL 섹션을 참조하세요.

각 DB 엔진에는 지원되는 고유한 기능이 있으며, DB 엔진의 각 버전에는 특정 기능이 포함될 수 있습니다. Amazon RDS 기능에 대한 지원은 AWS 리전 및 각 DB 엔진의 특정 버전에 따라 다릅니다. 다양한 엔진 버전 및 리전의 기능 지원을 확인하려면 AWS 리전 및 DB 엔진별 Amazon RDS에서 지원되는 기능 섹션을 참조하세요.

또한 DB 엔진마다 관리하는 데이터베이스의 동작을 제어하는 DB 파라미터 그룹에 파라미터 집합이 있습니다. 파라미터 그룹에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하세요.

DB 인스턴스 클래스

DB 인스턴스 클래스는 DB 인스턴스의 컴퓨팅 및 메모리 용량을 결정하며, DB 인스턴스 클래스는 DB 인스턴스 클래스 유형과 크기로 구성됩니다. Amazon RDS는 다음 인스턴스 클래스 유형을 지원합니다. 여기서 별표(*)는 세대, 선택적 속성 및 크기를 나타냅니다.

- 범용 – db.m*
- 메모리 최적화 – db.z*, db.x*, db.r*
- 컴퓨팅 최적화 – db.c*
- 성능 버스트 가능 – db.t*

인스턴스 클래스마다 서로 다른 컴퓨팅, 메모리 및 스토리지 기능을 제공합니다. 예를 들어, db.m7g는 AWS Graviton3 프로세서로 구동되는 7세대 범용 DB 인스턴스 클래스 유형입니다. DB 인스턴스를 생성할 때 db.m7g.2xlarge와 같은 DB 인스턴스 클래스를 지정하게 됩니다. 여기서 2xlarge는 크기입니다. 다양한 인스턴스 클래스의 하드웨어 사양에 대한 자세한 내용은 에 대한 DB 인스턴스 클래스의 하드웨어 사양 섹션을 참조하세요.

요구 사항에 가장 적합한 DB 인스턴스 클래스를 선택할 수 있습니다. 시간이 지나면서 요구 사항이 바뀌면 DB 인스턴스 클래스를 변경할 수 있습니다. 예를 들어, db.m7g.2xlarge 인스턴스 규모를 db.m7g.4xlarge로 조정할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요.

> **Note**
>
> DB 인스턴스 클래스에 대한 요금 정보는 Amazon RDS 제품 페이지의 요금 단원을 참조하세요.

DB 인스턴스 스토리지

Amazon EBS는 내구성이 있는 블록 수준 스토리지 볼륨을 제공하여 실행 중인 인스턴스에 연결하는 것이 가능합니다. DB 인스턴스 스토리지는 다음과 같은 유형으로 제공됩니다.

- 범용(SSD)

이 비용 효율적 스토리지 유형은 중간 크기 DB 인스턴스에서 실행하는 광범위한 워크로드에 이상적입니다. 범용 스토리지는 개발 및 테스트 환경에 가장 적합합니다.

- 프로비저닝된 IOPS(PIOPS)

  이 스토리지 유형은 낮은 I/O 지연 시간과 일관된 I/O 처리량을 요구하는 I/O 집약적인 워크로드, 특히 데이터베이스 워크로드의 요구를 충족하도록 설계되었습니다. 프로비저닝된 IOPS 스토리지는 프로덕션 환경에 가장 적합합니다.

- Magnetic

  Amazon RDS는 이전 버전과의 호환성을 위해 마그네틱 스토리지를 지원합니다. 새 스토리지가 필요할 경우 범용 SSD 또는 프로비저닝된 IOPS SSD를 사용하는 것이 좋습니다.

스토리지 유형은 성능 특성과 가격이 다릅니다. 데이터베이스 요건에 따라 스토리지 성능과 비용을 조정할 수 있습니다.

DB 인스턴스는 각각 스토리지 유형과 지원하는 데이터베이스 엔진에 따라 최소/최대 스토리지 요구 사항이 있습니다. 충분한 스토리지를 보유하여 데이터베이스를 확장할 수 있는 여유를 확보하는 것이 중요합니다. 또한 스토리지가 충분하면 콘텐츠를 작성하거나 항목을 기록할 수 있는 기능을 DB 엔진에 도입할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 섹션을 참조하세요.

Amazon Virtual Private Cloud(VPC)의 DB 인스턴스

Amazon Virtual Private Cloud(Amazon VPC) 서비스를 사용해 가상 사설 클라우드(VPC)에서 DB 인스턴스를 실행할 수 있습니다. VPC를 사용하면 가상 네트워킹 환경을 완벽하게 제어할 수 있습니다. 자기만의 IP 주소 범위를 선택하고, 서브넷을 생성하고, 라우팅 및 액세스 제어 목록을 구성할 수 있습니다.

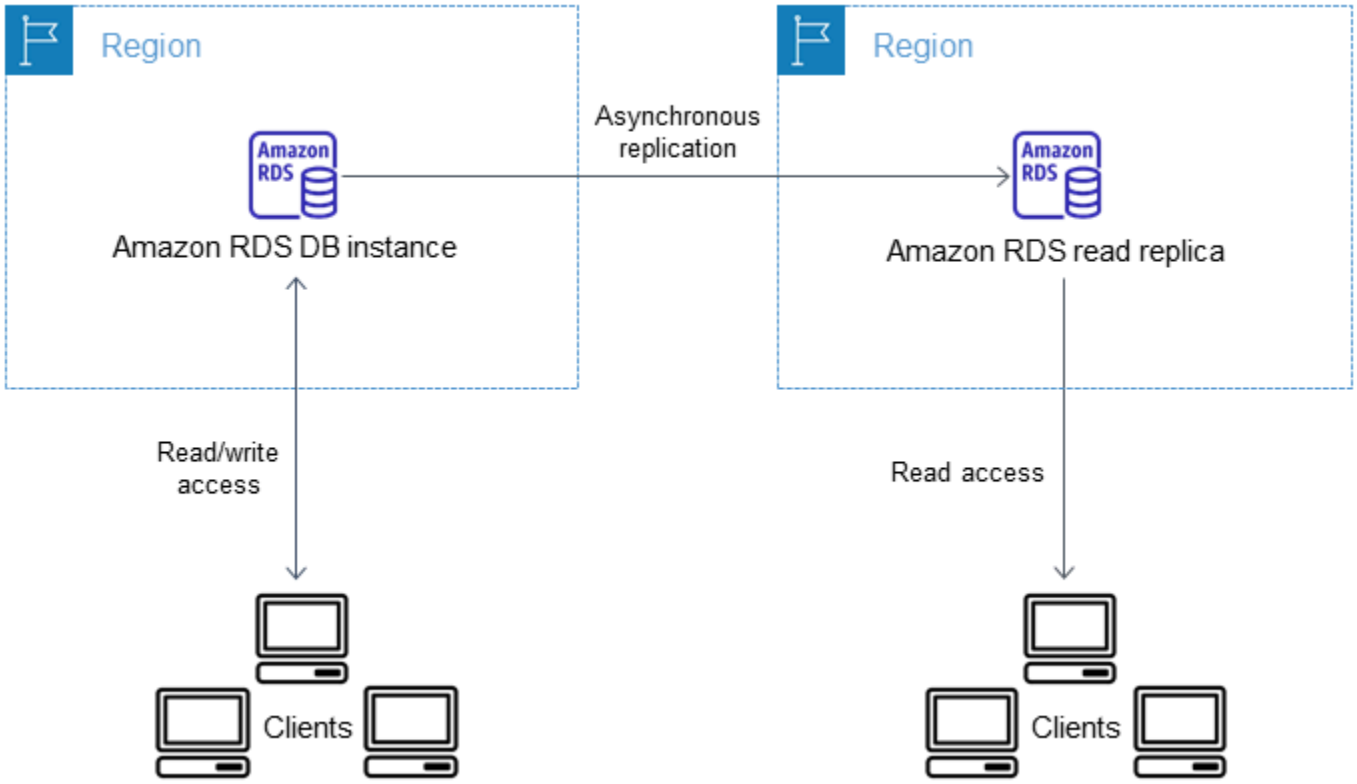
Amazon RDS의 기본 기능은 VPC 실행 중인지 여부에 관계없이 동일합니다. Amazon RDS는 백업, 소프트웨어 패치, 자동 장애 감지 및 복구를 관리합니다. VPC에서 DB 인스턴스를 실행하는 데는 추가 비용이 들지 않습니다. RDS와 Amazon VPC를 함께 사용하는 방법에 대한 자세한 내용은 Amazon VPC 및 Amazon RDS 단원을 참조하십시오.

Amazon RDS는 DB 인스턴스를 기준으로 시간을 동기화하는 데 NTP(Network Time Protocol)를 사용합니다.

AWS 리전 및 가용 영역

Amazon 클라우드 컴퓨팅 리소스는 전 세계 여러 리전의 가용성이 높은 데이터 센터 시설에 하우징됩니다(예: 북미, 유럽 또는 아시아). 각 데이터 센터 위치를 AWS 리전이라고 합니다. Amazon RDS를 사용하여 여러 리전에서 DB 인스턴스를 생성할 수 있습니다.

다음 시나리오는 한 리전의 RDS DB 인스턴스가 다른 리전의 대기 DB 인스턴스에 비동기적으로 복제되는 것을 보여줍니다. 한 리전을 사용할 수 없게 되더라도 다른 리전의 인스턴스는 계속 사용할 수 있습니다.



가용 영역

AWS 리전마다 가용 영역 또는 AZ라는 고유한 위치가 여러 개 포함됩니다. 각 가용 영역은 다른 가용 영역에서 발생한 장애에서 격리되도록 설계되었습니다. 각 가용 영역은 같은 AWS 리전에 있는 다른 가용 영역에 대해 저렴하고 지연 시간이 짧은 네트워크 연결을 제공하도록 설계되었습니다. 별도의 가용 영역에서 DB 인스턴스를 시작함으로써 단일 위치에서 장애가 발생할 경우 애플리케이션을 보호할 수 있습니다. 자세한 내용은 리전, 가용 영역 및 로컬 영역 섹션을 참조하세요.

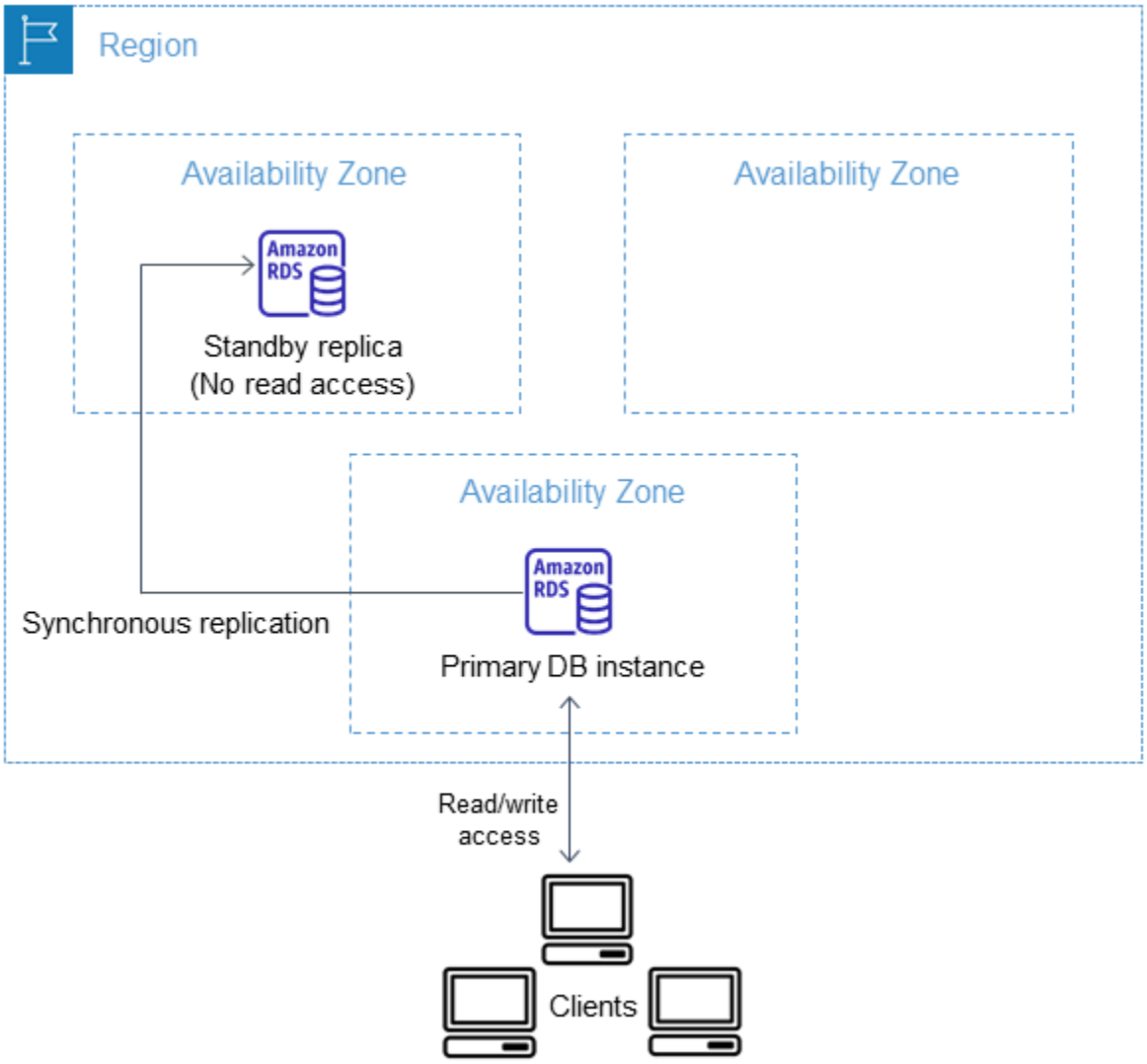
다중 AZ 배포

여러 가용 영역에서 DB 인스턴스를 실행할 수 있습니다. 다중 AZ 배포라는 옵션입니다. 이 옵션을 선택하면 Amazon은 다른 AZ에서 하나 이상의 보조 대기 DB 인스턴스를 자동으로 프로비저닝하고 유지합니다. 기본 DB 인스턴스는 가용 영역 전체에서 각 보조 DB 인스턴스로 복제됩니다.

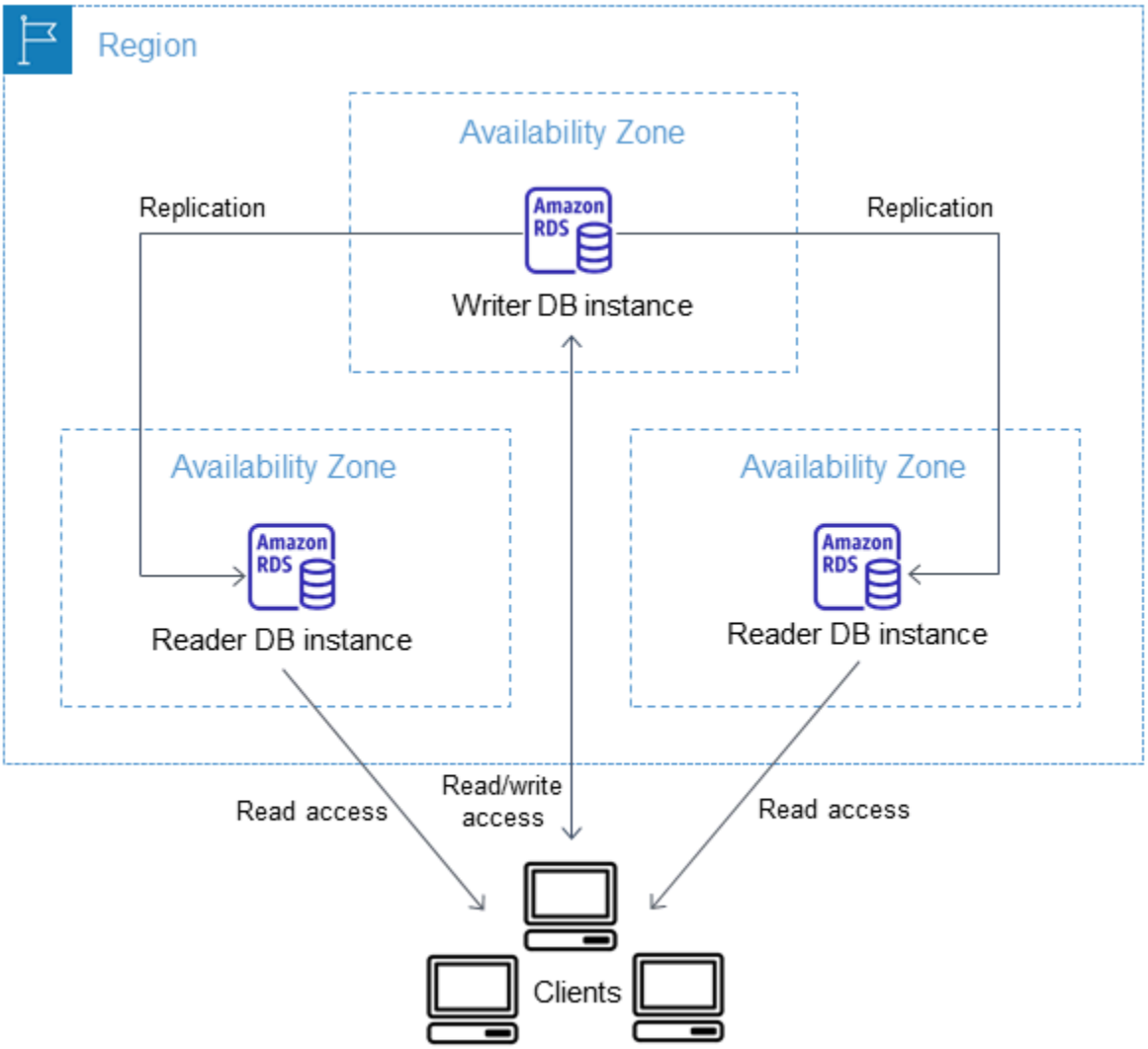
다중 AZ 배포는 다음과 같은 이점을 제공합니다.

- 데이터 중복성 및 장애 조치 지원 제공
- I/O 작동 중지 제거
- 시스템 백업 중 지연 시간 급증 최소화
- 보조 DB 인스턴스에서 읽기 트래픽 처리(다중 AZ DB 클러스터 배포만 해당)

다음 다이어그램은 Amazon RDS가 다른 가용 영역에서 동기식 대기 복제본을 자동으로 프로비저닝하고 유지 관리하는 다중 AZ DB 인스턴스 배포를 보여줍니다. 복제본 데이터베이스는 읽기 트래픽을 제공하지 않습니다.



다음 다이어그램은 동일한 AWS 리전에 있는 3개의 개별 가용 영역에 라이터 DB 인스턴스와 2개의 리더 DB 인스턴스가 있는 다중 AZ DB 클러스터 배포를 보여줍니다. 3개의 DB 인스턴스 모두 읽기 트래픽을 처리할 수 있습니다.

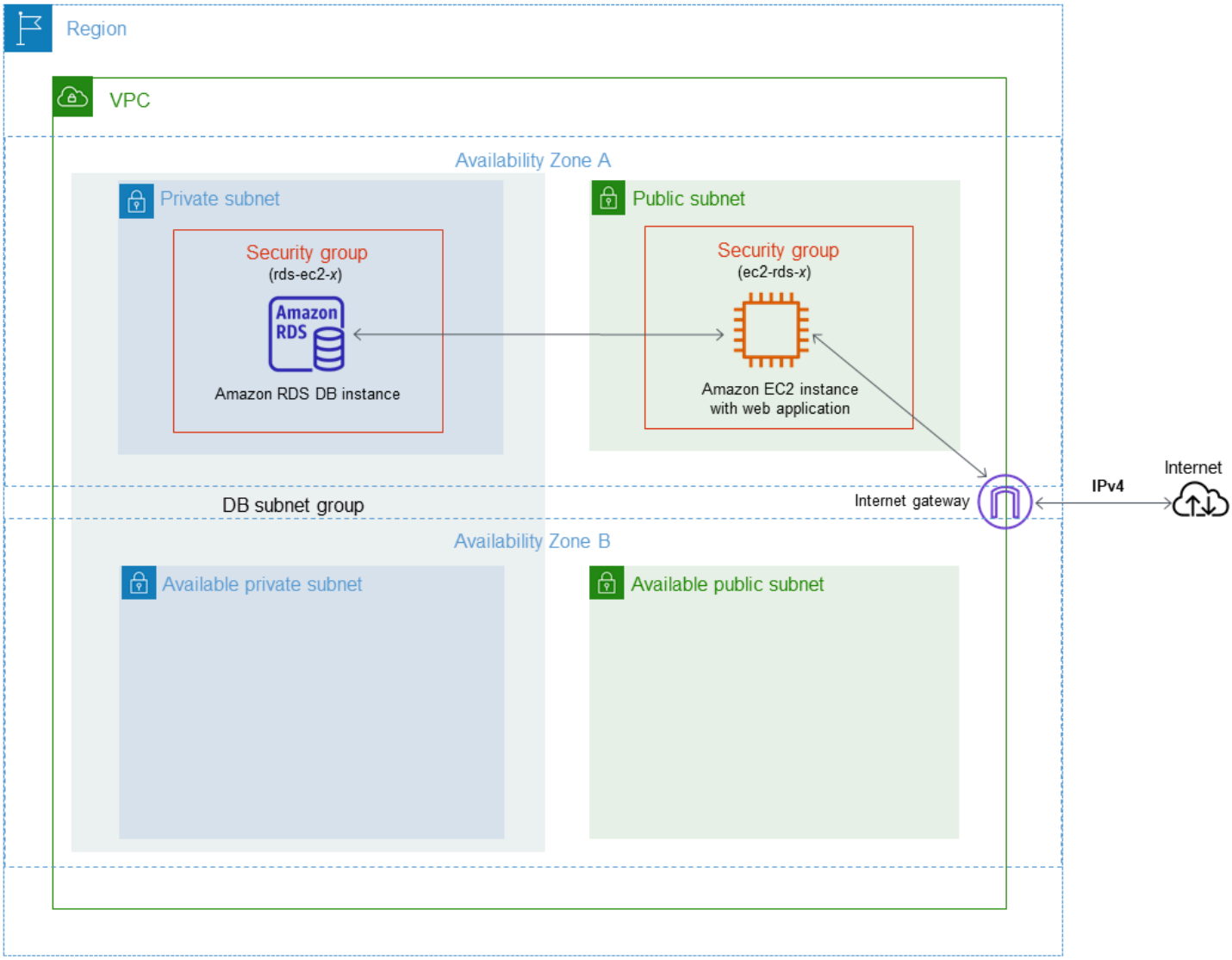


자세한 내용은 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 섹션을 참조하세요.

보안 그룹을 통한 액세스 제어

보안 그룹은 지정한 IP 주소 범위 또는 Amazon EC2 인스턴스에 대한 액세스를 허용하여 DB 인스턴스에 대한 액세스를 제어합니다. 보안 그룹을 하나 이상의 DB 인스턴스에 적용할 수 있습니다.

VPC에서 DB 인스턴스의 일반적인 용도는 동일한 VPC에 있는 애플리케이션 서버와 데이터를 공유하는 것입니다. 다음 예제에서는 `ec2-rds-`*x* VPC 보안 그룹을 사용하여 클라이언트 애플리케이션의 IP 주소를 소스로 사용하는 인바운드 규칙을 정의합니다. 애플리케이션 서버는 이 보안 그룹에 속합니다. `rds-ec2-`*x*라는 이름의 보조 보안 그룹은 `ec2-rds-`*x*를 소스로 지정하고 RDS DB 인스턴스에 연결합니다. 보안 그룹 규칙에 따르면 클라이언트 애플리케이션은 DB 인스턴스에 직접 액세스할 수 없지만, EC2 인스턴스는 DB 인스턴스에 액세스할 수 있습니다.



보안 그룹에 대한 자세한 내용은 Amazon RDS의 보안를(을) 참조하세요.

Amazon RDS 모니터링

모니터링은 Amazon RDS 및 기타 AWS 솔루션의 신뢰성, 가용성 및 성능을 유지하는 데 있어서 중요한 부분입니다. AWS에서는 Amazon RDS를 모니터링하고, 이상이 있을 때 이를 보고하고, 적절할 경우 자동 조치를 취할 수 있도록 여러 모니터링 도구를 제공합니다.

다양한 자동 및 수동 도구를 사용하여 DB 인스턴스의 성능과 상태를 추적할 수 있습니다.

Amazon RDS DB 인스턴스 상태 및 권장 사항

Amazon RDS 콘솔, AWS CLI 또는 RDS API를 사용하여 인스턴스의 현재 상태에 대한 세부 정보를 봅니다. 또한 DB 인스턴스, 읽기 전용 복제본, DB 파라미터 그룹과 같은 데이터베이스 리소스에 대한 자동화된 권장 사항에 응답할 수 있습니다. 자세한 내용은 Amazon RDS의 권장 사항 섹션을 참조하세요.

Amazon RDS에 대한 Amazon CloudWatch 지표

Amazon CloudWatch 서비스를 사용하여 DB 인스턴스의 성능 및 상태를 모니터링할 수 있습니다. Amazon RDS 콘솔에 CloudWatch 성능 차트가 표시됩니다. Amazon RDS는 각각의 활성 데이터베이스 인스턴스에 대해 1분마다 CloudWatch로 지표를 자동 전송합니다. CloudWatch에서 Amazon RDS 지표에 대한 추가 요금은 표시되지 않습니다.

Amazon CloudWatch 경보를 사용하면 특정 기간 동안 단일 Amazon RDS 지표를 볼 수 있습니다. 그런 다음 설정한 임계값과 지표 값을 비교하여 하나 이상의 작업을 수행할 수 있습니다. 자세한 내용은 Amazon CloudWatch로 Amazon RDS 지표 모니터링 섹션을 참조하세요.

Amazon RDS 성능 개선 도우미 및 운영 체제 모니터링

성능 개선 도우미는 데이터베이스의 부하를 평가하고 조치를 취할 시점과 위치를 결정합니다. 자세한 내용은 성능 개선 도우미를 통한 Amazon RDS 모니터링 섹션을 참조하세요. Amazon RDS 확장 모니터링은 운영 체제에 대한 지표를 실시간으로 확인합니다. 자세한 내용은 Enhanced Monitoring을 사용하여 OS 지표 모니터링 섹션을 참조하세요.

통합 AWS 서비스

Amazon RDS는 Amazon EventBridge, Amazon CloudWatch Logs, Amazon DevOps Guru와 통합됩니다. 자세한 내용은 Amazon RDS 인스턴스에서 지표 모니터링 섹션을 참조하세요.

Amazon RDS에 대한 사용자 인터페이스

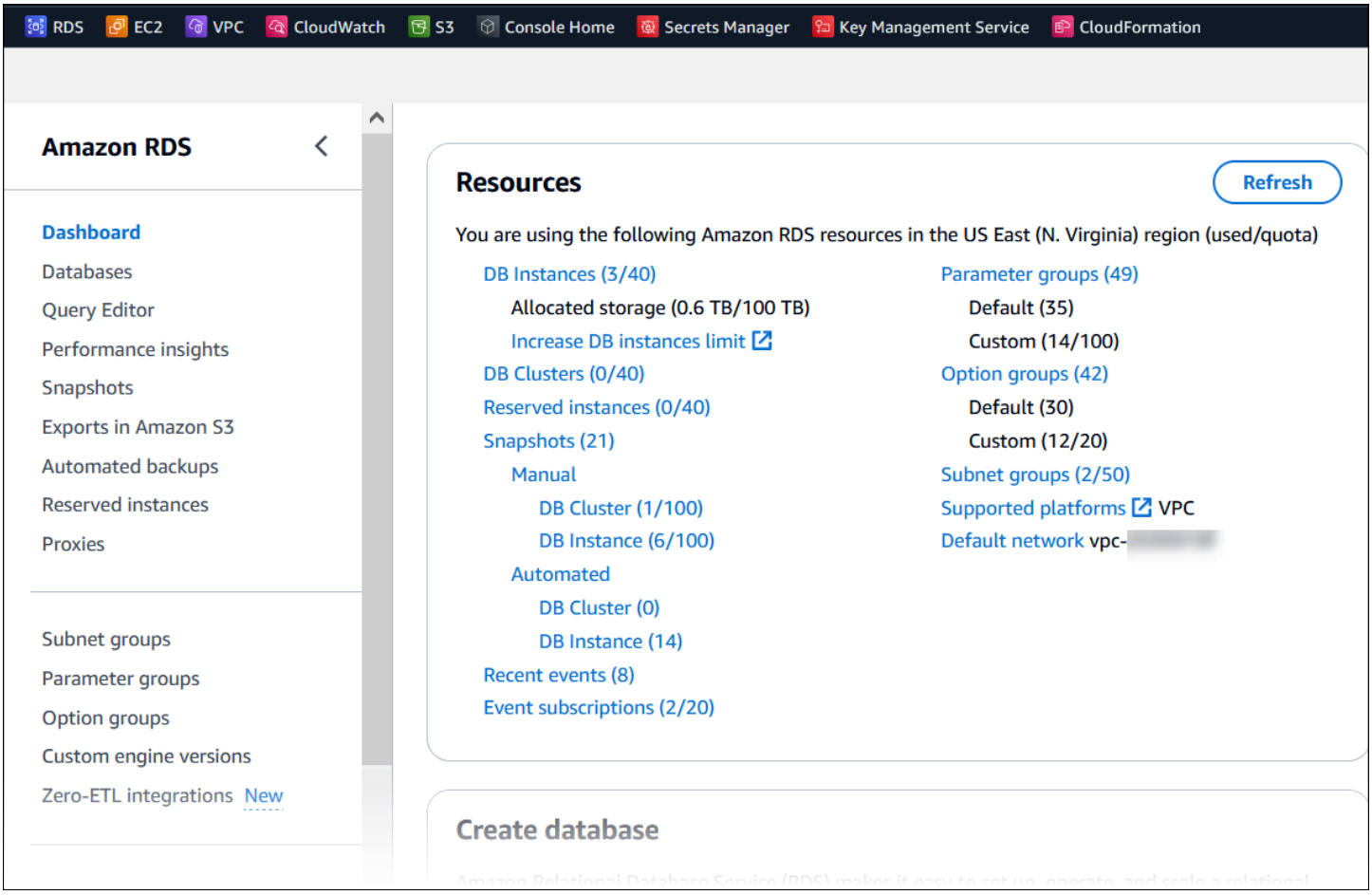
Amazon RDS와 상호 작용하는 방법에는 여러 가지가 있습니다.

주제

- AWS Management Console
- 명령줄 인터페이스
- Amazon RDS API

AWS Management Console

AWS Management Console은 간단한 웹 기반 사용자 인터페이스입니다. 콘솔에서 프로그래밍 없이 DB 인스턴스를 관리할 수 있습니다. Amazon RDS 콘솔에 액세스하려면 AWS Management Console에 로그인하고 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.



명령줄 인터페이스

AWS Command Line Interface(AWS CLI)를 사용하여 Amazon RDS API에 대화식으로 액세스할 수 있습니다. AWS CLI를 설치하려면 AWS 명령줄 인터페이스 설치를 참조하세요. RDS에 대해 AWS CLI 사용을 시작하려면 Amazon RDS용 AWS Command Line Interface 참조를 확인하세요.

Amazon RDS API

개발자라면 API를 사용하여 프로그래밍 방식으로 Amazon RDS에 액세스할 수 있습니다. 자세한 내용은 Amazon RDS API 참조 섹션을 참조하세요.

애플리케이션을 개발하는 경우에는 AWS 소프트웨어 개발 키트(SDK) 중 하나를 사용하는 것이 좋습니다. AWS SDK가 인증, 재시도 로직, 오류 처리 등 낮은 레벨 정보를 처리하기 때문에 개발자는 애플리케이션 로직에 더욱 집중할 수 있습니다. AWS SDK는 다양한 언어로 제공됩니다. 자세한 정보는 Amazon Web Services용 도구를 참조하십시오.

AWS그 밖에도 는 라이브러리, 샘플 코드, 자습서 등 보다 쉽게 시작하는 데 도움이 되는 리소스를 제공합니다. 자세한 정보는 샘플 코드 및 라이브러리를 참조하십시오.

Amazon RDS에 대한 요금이 부과되는 방법

Amazon RDS를 사용하는 경우 온디맨드 DB 인스턴스 또는 예약된 DB 인스턴스를 선택하여 사용할 수 있습니다. 자세한 내용은 Amazon RDS에 대한 DB 인스턴스 결제 섹션을 참조하세요.

Amazon RDS 요금에 대한 자세한 정보는 Amazon RDS 제품 페이지를 참조하십시오.

Amazon RDS의 AWS 프리 티어

Amazon RDS에서 다음 엔진 및 DB 인스턴스 클래스에 AWS 프리 티어를 사용할 수 있습니다.

- 엔진 유형 - MariaDB, MySQL, PostgreSQL 또는 SQL Server Express Edition
- DB 인스턴스 클래스
  - t3.micro – 모든 엔진 유형
  - t4g.micro – SQL Server Express Edition을 제외한 모든 엔진 유형

Amazon RDS에서 AWS 프리 티어로 다음 기능, 리소스 및 작업을 사용할 수 없습니다.

- Amazon RDS 사용자 지정 엔진 옵션
- 예약 DB 인스턴스
- 단일 AZ 이외의 배포 옵션
- 스냅샷 마이그레이션
- 쿼리 편집기
- Aurora 읽기 전용 복제본 생성

이러한 제한은 AWS 프리 티어 프로그램이 발전함에 따라 변경될 수 있습니다. AWS 프리 티어에 대한 자세한 내용은 Explore AWS services with AWS 프리 티어 섹션을 참조하세요. Amazon RDS 프리 티어에 대한 자세한 내용은 Amazon RDS 프리 티어를 참조하세요.

2025년 7월 17일 이전에 AWS 프리 티어 고객이었던 경우 12개월의 무료 사용 기간이 만료될 때까지 Amazon RDS 프리 티어를 계속 사용할 수 있습니다. 이 기간 동안 다음 사용 자격을 유지합니다.

- Amazon RDS에서 MySQL, MariaDB 또는 PostgreSQL을 실행하는 단일 AZ db.t3.micro 및 db.t4g.micro 인스턴스 클래스 매월 750시간
- Amazon RDS에서 SQL Server Express Edition을 실행하는 db.t3.micro 인스턴스 클래스 매월 750시간

2025년 7월 17일 후에는 AWS에서 생성하는 모든 새 AWS 프리 티어 리소스가 새 AWS 프리 티어 오퍼링을 사용합니다.

다음 단계

이전 단계에서는 RDS에서 제공하는 기본 인프라 구성 요소를 소개했습니다. 다음으로 무엇을 해야 할까요?

시작하기

Amazon RDS 시작하기의 지침에 따라 DB 인스턴스를 생성하세요.

데이터베이스 엔진과 관련된 주제

다음 섹션에서 특정 DB 엔진별 정보를 검토할 수 있습니다.

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

Amazon RDS DB 인스턴스

DB 인스턴스는 클라우드에서 실행하는 격리된 데이터베이스 환경입니다. 이것은 Amazon RDS의 기본 구성 요소입니다. DB 인스턴스에는 여러 사용자가 만든 데이터베이스가 포함될 수 있으며, 독립 실행형 데이터베이스 인스턴스에 액세스할 때 사용하는 도구 및 애플리케이션을 사용해 액세스할 수 있습니다. AWS 명령줄 도구, Amazon RDS API 작업 또는 AWS Management Console을 사용해 간단히 DB 인스턴스를 만들고 수정할 수 있습니다.

> **Note**
>
> Amazon RDS는 표준 SQL 클라이언트 애플리케이션을 사용하여 데이터베이스에 대한 액세스를 지원합니다. Amazon RDS는 RDS Custom을 제외하고 호스트에 대한 직접적인 액세스를 허용하지 않습니다.

최대 40개의 Amazon RDS DB 인스턴스를 보유할 수 있으며, 다음과 같은 제한 사항이 있습니다.

- '라이선스 포함' 모델에서 각 SQL Server 에디션(Enterprise, Standard, Web, 및 Express)의 경우 10개
- '라이선스 포함' 모델에서 Oracle의 경우 10개
- 'BYOL(bring-your-own-license)' 모델에서 Db2의 경우 40개
- MySQL, MariaDB 또는 PostgreSQL의 경우 40개
- 'BYOL4(bring-your-own-license)' 라이선싱 모델에서 Oracle의 경우 40개

> **Note**
>
> 애플리케이션에 더 많은 DB 인스턴스가 요구되는 경우 이 양식을 사용하여 추가 DB 인스턴스를 요청할 수 있습니다.

DB 인스턴스마다 DB 인스턴스 식별자가 있습니다. 이 식별자는 고객이 제공하는 이름으로, Amazon RDS API 및 AWS CLI 명령과 상호 작용하는 경우 DB 인스턴스를 고유하게 식별합니다. DB 인스턴스 식별자는 한 AWS 리전 내의 해당 고객에 대해 고유해야 합니다.

DB 인스턴스 식별자는 RDS에서 인스턴스에 할당된 DNS 호스트 이름의 일부로 구성되어 있습니다. 예를 들어 db1을 DB 인스턴스 식별자로 지정하는 경우, RDS는 인스턴스의 DNS 엔드포인트를 자동

으로 할당합니다. 일례로 엔드포인트 *db1*.*abcdefghijkl*.*us-east-1*.rds.amazonaws.com은 *db1*이며 인스턴스 ID입니다.

예제에서 엔드포인트 *db1*.*abcdefghijkl*.*us-east-1*.rds.amazonaws.com은 문자열 *abcdefghijkl*이며 특정 AWS 리전 및 AWS 계정가 조합된 고유한 식별자입니다. 예제에서 식별자 *abcdefghijkl*은 RDS에서 내부적으로 생성되며 지정된 리전 및 계정 조합은 변경되지 않습니다. 따라서 이 리전의 모든 DB 인스턴스는 동일한 고정 식별자를 공유합니다. 고정 식별자의 다음 기능을 고려해 보세요.

- DB 인스턴스의 이름을 변경하면 엔드포인트는 다르지만 고정 식별자는 동일합니다. 예를 들어 *db1*의 이름을 *renamed-db1*로 바꾸면 새 인스턴스 엔드포인트는 *renamed-db1*.*abcdefghijkl*.*us-east-1*.rds.amazonaws.com입니다.
- 동일한 DB 인스턴스 식별자로 DB 인스턴스를 삭제하고 다시 만들면 엔드포인트는 동일합니다.
- 동일한 계정을 사용하여 다른 리전에 DB 인스턴스를 만들 경우, *db2*.*mnopqrstuvwx*.*us-west-1*.rds.amazonaws.com처럼 리전이 다르기 때문에 내부적으로 생성되는 식별자도 달라집니다.

각 DB 인스턴스는 데이터베이스 엔진을 지원합니다. Amazon RDS는 현재 Db2, MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL Server, Amazon Aurora 데이터베이스 엔진을 지원합니다.

DB 인스턴스를 만들 때, 일부 데이터베이스 엔진의 경우 데이터베이스 이름을 지정해야 합니다. 한 DB 인스턴스에서 여러 개의 데이터베이스 또는 여러 스키마를 포함하는 단일 Oracle 데이터베이스를 호스팅할 수 있습니다. 데이터베이스의 이름 값은 데이터베이스 엔진에 따라 다음과 같이 달라집니다.

- Db2 데이터베이스 엔진의 경우, 데이터베이스 이름은 DB 인스턴스에서 호스팅되는 데이터베이스의 이름입니다. 이 필드는 선택 사항입니다. 나중에 `rdsadmin.create_database` 저장 프로시저를 직접 호출하여 데이터베이스를 만들 수 있습니다. 자세한 내용은 the section called “데이터베이스 생성” 섹션을 참조하세요.
- MySQL 및 MariaDB 데이터베이스 엔진의 경우, 데이터베이스 이름은 DB 인스턴스에서 호스팅하는 데이터베이스의 이름입니다. 동일한 DB 인스턴스가 호스팅하는 데이터베이스는 해당 인스턴스 내부에서 고유 이름이 부여되어야 합니다.
- Oracle 데이터베이스 엔진의 경우, 데이터베이스 이름을 사용하여 ORACLE_SID의 값을 설정하며, 이는 Oracle RDS 인스턴스에 연결할 때 제공해야 합니다.
- Microsoft SQL Server 데이터베이스 엔진의 경우, 데이터베이스 이름은 지원되는 파라미터가 아닙니다.

- PostgreSQL 데이터베이스 엔진의 경우, 데이터베이스 이름은 DB 인스턴스에서 호스팅되는 데이터베이스의 이름입니다. DB 인스턴스를 만들 때 데이터베이스 이름을 지정할 필요가 없습니다. 동일한 DB 인스턴스가 호스팅하는 데이터베이스는 해당 인스턴스 내부에서 고유 이름이 부여되어야 합니다.

Amazon RDS는 DB 인스턴스를 생성하는 중에 DB 인스턴스의 마스터 사용자 계정을 생성합니다. 이 마스터 사용자는 데이터베이스 생성을 비롯하여 마스터 사용자가 생성하는 테이블의 생성, 삭제, 선택 및 업데이트 및 삽입 작업을 수행할 수 있는 권한을 가집니다. DB 인스턴스를 만들 때 마스터 사용자 암호를 설정해야 하지만 AWS CLI, Amazon RDS API 작업 또는 AWS Management Console을 사용하여 이를 언제든지 변경할 수 있습니다. 또한 SQL 표준 명령을 사용하여 마스터 사용자 암호를 변경하고 사용자를 관리할 수도 있습니다.

> **Note**
>
> 이 지침은 Aurora 이외의 다른 Amazon RDS 데이터베이스 엔진에도 적용됩니다. Amazon Aurora 사용에 대한 자세한 정보는 Amazon Aurora 사용 설명서를 참조하십시오.

DB 인스턴스 클래스

DB 인스턴스 클래스는 Amazon RDS DB 인스턴스의 컴퓨팅 및 메모리 용량을 결정합니다. 필요한 DB 인스턴스 클래스는 DB 인스턴스의 처리력 및 메모리 요구 사항에 따라 다릅니다.

DB 인스턴스 클래스는 DB 인스턴스 클래스 유형과 크기로 구성됩니다. 예를 들어 db.r6g는 AWS Graviton2 프로세서로 구동되는 메모리 최적화 DB 인스턴스 클래스 유형입니다. db.r6g 인스턴스 유형 내에서 db.r6g.2xlarge는 DB 인스턴스 클래스입니다. 이 클래스의 크기는 2xlarge입니다.

인스턴스 클래스 요금에 대한 자세한 내용은 Amazon RDS 요금을 참조하세요.

DB 인스턴스 클래스 유형, 지원되는 DB 엔진, 지원되는 AWS 리전, DB 인스턴스 클래스 변경, RDS for Oracle용 프로세서 구성 또는 DB 인스턴스 클래스의 하드웨어 사양에 대한 자세한 내용은 다음 섹션을 참조하세요.

주제

- DB 인스턴스 클래스 유형
- DB 인스턴스 클래스에 지원되는 DB 엔진
- AWS 리전에서 DB 인스턴스 클래스 지원 확인
- DB 인스턴스 클래스 변경
- RDS for Oracle에서 DB 인스턴스 클래스의 프로세서 구성
- 에 대한 DB 인스턴스 클래스의 하드웨어 사양

DB 인스턴스 클래스 유형

Amazon RDS는 다음 사용 사례를 위한 DB 인스턴스 클래스를 지원합니다.

- 범용
- 메모리 최적화
- 컴퓨팅 최적화
- 버스트 가능한 성능
- 최적화된 읽기

Amazon EC2 인스턴스 유형에 대한 자세한 내용은 Amazon EC2 설명서의 인스턴스 유형을 참조하세요.

범용 인스턴스 클래스 유형

다음은 사용 가능한 범용 DB 인스턴스 클래스입니다.

- db.m8g - AWS Graviton4 프로세서로 구동되는 범용 DB 인스턴스 클래스입니다. 이러한 인스턴스 클래스는 광범위한 범용 워크로드에 균형 잡힌 컴퓨팅, 메모리 및 네트워킹을 제공합니다. 7세대 AWS Graviton3 기반 M7g 인스턴스에 비해 이러한 새로운 클래스는 최대 3배 더 많은 vCPU 및 메모리를 통해 더 큰 인스턴스 크기를 제공합니다.

  AWS Graviton4 프로세서로 구동되는 DB 인스턴스 클래스 중 하나를 사용하도록 DB 인스턴스를 수정할 수 있습니다. 이렇게 하려면 다른 DB 인스턴스 수정과 동일한 단계를 완료해야 합니다.
- db.m7i - 4세대 인텔 제온 스케일러블 프로세서로 작동되는 범용 DB 인스턴스 클래스 db.m7i는 SAP 인증을 받았으며 더 큰 인스턴스 크기 또는 높은 연속 CPU 사용량이 필요한 엔터프라이즈 애플리케이션을 지원하는 데 적합합니다. 이러한 인스턴스 클래스는 광범위한 범용 워크로드에 균형 잡힌 컴퓨팅, 메모리 및 네트워킹을 제공합니다. 이 인스턴스 클래스 유형은 최대 40,000Mbps EBS 대역폭과 최대 50Gbps 네트워크 대역폭을 제공합니다. db.m7i 인스턴스는 db.m6i 인스턴스에 비해 최대 15% 더 나은 가격 성능을 제공합니다.
- db.m7g - AWS Graviton3 프로세서로 구동되는 범용 DB 인스턴스 클래스. 이러한 인스턴스 클래스는 광범위한 범용 워크로드에 균형 잡힌 컴퓨팅, 메모리 및 네트워킹을 제공합니다.

  AWS Graviton3 프로세서로 구동되는 DB 인스턴스 클래스 중 하나를 사용하도록 DB 인스턴스를 수정할 수 있습니다. 이렇게 하려면 다른 DB 인스턴스 수정과 동일한 단계를 완료해야 합니다.
- db.m6g - AWS Graviton2 프로세서로 구동되는 범용 DB 인스턴스 클래스 이러한 인스턴스는 광범위한 범용 워크로드에 균형 잡힌 컴퓨팅, 메모리 및 네트워킹을 제공합니다. db.m6gd 인스턴스 클래스는 빠르고 지연 시간이 짧은 로컬 스토리지가 필요한 애플리케이션용 로컬 NVMe 기반 SSD 블록 수준 스토리지를 제공합니다.

  AWS Graviton2 프로세서로 구동되는 DB 인스턴스 클래스 중 하나를 사용하도록 DB 인스턴스를 수정할 수 있습니다. 이렇게 하려면 다른 DB 인스턴스 수정과 동일한 단계를 완료해야 합니다.
- db.m6i - 3세대 인텔 제온 스케일러블 프로세서로 구동되는 범용 DB 인스턴스 클래스입니다. SAP 인증을 받은 이러한 인스턴스는 엔터프라이즈 애플리케이션을 지원하는 백엔드 서버, 게임 서버, 캐싱 플릿, 애플리케이션 개발 환경 등의 워크로드에 적합합니다. db.m6id 및 db.m6idn 인스턴스 클래스는 최대 7.6TB의 로컬 NVMe 기반 SSD 스토리지를 제공하는 반면 db.m6in은 EBS 전용 스토리지를 제공합니다. db.m6in 및 db.m6idn 클래스는 최대 200Gbps의 네트워크 대역폭을 제공합니다.
- db.m5 – 컴퓨팅, 메모리 및 네트워크 리소스가 균형잡힌 범용 DB 인스턴스 클래스로, 대부분의 애플리케이션에 적합합니다. db.m5d 인스턴스 클래스는 호스트 서버에 물리적으로 연결된 NVMe 기반 SSD 스토리지를 제공합니다. db.m5 인스턴스 클래스는 이전의 db.m4 인스턴스 클래스보다 더 많은

컴퓨팅 용량을 제공합니다. 전용 하드웨어 및 경량 하이퍼바이저 결합된 AWS Nitro System을 기반으로 합니다.

- db.m4 – 이전 db.m3 인스턴스 클래스에 비해 더 많은 컴퓨팅 용량을 제공하는 범용 DB 인스턴스 클래스입니다.

  RDS for Oracle DB 엔진의 경우 Amazon RDS는 이제 db.m4 DB 인스턴스 클래스를 지원하지 않습니다. 이전에 RDS for Oracle db.m4 DB 인스턴스를 생성한 경우, Amazon RDS는 해당 DB 인스턴스를 이에 상응하는 db.m5 DB 인스턴스 클래스로 자동 업그레이드합니다.

  RDS for MariaDB, RDS for MySQL, RDS for SQL Server 및 RDS for PostgreSQL DB 엔진의 경우, Amazon RDS는 다음 일정에 따라 이 DB 인스턴스 클래스에 대한 지원 종료 프로세스를 시작했습니다. 이 인스턴스 클래스를 사용하는 모든 RDS DB 인스턴스의 경우 가능한 한 빨리 최신 세대 DB 인스턴스 클래스로 업그레이드하는 것이 좋습니다.

작업 또는 권장 사항	날짜
이 날짜부터 Amazon RDS는 db.m4를 사용하는 인스턴스를 최신 세대 db.m5 인스턴스 클래스로 자동 업그레이드하기 시작했습니다. db.m4 인스턴스 클래스를 사용하여 DB 인스턴스를 생성하는 것은 더 이상 지원되지 않습니다.	2024년 6월 1일
Amazon RDS는 db.m4에 대한 지원을 종료합니다.	2024년 12월 31일

- db.m3 – 이전 db.m1 인스턴스 클래스에 비해 더 많은 컴퓨팅 용량을 제공하는 범용 DB 인스턴스 클래스입니다.

  RDS for MariaDB, RDS for MySQL, RDS for PostgreSQL DB 엔진의 경우 Amazon RDS는 업그레이드 권장 사항이 포함된 다음 일정에 따라 db.m3 DB 인스턴스 클래스의 수명 종료 프로세스를 시작했습니다. db.m3 DB 인스턴스 클래스를 사용하는 모든 RDS DB 인스턴스의 경우 최대한 빨리 더 높은 버전의 DB 인스턴스 클래스로 업그레이드하는 것이 좋습니다.

작업 또는 권장 사항	날짜
이제 db.m3 DB 인스턴스 클래스를 사용하는 RDS DB 인스턴스를 만들 수 없습니다.	NOW

작업 또는 권장 사항	날짜
Amazon RDS는 db.m5 DB 인스턴스 클래스에 해당하는 db.m3 DB 인스턴스를 사용하는 RDS DB 인스턴스의 자동 업그레이드를 시작했습니다.	2023년 2월 1일

메모리 최적화 인스턴스 클래스 유형

메모리 최적화 Z 패밀리는 다음과 같은 인스턴스 클래스를 지원합니다.

- db.z1d – 메모리 집약적 애플리케이션에 최적화된 인스턴스 클래스. 이러한 인스턴스 클래스는 높은 컴퓨팅 용량과 큰 메모리 공간을 제공합니다. 고주파수 z1d 인스턴스는 최대 4.0GHz의 일관된 올코어 주파수를 제공합니다.

메모리 최적화 X 패밀리는 다음과 같은 인스턴스 클래스를 지원합니다.

- db.x2g - 메모리 집약적 애플리케이션에 최적화되고 AWS Graviton2 프로세서로 구동되는 인스턴스 클래스입니다. 이러한 인스턴스 클래스는 메모리 GiB당 낮은 비용을 제공합니다.

  AWS Graviton2 프로세서로 구동되는 DB 인스턴스 클래스 중 하나를 사용하도록 DB 인스턴스를 수정할 수 있습니다. 이렇게 하려면 다른 DB 인스턴스 수정과 동일한 단계를 완료해야 합니다.
- db.x2i – 메모리 집약적 애플리케이션에 최적화된 인스턴스 클래스. db.x2iedn 및 db.x2idn 인스턴스 클래스 유형은 3세대 인텔 제온 스케일러블 프로세서(Ice Lake)로 구동됩니다. 최대 3.8TB의 로컬 NVMe SSD 스토리지, 최대 100Gbps의 네트워킹 대역폭, 최대 4TiB(db.x2iden) 또는 2TiB(db.x2idn) 메모리가 포함되어 있습니다. db.x2iezn 유형은 2세대 인텔 제온 스케일러블 프로세서(Cascade Lake)로 구동되며 올코어 터보 주파수는 최대 4.5GHz이고 메모리는 최대 1.5TiB입니다.
- db.x1 – 메모리 집약적 애플리케이션에 최적화된 인스턴스 클래스. 이러한 인스턴스 클래스는 RAM GiB당 비용이 가장 낮은 인스턴스 클래스이며 최대 1,952GiB의 DRAM 기반 인스턴스 메모리를 제공합니다. db.x1e 인스턴스 클래스 유형은 최대 3,904GiB의 DRAM 기반 인스턴스 메모리를 제공합니다.

메모리 최적화 R 패밀리는 다음과 같은 인스턴스 클래스 유형을 지원합니다.

- db.r8g - AWS Graviton4 프로세서로 구동되는 인스턴스 클래스입니다. 이러한 인스턴스 클래스는 MySQL 및 PostgreSQL과 같은 오픈 소스 데이터베이스에서 메모리 사용량이 많은 워크로드를 실행

하는 데 적합합니다. 이러한 인스턴스는 7세대 AWS Graviton3 기반 db.r7g 인스턴스보다 3배 더 많은 vCPU와 메모리를 통해 더 큰 인스턴스 크기를 제공합니다.

AWS Graviton4 프로세서로 구동되는 DB 인스턴스 클래스 중 하나를 사용하도록 DB 인스턴스를 수정할 수 있습니다. 이렇게 하려면 다른 DB 인스턴스 수정과 동일한 단계를 완료해야 합니다.

- db.r7g - AWS Graviton3 프로세서로 구동되는 인스턴스 클래스. 이러한 인스턴스 클래스는 MySQL 및 PostgreSQL과 같은 오픈 소스 데이터베이스에서 메모리 사용량이 많은 워크로드를 실행하는 데 적합합니다.

  AWS Graviton3 프로세서로 구동되는 DB 인스턴스 클래스 중 하나를 사용하도록 DB 인스턴스를 수정할 수 있습니다. 이렇게 하려면 다른 DB 인스턴스 수정과 동일한 단계를 완료해야 합니다.
- db.r7i - 4세대 인텔 제온 스케일러블 프로세서로 구동되는 인스턴스 클래스입니다. 이러한 인스턴스는 SAP 인증을 받았으며 MySQL 및 PostgreSQL과 같은 오픈 소스 데이터베이스에서 메모리 집약적인 워크로드를 실행하는 데 적합합니다. 4세대 인텔 제온 스케일러블 프로세서로 구동되는 DB 인스턴스 클래스 중 하나를 사용하도록 DB 인스턴스를 수정할 수 있습니다. 이렇게 하려면 다른 DB 인스턴스 수정과 동일한 단계를 완료해야 합니다.
- db.r6g - AWS Graviton2 프로세서로 구동되는 인스턴스 클래스 이러한 인스턴스 클래스는 MySQL 및 PostgreSQL과 같은 오픈 소스 데이터베이스에서 메모리 사용량이 많은 워크로드를 실행하는 데 적합합니다. db.r6gd 유형은 빠르고 대기 시간이 짧은 로컬 스토리지가 필요한 애플리케이션을 위한 로컬 NVMe 기반 SSD 블록 스토리지를 제공합니다.

  AWS Graviton2 프로세서로 구동되는 DB 인스턴스 클래스 중 하나를 사용하도록 DB 인스턴스를 수정할 수 있습니다. 이렇게 하려면 다른 DB 인스턴스 수정과 동일한 단계를 완료해야 합니다.
- db.r6i - 3세대 인텔 제온 스케일러블 프로세서로 구동되는 인스턴스 클래스입니다. SAP 인증을 받은 이러한 인스턴스 클래스는 MySQL 및 PostgreSQL과 같은 오픈 소스 데이터베이스에서 메모리 사용량이 많은 워크로드를 실행하는 데 적합합니다. db.r6id, db.r6in 및 db.r6idn 인스턴스 클래스의 메모리 대 vCPU 비율은 8:1이고 최대 메모리는 1TiB입니다. db.r6id 및 db.r6idn 클래스는 최대 7.6TB의 직접 연결 NVMe 기반 SSD 스토리지를 제공하는 한편, db.r6in은 EBS 전용 스토리지를 제공합니다. db.r6idn 및 db.r6in 클래스는 최대 200Gbps의 네트워크 대역폭을 제공합니다.
- db.r5b – 처리량이 많은 애플리케이션에 대해 메모리 최적화된 인스턴스 클래스. AWS Nitro 시스템을 기반으로 하는 db.r5b 인스턴스는 최대 60Gbps의 대역폭과 260,000 IOPS의 EBS 성능을 제공합니다. 이는 EC2에서 가장 빠른 블록 스토리지 성능입니다.
- db.r5d – 짧은 지연 시간, 매우 높은 임의 I/O 성능, 높은 순차 읽기 처리량에 최적화된 인스턴스 클래스.

- db.r5 – 메모리 집약적 애플리케이션에 최적화된 인스턴스 클래스. 이러한 인스턴스 클래스는 향상된 네트워킹과 성능을 제공합니다. 전용 하드웨어 및 경량 하이퍼바이저 결합된 AWS Nitro System을 기반으로 합니다.
- db.r4 - 이전 db.r3 인스턴스 클래스보다 향상된 네트워킹을 제공하는 인스턴스 클래스입니다.

  RDS for Oracle DB 엔진의 경우 Amazon RDS는 업그레이드 권장 사항이 포함된 다음 일정을 사용하여 db.r4 DB 인스턴스 클래스의 수명 종료 프로세스를 시작했습니다. db.r4 인스턴스 클래스를 사용하는 RDS for Oracle DB 인스턴스의 경우 최대한 빨리 더 높은 버전의 인스턴스 클래스로 업그레이드하는 것이 좋습니다.

작업 또는 권장 사항	날짜
이제 db.r4 DB 인스턴스 클래스를 사용하는 RDS for Oracle DB 인스턴스를 만들 수 없습니다.	NOW
Amazon RDS는 db.r5 DB 인스턴스 클래스에 해당하는 db.r4 DB 인스턴스를 사용하는 RDS for Oracle DB 인스턴스의 자동 업그레이드를 시작했습니다.	2023년 4월 17일

  RDS for MariaDB, RDS for MySQL, RDS for SQL Server 및 RDS for PostgreSQL DB 엔진의 경우, Amazon RDS는 다음 일정에 따라 이 DB 인스턴스 클래스에 대한 지원 종료 프로세스를 시작했습니다. 이 인스턴스 클래스를 사용하는 모든 RDS DB 인스턴스의 경우 가능한 한 빨리 최신 세대 DB 인스턴스 클래스로 업그레이드하는 것이 좋습니다.

작업 또는 권장 사항	날짜
이 날짜부터 Amazon RDS는 db.r4를 사용하는 인스턴스를 최신 세대 db.r5 인스턴스 클래스로 자동 업그레이드하기 시작했습니다. db.m4 인스턴스 클래스를 사용하여 DB 인스턴스를 생성하는 것은 더 이상 지원되지 않습니다.	2024년 6월 1일
Amazon RDS는 db.r4에 대한 지원을 종료합니다.	2024년 12월 31일

- db.r3 – 메모리 최적화를 제공하는 인스턴스 클래스.

RDS for MariaDB, RDS for MySQL, RDS for PostgreSQL DB 엔진의 경우 Amazon RDS는 업그레이드 권장 사항이 포함된 다음 일정에 따라 db.r3 DB 인스턴스 클래스의 수명 종료 프로세스를 시작했습니다. db.r3 DB 인스턴스 클래스를 사용하는 모든 RDS DB 인스턴스의 경우 최대한 빨리 더 높은 버전의 DB 인스턴스 클래스로 업그레이드하는 것이 좋습니다.

작업 또는 권장 사항	날짜
이제 db.r3 DB 인스턴스 클래스를 사용하는 RDS DB 인스턴스를 만들 수 없습니다.	현재
Amazon RDS는 db.r5 DB 인스턴스 클래스에 해당하는 db.r3 DB 인스턴스를 사용하는 RDS DB 인스턴스의 자동 업그레이드를 시작했습니다.	2023년 2월 1일

컴퓨팅 최적화 인스턴스 클래스 유형

다음은 사용 가능한 컴퓨팅 최적화 인스턴스 클래스 유형입니다.

- db.c6gd - 고급 컴퓨팅 집약적 워크로드를 실행하는 데 적합한 인스턴스 클래스입니다. AWS Graviton2 프로세서로 지원되는 이 인스턴스 클래스는 빠르고 대기 시간이 짧은 로컬 스토리지가 필요한 애플리케이션을 위한 로컬 NVMe 기반 SSD 블록 스토리지를 제공합니다.

> **Note**
>
> c6gd 인스턴스 클래스는 다중 AZ DB 클러스터 배포에만 지원됩니다. `medium` 인스턴스 크기를 제공하는 다중 AZ DB 클러스터에 유일하게 지원되는 인스턴스 클래스입니다. 자세한 내용은 the section called “다중 AZ DB 클러스터 배포” 섹션을 참조하세요.

버스트 가능한 성능 인스턴스 클래스 유형

다음은 사용 가능한 버스트 가능 성능 DB 인스턴스 클래스 유형입니다.

- db.t4g - Arm 기반 AWS Graviton2 프로세서로 구동되는 범용 인스턴스 클래스입니다. 이러한 인스턴스 클래스는 광범위한 범용 워크로드 집합에 대해 이전의 버스트 가능 성능 DB 인스턴스 클래스보다 더 나은 가성비를 제공합니다. Amazon RDS db.t4g 인스턴스는 무제한 모드로 구성됩니다. 즉, 추가 요금을 지불하면 24시간 동안 기준 이상으로 높일 수 있습니다.

AWS Graviton2 프로세서로 구동되는 DB 인스턴스 클래스 중 하나를 사용하도록 DB 인스턴스를 수정할 수 있습니다. 이렇게 하려면 다른 DB 인스턴스 수정과 동일한 단계를 완료해야 합니다.

- db.t3 – CPU 사용률을 최대로 버스트할 수 있는 기능으로 기준 성능 수준을 제공하는 인스턴스 클래스입니다. db.t3 인스턴스는 무제한 모드로 구성됩니다. 이 인스턴스 클래스는 이전의 db.t2 인스턴스 클래스보다 더 많은 컴퓨팅 용량을 제공합니다. 전용 하드웨어 및 경량 하이퍼바이저 결합된 AWS Nitro System을 기반으로 합니다.
- db.t2 – CPU 사용률을 최대로 버스트할 수 있는 기능으로 기준 성능 수준을 제공하는 인스턴스 클래스입니다. db.t2 인스턴스는 무제한 모드로 구성됩니다. 이러한 인스턴스 클래스는 개발 및 테스트 서버 또는 기타 비 프로덕션 서버에만 사용하는 것이 좋습니다.

  RDS for MariaDB, RDS for MySQL, RDS for SQL Server 및 RDS for PostgreSQL DB 엔진의 경우, Amazon RDS는 다음 일정에 따라 이 DB 인스턴스 클래스에 대한 지원 종료 프로세스를 시작했습니다. 이 인스턴스 클래스를 사용하는 모든 RDS DB 인스턴스의 경우 가능한 한 빨리 최신 세대 DB 인스턴스 클래스로 업그레이드하는 것이 좋습니다.

작업 또는 권장 사항	날짜
이 날짜부터 Amazon RDS는 db.t2를 사용하는 인스턴스를 최신 세대 db.t3 인스턴스 클래스로 자동 업그레이드하기 시작했습니다. db.t2 인스턴스 클래스를 사용하여 DB 인스턴스를 생성하는 것은 더 이상 지원되지 않습니다.	2024년 6월 1일
Amazon RDS는 db.t2에 대한 지원을 종료합니다.	2024년 12월 31일

> **Note**
>
> AWS Nitro System을 사용하는 DB 인스턴스 클래스(db.m5, db.r5, db.t3)는 결합된 읽기 및 쓰기 워크로드에 대해 조절됩니다.

DB 인스턴스 클래스의 하드웨어 사양은 에 대한 DB 인스턴스 클래스의 하드웨어 사양 단원을 참조하십시오.

최적화된 읽기 인스턴스 클래스 유형

다음은 사용 가능한 최적화된 읽기 인스턴스 클래스 유형입니다.

- db.r6g - AWS Graviton2 프로세서로 구동되는 인스턴스 클래스 이러한 인스턴스 클래스는 메모리 사용량이 많은 워크로드를 실행하는 데 이상적이며 빠르고 지연 시간이 짧은 로컬 스토리지가 필요한 애플리케이션에 로컬 NVMe 기반 SSD 블록 스토리지를 제공합니다.
- db.r6id - 3세대 인텔 제온 스케일러블 프로세서로 구동되는 인스턴스 클래스입니다. SAP 인증을 받은 이러한 인스턴스 클래스는 메모리 사용량이 많은 워크로드에 적합합니다. 최대 1TiB의 메모리 및 최대 7.6TB의 직접 연결 NVMe 기반 SSD 스토리지를 제공합니다.

DB 인스턴스 클래스에 지원되는 DB 엔진

다음은 DB 인스턴스 클래스에 대한 DB 엔진 특정 고려 사항입니다.

Db2

DB 인스턴스 클래스 지원은 Db2의 버전과 에디션에 따라 달라집니다. 버전 및 에디션별 인스턴스 클래스 지원은 Amazon RDS for Db2 인스턴스 클래스 단원을 참조하세요.

Microsoft SQL Server

DB 인스턴스 클래스 지원은 SQL Server의 버전과 에디션에 따라 달라집니다. 버전 및 에디션별 인스턴스 클래스 지원은 Microsoft SQL Server를 위한 DB 인스턴스 클래스 지원 단원을 참조하세요.

Oracle

DB 인스턴스 클래스 지원은 Oracle 데이터베이스 버전과 에디션에 따라 달라집니다.
RDS for Oracle은 추가 메모리 최적화 인스턴스 클래스를 지원합니다. 이러한 클래스에는 db.r5.*instance_size*.*tpctreads_per_core*.mem*ratio* 형식의 이름이 있습니다. 최적화된 각 클래스에 대한 vCPU 수 및 메모리 할당의 경우 지원되는 RDS for Oracle DB 인스턴스 클래스 섹션을 참조하세요.

RDS Custom

RDS Custom에서 지원되는 DB 인스턴스 클래스에 대한 자세한 내용은 RDS Custom for Oracle에 대한 DB 인스턴스 클래스 지원 및 RDS Custom for SQL Server DB 인스턴스 클래스 지원 섹션을 참조하세요.

아래 표에서 각 Amazon RDS DB 엔진에 지원되는 Amazon RDS DB 인스턴스 클래스의 세부 정보를 확인하실 수 있습니다. 각 엔진의 셀에는 다음 값 중 하나가 포함됩니다.

예

인스턴스 클래스는 모든 버전의 DB 엔진에서 지원됩니다.

아니요

인스턴스 클래스는 DB 엔진에 대해 지원되지 않습니다.

specific-versions

인스턴스 클래스는 DB 엔진의 지정된 데이터베이스 버전에서만 지원됩니다.

Amazon RDS는 정기적으로 메이저 버전과 마이너 DB 엔진 버전을 지원 중단합니다. 모든 AWS 리전에서 이전 엔진 버전을 지원하는 것은 아닙니다. 현재 지원되는 버전에 대한 자세한 내용은 개별 DB 엔진에 대한 항목인 MariaDB 버전, Microsoft SQL Server 버전, MySQL 버전, Oracle 버전 및 PostgreSQL 버전을 참조하세요.

주제

- 범용 인스턴스 클래스에서 지원되는 DB 엔진
- 메모리 최적화 인스턴스 클래스에 지원되는 DB 엔진
- 컴퓨팅 최적화 인스턴스 클래스에서 지원되는 DB 엔진
- 성능 버스트 가능 인스턴스 클래스에서 지원되는 DB 엔진
- 최적화된 읽기 인스턴스 클래스에서 지원되는 DB 엔진

범용 인스턴스 클래스에서 지원되는 DB 엔진

다음 테이블에는 범용 인스턴스 클래스에서 지원되는 데이터베이스 및 데이터베이스 버전이 나와 있습니다.

db.m7g - AWS Graviton3 프로세서로 구동되는 범용 인스턴스 클래스

인스턴스 클래스	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.m7g.16 xlarge	아니요	MariaDB 10.11 버전, 10.6.10 이상의 10.6 버전, 10.5.17 이상의 10.5 버전, 10.4.26 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.4 이상 13 버전
db.m7g.12 xlarge	아니요	MariaDB 10.11 버전, 10.6.10 이상의 10.6 버전, 10.5.17 이상의 10.5 버전, 10.4.26 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.4 이상 13 버전
db.m7g.8x large	아니요	MariaDB 10.11 버전, 10.6.10 이상의 10.6 버전, 10.5.17 이상의 10.5 버전, 10.4.26 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.4 이상 13 버전
db.m7g.4x large	아니요	MariaDB 10.11 버전, 10.6.10 이상의 10.6 버전, 10.5.17 이상의 10.5 버전, 10.4.26 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.4 이상 13 버전
db.m7g.2x large	아니요	MariaDB 10.11 버전, 10.6.10 이상의 10.6 버전, 10.5.17 이상의 10.5 버전, 10.4.26 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.4 이상 13 버전
db.m7g.xlarge	아니요	MariaDB 10.11 버전, 10.6.10 이상의 10.6 버전, 10.5.17 이상의 10.5 버전, 10.4.26 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.4 이상 13 버전
db.m7g.large	아니요	MariaDB 10.11 버전, 10.6.10 이상의 10.6 버전, 10.5.17 이상의 10.5 버전, 10.4.26 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.4 이상 13 버전

db.m6g - AWS Graviton2 프로세서로 구동되는 범용 인스턴스 클래스

인스턴스 클래스	Db2	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.m6g.1(xlarge	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.23 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.m6g.1: xlarge	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.23 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.m6g.8: large	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.23 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.m6g.4: large	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.23 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.m6g.2: large	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.23 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.m6g.xl arge	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.23 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.m6g.la rge	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.23 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전

db.m6gd - AWS Graviton2 프로세서 및 SSD 스토리지로 구동되는 범용 인스턴스 클래스

인스턴스 클래스	Db2	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.m6gd.16xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16, 15, 14 버전, 13.7 이상의 13 버전 및 13.4 버전
db.m6gd.12xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16, 15, 14 버전, 13.7 이상의 13 버전 및 13.4 버전
db.m6gd.8xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16, 15, 14 버전, 13.7 이상의 13 버전 및 13.4 버전
db.m6gd.4xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16, 15, 14 버전, 13.7 이상의 13 버전 및 13.4 버전
db.m6gd.2xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16, 15, 14 버전, 13.7 이상의 13 버전 및 13.4 버전
db.m6gd.xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16, 15, 14 버전, 13.7 이상의 13 버전 및 13.4 버전

인스턴스 클래스	Db2	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
		전, 10.4.25 이상의 10.4 버전				
db.m6gd.large	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16, 15, 14 버전, 13.7 이상의 13 버전 및 13.4 버전

db.m6id - 3세대 인텔 제온 스케일러블 프로세서 및 SSD 스토리지로 구동되는 범용 인스턴스 클래스

인스턴스 클래스	Db2	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.m6id.32xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.m6id.24xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.m6id.16xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.m6id.12xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전

인스턴스 클래스	Db2	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
		10.5 버전, 10.4.25 이상의 10.4 버전				
db.m6id.8xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.m6id.4xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.m6id.2xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.m6id.xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.m6id.large	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전

db.m6idn - 3세대 인텔 제온 스케일러블 프로세서, SSD 스토리지 및 네트워크 최적화를 포함하는 범용 인스턴스 클래스

인스턴스 클래스	Db	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.m6idn.32xlarge	아니요	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.m6idn.24xlarge	아니요	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.m6idn.16xlarge	아니요	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.m6idn.12xlarge	아니요	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.m6idn.8xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.m6idn.4xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.m6idn.2xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전

인스턴스 클래스	Db	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.m6idn.xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.m6idn.large	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전

db.m6in - 3세대 인텔 제온 스케일러블 프로세서 및 네트워크 최적화로 구동되는 범용 인스턴스 클래스

인스턴스 클래스	Db	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.m6in.32xlarge	아니요	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.3 이상 14 버전, 13.7 이상 13 버전, 12.11 이상 12 버전, 11.16 이상 11 버전
db.m6in.24xlarge	아니요	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.3 이상 14 버전, 13.7 이상 13 버전, 12.11 이상 12 버전, 11.16 이상 11 버전
db.m6in.16xlarge	아니요	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.3 이상 14 버전, 13.7 이상 13 버전, 12.11

인스턴스 클래스	Db	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
						이상 12 버전, 11.16 이상 11 버전
db.m6in.12xlarge	아니요	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.3 이상 14 버전, 13.7 이상 13 버전, 12.11 이상 12 버전, 11.16 이상 11 버전
db.m6in.8xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.3 이상 14 버전, 13.7 이상 13 버전, 12.11 이상 12 버전, 11.16 이상 11 버전
db.m6in.4xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.3 이상 14 버전, 13.7 이상 13 버전, 12.11 이상 12 버전, 11.16 이상 11 버전
db.m6in.2xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.3 이상 14 버전, 13.7 이상 13 버전, 12.11 이상 12 버전, 11.16 이상 11 버전
db.m6in.xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.3 이상 14 버전, 13.7 이상 13 버전, 12.11 이상 12 버전, 11.16 이상 11 버전

인스턴스 클래스	Db	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
db.m6in.l arge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.3 이상 14 버전, 13.7 이상 13 버전, 12.11 이상 12 버전, 11.16 이상 11 버전

db.m6i - 3세대 인텔 제온 스케일러블 프로세서로 구동되는 범용 인스턴스 클래스

인스턴스 클래스	Db	MariaDB	Micros SQL Serve	MySQL	Oracle	PostgreSQL
db.m6i.32 xlarge	예	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.15 이상의 10.5 버전, 10.4.24 이상의 10.4 버전	예	MySQL 버전 8.0.28 이상	Oracle Databa 19c	모든 PostgreSQL 16, 15, 14 버전 및 13.4, 12.8, 11.13 이상 11 버전
db.m6i.24 xlarge	예	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.15 이상의 10.5 버전, 10.4.24 이상의 10.4 버전	예	MySQL 버전 8.0.28 이상	Oracle Databa 19c	모든 PostgreSQL 16, 15, 14 버전 및 13.4, 12.8, 11.13 이상 11 버전
db.m6i.16 xlarge	예	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.15 이상의 10.5 버전, 10.4.24 이상의 10.4 버전	예	MySQL 버전 8.0.28 이상	Oracle Databa 19c	모든 PostgreSQL 16, 15, 14 버전 및 13.4, 12.8, 11.13 이상 11 버전
db.m6i.12 xlarge	예	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.15 이상의 10.5	예	MySQL 버전 8.0.28 이상	Oracle Databa 19c	모든 PostgreSQL 16, 15, 14 버전 및 13.4, 12.8, 11.13 이상 11 버전

인스턴스 클래스	Db	MariaDB	Micros SQL Server	MySQL	Oracle	PostgreSQL
		버전, 10.4.24 이상의 10.4 버전				
db.m6i.8x large	예	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.15 이상의 10.5 버전, 10.4.24 이상의 10.4 버전	예	MySQL 버전 8.0.28 이상	Oracle Databa 19c	모든 PostgreSQL 16, 15, 14 버전 및 13.4, 12.8, 11.13 이상 11 버전
db.m6i.4x large	예	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.15 이상의 10.5 버전, 10.4.24 이상의 10.4 버전	예	MySQL 버전 8.0.28 이상	Oracle Databa 19c	모든 PostgreSQL 16, 15, 14 버전 및 13.4, 12.8, 11.13 이상 11 버전
db.m6i.2x large	예	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.15 이상의 10.5 버전, 10.4.24 이상의 10.4 버전	예	MySQL 버전 8.0.28 이상	Oracle Databa 19c	모든 PostgreSQL 16, 15, 14 버전 및 13.4, 12.8, 11.13 이상 11 버전
db.m6i.xl arge	예	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.15 이상의 10.5 버전, 10.4.24 이상의 10.4 버전	예	MySQL 버전 8.0.28 이상	Oracle Databa 19c	모든 PostgreSQL 16, 15, 14 버전 및 13.4, 12.8, 11.13 이상 11 버전
db.m6i.la rge	예	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.15 이상의 10.5 버전, 10.4.24 이상의 10.4 버전	예	MySQL 버전 8.0.28 이상	Oracle Databa 19c	모든 PostgreSQL 16, 15, 14 버전 및 13.4, 12.8, 11.13 이상 11 버전

db.m5d - 인텔 제온 플래티넘 프로세서 및 SSD 스토리지로 구동되는 범용 인스턴스 클래스

인스턴스 클래스	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.m5d.24xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	예	MySQL 8.0.28 이상	예	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.m5d.16xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	예	MySQL 8.0.28 이상	예	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.m5d.12xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	예	MySQL 8.0.28 이상	예	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.m5d.8xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	예	MySQL 8.0.28 이상	예	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.m5d.4xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	예	MySQL 8.0.28 이상	예	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.m5d.2xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버	예	MySQL 8.0.28 이상	예	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4

인스턴스 클래스	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
		전, 10.4.25 이상의 10.4 버전				
db.m5d.xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	예	MySQL 8.0.28 이상	예	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.m5d.large	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	예	MySQL 8.0.28 이상	예	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4

db.m5 - 2.5GHz 인텔 제온 플래티넘 프로세서로 구동되는 범용 인스턴스 클래스

인스턴스 클래스	Db2	Maria	Microsoft SQL Server	MyS	Oracl	PostgreSQL
db.m5.24xlarge	아니요	예	예	예	예	모든 PostgreSQL 16, 15, 14, 13, 12, 11 버전, 10.17 이상 10 버전, 9.6.22 이상 9 버전
db.m5.16xlarge	아니요	예	예	예	예	모든 PostgreSQL 16, 15, 14, 13, 12, 11 버전, 10.17 이상 10 버전, 9.6.22 이상 9 버전
db.m5.12xlarge	아니요	예	예	예	예	모든 PostgreSQL 16, 15, 14, 13, 12, 11 버전, 10.17 이상 10 버전, 9.6.22 이상 9 버전

인스턴스 클래스	Db2	Maria	Microsoft SQL Server	MyS	Oracl	PostgreSQL
db.m5.8xlarge	아니요	예	예	예	예	모든 PostgreSQL 16, 15, 14, 13, 12, 11 버전, 10.17 이상 10 버전, 9.6.22 이상 9 버전
db.m5.4xlarge	아니요	예	예	예	예	모든 PostgreSQL 16, 15, 14, 13, 12, 11 버전, 10.17 이상 10 버전, 9.6.22 이상 9 버전
db.m5.2xlarge	아니요	예	예	예	예	모든 PostgreSQL 16, 15, 14, 13, 12, 11 버전, 10.17 이상 10 버전, 9.6.22 이상 9 버전
db.m5.xlarge	아니요	예	예	예	예	모든 PostgreSQL 16, 15, 14, 13, 12, 11 버전, 10.17 이상 10 버전, 9.6.22 이상 9 버전
db.m5.large	아니요	예	예	예	예	모든 PostgreSQL 16, 15, 14, 13, 12, 11 버전, 10.17 이상 10 버전, 9.6.22 이상 9 버전

db.m4 - 인텔 제온 프로세서 포함 범용 인스턴스 클래스

인스턴스 클래스	Db	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.m4.16large	아니요	Deprecated	Deprecated	Deprecated	Deprecated	Deprecated
db.m4.10large	아니요	Deprecated	Deprecated	Deprecated	Deprecated	Deprecated

인스턴스 클래스	Db	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.m4.4xlarge	아니요	Deprecated	Deprecated	Deprecated	Deprecated	Deprecated
db.m4.2xlarge	아니요	Deprecated	Deprecated	Deprecated	Deprecated	Deprecated
db.m4.xlarge	아니요	Deprecated	Deprecated	Deprecated	Deprecated	Deprecated
db.m4.large	아니요	Deprecated	Deprecated	Deprecated	Deprecated	Deprecated

db.m3 - 범용 인스턴스 클래스

인스턴스 클래스	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.m3.2xlarge	아니요	아니요	Deprecated	예	사용되지 않음	Deprecated
db.m3.xlarge	아니요	아니요	Deprecated	예	사용되지 않음	Deprecated
db.m3.large	아니요	아니요	Deprecated	예	사용되지 않음	Deprecated
db.m3.medium	아니요	아니요	Deprecated	예	사용되지 않음	Deprecated

메모리 최적화 인스턴스 클래스에 지원되는 DB 엔진

다음 테이블에는 메모리 최적화 인스턴스 클래스에서 지원되는 데이터베이스 및 데이터베이스 버전이 나와 있습니다.

db.z1d – 메모리 최적화 인스턴스 클래스

인스턴스 클래스	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.z1d.1 xlarge	아니요	아니요	예	아니요	예	아니요
db.z1d.6 large	아니요	아니요	예	아니요	예	아니요
db.z1d.3 large	아니요	아니요	예	아니요	예	아니요
db.z1d.2 large	아니요	아니요	예	아니요	예	아니요
db.z1d.xl arge	아니요	아니요	예	아니요	예	아니요
db.z1d.la rge	아니요	아니요	예	아니요	예	아니요

db.x2g - AWS Graviton2 프로세서로 구동되는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	Db	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
db.x2g.1 xlarge	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.25 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전

인스턴스 클래스	Db	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
db.x2g.1 xlarge	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.25 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.x2g.8 large	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.25 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.x2g.4 large	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.25 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.x2g.2 large	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.25 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.x2g.x arge	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.25 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.x2g.l rge	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.25 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전

db.x2idn - 3세대 인텔 제온 스케일러블 프로세서로 구동되는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	Db	MariaDB	Micros SQL Serve	MySQL	Oracle	PostgreSQL
db.x2idn. 32xlarge	아니요	모든 MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의	아니요	MySQL 8.0.28 이상	Enterp e	PostgreSQL 15 버전, 14.6 버전, 13.9 버전

인스턴스 클래스	DI	MariaDB	Micros SQL Server	MySQL	Oracle	PostgreSQL
		10.5 버전, 10.4.25 이상의 10.4 버전			Edition 전용	
db.x2idn.24xlarge	아니요	모든 MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	Enterp e Edition 전용	PostgreSQL 15 버전, 14.6 버전, 13.9 버전
db.x2idn.16xlarge	아니요	모든 MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	Enterp e Edition 전용	PostgreSQL 15 버전, 14.6 버전, 13.9 버전

db.x2iedn — 3세대 인텔 제온 스케일러블 프로세서로 구동되며 로컬 NVMe 기반 SSD를 장착한 메모리 최적화 인스턴스 클래스

인스턴스 클래스	DI	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.x2iedn.32xlarge	예	모든 MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	Enterprise 및 Standard Editions 전용, SQL Server 2,014 12.00 이상	MySQL 8.0.28 이상	Enterp e Edition 전용	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.x2iedn.24xlarge	예	모든 MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	Enterprise 및 Standard Editions	MySQL 8.0.28 이상	Enterp e Edition 전용	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4

인스턴스 클래스	DI	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
			전용, SQL Server 2,014 12.00 이상			
db.x2iedn.16xlarge	예	모든 MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	Enterprise 및 Standard Editions 전용, SQL Server 2,014 12.00 이상	MySQL 8.0.28 이상	Enterprise Edition 전용	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.x2iedn.8xlarge	예	모든 MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	Enterprise 및 Standard Editions 전용, SQL Server 2,014 12.00 이상	MySQL 8.0.28 이상	Enterprise Edition 전용	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.x2iedn.4xlarge	예	모든 MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	Enterprise 및 Standard Editions 전용, SQL Server 2,014 12.00 이상	MySQL 8.0.28 이상	Enterprise Edition 및 Standa Edition 2(SE2	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4

인스턴스 클래스	DI	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.x2iedn.2xlarge	예	모든 MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	Enterprise 및 Standard Editions 전용, SQL Server 2,014 12.00 이상	MySQL 8.0.28 이상	Enterprise Edition 및 Standard Edition 2(SE2	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.x2iedn.xlarge	예	모든 MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	Enterprise 및 Standard Editions 전용, SQL Server 2,014 12.00 이상	MySQL 8.0.28 이상	Enterprise Edition 및 Standard Edition 2(SE2	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4

db.x2iezn - 2세대 인텔 제온 스케일러블 프로세서로 구동되는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.x2iezn.8xlarge	아니요	아니요	아니요	아니요	Enterprise Edition 전용	아니요
db.x2iezn.6xlarge	아니요	아니요	아니요	아니요	Enterprise Edition 전용	아니요
db.x2iezn.4xlarge	아니요	아니요	아니요	아니요	Enterprise Edition 및 Standard Edition 2(SE2)	아니요

인스턴스 클래스	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.x2iezn.2xlarge	아니요	아니요	아니요	아니요	Enterprise Edition 및 Standard Edition 2(SE2)	아니요

db.x1e – 메모리 최적화 인스턴스 클래스

인스턴스 클래스	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.x1e.32xlarge	아니요	아니요	예	아니요	예	아니요
db.x1e.16xlarge	아니요	아니요	예	아니요	예	아니요
db.x1e.8xlarge	아니요	아니요	예	아니요	예	아니요
db.x1e.4xlarge	아니요	아니요	예	아니요	예	아니요
db.x1e.2xlarge	아니요	아니요	예	아니요	예	아니요
db.x1e.xlarge	아니요	아니요	예	아니요	예	아니요

db.x1 – 메모리 최적화 인스턴스 클래스

인스턴스 클래스	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.x1.32xlarge	아니요	아니요	예	아니요	예	아니요

인스턴스 클래스	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.x1.16xlarge	아니요	아니요	예	아니요	예	아니요

db.r7g – AWS Graviton3 프로세서로 구동되는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	Db	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r7g.1xlarge	아니요	MariaDB 10.11 버전, 10.6.10 이상의 10.6 버전, 10.5.17 이상의 10.5 버전, 10.4.26 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.4 이상 13 버전
db.r7g.1xlarge	아니요	MariaDB 10.11 버전, 10.6.10 이상의 10.6 버전, 10.5.17 이상의 10.5 버전, 10.4.26 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.4 이상 13 버전
db.r7g.8large	아니요	MariaDB 10.11 버전, 10.6.10 이상의 10.6 버전, 10.5.17 이상의 10.5 버전, 10.4.26 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.4 이상 13 버전
db.r7g.4large	아니요	MariaDB 10.11 버전, 10.6.10 이상의 10.6 버전, 10.5.17 이상의 10.5 버전, 10.4.26 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.4 이상 13 버전
db.r7g.2large	아니요	MariaDB 10.11 버전, 10.6.10 이상의 10.6 버전, 10.5.17 이상의 10.5 버전, 10.4.26 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.4 이상 13 버전

인스턴스 클래스	DI	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r7g.xl arge	아니요	MariaDB 10.11 버전, 10.6.10 이상의 10.6 버전, 10.5.17 이상의 10.5 버전, 10.4.26 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.4 이상 13 버전
db.r7g.la rge	아니요	MariaDB 10.11 버전, 10.6.10 이상의 10.6 버전, 10.5.17 이상의 10.5 버전, 10.4.26 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.4 이상 13 버전

db.r6g – AWS Graviton2 프로세서로 구동되는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	Db2	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6g.16 xlarge	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.23 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.r6g.12 xlarge	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.23 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.r6g.8x large	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.23 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.r6g.4x large	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.23 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전

인스턴스 클래스	Db2	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6g.2x large	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.23 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.r6g.xl arge	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.23 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.r6g.la rge	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.23 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전

db.r6g - AWS Graviton2 프로세서로 구동되는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	D	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
db.r6gd.1 6xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.r6gd.1 2xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.r6gd.8 xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4

인스턴스 클래스	DI	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6gd.4xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.r6gd.2xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.r6gd.xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.r6gd.large	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4

db.r6id - 3세대 인텔 제온 스케일러블 프로세서로 구동되는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	Db	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6id.32xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전

인스턴스 클래스	Db	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6id.24xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.r6id.16xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.r6id.12xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.r6id.8xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.r6id.4xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.r6id.2xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.r6id.xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전

인스턴스 클래스	Db	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6id.large	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전

db.r6idn - 3세대 인텔 제온 스케일러블 프로세서로 구동되는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	Db	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6idn.32xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.r6idn.24xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.r6idn.16xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.r6idn.12xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전

인스턴스 클래스	Db2	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6idn.8xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.r6idn.4xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.r6idn.2xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.r6idn.xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전

db.r6in - 3세대 인텔 제온 스케일러블 프로세서로 구동되는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	Db2	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6in.32xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.3 이상 14 버전, 13.7 이상 13 버전, 12.11 이상 12 버전, 11.16 이상 11 버전

인스턴스 클래스	Db	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6in.24xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.3 이상 14 버전, 13.7 이상 13 버전, 12.11 이상 12 버전, 11.16 이상 11 버전
db.r6in.16xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.3 이상 14 버전, 13.7 이상 13 버전, 12.11 이상 12 버전, 11.16 이상 11 버전
db.r6in.12xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.3 이상 14 버전, 13.7 이상 13 버전, 12.11 이상 12 버전, 11.16 이상 11 버전
db.r6in.8xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.3 이상 14 버전, 13.7 이상 13 버전, 12.11 이상 12 버전, 11.16 이상 11 버전
db.r6in.4xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.3 이상 14 버전, 13.7 이상 13 버전, 12.11 이상 12 버전, 11.16 이상 11 버전
db.r6in.2xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.3 이상 14 버전, 13.7 이상 13 버전, 12.11 이상 12 버전, 11.16 이상 11 버전
db.r6in.xlarge	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.3 이상 14 버전, 13.7 이상 13 버전, 12.11 이상 12 버전, 11.16 이상 11 버전

인스턴스 클래스	Db	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6in.large	예	MariaDB 10.6.8 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.3 이상 14 버전, 13.7 이상 13 버전, 12.11 이상 12 버전, 11.16 이상 11 버전

db.r6i – 대용량 메모리, 스토리지 및 I/O용으로 사전 구성된 최신 메모리 최적화 인스턴스 클래스

인스턴스 클래스	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r6i.8xlarge.tpc2.mem4x	아니요	아니요	아니요	아니요	EE 전용	아니요
db.r6i.8xlarge.tpc2.mem3x	아니요	아니요	아니요	아니요	EE 전용	아니요
db.r6i.6xlarge.tpc2.mem4x	아니요	아니요	아니요	아니요	EE 전용	아니요
db.r6i.4xlarge.tpc2.mem4x	아니요	아니요	아니요	아니요	EE 및 SE2 BYOL	아니요
db.r6i.4xlarge.tpc2.mem3x	아니요	아니요	아니요	아니요	EE 및 SE2 BYOL	아니요

인스턴스 클래스	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r6i.4xlarge.tpc2.mem2x	아니요	아니요	아니요	아니요	EE 및 SE2 BYOL	아니요
db.r6i.2xlarge.tpc2.mem8x	아니요	아니요	아니요	아니요	EE 및 SE2 BYOL	아니요
db.r6i.2xlarge.tpc2.mem4x	아니요	아니요	아니요	아니요	EE 및 SE2 BYOL	아니요
db.r6i.2xlarge.tpc1.mem2x	아니요	아니요	아니요	아니요	EE 및 SE2 BYOL	아니요
db.r6i.xlarge.tpc2.mem4x	아니요	아니요	아니요	아니요	EE 및 SE2 BYOL	아니요
db.r6i.xlarge.tpc2.mem2x	아니요	아니요	아니요	아니요	EE 및 SE2 BYOL	아니요
db.r6i.large.tpc1.mem2x	아니요	아니요	아니요	아니요	EE 및 SE2 BYOL	아니요

db.r6i – 메모리 최적화 인스턴스 클래스

인스턴스 클래스	DI	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
db.r6i.3 xlarge	여	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.15 이상의 10.5 버전, 10.4.24 이상의 10.4 버전	예	MySQL 버전 8.0.28 이상	예	모든 PostgreSQL 16, 15, 14 버전, 13.4 이상 13 버전, 12.8 이상 12 버전, 11.13 이상 11 버전, 10.21 이상 10 버전
db.r6i.2 xlarge	여	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.15 이상의 10.5 버전, 10.4.24 이상의 10.4 버전	예	MySQL 버전 8.0.28 이상	예	모든 PostgreSQL 16, 15, 14 버전, 13.4 이상 13 버전, 12.8 이상 12 버전, 11.13 이상 11 버전, 10.21 이상 10 버전
db.r6i.1 xlarge	여	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.15 이상의 10.5 버전, 10.4.24 이상의 10.4 버전	예	MySQL 버전 8.0.28 이상	예	모든 PostgreSQL 16, 15, 14 버전, 13.4 이상 13 버전, 12.8 이상 12 버전, 11.13 이상 11 버전, 10.21 이상 10 버전
db.r6i.1 xlarge	여	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.15 이상의 10.5 버전, 10.4.24 이상의 10.4 버전	예	MySQL 버전 8.0.28 이상	예	모든 PostgreSQL 16, 15, 14 버전, 13.4 이상 13 버전, 12.8 이상 12 버전, 11.13 이상 11 버전, 10.21 이상 10 버전
db.r6i.8 large	여	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.15 이상의 10.5 버전, 10.4.24 이상의 10.4 버전	예	MySQL 버전 8.0.28 이상	예	모든 PostgreSQL 16, 15, 14 버전, 13.4 이상 13 버전, 12.8 이상 12 버전, 11.13 이상 11 버전, 10.21 이상 10 버전
db.r6i.4 large	여	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.15 이	예	MySQL 버전	예	모든 PostgreSQL 16, 15, 14 버전, 13.4 이상 13 버전, 12.8 이상 12 버전, 11.13 이

인스턴스 클래스	DI	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
		상의 10.5 버전, 10.4.24 이상의 10.4 버전		8.0.28 이상		상 11 버전, 10.21 이상 10 버전
db.r6i.2 large	여	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.15 이상의 10.5 버전, 10.4.24 이상의 10.4 버전	예	MySQL 버전 8.0.28 이상	예	모든 PostgreSQL 16, 15, 14 버전, 13.4 이상 13 버전, 12.8 이상 12 버전, 11.13 이상 11 버전, 10.21 이상 10 버전
db.r6i.xl arge	여	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.15 이상의 10.5 버전, 10.4.24 이상의 10.4 버전	예	MySQL 버전 8.0.28 이상	예	모든 PostgreSQL 16, 15, 14 버전, 13.4 이상 13 버전, 12.8 이상 12 버전, 11.13 이상 11 버전, 10.21 이상 10 버전
db.r6i.la rge	여	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.15 이상의 10.5 버전, 10.4.24 이상의 10.4 버전	예	MySQL 버전 8.0.28 이상	예	모든 PostgreSQL 16, 15, 14 버전, 13.4 이상 13 버전, 12.8 이상 12 버전, 11.13 이상 11 버전, 10.21 이상 10 버전

db.r5d – 메모리 최적화 인스턴스 클래스

인스턴스 클래스	DI	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
db.r5d.2 xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	예	MySQL 8.0.28 이상	예	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4

인스턴스 클래스	DI	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
db.r5d.1 xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	예	MySQL 8.0.28 이상	예	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.r5d.1 xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	예	MySQL 8.0.28 이상	예	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.r5d.8 large	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	예	MySQL 8.0.28 이상	예	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.r5d.4 large	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	예	MySQL 8.0.28 이상	예	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.r5d.2 large	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	예	MySQL 8.0.28 이상	예	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.r5d.x arge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	예	MySQL 8.0.28 이상	예	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4

인스턴스 클래스	Db	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
db.r5d.la rge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	예	MySQL 8.0.28 이상	예	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4

db.r5 – 대용량 메모리, 스토리지 및 I/O용으로 사전 구성된 최신 메모리 최적화 인스턴스 클래스

인스턴스 클래스	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r5b.8xlarge.tpc2.mem3x	아니요	아니요	아니요	아니요	예	아니요
db.r5b.6xlarge.tpc2.mem4x	아니요	아니요	아니요	아니요	예	아니요
db.r5b.4xlarge.tpc2.mem4x	아니요	아니요	아니요	아니요	예	아니요
db.r5b.4xlarge.tpc2.mem3x	아니요	아니요	아니요	아니요	예	아니요
db.r5b.4xlarge.tpc2.mem2x	아니요	아니요	아니요	아니요	예	아니요

인스턴스 클래스	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r5b.2xlarge.tpc2.mem8x	아니요	아니요	아니요	아니요	예	아니요
db.r5b.2xlarge.tpc2.mem4x	아니요	아니요	아니요	아니요	예	아니요
db.r5b.2xlarge.tpc1.mem2x	아니요	아니요	아니요	아니요	예	아니요
db.r5b.xlarge.tpc2.mem4x	아니요	아니요	아니요	아니요	예	아니요
db.r5b.xlarge.tpc2.mem2x	아니요	아니요	아니요	아니요	예	아니요
db.r5b.large.tpc1.mem2x	아니요	아니요	아니요	아니요	예	아니요

db.r5b – 메모리 최적화 인스턴스 클래스

인스턴스 클래스	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r5b.24xlarge	아니요	MariaDB 10.11 버전, 10.6.5 이상의 10.6 버전, 10.5.12 이상의 10.5 버전, 10.4.24 이상	예	MySQL 8.0.25 이상	예	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전

인스턴스 클래스	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
		의 10.4 버전, 10.3.34 이상의 10.3 버전				
db.r5b.16xlarge	아니요	MariaDB 10.11 버전, 10.6.5 이상의 10.6 버전, 10.5.12 이상의 10.5 버전, 10.4.24 이상의 10.4 버전, 10.3.34 이상의 10.3 버전	예	MySQL 8.0.25 이상	예	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.r5b.12xlarge	아니요	MariaDB 10.11 버전, 10.6.5 이상의 10.6 버전, 10.5.12 이상의 10.5 버전, 10.4.24 이상의 10.4 버전, 10.3.34 이상의 10.3 버전	예	MySQL 8.0.25 이상	예	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.r5b.8xlarge	아니요	MariaDB 10.11 버전, 10.6.5 이상의 10.6 버전, 10.5.12 이상의 10.5 버전, 10.4.24 이상의 10.4 버전, 10.3.34 이상의 10.3 버전	예	MySQL 8.0.25 이상	> 예	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.r5b.4xlarge	아니요	MariaDB 10.11 버전, 10.6.5 이상의 10.6 버전, 10.5.12 이상의 10.5 버전, 10.4.24 이상의 10.4 버전, 10.3.34 이상의 10.3 버전	예	MySQL 8.0.25 이상	예	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.r5b.2xlarge	아니요	MariaDB 10.11 버전, 10.6.5 이상의 10.6 버전, 10.5.12 이상의 10.5 버전, 10.4.24 이상의 10.4 버전, 10.3.34 이상의 10.3 버전	예	MySQL 8.0.25 이상	예	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전

인스턴스 클래스	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.r5b.xlarge	아니요	MariaDB 10.11 버전, 10.6.5 이상의 10.6 버전, 10.5.12 이상의 10.5 버전, 10.4.24 이상의 10.4 버전, 10.3.34 이상의 10.3 버전	예	MySQL 8.0.25 이상	예	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.r5b.large	아니요	MariaDB 10.11 버전, 10.6.5 이상의 10.6 버전, 10.5.12 이상의 10.5 버전, 10.4.24 이상의 10.4 버전, 10.3.34 이상의 10.3 버전	예	MySQL 8.0.25 이상	예	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전

db.r5 – 대용량 메모리, 스토리지 및 I/O용으로 사전 구성된 최신 메모리 최적화 인스턴스 클래스

인스턴스 클래스	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r5.12xlarge.tpc2.mem2x	아니요	아니요	아니요	아니요	예	아니요
db.r5.8xlarge.tpc2.mem3x	아니요	아니요	아니요	아니요	예	아니요
db.r5.6xlarge.tpc2.mem4x	아니요	아니요	아니요	아니요	예	아니요
db.r5.4xlarge.tpc2.mem4x	아니요	아니요	아니요	아니요	예	아니요

인스턴스 클래스	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r5.4xlarge.tpc2.mem3x	아니요	아니요	아니요	아니요	예	아니요
db.r5.4xlarge.tpc2.mem2x	아니요	아니요	아니요	아니요	예	아니요
db.r5.2xlarge.tpc2.mem8x	아니요	아니요	아니요	아니요	예	아니요
db.r5.2xlarge.tpc2.mem4x	아니요	아니요	아니요	아니요	예	아니요
db.r5.2xlarge.tpc1.mem2x	아니요	아니요	아니요	아니요	예	아니요
db.r5.xlarge.tpc2.mem4x	아니요	아니요	아니요	아니요	예	아니요
db.r5.xlarge.tpc2.mem2x	아니요	아니요	아니요	아니요	예	아니요
db.r5.large.tpc1.mem2x	아니요	아니요	아니요	아니요	예	아니요

db.r5 – 메모리 최적화 인스턴스 클래스

인스턴스 클래스	Db2	MariaD	Microsoft SQL Server	MySC	Oracle	PostgreSQL
db.r5.24xlarge	아니요	예	예	예	예	모든 PostgreSQL 16, 15, 14, 13, 12, 11 버전, 10.17 이상 10 버전, 9.6.22 이상 9 버전
db.r5.16xlarge	아니요	예	예	예	예	모든 PostgreSQL 16, 15, 14, 13, 12, 11 버전, 10.17 이상 10 버전, 9.6.22 이상 9 버전
db.r5.12xlarge	아니요	예	예	예	예	모든 PostgreSQL 16, 15, 14, 13, 12, 11 버전, 10.17 이상 10 버전, 9.6.22 이상 9 버전
db.r5.8xlarge	아니요	예	예	예	예	모든 PostgreSQL 16, 15, 14, 13, 12, 11 버전, 10.17 이상 10 버전, 9.6.22 이상 9 버전
db.r5.4xlarge	아니요	예	예	예	예	모든 PostgreSQL 16, 15, 14, 13, 12, 11 버전, 10.17 이상 10 버전, 9.6.22 이상 9 버전
db.r5.2xlarge	아니요	예	예	예	예	모든 PostgreSQL 16, 15, 14, 13, 12, 11 버전, 10.17 이상 10 버전, 9.6.22 이상 9 버전
db.r5.xlarge	아니요	예	예	예	예	모든 PostgreSQL 16, 15, 14, 13, 12, 11 버전, 10.17 이상 10 버전, 9.6.22 이상 9 버전
db.r5.large	아니요	예	예	예	예	모든 PostgreSQL 16, 15, 14, 13, 12, 11 버전, 10.17 이상 10 버전, 9.6.22 이상 9 버전

db.r4 – 메모리 최적화 인스턴스 클래스

인스턴스 클래스	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r4.16xlarge	아니요	Deprecated	Deprecated	Deprecated	Deprecated	Deprecated
db.r4.8xlarge	아니요	Deprecated	Deprecated	Deprecated	Deprecated	Deprecated
db.r4.4xlarge	아니요	Deprecated	Deprecated	Deprecated	Deprecated	Deprecated
db.r4.2xlarge	아니요	Deprecated	Deprecated	Deprecated	Deprecated	Deprecated
db.r4.xlarge	아니요	Deprecated	Deprecated	Deprecated	Deprecated	Deprecated
db.r4.large	아니요	Deprecated	Deprecated	Deprecated	Deprecated	Deprecated

db.r3 – 메모리 최적화 인스턴스 클래스

인스턴스 클래스	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r3.8xlarge**	아니요	모든 MariaDB 10.6, 10.5, 10.4, 10.3 버전	Deprecated	예	사용되지 않음	Deprecated

인스턴스 클래스	Db2	MariaDB	Microsoft SQL Server	MySQ	Oracle	PostgreSQL
db.r3.4xlarge	아니요	모든 MariaDB 10.6, 10.5, 10.4, 10.3 버전	Deprecated	예	사용되지 않음	Deprecated
db.r3.2xlarge	아니요	모든 MariaDB 10.6, 10.5, 10.4, 10.3 버전	Deprecated	예	사용되지 않음	Deprecated
db.r3.xlarge	아니요	모든 MariaDB 10.6, 10.5, 10.4, 10.3 버전	Deprecated	예	사용되지 않음	Deprecated
db.r3.large	아니요	모든 MariaDB 10.6, 10.5, 10.4, 10.3 버전	Deprecated	예	사용되지 않음	Deprecated

컴퓨팅 최적화 인스턴스 클래스에서 지원되는 DB 엔진

다음 테이블에는 컴퓨팅 최적화 인스턴스 클래스에서 지원되는 데이터베이스 및 데이터베이스 버전이 나와 있습니다.

db.c6gd - 컴퓨팅 최적화 인스턴스 클래스(다중 AZ DB 클러스터 배포만 해당)

인스턴스 클래스	Db2	Maria	Microsoft SQL Server	MySQL	Orac	PostgreSQL
db.c6gd.16xlarge	아니요	아니요	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.4 및 13.7 이상 13 버전
db.c6gd.12xlarge	아니요	아니요	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.4 및 13.7 이상 13 버전

인스턴스 클래스	Db2	Maria	Microsoft SQL Server	MySQL	Orac	PostgreSQL
db.c6gd.8xlarge	아니요	아니요	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.4 및 13.7 이상 13 버전
db.c6gd.4xlarge	아니요	아니요	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.4 및 13.7 이상 13 버전
db.c6gd.2xlarge	아니요	아니요	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.4 및 13.7 이상 13 버전
db.c6gd.xlarge	아니요	아니요	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.4 및 13.7 이상 13 버전
db.c6gd.large	아니요	아니요	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.4 및 13.7 이상 13 버전
db.c6gd.medium	아니요	아니요	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.4 및 13.7 이상 13 버전

성능 버스트 가능 인스턴스 클래스에서 지원되는 DB 엔진

다음 테이블에는 성능 버스트 가능 인스턴스 클래스에서 지원되는 데이터베이스 및 데이터베이스 버전이 나와 있습니다.

db.t4g - AWS Graviton2 프로세서로 구동되는 버스트 가능 성능 인스턴스 클래스

인스턴스 클래스	Db2	MariaDB	Microsoft SQL Server	MySQL	Orac	PostgreSQL
db.t4g - AWS Graviton2 프로세서로 구동되는 버스트 가능 성능 인스턴스 클래스						
db.t4g.2xlarge	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.25 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.t4g.xlarge	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.25 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.t4g.xlarge	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.25 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.t4g.medium	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.25 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.t4g.small	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.25 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전
db.t4g.micro	아니요	모든 MariaDB 10.11, 10.6, 10.5, 10.4 버전	아니요	MySQL 8.0.25 이상	아니요	모든 PostgreSQL 16, 15, 14, 13 버전, 12.7 이상 12 버전

db.t3 – 버스트 가능 성능 인스턴스 클래스

인스턴스 클래스	Db2	Maria	Microsoft SQL Server	MySQ	Oracle	PostgreSQL
db.t3.2xlarge	예	예	예	예	예	모든 PostgreSQL 16, 15, 14, 13, 12, 11, 10 버전, 9.6.22 이상 9 버전
db.t3.xlarge	예	예	예	예	예	모든 PostgreSQL 16, 15, 14, 13, 12, 11, 10 버전, 9.6.22 이상 9 버전
db.t3.large	예	예	예	예	예	모든 PostgreSQL 16, 15, 14, 13, 12, 11, 10 버전, 9.6.22 이상 9 버전
db.t3.medium	예	예	예	예	예	모든 PostgreSQL 16, 15, 14, 13, 12, 11, 10 버전, 9.6.22 이상 9 버전
db.t3.small	예	예	예	예	예	모든 PostgreSQL 16, 15, 14, 13, 12, 11, 10 버전, 9.6.22 이상 9 버전
db.t3.micro	아니요	예	예	예	Oracle Database 12c 릴리스 1(12.1.0.2)(더 이상 사용되지 않음)에서만	모든 PostgreSQL 16, 15, 14, 13, 12, 11, 10 버전, 9.6.22 이상 9 버전

db.t2 – 버스트 가능 성능 인스턴스 클래스

인스턴스 클래스	Db2	MariaDB	Microsof SQL Server	MySQL	Oracle	PostgreSQL
db.t2.2xlarge	아니요	Deprecated	아니요	Deprecated	Deprecated	Deprecated

인스턴스 클래스	Db2	MariaDB	Microsof SQL Server	MySQL	Oracle	PostgreSQL
db.t2.xla rge	아니요	Deprecated	아니요	Deprecate d	Deprec d	Deprecated
db.t2.lar ge	아니요	Deprecated	Deprecat d	Deprecate d	Deprec d	Deprecated
db.t2.med ium	아니요	Deprecated	Deprecat d	Deprecate d	Deprec d	Deprecated
db.t2.sma ll	아니요	Deprecated	Deprecat d	Deprecate d	Deprec d	Deprecated
db.t2.mic ro	아니요	Deprecated	Deprecat d	Deprecate d	Deprec d	Deprecated

최적화된 읽기 인스턴스 클래스에서 지원되는 DB 엔진

다음 테이블에는 최적화된 읽기 인스턴스 클래스에서 지원되는 데이터베이스 및 데이터베이스 버전이 나와 있습니다.

db.r6gd - AWS Graviton2 프로세서로 구동되고 최적화된 읽기를 지원하는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	DI	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6gd.1 6xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.r6gd.1 2xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.r6gd.8 xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.r6gd.4 xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.r6gd.2 xlarge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.r6gd.x large	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4
db.r6gd.l arge	아니요	MariaDB 10.11 버전, 10.6.7 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전, 13.4

db.r6id - 3세대 인텔 제온 스케일러블 프로세서로 구동되고 최적화된 읽기를 지원하는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	Db	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6id.32xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.r6id.24xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.r6id.16xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.r6id.12xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.r6id.8xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.r6id.4xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전

인스턴스 클래스	Db	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6id.2xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.r6id.xlarge	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전
db.r6id.large	아니요	MariaDB 10.6.10 이상의 10.6 버전, 10.5.16 이상의 10.5 버전, 10.4.25 이상의 10.4 버전	아니요	MySQL 버전 8.0.28 이상	아니요	모든 PostgreSQL 16 및 15 버전, 14.5 이상 14 버전, 13.7 이상 13 버전

AWS 리전에서 DB 인스턴스 클래스 지원 확인

특정 AWS 리전의 각 DB 엔진에서 지원하는 DB 인스턴스 클래스를 확인하려면 여러 접근 방식 중 하나를 사용할 수 있습니다. AWS Management Console, Amazon RDS 요금 페이지 또는 AWS Command Line Interface(AWS CLI)의 경우 describe-orderable-db-instance-options 명령을 사용할 수 있습니다.

> **Note**
>
> AWS Management Console에서 작업을 수행하면 특정 DB 엔진, DB 엔진 버전 및 AWS 리전에서 지원되는 DB 인스턴스 클래스가 자동으로 표시됩니다. 수행할 수 있는 작업의 예로는 DB 인스턴스 생성 및 수정 등이 있습니다.

목차

- Amazon RDS 요금 페이지를 사용하여 AWS 리전에서 DB 인스턴스 클래스 지원 확인
- AWS CLI를 사용하여 AWS 리전에서 DB 인스턴스 클래스 지원 확인

- AWS 리전의 특정 DB 엔진 버전에서 지원하는 DB 인스턴스 클래스 나열
- AWS 리전에서 특정 DB 인스턴스 클래스를 지원하는 DB 엔진 버전 나열
- 특정 DB 엔진 및 인스턴스 클래스를 지원하는 AWS 리전 나열

Amazon RDS 요금 페이지를 사용하여 AWS 리전에서 DB 인스턴스 클래스 지원 확인

Amazon RDS 요금 페이지를 사용하여 특정 AWS 리전의 각 DB 엔진에서 지원하는 DB 인스턴스 클래스를 확인할 수 있습니다.

요금 페이지를 사용하여 리전의 각 엔진에서 지원하는 DB 인스턴스 클래스를 확인하려면

1. [Amazon RDS 요금(Amazon RDS Pricing)]으로 이동합니다.
2. AWS Pricing Calculator for Amazon RDS 섹션에서 지금 맞춤형 예상 비용 생성을 선택합니다.
3. 리전 선택에서 AWS 리전을 선택합니다.
4. 서비스 찾기에 **`Amazon RDS`**를 입력합니다.
5. 구성 옵션 및 DB 엔진에 대해 구성을 선택합니다.
6. 호환되는 인스턴스 섹션을 사용하여 지원되는 DB 인스턴스 클래스를 확인합니다.
7. (선택 사항) 계산기에서 다른 옵션을 선택한 다음 요약 저장 및 보기 또는 서비스 저장 및 추가를 선택합니다.

AWS CLI를 사용하여 AWS 리전에서 DB 인스턴스 클래스 지원 확인

AWS CLI를 사용하여 AWS 리전에서 특정 DB 엔진 및 DB 엔진 버전에 대해 지원되는 DB 인스턴스 클래스를 확인할 수 있습니다. 다음 표에는 유효한 DB 엔진 값이 나와 있습니다.

엔진 이름	CLI 명령의 엔진 값	버전에 대한 추가 정보
Db2	`db2-ae` `db2-se`	Amazon RDS의 Db2 버전
MariaDB	`mariadb`	Amazon RDS MariaDB 버전
Microsoft SQL Server	`sqlserver-ee` `sqlserver-se`	Amazon RDS의 Microsoft SQL Server 버전

엔진 이름	CLI 명령의 엔진 값	버전에 대한 추가 정보
	`sqlserver-ex` `sqlserver-web`	
MySQL	`mysql`	Amazon RDS의 MySQL 버전
Oracle	`oracle-ee` `oracle-se2`	Amazon RDS for Oracle 릴리스 정보
PostgreSQL	`postgres`	사용 가능한 PostgreSQL 데이터베이스 버전

AWS 리전 이름에 대한 자세한 내용은 AWS 리전 섹션을 참조하세요.

다음 예는 describe-orderable-db-instance-options AWS 리전 명령을 사용하여 AWS CLI 리전에서 DB 인스턴스 클래스 지원을 확인하는 방법을 보여줍니다.

> **Note**
>
> 출력을 제한하기 위해 다음 예시에서는 범용 SSD(gp2) 스토리지 유형에 대한 결과만 보여줍니다. 필요한 경우 명령에서 스토리지 유형을 범용 SSD(gp3), 프로비저닝된 IOPS(io1), 프로비저닝된 IOPS Block Express(io2) 또는 마그네틱(standard)으로 변경할 수 있습니다.

주제

- AWS 리전의 특정 DB 엔진 버전에서 지원하는 DB 인스턴스 클래스 나열
- AWS 리전에서 특정 DB 인스턴스 클래스를 지원하는 DB 엔진 버전 나열
- 특정 DB 엔진 및 인스턴스 클래스를 지원하는 AWS 리전 나열

AWS 리전의 특정 DB 엔진 버전에서 지원하는 DB 인스턴스 클래스 나열

AWS 리전 리전의 특정 DB 엔진 버전에서 지원하는 DB 인스턴스 클래스를 나열하려면 다음 명령을 실행합니다.

대상 LinuxmacOS, 또는Unix:

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

Windows의 경우:

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

예를 들어 다음 명령은 미국 동부(버지니아 북부)에서 RDS for PostgreSQL의 DB 엔진 버전 13.6에 대해 지원되는 DB 인스턴스 클래스를 나열합니다.

대상 LinuxmacOS, 또는Unix:

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

Windows의 경우:

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

AWS 리전에서 특정 DB 인스턴스 클래스를 지원하는 DB 엔진 버전 나열

AWS 리전 리전에서 특정 DB 인스턴스 클래스를 지원하는 DB 엔진 버전을 나열하려면 다음 명령을 실행합니다.

대상 LinuxmacOS, 또는Unix:

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

Windows의 경우:

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

예를 들어 다음 명령은 US East (N. Virginia)에서 db.r5.large DB 인스턴스 클래스를 지원하는 RDS for PostgreSQL DB 엔진의 DB 엔진 버전을 나열합니다.

대상 LinuxmacOS, 또는Unix:

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

Windows의 경우:

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

특정 DB 엔진 및 인스턴스 클래스를 지원하는 AWS 리전 나열

다음 bash 스크립트는 DB 엔진과 인스턴스 클래스의 특정 조합을 지원하는 모든 AWS 리전을 나열합니다.

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

if [ $# -ne 2 ]; then
  echo "Usage: $0 <db-engine> <db-instance-class>"
  exit 1
fi
ENGINE="$1"
INSTANCE_CLASS="$2"
REGIONS=$(aws ec2 describe-regions --query "Regions[].RegionName" --output text)
for region in $REGIONS; do
  supported_count=$(aws rds describe-orderable-db-instance-options \
    --region "$region" \
    --engine "$ENGINE" \
    --db-instance-class "$INSTANCE_CLASS" \
    --query 'length(OrderableDBInstanceOptions)' \
    --output text 2>/dev/null || echo "0")
  if [ "$supported_count" -gt 0 ]; then
    echo "$region supports $INSTANCE_CLASS for $ENGINE."
  else
    echo "$region doesn't support $INSTANCE_CLASS for $ENGINE."
  fi
done
```

다음 샘플 출력은 db.r8g.large 인스턴스 클래스를 사용하여 RDS for MySQL에 대한 리전 지원을 확인합니다.

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

```
ap-southeast-1 doesn't support db.r8g.large for mysql.
ap-southeast-2 doesn't support db.r8g.large for mysql.
eu-central-1 supports db.r8g.large for mysql.
us-east-1 supports db.r8g.large for mysql.
us-east-2 supports db.r8g.large for mysql.
us-west-1 doesn't support db.r8g.large for mysql.
us-west-2 supports db.r8g.large for mysql.
```

DB 인스턴스 클래스 변경

DB 인스턴스 클래스를 변경하여 DB 인스턴스에서 사용 가능한 CPU 및 메모리를 변경할 수 있습니다. DB 인스턴스 클래스를 변경하려면 Amazon RDS DB 인스턴스 수정의 지침에 따라 DB 인스턴스를 수정합니다.

RDS for Oracle에서 DB 인스턴스 클래스의 프로세서 구성

Amazon RDS DB 인스턴스 클래스는 여러 개의 스레드를 하나의 Intel Xeon CPU 코어에서 동시에 실행할 수 있는 하이퍼 스레딩 기술을 지원합니다. 각 스레드는 DB 인스턴스에서 가상 CPU(vCPU)로 표현됩니다. DB 인스턴스에는 DB 인스턴스 클래스에 따라 다른 기본 CPU 코어 수가 있습니다. 예를 들어, db.m4.xlarge DB 인스턴스 클래스에는 기본적으로 2개의 CPU 코어 및 코어당 2개의 스레드 - 총 4개의 vCPU가 있습니다.

> **Note**
>
> 각 vCPU는 Intel Xeon CPU 코어의 하이퍼 스레딩입니다.

주제

- RDS for Oracle의 프로세서 구성 개요
- 프로세서 구성을 지원하는 DB 인스턴스 클래스
- DB 인스턴스 클래스의 CPU 코어 및 CPU 코어당 스레드 설정

RDS for Oracle의 프로세서 구성 개요

RDS for Oracle을 사용하는 경우 보통 워크로드에 적합하게 메모리와 vCPU 수가 결합된 DB 인스턴스 클래스를 확인할 수 있습니다. 하지만 다음 프로세서 기능을 지정하여 특정 워크로드 또는 비즈니스 필요에 맞게 RDS for Oracle DB 인스턴스를 최적화할 수도 있습니다.

- CPU 코어 수 – DB 인스턴스에 대한 CPU 코어 수를 사용자 지정할 수 있습니다. 이를 통해, 메모리 집약 워크로드용 RAM이 충분하면서도 CPU 코어를 적게 사용하는 DB 인스턴스의 소프트웨어 라이선스 비용을 잠재적으로 최적화할 수 있습니다.
- 코어당 스레드 – CPU 코어당 단일 스레드를 지정하여 인텔 하이퍼 스레딩 기술을 비활성화할 수 있습니다. HPC(고성능 컴퓨팅) 워크로드와 같은 특정 워크로드에 대해 이 작업을 수행할 수 있습니다.

각 코어에 대해 개별적으로 CPU 코어 및 스레드 수를 제어할 수 있습니다. 요청에서 하나의 값 또는 두 값을 설정할 수 있습니다. 설정이 DB 인스턴스와 연결된 후에는 변경할 때까지 설정이 지속됩니다.

DB 인스턴스에 대한 프로세서 설정은 DB 인스턴스의 스냅샷과 연결됩니다. 스냅샷이 복원되면 복원된 DB 인스턴스는 스냅샷을 생성할 때 사용된 프로세서 기능 설정을 사용합니다.

DB 인스턴스의 DB 인스턴스 클래스를 기본값이 아닌 프로세서 설정으로 수정하는 경우 수정 시 기본 프로세서 설정을 지정하거나 프로세서 설정을 명시적으로 지정하십시오. 이 요구 사항을 준수하면 DB 인스턴스를 수정할 때 발생할 수 있는 타사 라이선스 비용을 알 수 있습니다.

RDS for Oracle DB 인스턴스에서 프로세서 기능 지정에 대한 추가 요금이나 비용 경감은 없습니다. 기본 CPU 구성으로 시작한 DB 인스턴스와 동일한 요금이 청구됩니다.

프로세서 구성을 지원하는 DB 인스턴스 클래스

다음 조건이 충족되는 경우에만 코어당 CPU 코어 수 및 코어당 스레드 수를 구성할 수 있습니다.

- RDS for Oracle DB 인스턴스를 구성하고 있습니다. 다른 Oracle 데이터베이스 에디션에서 지원되는 DB 인스턴스 클래스에 대한 자세한 내용은 RDS for Oracle DB 인스턴스 클래스 섹션을 참조하세요.
- DB 인스턴스에서 RDS for Oracle용 기존 보유 라이선스 사용(BYOL) 라이선스 옵션을 사용하고 있습니다. Oracle 라이선스 옵션에 대한 자세한 내용은 RDS for Oracle 라이선스 옵션 단원을 참조하십시오.
- DB 인스턴스가 미리 정의된 프로세서 구성을 가진 db.r5b 또는 db.r5 인스턴스 클래스에 속하지 않습니다. 이러한 인스턴스 클래스에는 db.r5.*instance_size*.tpc*threads_per_core*.mem*ratio* 또는 db.r5b.*instance_size*.tpc*threads_per_core*.mem*ratio* 형식의 이름이 부여됩니다. 예를 들어 db.r5b.xlarge.tpc2.mem4x는 표준 db.r5b.xlarge 인스턴스 클래스보다 코어당 2개의 스레드(tpc2)와 4배 많은 메모리로 미리 구성됩니다. 이러한 최적화된 인스턴스 클래스의 프로세서 기능을 구성할 수 없습니다. 자세한 내용은 지원되는 RDS for Oracle DB 인스턴스 클래스 섹션을 참조하세요.

다음 AWS CLI 명령을 사용하여 인스턴스 클래스에 대한 기본 vCPU, 코어, 코어당 스레드 및 유효한 코어 수를 표시할 수 있습니다. 샘플 명령의 *r7i.48xlarge*를 인스턴스 클래스의 이름으로 바꿉니다.

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

다음 표에서는 다수의 CPU 코어 및 코어당 CPU 스레드 설정을 지원하는 DB 인스턴스 클래스를 확인할 수 있습니다. 각 DB 인스턴스 클래스에 대한 CPU 코어 및 코어당 CPU 스레드 수의 기본값과 유효한 값도 확인할 수 있습니다.

DB 인스턴스 클래스	기본 vCPU	기본 CPU 코어	코어당 기본 스레드	유효한 CPU 코어 수	코어당 유효한 스레드 수
db.m6i – 메모리 최적화 인스턴스 클래스					
db.m6i.large	2	1	2	1	1, 2
db.m6i.xlarge	4	2	2	2	1, 2
db.m6i.2xlarge	8	4	2	2, 4	1, 2
db.m6i.4xlarge	16	8	2	2, 4, 6, 8	1, 2
db.m6i.4xlarge	16	8	2	2, 4, 6, 8	1, 2
db.m6i.8xlarge	32	16	2	2, 4, 6, 8, 10, 12, 14, 16	1, 2
db.m6i.12xlarge	48	24	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24	1, 2
db.m6i.16xlarge	64	32	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32	1, 2

DB 인스턴스 클래스	기본 vCPU	기본 CPU 코어	코어당 기본 스레드	유효한 CPU 코어 수	코어당 유효한 스레드 수
db.m6i.24xlarge	96	48	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48	1, 2
db.m6i.32xlarge	128	64	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64	1, 2
db.m5 - 범용 인스턴스 클래스					
db.m5.large	2	1	2	1	1, 2
db.m5.xlarge	4	2	2	2	1, 2
db.m5.2xlarge	8	4	2	2, 4	1, 2
db.m5.4xlarge	16	8	2	2, 4, 6, 8	1, 2
db.m5.8xlarge	32	16	2	2, 4, 6, 8, 10, 12, 14, 16	1, 2
db.m5.12xlarge	48	24	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24	1, 2
db.m5.16xlarge	64	32	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32	1, 2
db.m5.24xlarge	96	48	2	4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48	1, 2

DB 인스턴스 클래스	기본 vCPU	기본 CPU 코어	코어당 기본 스레드	유효한 CPU 코어 수	코어당 유효한 스레드 수
db.m5d - 범용 인스턴스 클래스					
db.m5d.large	2	1	2	1	1, 2
db.m5d.xlarge	4	2	2	2	1, 2
db.m5d.2xlarge	8	4	2	2, 4	1, 2
db.m5d.4xlarge	16	8	2	2, 4, 6, 8	1, 2
db.m5d.8xlarge	32	16	2	2, 4, 6, 8, 10, 12, 14, 16	1, 2
db.m5d.12xlarge	48	24	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24	1, 2
db.m5d.16xlarge	64	32	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32	1, 2
db.m5d.24xlarge	96	48	2	4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48	1, 2
db.m4 - 범용 인스턴스 클래스					
db.m4.10xlarge	40	20	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20	1, 2
db.m4.16xlarge	64	32	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32	1, 2

DB 인스턴스 클래스	기본 vCPU	기본 CPU 코어	코어당 기본 스레드	유효한 CPU 코어 수	코어당 유효한 스레드 수
db.r7i – 메모리 최적화 인스턴스 클래스					
db.r7i.large	2	1	2	1	1, 2
db.r7i.xlarge	4	2	2	1, 2	1, 2
db.r7i.2xlarge	8	4	2	1, 2, 3, 4	1, 2
db.r7i.4xlarge	16	8	2	1, 2, 3, 4, 5, 6, 7, 8	1, 2
db.r7i.8xlarge	32	16	2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16	1, 2
db.r7i.12xlarge	48	24	2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24	1, 2
db.r7i.16xlarge	64	32	2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32	1, 2
db.r7i.24xlarge	96	48	2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48	1, 2

DB 인스턴스 클래스	기본 vCPU	기본 CPU 코어	코어당 기본 스레드	유효한 CPU 코어 수	코어당 유효한 스레드 수
db.r7i.48xlarge	192	96	2	4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96	1, 2
db.r6i – 메모리 최적화 인스턴스 클래스					
db.r6i.large	2	1	2	1	1, 2
db.r6i.xlarge	4	2	2	1, 2	1, 2
db.r6i.2xlarge	8	4	2	2, 4	1, 2
db.r6i.4xlarge	16	8	2	2, 4, 6, 8	1, 2
db.r6i.8xlarge	32	16	2	2, 4, 6, 8, 10, 12, 14, 16	1, 2
db.r6i.12xlarge	48	24	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24	1, 2
db.r6i.16xlarge	64	32	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32	1, 2
db.r6i.24xlarge	96	48	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48	1, 2

DB 인스턴스 클래스	기본 vCPU	기본 CPU 코어	코어당 기본 스레드	유효한 CPU 코어 수	코어당 유효한 스레드 수
db.r6i.32xlarge	128	64	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64	1, 2
db.r5 – 메모리 최적화 인스턴스 클래스					
db.r5.large	2	1	2	1	1, 2
db.r5.xlarge	4	2	2	2	1, 2
db.r5.2xlarge	8	4	2	2, 4	1, 2
db.r5.4xlarge	16	8	2	2, 4, 6, 8	1, 2
db.r5.8xlarge	32	16	2	2, 4, 6, 8, 10, 12, 14, 16	1, 2
db.r5.12xlarge	48	24	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24	1, 2
db.r5.16xlarge	64	32	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32	1, 2
db.r5.24xlarge	96	48	2	4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48	1, 2
db.r5 – 메모리 최적화 인스턴스 클래스					
db.r5b.large	2	1	2	1	1, 2
db.r5b.xlarge	4	2	2	2	1, 2

DB 인스턴스 클래스	기본 vCPU	기본 CPU 코어	코어당 기본 스레드	유효한 CPU 코어 수	코어당 유효한 스레드 수
db.r5b.2xlarge	8	4	2	2, 4	1, 2
db.r5b.4xlarge	16	8	2	2, 4, 6, 8	1, 2
db.r5b.8xlarge	32	16	2	2, 4, 6, 8, 10, 12, 14, 16	1, 2
db.r5b.12xlarge	48	24	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24	1, 2
db.r5b.16xlarge	64	32	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32	1, 2
db.r5b.24xlarge	96	48	2	4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48	1, 2
db.r5d – 메모리 최적화 인스턴스 클래스					
db.r5d.large	2	1	2	1	1, 2
db.r5d.xlarge	4	2	2	2	1, 2
db.r5d.2xlarge	8	4	2	2, 4	1, 2
db.r5d.4xlarge	16	8	2	2, 4, 6, 8	1, 2
db.r5d.8xlarge	32	16	2	2, 4, 6, 8, 10, 12, 14, 16	1, 2
db.r5d.12xlarge	48	24	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24	1, 2
db.r5d.16xlarge	64	32	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32	1, 2

DB 인스턴스 클래스	기본 vCPU	기본 CPU 코어	코어당 기본 스레드	유효한 CPU 코어 수	코어당 유효한 스레드 수
db.r5d.24xlarge	96	48	2	4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48	1, 2
db.r4 – 메모리 최적화 인스턴스 클래스					
db.r4.large	2	1	2	1	1, 2
db.r4.xlarge	4	2	2	1, 2	1, 2
db.r4.2xlarge	8	4	2	1, 2, 3, 4	1, 2
db.r4.4xlarge	16	8	2	1, 2, 3, 4, 5, 6, 7, 8	1, 2
db.r4.8xlarge	32	16	2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16	1, 2
db.r4.16xlarge	64	32	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32	1, 2
db.r3 – 메모리 최적화 인스턴스 클래스					
db.r3.large	2	1	2	1	1, 2
db.r3.xlarge	4	2	2	1, 2	1, 2
db.r3.2xlarge	8	4	2	1, 2, 3, 4	1, 2
db.r3.4xlarge	16	8	2	1, 2, 3, 4, 5, 6, 7, 8	1, 2
db.r3.8xlarge	32	16	2	2, 4, 6, 8, 10, 12, 14, 16	1, 2
db.x2idn – 메모리 최적화 인스턴스 클래스					

DB 인스턴스 클래스	기본 vCPU	기본 CPU 코어	코어당 기본 스레드	유효한 CPU 코어 수	코어당 유효한 스레드 수
db.x2idn.16xlarge	64	32	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32	1, 2
db.x2idn.24xlarge	96	48	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48	1, 2
db.x2idn.32xlarge	128	64	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64	1, 2
db.x2iedn – 메모리 최적화 인스턴스 클래스					
db.x2iedn.xlarge	4	2	2	1, 2	1, 2
db.x2iedn.2xlarge	8	4	2	2, 4	1, 2
db.x2iedn.4xlarge	16	8	2	2, 4, 6, 8	1, 2
db.x2iedn.8xlarge	32	16	2	2, 4, 6, 8, 10, 12, 14, 16	1, 2
db.x2iedn.16xlarge	64	32	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32	1, 2

DB 인스턴스 클래스	기본 vCPU	기본 CPU 코어	코어당 기본 스레드	유효한 CPU 코어 수	코어당 유효한 스레드 수
db.x2iedn .24xlarge	96	48	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48	1, 2
db.x2iedn .32xlarge	128	64	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64	1, 2
db.x2iezn – 메모리 최적화 인스턴스 클래스					
db.x2iezn .2xlarge	8	4	2	2, 4	1, 2
db.x2iezn .4xlarge	16	8	2	2, 4, 6, 8	1, 2
db.x2iezn .6xlarge	24	12	2	2, 4, 6, 8, 10, 12	1, 2
db.x2iezn .8xlarge	32	16	2	2, 4, 6, 8, 10, 12, 14, 16	1, 2
db.x2iezn .12xlarge	48	24	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24	1, 2
db.x1 – 메모리 최적화 인스턴스 클래스					
db.x1.16xlarge	64	32	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32	1, 2

DB 인스턴스 클래스	기본 vCPU	기본 CPU 코어	코어당 기본 스레드	유효한 CPU 코어 수	코어당 유효한 스레드 수
db.x1.32xlarge	128	64	2	4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 64	1, 2
db.x1e – 메모리 최적화 인스턴스 클래스					
db.x1e.xlarge	4	2	2	1, 2	1, 2
db.x1e.2xlarge	8	4	2	1, 2, 3, 4	1, 2
db.x1e.4xlarge	16	8	2	1, 2, 3, 4, 5, 6, 7, 8	1, 2
db.x1e.8xlarge	32	16	2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16	1, 2
db.x1e.16 xlarge	64	32	2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32	1, 2
db.x1e.32 xlarge	128	64	2	4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 64	1, 2
db.z1d – 메모리 최적화 인스턴스 클래스					
db.z1d.large	2	1	2	1	1, 2
db.z1d.xlarge	4	2	2	2	1, 2
db.z1d.2xlarge	8	4	2	2, 4	1, 2
db.z1d.3xlarge	12	6	2	2, 4, 6	1, 2
db.z1d.6xlarge	24	12	2	2, 4, 6, 8, 10, 12	1, 2

DB 인스턴스 클래스	기본 vCPU	기본 CPU 코어	코어당 기본 스레드	유효한 CPU 코어 수	코어당 유효한 스레드 수
db.z1d.12xlarge	48	24	2	4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24	1, 2

> **Note**
>
> AWS CloudTrail을 사용해 Amazon RDS for Oracle DB 인스턴스의 프로세스 구성에 대한 변경 사항을 모니터링하고 감사할 수 있습니다. CloudTrail 사용에 관한 자세한 내용은 AWS CloudTrail에서 Amazon RDS API 호출 모니터링 단원을 참조하십시오.

DB 인스턴스 클래스의 CPU 코어 및 CPU 코어당 스레드 설정

다음 작업을 수행할 때 DB 인스턴스 클래스의 CPU 코어 및 코어당 스레드 수를 구성할 수 있습니다.

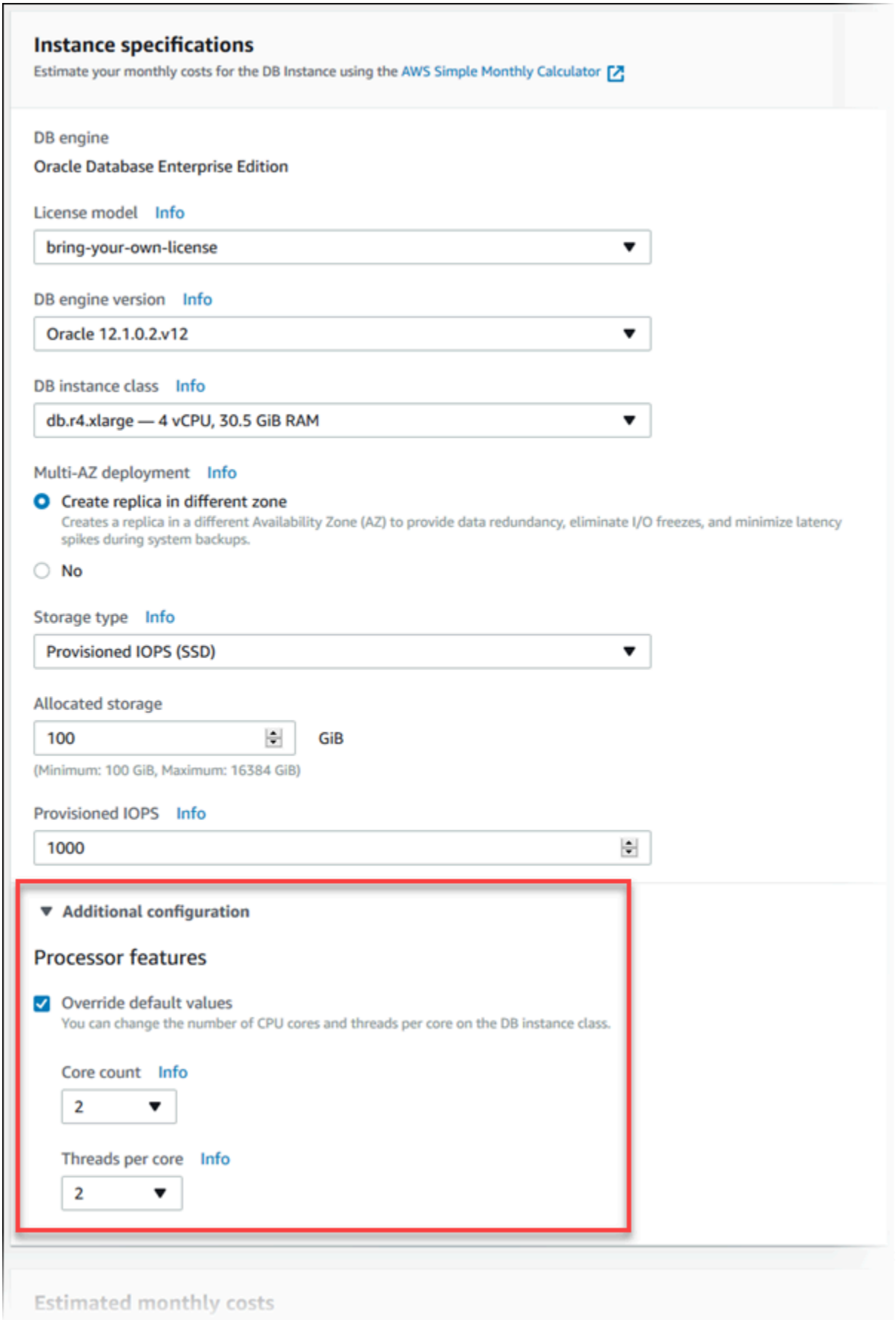
- Amazon RDS DB 인스턴스 생성
- Amazon RDS DB 인스턴스 수정
- DB 인스턴스 복원
- Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원

> **Note**
>
> CPU 코어 또는 코어당 스레드 수를 구성하기 위해 DB 인스턴스를 수정할 때 DB 인스턴스가 잠시 중단됩니다.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB 인스턴스 클래스의 CPU 코어 및 CPU 코어당 스레드를 설정할 수 있습니다.

콘솔

DB 인스턴스를 생성, 수정 또는 복원할 때 AWS Management Console에서 DB 인스턴스 클래스를 설정합니다. 인스턴스 사양 단원에는 프로세서에 대한 옵션이 표시됩니다. 다음 이미지는 프로세서 기능 옵션을 보여줍니다.

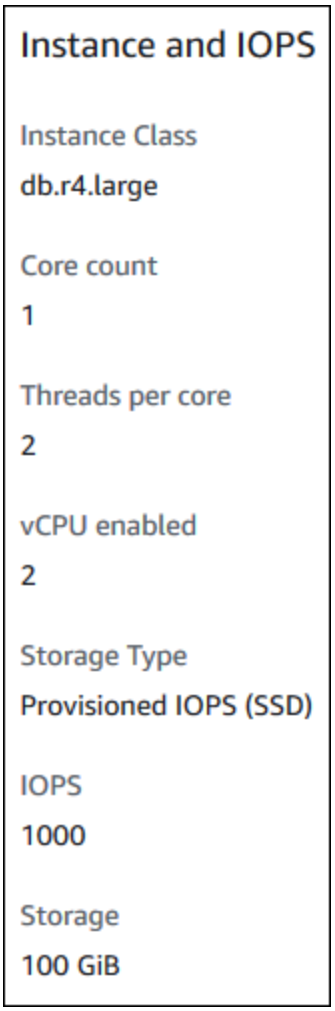

Instance specifications
Estimate your monthly costs for the DB Instance using the AWS Simple Monthly Calculator
DB engine
Oracle Database Enterprise Edition
License model Info
bring-your-own-license
DB engine version Info
Oracle 12.1.0.2.v12
DB instance class Info
db.r4.xlarge — 4 vCPU, 30.5 GiB RAM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

Processor features(프로세서 기능)에서 다음 옵션을 DB 인스턴스 클래스에 적절한 값으로 설정합니다.

- Core count(코어 수) – 이 옵션을 사용하여 CPU 코어 수를 설정합니다. 이 값은 DB 인스턴스의 최대 CPU 코어 수보다 작거나 같아야 합니다.
- Threads per core(코어당 스레드) – 코어당 여러 스레드를 활성화하려면 2를 지정하거나, 코어다 여러 스레드를 비활성화하려면 1을 설정합니다.

DB 인스턴스를 수정하거나 복원할 때 CPU 코어 및 CPU 코어당 스레드를 인스턴스 클래스의 기본값으로 설정할 수도 있습니다.

콘솔에서 DB 인스턴스에 대한 세부 정보를 볼 때 구성 탭에서 DB 인스턴스 클래스에 대한 프로세서 정보를 볼 수 있습니다. 다음 이미지는 CPU 코어 하나 및 코어당 여러 스레드가 있는 DB 인스턴스 클래스를 보여 줍니다.



Oracle DB 인스턴스의 경우 BYOL(기존 보유 라이선스 사용) DB 인스턴스에 대한 프로세서 정보만 나타납니다.

AWS CLI

다음 AWS CLI 명령 중 하나를 실행할 때 DB 인스턴스의 프로세서 기능을 설정할 수 있습니다.

- create-db-instance
- modify-db-instance
- restore-db-instance-from-db-snapshot
- restore-db-instance-from-s3
- restore-db-instance-to-point-in-time

AWS CLI를 사용하여 DB 인스턴스에 대한 DB 인스턴스 클래스의 프로세서를 구성하려면 `--processor-features` 옵션을 명령에 포함시킵니다. `coreCount` 기능 이름을 사용하여 CPU 코어 수를 지정하고, `threadsPerCore` 기능 이름을 사용하여 코어당 여러 스레드가 활성화되는지 여부를 지정합니다.

옵션에는 다음과 같은 구문이 있습니다.

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

다음은 프로세서를 구성하는 예제입니다.

예제:

- DB 인스턴스의 CPU 코어 수 설정
- DB 인스턴스의 CPU 코어 수 설정 및 여러 스레드 비활성화
- DB 인스턴스 클래스에 유효한 프로세서 값 보기
- DB 인스턴스의 기본 프로세서 설정으로 돌아가기
- DB 인스턴스의 기본 CPU 코어 수로 돌아가기
- DB 인스턴스의 코어당 기본 스레드 수로 돌아가기

DB 인스턴스의 CPU 코어 수 설정

Example

다음 예에서는 CPU 코어 수를 4로 설정하여 mydbinstance를 수정합니다. --apply-immediately를 사용하면 변경 내용이 즉시 적용됩니다. 다음 예약 유지 관리 기간 중에 변경 내용을 적용하려는 경우 --apply-immediately 옵션을 생략합니다.

대상 LinuxmacOS, 또는Unix:

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

Windows의 경우:

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

DB 인스턴스의 CPU 코어 수 설정 및 여러 스레드 비활성화

Example

다음 예제에서는 CPU 코어 수를 mydbinstance로 설정하고 코어당 여러 스레드를 비활성화하여 4를 수정합니다. --apply-immediately를 사용하면 변경 내용이 즉시 적용됩니다. 다음 예약 유지 관리 기간 중에 변경 내용을 적용하려는 경우 --apply-immediately 옵션을 생략합니다.

대상 LinuxmacOS, 또는Unix:

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

Windows의 경우:

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

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

DB 인스턴스 클래스에 유효한 프로세서 값 보기

Example

describe-orderable-db-instance-options 명령을 실행하고 --db-instance-class 옵션의 인스턴스 클래스를 지정하여 특정 DB 인스턴스에 유효한 프로세서 값을 볼 수 있습니다. 예를 들어, 다음 명령의 출력은 db.r3.large 인스턴스 클래스의 프로세서 옵션을 보여 줍니다.

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

다음은 JSON 형식 명령의 샘플 출력입니다.

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

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

또한 DB 인스턴스 클래스 프로세서 정보에 대해 다음 명령을 실행할 수 있습니다.

- describe-db-instances – 지정된 DB 인스턴스에 대한 프로세서 정보를 보여 줍니다.
- describe-db-snapshots – 지정된 DB 스냅샷에 대한 프로세서 정보를 보여 줍니다.
- describe-valid-db-instance-modifications – 지정된 DB 인스턴스에 유효한 프로세서 수정 사항을 보여 줍니다.

앞의 명령의 출력에서 프로세서 기능 값은 다음 조건이 충족되는 경우에만 null이 아닙니다.

- RDS for Oracle DB 인스턴스를 사용하고 있습니다.
- RDS for Oracle DB 인스턴스가 프로세서 값 변경을 지원합니다.
- 현재 CPU 코어 및 스레드 설정이 기본값이 아닌 값으로 설정됩니다.

앞의 조건이 충족되지 않으면 describe-db-instance를 사용하여 인스턴스 유형을 가져올 수 있습니다. EC2 작업 describe-instance-types를 실행하여 이 인스턴스 유형에 대한 프로세서 정보를 얻을 수 있습니다.

DB 인스턴스의 기본 프로세서 설정으로 돌아가기

Example

다음 예제에서는 DB 인스턴스 클래스를 기본 프로세서 값으로 되돌려서 mydbinstance를 수정합니다. --apply-immediately를 사용하면 변경 내용이 즉시 적용됩니다. 다음 예약 유지 관리 기간 중에 변경 내용을 적용하려는 경우 --apply-immediately 옵션을 생략합니다.

대상 LinuxmacOS, 또는Unix:

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

Windows의 경우:

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

DB 인스턴스의 기본 CPU 코어 수로 돌아가기

Example

다음 예제에서는 DB 인스턴스 클래스를 기본 CPU 코어 수로 되돌려서 mydbinstance를 수정합니다. 코어당 스레드 설정은 변경되지 않습니다. --apply-immediately를 사용하면 변경 내용이 즉시 적용됩니다. 다음 예약 유지 관리 기간 중에 변경 내용을 적용하려는 경우 --apply-immediately 옵션을 생략합니다.

대상 LinuxmacOS, 또는Unix:

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

Windows의 경우:

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

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

DB 인스턴스의 코어당 기본 스레드 수로 돌아가기

Example

다음 예제에서는 DB 인스턴스 클래스를 코어당 기본 스레드 수로 되돌려서 mydbinstance를 수정합니다. CPU 코어 수 설정은 변경되지 않습니다. --apply-immediately를 사용하면 변경 내용이 즉시 적용됩니다. 다음 예약 유지 관리 기간 중에 변경 내용을 적용하려는 경우 --apply-immediately 옵션을 생략합니다.

대상 LinuxmacOS, 또는Unix:

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

Windows의 경우:

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

RDS API

다음 Amazon RDS API 작업 중 하나를 호출할 때 DB 인스턴스의 프로세서 기능을 설정할 수 있습니다.

- CreateDBInstance
- ModifyDBInstance
- RestoreDBInstanceFromDBSnapshot
- RestoreDBInstanceFromS3
- RestoreDBInstanceToPointInTime

Amazon RDS API를 사용하여 DB 인스턴스에 대한 DB 인스턴스 클래스의 프로세서 기능을 구성하려면 ProcessFeatures 파라미터를 호출에 포함시킵니다.

파라미터의 구문은 다음과 같습니다.

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

`coreCount` 기능 이름을 사용하여 CPU 코어 수를 지정하고, `threadsPerCore` 기능 이름을 사용하여 코어당 여러 스레드가 활성화되는지 여부를 지정합니다.

DescribeOrderableDBInstanceOptions 작업을 실행하고 `DBInstanceClass` 파라미터의 인스턴스 클래스를 지정하여 특정 DB 인스턴스 클래스에 유효한 프로세서 값을 볼 수 있습니다. 다음 작업을 사용할 수도 있습니다.

- DescribeDBInstances – 지정된 DB 인스턴스에 대한 프로세서 정보를 보여 줍니다.
- DescribeDBSnapshots – 지정된 DB 스냅샷에 대한 프로세서 정보를 보여 줍니다.
- DescribeValidDBInstanceModifications – 지정된 DB 인스턴스에 유효한 프로세서 수정 사항을 보여 줍니다.

앞의 작업의 출력에서 프로세서 기능 값은 다음 조건이 충족되는 경우에만 null이 아닙니다.

- RDS for Oracle DB 인스턴스를 사용하고 있습니다.
- RDS for Oracle DB 인스턴스가 프로세서 값 변경을 지원합니다.
- 현재 CPU 코어 및 스레드 설정이 기본값이 아닌 값으로 설정됩니다.

앞의 조건이 충족되지 않으면 DescribeDBInstances를 사용하여 인스턴스 유형을 가져올 수 있습니다. EC2 작업 DescribeInstanceTypes를 실행하여 이 인스턴스 유형에 대한 프로세서 정보를 얻을 수 있습니다.

에 대한 DB 인스턴스 클래스의 하드웨어 사양

이 섹션의 여러 표에서 Amazon RDS DB 인스턴스 클래스에 대한 하드웨어 세부 정보를 확인할 수 있습니다.

각 DB 인스턴스 클래스에 대한 Amazon RDS DB 엔진 지원에 관한 자세한 내용은 DB 인스턴스 클래스에 지원되는 DB 엔진 단원을 참조하십시오.

주제

- DB 인스턴스 클래스의 하드웨어 용어
- 범용 인스턴스 클래스에 대한 하드웨어 사양

- 메모리 최적화 인스턴스 클래스에 대한 하드웨어 사양
- 컴퓨팅 최적화 인스턴스 클래스에 대한 하드웨어 사양
- 성능 버스트 가능 인스턴스 클래스에 대한 하드웨어 사양

DB 인스턴스 클래스의 하드웨어 용어

다음 용어는 DB 인스턴스 클래스의 하드웨어 사양을 기술하는 데 사용됩니다.

vCPU

가상 CPU(중앙 처리 유닛)의 수입니다. 가상 CPU는 DB 인스턴스 클래스를 비교하는 데 사용할 수 있는 용량 단위입니다. 특정 프로세서를 구매하거나 임차해 몇 개월 또는 몇 년간 사용하는 것이 아니라, 시간 단위로 용량을 임대합니다. 목표는 실제 기본 하드웨어의 제한 내에서 일정하고 구체적인 CPU 용량을 제공하는 것입니다.

ECU

Amazon EC2 인스턴스의 정수 처리 파워에 대한 상대적인 척도입니다. 개발자들이 다양한 인스턴스 클래스 간에 CPU 용량을 손쉽게 비교할 수 있도록 Amazon EC2 컴퓨팅 유닛(ECU)을 정의했습니다. 특정 인스턴스에 할당된 CPU의 용량은 이러한 EC2 컴퓨팅 유닛(ECU)으로 표현됩니다. 현재 ECU 한 개당 제공하는 CPU 용량은 1.0–1.2GHz 2007 Opteron 또는 2007 Xeon 프로세서와 동일합니다.

메모리(GiB)

DB 인스턴스에 할당되는 RAM(단위: 기비바이트)입니다. 메모리와 vCPU 간 일정한 비율이 존재하는 경우가 많다는 점에 유의하십시오. db.r5 인스턴스 클래스와 비슷한 vCPU 비율에 대한 메모리가 있는 db.r4 인스턴스 클래스를 예로 들 수 있습니다. 그러나 대부분의 사용 db.r5 인스턴스 클래스는 db.r4 인스턴스 클래스보다 더 낮고 더 일관성 있는 성능을 제공합니다.

EBS 최적화

DB 인스턴스는 최적화된 구성 스택을 사용하며 I/O에 대한 전용 용량을 추가로 제공합니다. 이 최적화는 I/O와 인스턴스 간의 경합을 최소화하여 최상의 성능을 제공합니다. Amazon EBS 최적화 인스턴스에 대한 자세한 내용은 Amazon EC2 사용 설명서에서 Amazon EBS 최적화 인스턴스를 참조하세요.

EBS 최적화 인스턴스에는 기준 및 최대 IOPS 속도가 있습니다. 최대 IOPS 속도가 DB 인스턴스 수준에서 강제 적용됩니다. EBS 볼륨이 결합되어 최댓값보다 높은 하나의 IOPS 속도를 갖는 EBS 볼륨 세트는 인스턴스 수준 임계값을 초과할 수 없습니다. 예를 들어 특정 DB 인스턴스 클래스의 최대 IOPS가 40,000이고 64,000 IOPS EBS 볼륨 4개를 연결하는 경우 최대 IOPS는 256,000이 아니

라 40,000입니다. 각 EC2 인스턴스 유형별 최대 IOPS는 Linux 인스턴스용 Amazon EC2 사용 설명서의 지원되는 인스턴스 유형을 참조하세요.

최대 EBS 대역폭(Mbps)

초당 메가비트 단위의 최대 EBS 대역폭입니다. 이 값을 8로 나누면 초당 메가바이트 단위로 예상되는 처리량을 구할 수 있습니다.

> **Important**
>
> Amazon RDS DB 인스턴스의 범용 SSD(gp2) 볼륨은 대부분의 경우 처리량 제한이 250MiB/s입니다. 그러나 처리량 제한은 볼륨 크기에 따라 달라질 수 있습니다. 자세한 내용을 알아보려면 Amazon EC2 사용 설명서의 Amazon EBS 볼륨 유형을 참조하세요.

네트워크 대역폭

다른 DB 인스턴스 클래스 대비 네트워크 속도입니다.

범용 인스턴스 클래스에 대한 하드웨어 사양

다음 표에는 범용 인스턴스 클래스에 대한 컴퓨팅, 메모리, 스토리지 및 대역폭 사양이 나와 있습니다.

db.m8g - AWS Graviton4 프로세서로 구동되는 범용 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.m8g.48xlarge	192	—	768	EBS 최적화 전용	40,000	50
db.m8g.24xlarge	96	—	512	EBS 최적화 전용	30,000개	40
db.m8g.16xlarge	64	—	384	EBS 최적화 전용	20,000건	30
db.m8g.12xlarge	48	—	256	EBS 최적화 전용	15,000	22.5

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.m8g.8xlarge	32	—	128	EBS 최적화 전용	10,000	15
db.m8g.4xlarge*	16	—	64	EBS 최적화 전용	최대 10,000	최대 15
db.m8g.2xlarge*	8	—	32	EBS 최적화 전용	최대 10,000	최대 15
db.m8g.xlarge*	4	—	16	EBS 최적화 전용	최대 10,000	최대 12.5
db.m8g.large*	2	—	8	EBS 최적화 전용	최대 10,000	최대 12.5

db.m7i - 4세대 인텔 제온 스케일러블 프로세서로 구동되는 범용 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.m7i.48xlarge	192	—	768	EBS 최적화 전용	40,000	50
db.m7i.24xlarge	96	—	384	EBS 최적화 전용	30,000개	37.5
db.m7i.16xlarge	64	—	256	EBS 최적화 전용	20,000건	25
db.m7i.12xlarge	48	—	192	EBS 최적화 전용	15,000	18.75
db.m7i.8xlarge	32	—	128	EBS 최적화 전용	10,000	12.5

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.m7i.4xlarge	16	—	64	EBS 최적화 전용	최대 10,000	최대 12.5
db.m7i.2xlarge	8	—	32	EBS 최적화 전용	최대 10,000	최대 12.5
db.m7i.xlarge	4	—	16	EBS 최적화 전용	최대 10,000	최대 12.5
db.m7i.large	2	—	8	EBS 최적화 전용	최대 10,000	최대 12.5

db.m7g - AWS Graviton3 프로세서로 구동되는 범용 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.m7g.16xlarge	64	—	256	EBS 최적화 전용	20,000건	30
db.m7g.12xlarge	48	—	192	EBS 최적화 전용	15,000	22.5
db.m7g.8xlarge	32	—	128	EBS 최적화 전용	10,000	15
db.m7g.4xlarge	16	—	64	EBS 최적화 전용	최대 10,000	최대 15
db.m7g.2xlarge*	8	—	32	EBS 최적화 전용	최대 10,000	최대 15
db.m7g.xlarge*	4	—	16	EBS 최적화 전용	최대 10,000	최대 12.5

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.m7g.large*	2	—	8	EBS 최적화 전용	최대 10,000	최대 12.5

db.m6g - AWS Graviton2 프로세서로 구동되는 범용 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.m6g.16xlarge	64	—	256	EBS 최적화 전용	19,000	25
db.m6g.12xlarge	48	—	192	EBS 최적화 전용	13,500	20
db.m6g.8xlarge	32	—	128	EBS 최적화 전용	9,000	12
db.m6g.4xlarge	16	—	64	EBS 최적화 전용	4,750	최대 10
db.m6g.2xlarge*	8	—	32	EBS 최적화 전용	최대 4,750	최대 10
db.m6g.xlarge*	4	—	16	EBS 최적화 전용	최대 4,750	최대 10
db.m6g.large*	2	—	8	EBS 최적화 전용	최대 4,750	최대 10

db.m6gd - AWS Graviton2 프로세서 및 SSD 스토리지로 구동되는 범용 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.m6gd.16xlarge	64	—	256	2 x 1900 NVMe SSD	19,000	25
db.m6gd.12xlarge	48	—	192	2 x 1425 NVMe SSD	13,500	20
db.m6gd.8xlarge	32	—	128	1 x 1900 NVMe SSD	9,000	12
db.m6gd.4xlarge*	16	—	64	1 x 950 NVMe SSD	4,750	최대 10
db.m6gd.2xlarge*	8	—	32	1 x 474 NVMe SSD	최대 4,750	최대 10
db.m6gd.xlarge*	4	—	16	1 x 237 NVMe SSD	최대 4,750	최대 10
db.m6gd.large*	2	—	8	1 x 118 NVMe SSD	최대 4,750	최대 10

db.m6id - 3세대 인텔 제온 스케일러블 프로세서 및 SSD 스토리지로 구동되는 범용 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.m6id.32xlarge	128	—	512	4 x 1900 NVMe SSD	40,000	50
db.m6id.24xlarge	96	—	384	4 x 1425 NVMe SSD	30,000개	37.5
db.m6id.16xlarge	64	—	256	2 x 1900 NVMe SSD	20,000건	25

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.m6id.12xlarge	48	—	192	2 x 1425 NVMe SSD	15,000	18.75
db.m6id.8xlarge	32	—	128	1 x 1900 NVMe SSD	10,000	12.5
db.m6id.4xlarge*	16	—	64	1 x 950 NVMe SSD	최대 10,000	최대 12.5
db.m6id.2xlarge*	8	—	32	1 x 474 NVMe SSD	최대 10,000	최대 12.5
db.m6id.xlarge*	4	—	16	1 x 237 NVMe SSD	최대 10,000	최대 12.5
db.m6id.large*	2	—	8	1 x 118 NVMe SSD	최대 10,000	최대 12.5

db.m6idn - 3세대 인텔 제온 스케일러블 프로세서, SSD 스토리지 및 네트워크 최적화를 포함하는 범용 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.m6idn.32xlarge	128	—	512	4 x 1900 NVMe SSD	80,000	200
db.m6idn.24xlarge	96	—	384	4 x 1425 NVMe SSD	60,000	150
db.m6idn.16xlarge	64	—	256	2 x 1900 NVMe SSD	40,000	100
db.m6idn.12xlarge	48	—	192	2 x 1425 NVMe SSD	30,000개	75

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.m6idn.8xlarge	32	—	128	1 x 1900 NVMe SSD	20,000건	50
db.m6idn.4xlarge*	16	—	64	1 x 950 NVMe SSD	최대 20,000개	최대 50개
db.m6idn.2xlarge*	8	—	32	1 x 474 NVMe SSD	최대 20,000개	최대 40개
db.m6idn.xlarge*	4	—	16	1 x 237 NVMe SSD	최대 20,000개	최대 30개
db.m6idn.large*	2	—	8	1 x 118 NVMe SSD	최대 20,000개	최대 25개

db.m6in - 3세대 인텔 제온 스케일러블 프로세서 및 네트워크 최적화로 구동되는 범용 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.m6in.32xlarge	128	—	512	EBS 최적화 전용	80,000	200
db.m6in.24xlarge	96	—	384	EBS 최적화 전용	60,000	150
db.m6in.16xlarge	64	—	256	EBS 최적화 전용	40,000	100
db.m6in.12xlarge	48	—	192	EBS 최적화 전용	30,000개	75
db.m6in.8xlarge	32	—	128	EBS 최적화 전용	20,000건	50

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.m6in.4xlarge*	16	—	64	EBS 최적화 전용	최대 20,000개	최대 50개
db.m6in.2xlarge*	8	—	32	EBS 최적화 전용	최대 20,000개	최대 40개
db.m6in.xlarge*	4	—	16	EBS 최적화 전용	최대 20,000개	최대 30개
db.m6in.large*	2	—	8	EBS 최적화 전용	최대 20,000개	최대 25개

db.m6i - 3세대 인텔 제온 스케일러블 프로세서로 구동되는 범용 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.m6i.32xlarge	128	—	512	EBS 최적화 전용	40,000	50
db.m6i.24xlarge	96	—	384	EBS 최적화 전용	30,000개	37.5
db.m6i.16xlarge	64	—	256	EBS 최적화 전용	20,000건	25
db.m6i.12xlarge	48	—	192	EBS 최적화 전용	15,000	18.75
db.m6i.8xlarge	32	—	128	EBS 최적화 전용	10,000	12.5
db.m6i.4xlarge*	16	—	64	EBS 최적화 전용	최대 10,000	최대 12.5

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.m6i.2xlarge*	8	—	32	EBS 최적화 전용	최대 10,000	최대 12.5
db.m6i.xlarge*	4	—	16	EBS 최적화 전용	최대 10,000	최대 12.5
db.m6i.large*	2	—	8	EBS 최적화 전용	최대 10,000	최대 12.5

db.m5d - 인텔 제온 플래티넘 프로세서 및 SSD 스토리지로 구동되는 범용 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.m5d.24xlarge	96	345	384	4 x 900 NVMe SSD	19,000	25
db.m5d.16xlarge	64	262	256	4 x 600 NVMe SSD	13,600	20
db.m5d.12xlarge	48	173	192	2 x 900 NVMe SSD	9,500	10
db.m5d.8xlarge	32	131	128	2 x 600 NVMe SSD	6,800	10
db.m5d.4xlarge	16	61	64	2 x 300 NVMe SSD	4,750	최대 10
db.m5d.2xlarge*	8	31	32	1 x 300 NVMe SSD	최대 4,750	최대 10
db.m5d.xlarge*	4	15	16	1 x 150 NVMe SSD	최대 4,750	최대 10

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.m5d.large*	2	10	8	1 x 75 NVMe SSD	최대 4,750	최대 10

db.m5 - 인텔 제온 플래티넘 프로세서를 포함하는 범용 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.m5.24xlarge	96	345	384	EBS 최적화 전용	19,000	25
db.m5.16xlarge	64	262	256	EBS 최적화 전용	13,600	20
db.m5.12xlarge	48	173	192	EBS 최적화 전용	9,500	10
db.m5.8xlarge	32	131	128	EBS 최적화 전용	6,800	10
db.m5.4xlarge	16	61	64	EBS 최적화 전용	4,750	최대 10
db.m5.2xlarge*	8	31	32	EBS 최적화 전용	최대 4,750	최대 10
db.m5.xlarge*	4	15	16	EBS 최적화 전용	최대 4,750	최대 10
db.m5.large*	2	10	8	EBS 최적화 전용	최대 4,750	최대 10

db.m4 - 인텔 제온 스케일러블 프로세서를 지원하는 범용 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.m4.16xlarge	64	188	256	EBS 최적화 전용	10,000	25
db.m4.10xlarge	40	124.5	160	EBS 최적화 전용	4,000	10
db.m4.4xlarge	16	53.5	64	EBS 최적화 전용	2,000	높음
db.m4.2xlarge	8	25.5	32	EBS 최적화 전용	1,000	높음
db.m4.xlarge	4	13	16	EBS 최적화 전용	750	높음
db.m4.large	2	6.5	8	EBS 최적화 전용	450	보통

db.m3 - 범용 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.m3.2xlarge	8	26	30	EBS 최적화 전용	1,000	높음
db.m3.xlarge	4	13	15	EBS 최적화 전용	500	높음
db.m3.large	2	6.5	7.5	EBS 전용	—	보통
db.m3.medium	1	3	3.75	EBS 전용	—	보통

* 이러한 DB 인스턴스 클래스에서는 24시간마다 최소 한 번씩 30분간 최대 성능을 지원합니다. 기본 EC2 인스턴스 유형의 기본 성능에 대한 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EBS 최적화 인스턴스를 참조하세요.

메모리 최적화 인스턴스 클래스에 대한 하드웨어 사양

다음 표에는 메모리 최적화 인스턴스 클래스에 대한 컴퓨팅, 메모리, 스토리지 및 대역폭 사양이 나와 있습니다.

db.z1d – 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.z1d.12xlarge	48	271	384	2 x 900 NVMe SSD	14,000	25
db.z1d.6xlarge	24	134	192	1 x 900 NVMe SSD	7,000	10
db.z1d.3xlarge	12	75	96	1 x 450 NVMe SSD	3,500	최대 10
db.z1d.2xlarge	8	53	64	1 x 300 NVMe SSD	2,333	최대 10
db.z1d.xlarge*	4	28	32	1 x 150 NVMe SSD	최대 2,333	최대 10
db.z1d.large*	2	15	16	1 x 75 NVMe SSD	최대 2,333	최대 10

db.x2g – AWS Graviton2 프로세서를 지원하는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.x2g.16xlarge	64	—	1024	EBS 최적화 전용	19,000	25

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.x2g.12xlarge	48	—	768	EBS 최적화 전용	14,250	20
db.x2g.8xlarge	32	—	512	EBS 최적화 전용	9,500	12
db.x2g.4xlarge	16	—	256	EBS 최적화 전용	4,750	최대 10
db.x2g.2xlarge	8	—	128	EBS 최적화 전용	최대 4,750	최대 10
db.x2g.xlarge	4	—	64	EBS 최적화 전용	최대 4,750	최대 10
db.x2g.large	2	—	32	EBS 최적화 전용	최대 4,750	최대 10

db.x2idn - 3세대 인텔 제온 스케일러블 프로세서를 지원하는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.x2idn.32xlarge	128	—	2,048	2 x 1900 NVMe SSD	80,000	100
db.x2idn.24xlarge	96	—	1,536	2 x 1425 NVMe SSD	60,000	75
db.x2idn.16xlarge	64	—	1,024	1 x 1900 NVMe SSD	40,000	50

db.x2iedn – 3세대 인텔 제온 스케일러블 프로세서 및 로컬 NVMe 기반 SSD를 지원하는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.x2iedn.32xlarge	128	—	4,096	2 x 1900 NVMe SSD	80,000	100
db.x2iedn.24xlarge	96	—	3,072	2 x 1425 NVMe SSD	60,000	75
db.x2iedn.16xlarge	64	—	2,048	1 x 1900 NVMe SSD	40,000	50
db.x2iedn.8xlarge	32	—	1,024	1 x 950 NVMe SSD	20,000건	25
db.x2iedn.4xlarge	16	—	512	1 x 475 NVMe SSD	최대 20,000개	최대 25개
db.x2iedn.2xlarge	8	—	256	1 x 237 NVMe SSD	최대 20,000개	최대 25개
db.x2iedn.xlarge	4	—	128	1 x 118 NVMe SSD	최대 20,000개	최대 25개

db.x2iezn - 2세대 인텔 제온 스케일러블 프로세서를 지원하는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.x2iezn.12xlarge	>48	—	1,536	EBS 최적화 전용	19,000	100
db.x2iezn.8xlarge	32	—	1,024	EBS 최적화 전용	12,000	75
db.x2iezn.6xlarge	24	—	768	EBS 최적화 전용	최대 9,500	50

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.x2iezn.4xlarge	16	—	512	EBS 최적화 전용	최대 4,750	최대 25개
db.x2iezn.2xlarge	8	—	256	EBS 최적화 전용	최대 3,170개	최대 25개

db.x1e – 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.x1e.32xlarge	128	340	3,904	EBS 최적화 전용	14,000	25
db.x1e.16xlarge	64	179	1,952	EBS 최적화 전용	7,000	10
db.x1e.8xlarge	32	91	976	EBS 최적화 전용	3,500	최대 10
db.x1e.4xlarge	16	47	488	EBS 최적화 전용	1,750	최대 10
db.x1e.2xlarge	8	23	244	EBS 최적화 전용	1,000	최대 10
db.x1e.xlarge	4	12	122	EBS 최적화 전용	500	최대 10

db.x1 – 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.x1.32xlarge	128	349	1,952	EBS 최적화 전용	14,000	25
db.x1.16xlarge	64	174.5	976	EBS 최적화 전용	7,000	10

db.r8g – AWS Graviton4 프로세서를 포함되는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r8g.48xlarge	192	—	1536	EBS 최적화 전용	40,000	50
db.r8g.24xlarge	96	—	768	EBS 최적화 전용	30,000개	40
db.r8g.16xlarge	64	—	512	EBS 최적화 전용	20,000건	30
db.r8g.12xlarge	48	—	384	EBS 최적화 전용	15,000	22.5
db.r8g.8xlarge	32	—	256	EBS 최적화 전용	10,000	15
db.r8g.4xlarge*	16	—	128	EBS 최적화 전용	최대 10,000	최대 15
db.r8g.2xlarge*	8	—	64	EBS 최적화 전용	최대 10,000	최대 15
db.r8g.xlarge*	4	—	32	EBS 최적화 전용	최대 10,000	최대 12.5

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r8g.large*	2	—	16	EBS 최적화 전용	최대 10,000	최대 12.5

db.r7i - 4세대 인텔 제온 스케일러블 프로세서로 구동되는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r7i.48xlarge	192	—	1536	EBS 최적화 전용	40,000	50
db.r7i.24xlarge	96	—	768	EBS 최적화 전용	30,000개	37.5
db.r7i.16xlarge	64	—	512	EBS 최적화 전용	20,000건	25
db.r7i.12xlarge	48	—	384	EBS 최적화 전용	15,000	18.75
db.r7i.8xlarge	32	—	256	EBS 최적화 전용	10,000	12.5
db.r7i.4xlarge	16	—	128	EBS 최적화 전용	최대 10,000	최대 12.5
db.r7i.2xlarge	8	—	64	EBS 최적화 전용	최대 10,000	최대 12.5
db.r7i.xlarge	4	—	32	EBS 최적화 전용	최대 10,000	최대 12.5
db.r7i.large	2	—	16	EBS 최적화 전용	최대 10,000	최대 12.5

db.r7g – AWS Graviton3 프로세서를 포함되는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r7g.16xlarge	64	—	512	EBS 최적화 전용	20,000건	30
db.r7g.12xlarge	48	—	384	EBS 최적화 전용	15,000	22.5
db.r7g.8xlarge	32	—	256	EBS 최적화 전용	10,000	15
db.r7g.4xlarge	16	—	128	EBS 최적화 전용	최대 10,000	최대 15
db.r7g.2xlarge*	8	—	64	EBS 최적화 전용	최대 10,000	최대 15
db.r7g.xlarge*	4	—	32	EBS 최적화 전용	최대 10,000	최대 12.5
db.r7g.large*	2	—	16	EBS 최적화 전용	최대 10,000	최대 12.5

db.r6g – AWS Graviton2 프로세서를 포함되는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r6g.16xlarge	64	—	512	EBS 최적화 전용	19,000	25
db.r6g.12xlarge	48	—	384	EBS 최적화 전용	13,500	20

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r6g.8xlarge	32	—	256	EBS 최적화 전용	9,000	12
db.r6g.4xlarge	16	—	128	EBS 최적화 전용	4,750	최대 10
db.r6g.2xlarge*	8	—	64	EBS 최적화 전용	최대 4,750	최대 10
db.r6g.xlarge*	4	—	32	EBS 최적화 전용	최대 4,750	최대 10
db.r6g.large*	2	—	16	EBS 최적화 전용	최대 4,750	최대 10

db.r6gd – AWS Graviton2 프로세서 및 SSD 스토리지를 포함하는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r6gd.16xlarge	64	—	512	2 x 1900 NVMe SSD	19,000	25
db.r6gd.12xlarge	48	—	384	2 x 1425 NVMe SSD	13,500	20
db.r6gd.8xlarge	32	—	256	1 x 1900 NVMe SSD	9,000	12
db.r6gd.4xlarge	16	—	128	1 x 950 NVMe SSD	4,750	최대 10
db.r6gd.2xlarge	8	—	64	1 x 474 NVMe SSD	최대 4,750	최대 10

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r6gd.xlarge	4	—	32	1 x 237 NVMe SSD	최대 4,750	최대 10
db.r6gd.large	2	—	16	1 x 118 NVMe SSD	최대 4,750	최대 10

db.r6id - 3세대 인텔 제온 스케일러블 프로세서 및 SSD 스토리지를 지원하는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r6id.32xlarge	128	—	1,024	4 x 1900 NVMe SSD	40,000	50
db.r6id.24xlarge	96	—	768	4 x 1425 NVMe SSD	30,000개	37.5
db.r6id.16xlarge	64	—	512	2 x 1900 NVMe SSD	20,000건	25
db.r6id.12xlarge	48	—	384	2 x 1425 NVMe SSD	15,000	18.75
db.r6id.8xlarge	32	—	256	1 x 1900 NVMe SSD	10,000	12.5
db.r6id.4xlarge*	16	—	128	1 x 950 NVMe SSD	최대 10,000	최대 12.5
db.r6id.2xlarge*	8	—	64	1 x 474 NVMe SSD	최대 10,000	최대 12.5
db.r6id.xlarge*	4	—	32	1 x 237 NVMe SSD	최대 10,000	최대 12.5

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r6id.large*	2	—	16	1 x 118 NVMe SSD	최대 10,000	최대 12.5

db.r6idn - 3세대 인텔 제온 스케일러블 프로세서, SSD 스토리지 및 네트워크 최적화를 지원하는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r6idn.32xlarge	128	—	1,024	4 x 1900 NVMe SSD	80,000	200
db.r6idn.24xlarge	96	—	768	4 x 1425 NVMe SSD	60,000	150
db.r6idn.16xlarge	64	—	512	2 x 1900 NVMe SSD	40,000	100
db.r6idn.12xlarge	48	—	384	2 x 1425 NVMe SSD	30,000개	75
db.r6idn.8xlarge	32	—	256	1 x 1900 NVMe SSD	20,000건	50
db.r6idn.4xlarge*	16	—	128	1 x 950 NVMe SSD	최대 20,000개	최대 50개
db.r6idn.2xlarge*	8	—	64	1 x 474 NVMe SSD	최대 20,000개	최대 40개
db.r6idn.xlarge*	4	—	32	1 x 237 NVMe SSD	최대 20,000개	최대 30개
db.r6idn.large*	2	—	16	1 x 118 NVMe SSD	최대 20,000개	최대 25개

db.r6in - 3세대 인텔 제온 스케일러블 프로세서 및 네트워크 최적화를 지원하는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r6in.32xlarge	128	—	1,024	EBS 최적화 전용	80,000	200
db.r6in.24xlarge	96	—	768	EBS 최적화 전용	60,000	150
db.r6in.16xlarge	64	—	512	EBS 최적화 전용	40,000	100
db.r6in.12xlarge	48	—	384	EBS 최적화 전용	30,000개	75
db.r6in.8xlarge	32	—	256	EBS 최적화 전용	20,000건	50
db.r6in.4xlarge*	16	—	128	EBS 최적화 전용	최대 20,000개	최대 50개
db.r6in.2xlarge*	8	—	64	EBS 최적화 전용	최대 20,000개	최대 40개
db.r6in.xlarge*	4	—	32	EBS 최적화 전용	최대 20,000개	최대 30개
db.r6in.large*	2	—	16	EBS 최적화 전용	최대 20,000개	최대 25개

db.r6i – 대용량 메모리, 스토리지 및 I/O용으로 사전 구성된 Oracle 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r6i.8xlarge.tpc2.mem4x	32	—	1024	EBS 최적화 전용	40,000	50
db.r6i.8xlarge.tpc2.mem3x	32	—	768	EBS 최적화 전용	30,000개	37.5
db.r6i.6xlarge.tpc2.mem4x	24	—	768	EBS 최적화 전용	30,000개	37.5
db.r6i.4xlarge.tpc2.mem4x	16	—	512	EBS 최적화 전용	20,000건	25
db.r6i.4xlarge.tpc2.mem3x	16	—	384	EBS 최적화 전용	15,000	18.75
db.r6i.4xlarge.tpc2.mem2x	16	—	256	EBS 최적화 전용	10,000	12.5
db.r6i.2xlarge.tpc2.mem8x	8	—	512	EBS 최적화 전용	20,000건	12.5
db.r6i.2xlarge.tpc2.mem4x	8	—	256	EBS 최적화 전용	10,000	12.5
db.r6i.2xlarge.tpc1.mem2x	8	—	128	EBS 최적화 전용	최대 10,000	12.5
db.r6i.xlarge.tpc2.mem4x	4	—	128	EBS 최적화 전용	최대 10,000	12.5
db.r6i.xlarge.tpc2.mem2x	4	—	64	EBS 최적화 전용	최대 10,000	12.5
db.r6i.large.tpc1.mem2x	2	—	32	EBS 최적화 전용	최대 10,000	12.5

db.r6i - 3세대 인텔 제온 스케일러블 프로세서를 포함하는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r6i.32xlarge	128	—	1,024	EBS 최적화 전용	40,000	50
db.r6i.24xlarge	96	—	768	EBS 최적화 전용	30,000개	37.5
db.r6i.16xlarge	64	—	512	EBS 최적화 전용	20,000건	25
db.r6i.12xlarge	48	—	384	EBS 최적화 전용	15,000	18.75
db.r6i.8xlarge	32	—	256	EBS 최적화 전용	10,000	12.5
db.r6i.4xlarge*	16	—	128	EBS 최적화 전용	최대 10,000	최대 12.5
db.r6i.2xlarge*	8	—	64	EBS 최적화 전용	최대 10,000	최대 12.5
db.r6i.xlarge*	4	—	32	EBS 최적화 전용	최대 10,000	최대 12.5
db.r6i.large*	2	—	16	EBS 최적화 전용	최대 10,000	최대 12.5

db.r5d - 인텔 제온 플래티넘 프로세서 및 SSD 스토리지를 포함하는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r5d.24xlarge	96	347	768	4 x 900 NVMe SSD	19,000	25
db.r5d.16xlarge	64	264	512	4 x 600 NVMe SSD	13,600	20
db.r5d.12xlarge	48	173	384	2 x 900 NVMe SSD	9,500	10
db.r5d.8xlarge	32	132	256	2 x 600 NVMe SSD	6,800	10
db.r5d.4xlarge	16	71	128	2 x 300 NVMe SSD	4,750	최대 10
db.r5d.2xlarge*	8	38	64	1 x 300 NVMe SSD	최대 4,750	최대 10
db.r5d.xlarge*	4	19	32	1 x 150 NVMe SSD	최대 4,750	최대 10
db.r5d.large*	2	10	16	1 x 75 NVMe SSD	최대 4,750	최대 10

db.r5b – 대용량 메모리, 스토리지 및 I/O용으로 사전 구성된 Oracle 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r5b.8xlarge.tpc2.mem3x	32	—	768	EBS 최적화 전용	60,000	25
db.r5b.6xlarge.tpc2.mem4x	24	—	768	EBS 최적화 전용	60,000	25

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r5b.4xlarge.tpc2.mem4x	16	—	512	EBS 최적화 전용	40,000	20
db.r5b.4xlarge.tpc2.mem3x	16	—	384	EBS 최적화 전용	30,000개	10
db.r5b.4xlarge.tpc2.mem2x	16	—	256	EBS 최적화 전용	20,000건	10
db.r5b.2xlarge.tpc2.mem8x	8	—	512	EBS 최적화 전용	40,000	20
db.r5b.2xlarge.tpc2.mem4x	8	—	256	EBS 최적화 전용	20,000건	10
db.r5b.2xlarge.tpc1.mem2x	8	—	128	EBS 최적화 전용	10,000	최대 10
db.r5b.xlarge.tpc2.mem4x	4	—	128	EBS 최적화 전용	10,000	최대 10
db.r5b.xlarge.tpc2.mem2x	4	—	64	EBS 최적화 전용	최대 10,000	최대 10
db.r5b.large.tpc1.mem2x	2	—	32	EBS 최적화 전용	최대 10,000	최대 10

db.r5b - 인텔 제온 플래티넘 프로세서 및 EBS 최적화를 포함하는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r5b.24xlarge	96	347	768	EBS 최적화 전용	60,000	25

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r5b.16xlarge	64	264	512	EBS 최적화 전용	40,000	20
db.r5b.12xlarge	48	173	384	EBS 최적화 전용	30,000개	10
db.r5b.8xlarge	32	132	256	EBS 최적화 전용	20,000건	10
db.r5b.4xlarge	16	71	128	EBS 최적화 전용	10,000	최대 10
db.r5b.2xlarge*	8	38	64	EBS 최적화 전용	최대 10,000	최대 10
db.r5b.xlarge*	4	19	32	EBS 최적화 전용	최대 10,000	최대 10
db.r5b.large*	2	10	16	EBS 최적화 전용	최대 10,000	최대 10

db.r5 – 대용량 메모리, 스토리지 및 I/O용으로 사전 구성된 Oracle 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r5.12xlarge.tpc2.mem2x	48	—	768	EBS 최적화 전용	19,000	25
db.r5.8xlarge.tpc2.mem3x	32	—	768	EBS 최적화 전용	19,000	25
db.r5.6xlarge.tpc2.mem4x	24	—	768	EBS 최적화 전용	19,000	25

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r5.4xlarge.tpc2.mem4x	16	—	512	EBS 최적화 전용	13,600	20
db.r5.4xlarge.tpc2.mem3x	16	—	384	EBS 최적화 전용	9,500	10
db.r5.4xlarge.tpc2.mem2x	16	—	256	EBS 최적화 전용	6,800	10
db.r5.2xlarge.tpc2.mem8x	8	—	512	EBS 최적화 전용	13,600	20
db.r5.2xlarge.tpc2.mem4x	8	—	256	EBS 최적화 전용	6,800	10
db.r5.2xlarge.tpc1.mem2x	8	—	128	EBS 최적화 전용	4,750	최대 10
db.r5.xlarge.tpc2.mem4x	4	—	128	EBS 최적화 전용	4,750	최대 10
db.r5.xlarge.tpc2.mem2x	4	—	64	EBS 최적화 전용	최대 4,750	최대 10
db.r5.large.tpc1.mem2x	2	—	32	EBS 최적화 전용	최대 4,750	최대 10

db.r5 – 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r5.24xlarge	96	347	768	EBS 최적화 전용	19,000	25

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r5.16xlarge	64	264	512	EBS 최적화 전용	13,600	20
db.r5.12xlarge	48	173	384	EBS 최적화 전용	9,500	12
db.r5.8xlarge	32	132	256	EBS 최적화 전용	6,800	10
db.r5.4xlarge	16	71	128	EBS 최적화 전용	4,750	최대 10
db.r5.2xlarge*	8	38	64	EBS 최적화 전용	최대 4,750	최대 10
db.r5.xlarge*	4	19	32	EBS 최적화 전용	최대 4,750	최대 10
db.r5.large*	2	10	16	EBS 최적화 전용	최대 4,750	최대 10

db.r4 - 인텔 제온 스케일러블 프로세서를 포함하는 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r4.16xlarge	64	195	488	EBS 최적화 전용	14,000	25
db.r4.8xlarge	32	99	244	EBS 최적화 전용	7,000	10
db.r4.4xlarge	16	53	122	EBS 최적화 전용	3,500	최대 10

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r4.2xlarge	8	27	61	EBS 최적화 전용	1,700	최대 10
db.r4.xlarge	4	13.5	30.5	EBS 최적화 전용	850	최대 10
db.r4.large	2	7	15.25	EBS 최적화 전용	425	최대 10

db.r3 – 메모리 최적화 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.r3.8xlarge**	32	104	244	EBS 전용	—	10
db.r3.4xlarge	16	52	122	EBS 최적화 전용	2,000	높음
db.r3.2xlarge	8	26	61	EBS 최적화 전용	1,000	높음
db.r3.xlarge	4	13	30.5	EBS 최적화 전용	500	보통
db.r3.large	2	6.5	15.25	EBS 최적화 전용	—	보통

* 이러한 DB 인스턴스 클래스에서는 24시간마다 최소 한 번씩 30분간 최대 성능을 지원합니다. 기본 EC2 인스턴스 유형의 기본 성능에 대한 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EBS 최적화 인스턴스를 참조하세요.

** r3.8xlarge DB 인스턴스 클래스는 전용 EBS 대역폭이 없으므로 EBS 최적화를 제공하지 않습니다. 이 인스턴스 클래스에서 네트워크 트래픽과 Amazon EBS 트래픽은 동일한 10기가비트 네트워크 인터페이스를 공유합니다.

컴퓨팅 최적화 인스턴스 클래스에 대한 하드웨어 사양

다음 표에는 컴퓨팅 최적화 인스턴스 클래스에 대한 컴퓨팅, 메모리, 스토리지 및 대역폭 사양이 나와 있습니다.

db.c6gd - 컴퓨팅 최적화 인스턴스 클래스(다중 AZ DB 클러스터 배포만 해당)

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.c6gd.16xlarge	64	—	128	2 x 1900 NVMe SSD	19,000	25
db.c6gd.12xlarge	48	—	96	2 x 1425 NVMe SSD	13,500	20
db.c6gd.8xlarge	32	—	64	1 x 1900 NVMe SSD	9,000	12
db.c6gd.4xlarge	16	—	32	1 x 950 NVMe SSD	4,750	최대 10
db.c6gd.2xlarge	8	—	16	1 x 474 NVMe SSD	최대 4,750	최대 10
db.c6gd.xlarge	4	—	8	1 x 237 NVMe SSD	최대 4,750	최대 10
db.c6gd.large	2	—	4	1 x 118 NVMe SSD	최대 4,750	최대 10
db.c6gd.medium	1	—	2	1 x 59 NVMe SSD	최대 4,750	최대 10

성능 버스트 가능 인스턴스 클래스에 대한 하드웨어 사양

다음 표에는 성능 버스트 가능 인스턴스 클래스에 대한 컴퓨팅, 메모리, 스토리지 및 대역폭 사양이 나와 있습니다.

db.t4g - AWS Graviton2 프로세서로 구동되는 버스트 가능 성능 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.t4g.2xlarge*	8	—	32	EBS 최적화 전용	최대 2,780	최대 5
db.t4g.xlarge*	4	—	16	EBS 최적화 전용	최대 2,780	최대 5
db.t4g.large*	2	—	8	EBS 최적화 전용	최대 2,780	최대 5
db.t4g.medium*	2	—	4	EBS 최적화 전용	최대 2,085개	최대 5
db.t4g.small*	2	—	2	EBS 최적화 전용	최대 2,085개	최대 5
db.t4g.micro*	2	—	1	EBS 최적화 전용	최대 2,085개	최대 5

db.t3 – 버스트 가능 성능 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리 (GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.t3.2xlarge*	8	변수	32	EBS 최적화 전용	최대 2,048개	최대 5
db.t3.xlarge*	4	변수	16	EBS 최적화 전용	최대 2,048개	최대 5
db.t3.large*	2	변수	8	EBS 최적화 전용	최대 2,048개	최대 5
db.t3.medium*	2	변수	4	EBS 최적화 전용	최대 1,536개	최대 5

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.t3.small*	2	변수	2	EBS 최적화 전용	최대 1,536개	최대 5
db.t3.micro*	2	변수	1	EBS 최적화 전용	최대 1,536개	최대 5

db.t2 – 버스트 가능 성능 인스턴스 클래스

인스턴스 클래스	vCPU	ECU	메모리(GiB)	인스턴스 스토리지(GiB)	최대 EBS 대역폭(Mbps)	네트워크 대역폭(Gbps)
db.t2.2xlarge	8	변수	32	EBS 전용	—	보통
db.t2.xlarge	4	변수	16	EBS 전용	—	보통
db.t2.large	2	변수	8	EBS 전용	—	보통
db.t2.medium	2	변수	4	EBS 전용	—	보통
db.t2.small	1	변수	2	EBS 전용	—	낮음
db.t2.micro	1	변수	1	EBS 전용	—	낮음

* 이러한 DB 인스턴스 클래스에서는 24시간마다 최소 한 번씩 30분간 최대 성능을 지원합니다. 기본 EC2 인스턴스 유형의 기본 성능에 대한 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EBS 최적화 인스턴스를 참조하세요.

Amazon RDS DB 인스턴스 스토리지

Amazon RDS for Db2, MariaDB, MySQL, PostgreSQL, Oracle, Microsoft SQL Server의 DB 인스턴스는 데이터베이스 및 로그 스토리지에 Amazon Elastic Block Store(Amazon EBS) 볼륨을 사용합니다.

경우에 따라서는 데이터베이스 워크로드가 프로비저닝한 IOPS의 100%를 달성할 수 없을 수 있습니다. 자세한 내용은 스토리지 성능에 영향을 끼치는 요인 단원을 참조하십시오.

인스턴스 스토리지 요금에 대한 자세한 내용은 Amazon RDS 요금을 참조하세요.

Amazon RDS 스토리지 유형

Amazon RDS에서는 프로비저닝된 IOPS SSD(io1 및 io2 Block Express라고도 함), 범용 SSD(gp2 및 gp3라고도 함), 마그네틱(표준이라고도 함) 등 세 가지 스토리지 유형을 제공합니다. 이러한 3가지 유형은 성능 특성과 가격이 다르므로 데이터베이스 워크로드 요건에 따라 스토리지 성능과 비용을 조정할 수 있습니다. 최대 64테라바이트(TiB) 스토리지의 Db2, MySQL, MariaDB, Oracle, SQL Server 및 PostgreSQL RDS DB 인스턴스를 만들 수 있습니다. RDS for Db2는 gp2 및 마그네틱 스토리지 유형을 지원하지 않습니다.

다음은 세 가지 스토리지 유형에 대한 간략한 설명입니다.

- 프로비저닝된 IOPS SSD – 프로비저닝된 IOPS 스토리지는 I/O 지연 시간이 짧고 I/O 처리량이 일정한 I/O 집약적 워크로드, 특히 데이터베이스 워크로드 요구 사항을 충족하도록 설계되었습니다. 프로비저닝된 IOPS 스토리지는 프로덕션 환경에 가장 적합합니다.

  스토리지 크기 범위를 포함하여 프로비저닝된 IOPS 스토리지에 대한 자세한 내용은 프로비저닝된 IOPS SSD 스토리지 단원을 참조하십시오.
- 범용 SSD – 범용 SSD 볼륨은 중간 크기 DB 인스턴스에서 실행하는 광범위한 워크로드에 이상적인 비용 효율적 스토리지를 제공합니다. 범용 스토리지는 개발 및 테스트 환경에 가장 적합합니다.

  스토리지 크기 범위를 포함하여 범용 SSD 스토리지에 대한 자세한 내용은 범용 SSD 스토리지 단원을 참조하십시오.
- 마그네틱 – Amazon RDS는 역호환성을 위해 마그네틱 스토리지도 지원합니다. 새 스토리지가 필요할 경우 범용 SSD 또는 프로비저닝된 IOPS SSD를 사용하는 것이 좋습니다. 마그네틱 스토리지에서는 DB 인스턴스에 허용되는 최대 스토리지 크기가 3TiB입니다. 자세한 내용은 마그네틱 스토리지(레거시, 권장하지 않음) 단원을 참조하십시오.

범용 SSD 또는 프로비저닝된 IOPS SSD를 선택하면 선택한 엔진과 요청된 스토리지의 양에 따라 Amazon RDS가 다음 표와 같이 여러 볼륨에 걸쳐 자동으로 스트라이핑하여 성능을 향상시킵니다.

데이터베이스 엔진	Amazon RDS 스토리지 크기	프로비저닝된 볼륨 수
Db2	400GiB 미만	1
Db2	400~65,536GiB	4
MariaDB, MySQL 및 PostgreSQL	400GiB 미만	1
MariaDB, MySQL 및 PostgreSQL	400~65,536GiB	4
Oracle	200GiB 미만	1
Oracle	200~65,536GiB	4
SQL Server	모두	1

범용 SSD 또는 프로비저닝된 IOPS SSD 볼륨을 수정하면 이는 상태 시퀀스를 거칩니다. 볼륨이 `optimizing` 상태에 있는 동안 볼륨 성능은 소스와 대상의 구성 사양 사이입니다. 일시적인 볼륨 성능은 두 사양 중 더 낮은 사양 이상입니다.

> **Important**
>
> 하나의 볼륨에서 네 개의 볼륨으로 전환되도록 인스턴스의 스토리지를 수정하거나 마그네틱 스토리지를 사용하여 인스턴스를 수정하면 Amazon RDS는 탄력적 볼륨 기능을 사용하지 않습니다. 대신 Amazon RDS는 새 볼륨을 프로비저닝하고 데이터를 이전 볼륨에서 새 볼륨으로 투명하게 이동합니다. 이 작업은 이전 볼륨과 새 볼륨 모두에서 상당한 양의 IOPS와 처리량을 소비합니다. 볼륨의 크기와 수정 중에 존재하는 데이터베이스 워크로드 양에 따라 이 작업은 많은 양의 IOPS를 소비하고, I/O 지연 시간을 크게 늘리며, 완료하는 데 몇 시간이 걸릴 수 있으며 RDS 인스턴스는 `Modifying` 상태를 유지합니다.

프로비저닝된 IOPS SSD 스토리지

빠르고 일관된 I/O 성능이 필요한 프로덕션 애플리케이션의 경우에는 프로비저닝된 IOPS 스토리지를 사용하는 것이 좋습니다. 프로비저닝된 IOPS 스토리지는 성능이 예측 가능하며, 일관적으로 지연 시간

이 짧은 스토리지 유형입니다. 프로비저닝된 IOPS 스토리지는 일관적인 성능이 필요한 온라인 트랜잭션 프로세싱(OLTP) 워크로드에 이상적입니다. 프로비저닝된 IOPS는 이런 워크로드의 성능 튜닝에도 효과적입니다.

DB 인스턴스를 생성할 때는 IOPS 속도와 볼륨 크기를 지정합니다. Amazon RDS는 사용자가 변경할 때까지 DB 인스턴스에 대해 해당 IOPS 속도를 제공합니다.

Amazon RDS는 io2 Block Express 스토리지(권장) 및 io1 스토리지(이전 세대)라는 2가지 유형의 프로비저닝된 IOPS SSD 스토리지를 제공합니다.

io2 Block Express 스토리지(권장)

I/O 집약적이고 지연 시간에 민감한 워크로드의 경우 프로비저닝된 IOPS SSD io2 Block Express 스토리지를 사용하여 초당 최대 25만 6,000개의 I/O 작업 수(IOPS)를 달성할 수 있습니다. io2 Block Express 볼륨의 처리량은 볼륨당 프로비저닝된 IOPS의 양과 실행 중인 I/O 작업의 크기에 따라 달라집니다.

AWS Nitro System 기반의 모든 RDS io2 볼륨은 io2 Block Express 볼륨이며 평균 지연 시간은 1밀리초 미만입니다. AWS Nitro System을 기반으로 하지 않는 DB 인스턴스는 io2 볼륨입니다.

다음 표는 각 데이터베이스 엔진 및 스토리지 크기 범위에 따른 프로비저닝된 IOPS 범위 및 최대 처리량을 나타냅니다.

데이터베이스 엔진	스토리지 크기 범위	프로비저닝된 IOPS 범위	최대 처리량
Db2, MariaDB, MySQL 및 PostgreSQL	100~65,536GiB	1,000–256,000 IOPS	16,000MiB/s
Oracle	100~199GiB	1,000~199,000IOPS	16,000MiB/s
Oracle	200~65,536GiB	1,000–256,000 IOPS	16,000MiB/s
SQL Server	20~65,536GiB	1,000–256,000 IOPS	16,000MiB/s

IOPS 및 스토리지 크기 범위에는 다음과 같은 제약이 있습니다.

- 할당된 스토리지에 대한 IOPS 비율(GiB)은 1000:1을 넘지 않아야 합니다. AWS Nitro System을 기반으로 하지 않는 DB 인스턴스의 경우 비율은 500:1입니다.

- 최대 IOPS는 256GiB 이상의 볼륨으로 프로비저닝될 수 있습니다(1,000 IOPS x 256GiB = 256,000 IOPS). AWS Nitro System을 기반으로 하지 않는 DB 인스턴스의 경우 최대 IOPS는 512GiB(500IOPS x 512GiB = 256,000IOPS)에서 달성됩니다.
- 처리량은 프로비저닝된 IOPS당 최대 0.256Mib/s까지 비례하여 확장됩니다. 16KiB I/O 크기의 경우 256,000IOPS, 256KiB 입출력 크기의 경우 16,000IOPS 이상에서 최대 4,000MiB/s의 처리량을 달성할 수 있습니다. AWS Nitro System을 기반으로 하지 않는 DB 인스턴스의 경우 16KiB I/O 크기의 경우 128,000IOPS에서 최대 2,000MiB/s의 처리량을 달성할 수 있습니다.
- 스토리지 자동 크기 조정을 사용하는 경우 IOPS와 최대 스토리지 임계값(GiB) 간에 동일한 비율이 적용됩니다. 스토리지 자동 크기 조정에 대한 자세한 내용은 Amazon RDS 스토리지 Autoscaling을 사용한 용량 자동 관리 섹션을 참조하세요.

Amazon RDS io2 Block Express 볼륨은 다음 AWS 리전에서 제공됩니다.

- 아시아 태평양(홍콩)
- 아시아 태평양(뭄바이)
- 아시아 태평양(서울)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(도쿄)
- 캐나다(중부)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- Europe (London)
- 유럽(스톡홀름)
- 중동(바레인)
- 미국 동부(오하이오)
- 미국 동부(버지니아 북부)
- 미국 서부(캘리포니아 북부)
- 미국 서부(오레곤)

io1 스토리지(이전 세대)

I/O 집약적 워크로드의 경우 프로비저닝된 IOPS SSD io1 스토리지를 사용하여 초당 최대 256,000개의 I/O 작업 수(IOPS)를 달성할 수 있습니다. io1 볼륨의 처리량은 볼륨당 프로비저닝된 IOPS의 양과 실행 중인 I/O 작업의 크기에 따라 달라집니다. 가능한 경우 io2 Block Express 스토리지를 사용하는 것이 좋습니다.

다음 표는 각 데이터베이스 엔진 및 스토리지 크기 범위에 따른 프로비저닝된 IOPS 범위 및 최대 처리량을 나타냅니다.

데이터베이스 엔진	스토리지 크기 범위	프로비저닝된 IOPS 범위	최대 처리량
Db2, MariaDB, MySQL 및 PostgreSQL	100~399GiB	1,000~19,950IOPS	500MiB/s
Db2, MariaDB, MySQL 및 PostgreSQL	400~65,536GiB	1,000–256,000 IOPS	4,000MiB/s
Oracle	100~199GiB	1,000~9,950IOPS	500MiB/s
Oracle	200~65,536GiB	1,000~256,000IOPS[1]	4,000MiB/s
SQL Server	20~16,384GiB	1,000~64,000IOPS[2]	1,000MiB/s

> **Note**
>
> [1] Oracle의 경우 r5b 인스턴스 유형에서만 최대 256,000 IOPS를 프로비저닝할 수 있습니다.
> [2] SQL Server의 경우 m5*, m6i, r5*, r6i, z1d 인스턴스 유형에 있는 Nitro 기반 인스턴스에서만 최대 64,000 IOPS가 보장됩니다. 다른 인스턴스 유형은 최대 32,000 IOPS의 성능을 보장합니다.

IOPS 및 스토리지 크기 범위에는 다음과 같은 제약이 있습니다.

- 할당된 스토리지에 대한 IOPS 비율(GiB)은 RDS for SQL Server에서 1~50, 기타 RDS DB 엔진에서 0.5~50이어야 합니다.

- 스토리지 자동 크기 조정을 사용하는 경우 IOPS와 최대 스토리지 임계값(GiB) 간에 동일한 비율이 적용됩니다.

  스토리지 자동 크기 조정에 대한 자세한 내용은 Amazon RDS 스토리지 Autoscaling을 사용한 용량 자동 관리 섹션을 참조하세요.

프로비저닝된 IOPS 스토리지를 다중 AZ 배포 또는 읽기 전용 복제본과 결합합니다.

프로덕션 OLTP 사용 사례에서 내결함성을 높이는 것이 목적일 때는 다중 AZ 배포를, 빠르고 예측 가능한 성능이 목적일 때는 프로비저닝된 IOPS 스토리지를 사용하는 것이 바람직합니다.

또한 MySQL, MariaDB 또는 PostgreSQL에 대한 읽기 전용 복제본과 함께 프로비저닝된 IOPS 스토리지를 사용할 수 있습니다. 읽기 전용 복제본의 스토리지 유형은 기본 DB 인스턴스의 스토리지 유형과 독립되어 있습니다. 예를 들어, 프로비저닝된 IOPS SSD 스토리지를 사용하는 기본 DB 인스턴스에 읽기 전용 복제본의 범용 SSD를 사용하여 비용을 절감할 수 있습니다. 그러나 이 경우의 읽기 전용 복제본의 성능은 기본 DB 인스턴스와 읽기 전용 복제본 모두 프로비저닝된 IOPS 스토리지를 사용하는 구성의 성능과 차이를 보일 수 있습니다.

프로비저닝된 IOPS 스토리지 비용

프로비저닝된 IOPS 스토리지는 한 달간 사용 여부에 상관없이 프로비저닝된 리소스에 대한 요금이 청구됩니다.

요금에 대한 자세한 내용은 Amazon RDS 요금을 참조하십시오.

Amazon RDS 프로비저닝된 IOPS 스토리지에서 최상의 성능 얻기

워크로드에 I/O 제약이 있으면 프로비저닝된 IOPS 스토리지의 사용에 따라 시스템이 동시에 처리할 수 있는 I/O 요청 수가 증가합니다. 이처럼 동시 처리 가능한 수가 증가하면 대기열의 I/O 요청 시간이 줄어들어 지연 시간이 감소하는 효과가 있습니다. 지연 시간 감소는 데이터베이스 커밋 속도의 향상으로 이어져 응답 시간이 개선되고 데이터베이스 처리 속도가 빨라집니다.

프로비저닝된 IOPS 스토리지는 IOPS를 지정하여 I/O 용량을 예약할 수 있습니다. 하지만, 다른 시스템 용량 속성과 마찬가지로 재하 시 최대 처리 속도는 처음 사용된 리소스에 따라 제약을 받게 됩니다. 이러한 리소스로는 네트워크 대역폭, CPU 메모리 또는 데이터베이스 내부 리소스가 있습니다.

범용 SSD 스토리지

범용 스토리지는 지연 시간에 민감하지 않은 대부분의 데이터베이스 워크로드에 적합한, 비용 효율적인 스토리지를 제공합니다.

> **Note**
>
> 범용 스토리지를 사용하는 DB 인스턴스는 프로비저닝된 IOPS 스토리지를 사용하는 인스턴스보다 지연 시간이 훨씬 길 수 있습니다. 이러한 작업 후 지연 시간이 최소인 DB 인스턴스가 필요하다면 프로비저닝된 IOPS SSD 스토리지를 사용하는 것이 좋습니다.

Amazon RDS는 gp3 스토리지(권장) 및 gp2 스토리지(이전 세대)라는 두 가지 유형의 범용 스토리지를 제공합니다.

gp3 스토리지(권장)

범용 gp3 스토리지 볼륨을 사용하면 스토리지 용량과 관계없이 스토리지 성능을 사용자 지정할 수 있습니다. 스토리지 성능은 초당 I/O 작업(IOPS)과 스토리지 볼륨에서 읽기 및 쓰기 작업을 수행하는 속도(스토리지 처리량)의 조합입니다. gp3 스토리지 볼륨에서 Amazon RDS는 3,000IOPS 및 125MiB/s의 기본 스토리지 성능을 제공합니다.

RDS for SQL Server를 제외한 모든 RDS DB 엔진에서, gp3 볼륨의 스토리지 크기가 특정 임계값에 도달하면 기본 스토리지 성능이 증가합니다. 스토리지가 하나가 아닌 네 개의 볼륨을 사용하는 볼륨 스트라이핑 덕분입니다. RDS for SQL Server는 볼륨 스트라이핑을 지원하지 않으며, 따라서 임계값이 없습니다. 스트라이프 볼륨에서 Amazon RDS는 12,000IOPS 및 500MiB/s의 기본 스토리지 성능을 제공합니다.

다음 표에는 임계값을 비롯한 Amazon RDS DB 엔진의 gp3 볼륨 스토리지 성능이 나와 있습니다.

DB 엔진	스토리지 크기	기본 스토리지 성능	프로비저닝된 IOPS 범위	프로비저닝된 스토리지 처리량 범위
Db2, MariaDB, MySQL 및 PostgreSQL	20~399GiB	3,000IOPS /125MiB/s	N/A	N/A
Db2, MariaDB, MySQL 및 PostgreSQL	400~65,536GiB	12,000IOPS/500MiB/s	12,000–64,000 IOPS	500~4,000MiB/s
Oracle	20~199GiB	3,000IOPS /125MiB/s	N/A	N/A

DB 엔진	스토리지 크기	기본 스토리지 성능	프로비저닝된 IOPS 범위	프로비저닝된 스토리지 처리량 범위
Oracle	200~65,536GiB	12,000IOPS/500MiB/s	12,000–64,000 IOPS	500~4,000MiB/s
SQL Server	20~16,384GiB	3,000IOPS/125MiB/s	3,000–16,000 IOPS	125~1,000MiB/s

RDS for SQL Server를 제외한 모든 DB 엔진에서는, 스토리지 크기가 임계값 이상일 때 추가 IOPS 및 스토리지 처리량을 프로비저닝할 수 있습니다. RDS for SQL Server의 경우 사용 가능한 스토리지 크기에 맞게 추가 IOPS 및 스토리지 처리량을 프로비저닝할 수 있습니다. 모든 DB 엔진은 추가로 프로비저닝된 스토리지 성능에 대해서만 비용을 지불하면 됩니다. 자세한 내용은 Amazon RDS 요금을 참조하십시오.

추가된 프로비저닝된 IOPS와 스토리지 처리량은 스토리지 크기와 무관하지만 서로 관련이 있습니다. MariaDB 및 MySQL의 IOPS를 32,000 이상으로 올리면 스토리지 처리량 값이 500MiBPS에서 자동으로 증가합니다. 예를 들어 RDS for MySQL에서 IOPS를 40,000으로 설정하면 스토리지 처리량은 625MiBPS 이상이어야 합니다. Db2, Oracle, PostgreSQL 및 SQL Server DB 인스턴스에서는 자동 증가가 진행되지 않습니다.

다중 AZ DB 클러스터의 경우 Amazon RDS는 프로비저닝한 IOPS를 기반으로 처리량 값을 자동으로 설정합니다. 처리량 값은 수정할 수 없습니다.

RDS의 gp3 볼륨에 대한 스토리지 성능 값에는 다음과 같은 제약이 적용됩니다.

- 지원되는 모든 DB 엔진의 스토리지 처리량 대 IOPS 최대 비율은 0.25입니다.
- RDS for SQL Server에서 IOPS 대 할당된 스토리지(GiB)의 최소 비율은 0.5입니다. 지원되는 다른 DB 엔진에는 최소 비율이 없습니다.
- 지원되는 모든 DB 엔진의 IOPS 대 스토리지 처리량 최대 비율은 500입니다.
- 스토리지 자동 크기 조정을 사용하는 경우 IOPS와 최대 스토리지 임계값(GiB) 간에 동일한 비율이 적용됩니다.

  스토리지 자동 크기 조정에 대한 자세한 내용은 Amazon RDS 스토리지 Autoscaling을 사용한 용량 자동 관리 섹션을 참조하세요.

gp2 스토리지(이전 세대)

애플리케이션에 뛰어난 스토리지 성능이 필요없다면 범용 SSD gp2 스토리지를 사용하는 것이 좋습니다. gp2 스토리지의 기본 I/O 성능은 1GiB당 3IOPS이며, 최소 100IOPS입니다. 이 관계는 볼륨이 클수록 성능이 높아진다는 의미입니다. 예를 들어, 100GiB 볼륨 하나에 대한 기준 성능은 300IOPS입니다. 1,000GiB 볼륨 하나에 대한 기준 성능은 3,000IOPS입니다.

1,000GiB보다 작은 크기의 개별 gp2 볼륨은 늘어난 시간에 대해 3,000IOPS로 확장될 수도 있습니다. 버스트 성능은 볼륨 I/O 크레딧 밸런스에 의해 결정됩니다. 기준 성능과 I/O 크레딧 밸런스가 성능에 미치는 영향에 대한 자세한 내용은 AWS 데이터베이스 블로그의 Understanding burst vs. baseline performance with Amazon RDS and gp2(Amazon RDS 및 gp2를 이용할 때의 버스트와 기존 성능 차이 이해하기)를 참조하세요.

대부분의 워크로드에서는 버스트 밸런스가 고갈되지 않습니다. 하지만, 일부 워크로드가 3,000IOPS 버스트 스토리지 크레딧 밸런스를 소비할 수 있으므로 워크로드의 요건에 맞게 스토리지 용량을 계획해야 합니다.

4,000GiB보다 큰 gp2 볼륨의 경우 기준 성능이 버스트 성능보다 좋습니다. 이러한 볼륨의 경우 기준 성능이 3,000 IOPS 버스트 성능보다 우수하기 때문에 버스트가 무관합니다. 그러나 특정 엔진 및 크기의 DB 인스턴스는 스토리지가 네 개의 볼륨에 걸쳐 스트라이핑되어 기준 처리량의 4배, 단일 볼륨 버스트 IOPS의 4배를 제공합니다.

다음 표에는 Amazon RDS DB 엔진에서 다양한 스토리지 크기의 gp2 볼륨 스토리지 성능이 나와 있습니다.

DB 엔진	RDS 스토리지 크기	기준 IOPS 범위	기준 처리량 범위	버스트 IOPS
MariaDB, MySQL 및 PostgreSQL	5~399GiB[1]	100~1,197IOPS	128~250MiB/s	3,000
MariaDB, MySQL 및 PostgreSQL	400~1,335GiB	1,200~4,005IOPS	512~1,000MiB/s	12,000
MariaDB, MySQL 및 PostgreSQL	1,336~3,999GiB	4,008~11,997IOPS	1,000MiB/s	12,000

DB 엔진	RDS 스토리지 크기	기준 IOPS 범위	기준 처리량 범위	버스트 IOPS
MariaDB, MySQL 및 PostgreSQL	4,000~65,536GiB	12,000~64,000IOPS	1,000MiB/s	N/A[2]
Oracle	20~199GiB	100~597IOPS	128~250MiB/s	3,000
Oracle	200~1,335GiB	600~4,005IOPS	500~1,000MiB/s	12,000
Oracle	1,336~3,999GiB	4,008~11,997IOPS	1,000MiB/s	12,000
Oracle	4,000~65,536GiB	12,000~64,000IOPS	1,000MiB/s	N/A[2]
SQL Server	20~333GiB	100~999IOPS	128~250MiB/s	3,000
SQL Server	334~999GiB	1,002~2,997IOPS	250MiB/s	3,000
SQL Server	1,000~16,384GiB	3,000–16,000 IOPS	250MiB/s	N/A[2]

> **Note**
>
> [1] AWS Management Console을 사용하여 프리 티어에서 db.t3.micro 및 db.t4g.micro DB 인스턴스 클래스에 대해 최소 스토리지 크기가 5GiB인 DB 인스턴스를 만들 수 있습니다. 그렇지 않으면 최소 스토리지 크기는 20GiB입니다. 이 제한은 AWS CLI 및 RDS API에는 적용되지 않습니다.
> [2] 볼륨의 기준 성능이 최대 버스트 성능을 초과합니다.

솔리드 스테이트 드라이브(SSD) 스토리지 유형 비교

다음 표는 Amazon RDS에서 사용하는 SSD 스토리지 볼륨의 사용 사례 및 성능 특성을 보여줍니다.

기능	프로비저닝된 IOPS(io2 Block Express)	프로비저닝된 IOPS(io1)	범용(gp3)	범용(gp2)
설명	RDS 스토리지 포트폴리오 내 최고 성능(IOPS, 처리량, 지연 시간) 지연 시간에 민감한 트랜잭션 워크로드용으로 설계됨	일관된 스토리지 성능 (IOPS, 처리량, 지연 시간) 지연 시간에 민감한 트랜잭션 워크로드용으로 설계됨	스토리지, IOPS 및 처리량을 독립적으로 프로비저닝할 수 있는 유연성 다양한 트랜잭션 워크로드를 위한 가성비	버스트 가능한 IOPS 제공 다양한 트랜잭션 워크로드를 위한 가성비
사용 사례	1밀리초 미만의 지연 시간과 최대 256,000IOPS의 지속적인 IOPS 성능을 요구하는 비즈니스 크리티컬 트랜잭션 워크로드	최대 256,000 IOPS의 지속적인 IOPS 성능을 요구하는 트랜잭션 워크로드	개발/테스트 환경의 중간 규모 관계형 데이터베이스에서 실행되는 광범위한 워크로드	개발/테스트 환경의 중간 규모 관계형 데이터베이스에서 실행되는 광범위한 워크로드
지연 시간	1밀리초 미만, 99.9%의 확률로 일관되게 제공	한 자리 밀리초, 99.9% 시간 동안 일관되게 제공	한 자리 밀리초, 99% 시간 동안 일관되게 제공	한 자리 밀리초, 99% 시간 동안 일관되게 제공
볼륨 크기	100~65,536GiB	100~65,536GiB(RDS for SQL Server에서는 20~16,384 GiB)	20~65,536 GiB(RDS for SQL Server에서는 16,384GiB)	20~65,536 GiB(RDS for SQL Server에서는 16,384GiB)
최대 IOPS	256,000	256,000(RDS for SQL Server에서는 64,000)	64,000(RDS for SQL Server에서는 16,000)	64,000(RDS for SQL Server에서는 16,000)

기능	프로비저닝된 IOPS(io2 Block Express)	프로비저닝된 IOPS(io1)	범용(gp3)	범용(gp2)
				Note gp2 스토리지에서 바로 IOPS를 프로비저닝할 수는 없습니다. IOPS는 할당된 스토리지 크기에 따라 달라집니다.

기능	프로비저닝된 IOPS(io2 Block Express)	프로비저닝된 IOPS(io1)	범용(gp3)	범용(gp2)
최대 처리량	프로비저닝된 IOPS를 기준으로 최대 4,000MB/s까지 확장 처리량은 프로비저닝된 IOPS당 최대 0.256Mib/s까지 비례하여 확장됩니다. 16KiB I/O 크기의 경우 256,000IOPS, 256KiB 입출력 크기의 경우 16,000IOPS 이상에서 최대 4,000MiB/s의 처리량을 달성할 수 있습니다. AWS Nitro System을 기반으로 하지 않는 인스턴스의 경우 16KiB I/O 크기의 경우 128,000IOPS에서 최대 2,000MiB/s의 처리량을 달성할 수 있습니다.	프로비저닝된 IOPS를 기준으로 최대 4,000MB/s까지 확장	최대 4,000MB/s의 추가 처리량 프로비저닝(RDS for SQL Server의 경우 1,000MB/s)	1000MB/s(RDS for SQL Server에서는 250 MB/s)
AWS CLI 및 RDS API 이름	io2	io1	gp3	gp2

마그네틱 스토리지(레거시, 권장하지 않음)

또한 Amazon RDS는 이전 버전과의 호환성을 위해 마그네틱 스토리지를 지원합니다. 새 스토리지가 필요할 경우 범용 SSD 또는 프로비저닝된 IOPS SSD를 사용하는 것이 좋습니다. 다음은 마그네틱 스토로지의 몇 가지 제한입니다.

- SQL 서버 데이터베이스 엔진을 사용할 경우 스토리지를 확장할 수 없습니다.
- SQL Server 데이터베이스 엔진을 사용할 경우 다른 스토리지로 전환할 수 없습니다.
- 스토리지 자동 조정을 지원하지 않습니다.
- 탄력적 볼륨을 지원하지 않습니다.
- 최대 3TiB 크기로 제한됩니다.
- 최대 1,000IOPS로 제한됩니다.

전용 로그 볼륨(DLV)

Amazon RDS 콘솔, AWS CLI 또는 Amazon RDS API를 사용하여 프로비저닝된 IOPS(PIOPS) 스토리지를 사용하는 DB 인스턴스 전용 로그 볼륨(DLV)을 사용할 수 있습니다. DLV는 PostgreSQL 데이터베이스 트랜잭션 로그 및 MySQL/MariaDB의 재실행 로그와 바이너리 로그를 데이터베이스 표가 들어 있는 볼륨과 분리된 스토리지 볼륨으로 옮깁니다. DLV는 트랜잭션 쓰기 로깅을 보다 효율적이고 일관되게 만듭니다. DLV는 할당된 스토리지가 크고 초당 I/O(IOPS) 요구 사항이 높거나 지연 시간에 민감한 워크로드가 있는 데이터베이스에 적합합니다.

DLV는 PIOPS 스토리지(io1 및 io2 Block Express)에 지원되며 1,000GiB의 고정 크기와 프로비저닝된 IOPS 3,000으로 생성됩니다.

> **Note**
>
> DLV는 범용 스토리지(gp2 및 gp3)에서 지원되지 않습니다.

Amazon RDS는 다음 버전의 경우 모든 AWS 리전에서 DLV를 지원합니다.

- MariaDB 10.6.7 이상의 10 버전
- MySQL 8.0.28 이상의 8 버전
- PostgreSQL 13.10 이상의 13 버전, 14.7 이상의 14 버전, 15.2 이상의 15 버전, 16.1 이상의 16 버전

RDS는 다중 AZ 배포와 함께 DLV를 지원합니다. 다중 AZ 인스턴스를 수정하거나 생성하면 기본 인스턴스와 보조 인스턴스 모두에 대해 DLV가 생성됩니다.

RDS는 읽기 전용 복제본이 있는 DLV를 지원합니다. 기본 DB 인스턴스에 DLV가 활성화되어 있는 경우 DLV를 활성화한 후 생성되는 모든 읽기 전용 복제본에도 DLV가 포함됩니다. DLV로 전환하기 전에 생성된 읽기 전용 복제본에는 명시적으로 수정하지 않는 한 DLV가 활성화되지 않습니다. DLV가 활성화되기 전에 기본 인스턴스에 연결된 모든 읽기 전용 복제본도 DLV를 사용하도록 수동으로 수정하는 것이 좋습니다.

DB 인스턴스의 DLV 설정을 수정하면 DB 인스턴스를 재부팅해야 합니다.

DLV 활성화에 대한 자세한 내용은 전용 로그 볼륨(DLV) 사용 섹션을 참조하세요.

스토리지 성능 모니터링

Amazon RDS는 DB 인스턴스의 성능을 측정할 수 있는 몇 가지 측정치가 있습니다. Amazon RDS 관리 콘솔의 인스턴스에 대한 요약 페이지에서 지표를 볼 수 있습니다. 또한 Amazon CloudWatch를 사용하여 이 측정치를 모니터링할 수도 있습니다. 자세한 내용은 Amazon RDS 콘솔에서 지표 보기 섹션을 참조하세요. 향상된 모니터링은 더 자세한 I/O 측정치를 제공합니다. 자세한 내용은 Enhanced Monitoring을 사용하여 OS 지표 모니터링 섹션을 참조하세요.

다음 지표는 DB 인스턴스용 스토리지를 모니터링하는 데 유용합니다.

- IOPS – 완료한 초당 I/O 작업 횟수입니다. 이 지표는 지정된 시간 간격 동안의 평균 IOPS로 보고됩니다. Amazon RDS는 1분의 간격을 두고 읽기 및 쓰기 IOPS를 따로 보고합니다. 총 IOPS는 읽기 IOPS와 쓰기 IOPS의 합계입니다. 일반적인 IOPS 값 범위는 초당 0에서 수만에 이릅니다.
- Latency(지연 시간) – I/O 요청을 제출하여 완료될 때까지 걸리는 시간입니다. 이 지표는 지정된 시간 간격 동안의 평균 대기 시간으로 보고됩니다. Amazon RDS는 1분의 간격을 두고 읽기 대기 시간과 쓰기 대기 시간을 따로 보고합니다. 일반적인 대기 시간 값은 밀리초(ms) 단위입니다.
- Throughput(처리량) – 디스크와 송/수신하는 초당 바이트 수입니다. 이 지표는 지정된 시간 간격 동안의 평균 처리량으로 보고됩니다. Amazon RDS는 1분 간격을 두고 초당 바이트(B/s) 단위의 읽기 처리량과 쓰기 처리량을 따로 보고합니다. 일반적인 처리량 값 범위는 0에서 I/O 채널의 최대 대역폭까지입니다.
- Queue Depth(대기열 깊이) – 대기열에서 처리를 기다리는 I/O 요청 수입니다. 애플리케이션에서 I/O 요청을 제출하였지만 디바이스가 다른 I/O 요청을 처리하고 있을 때는 전송되지 않는 경우가 있습니다. 이때 대기열에서 기다리게 되는데 이 대기 시간이 지연 시간과 서비스 시간(측정치로 사용하지 않음)을 구성합니다. 이 지표는 지정된 시간 간격 동안의 평균 대기열 깊이로 보고됩니다. Amazon RDS는 1분 간격으로 대기열 깊이를 보고합니다. 일반적인 대기열 깊이 값은 0에서 수백에 이릅니다.

측정된 IOPS 값은 개별 I/O 작업 크기와 독립적입니다. 따라서 I/O 성능을 측정할 경우, 단순한 I/O 연산 수가 아닌 인스턴스 처리량에 주목해야 합니다.

스토리지 성능에 영향을 끼치는 요인

시스템 활동, 데이터베이스 워크로드 및 DB 인스턴스 클래스가 스토리지 성능에 영향을 미칠 수 있습니다.

시스템 활동

다음 시스템 관련 활동은 I/O 용량을 사용하기 때문에 진행되는 동안에는 DB 인스턴스 성능이 떨어질 수 있습니다.

- 다중 AZ 대기 생성
- 읽기 전용 복제본 생성
- 스토리지 유형 변경

데이터베이스 워크로드

경우에 따라 데이터베이스 또는 애플리케이션 설계에 따라 동시 처리 가능한 연산 문제, 잠금 또는 그 밖에 데이터베이스 논쟁을 불러일으킬 수 있습니다. 이런 경우, 프로비저닝된 대역폭 모두를 직접 사용하지 못할 수 있습니다. 또한, 다음과 같은 워크로드 관련 상황이 발생할 수 있습니다.

- 기본 인스턴스 유형의 제한된 처리량에 도달했습니다.
- 애플리케이션이 충분한 I/O 작업을 실행하지 않기 때문에 대기열 깊이는 지속적으로 1 미만입니다.
- 몇 가지 I/O 용량을 사용하지 않아도 데이터베이스의 쿼리 문제가 있을 수 있습니다.

한도에 도달하거나 근접한 시스템 리소스가 없는 상태에서 스레드를 추가하더라도 데이터베이스 트랜잭션 속도가 높아지지 않는 경우도 있습니다. 이 경우 데이터베이스에서 발생하는 경합이 병목 지점일 가능성이 큽니다. 가장 공통적인 형식은 행 잠금과 인덱스 페이지 잠금이지만 여러 다른 방법도 있습니다. 이런 경우에는 데이터베이스 성능 튜닝 전문가에게 조언을 구하세요.

DB 인스턴스 클래스

Amazon RDS DB 인스턴스의 최대 성능을 얻으려면 자체 스토리지 유형을 지원하는 충분한 대역폭이 있는 현재 작업 인스턴스 유형을 선택합니다. 예를 들어, 10Gb 네트워크에 연결된 Amazon EBS에 최적화된 인스턴스를 선택해야 합니다.

> **Important**
>
> 사용 중인 인스턴스 클래스에 따라, IOPS 성능이 RDS를 사용하여 프로비저닝할 수 있는 최대 성능보다 낮을 수 있습니다. DB 인스턴스 클래스의 IOPS 성능에 대한 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EBS 최적화 인스턴스를 참조하세요. DB 인스턴스에 대해 프로비저닝된 IOPS 값을 설정하기 전에 인스턴스 클래스의 최대 IOPS를 결정하는 것이 좋습니다.

최상의 성능을 얻으려면 최신 세대 인스턴스를 사용할 것을 권장합니다. 이전 세대 DB 인스턴스는 최대 스토리지 한도가 더 낮을 수 있습니다.

일부 구형 32비트 파일 시스템은 스토리지 용량이 더 낮을 수 있습니다. DB 인스턴스의 스토리지 용량을 확인하려면 describe-valid-db-instance-modifications AWS CLI 명령을 사용합니다.

다음 목록은 각 데이터베이스 인스턴스에 대해 대부분의 DB 인스턴스 클래스가 확장할 수 있는 최대 스토리지를 보여줍니다.

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

다음 테이블에는 최대 스토리지(TiB)에 대한 몇 가지 예외가 나와 있습니다. io2 Block Express 스토리지를 제외한 모든 Microsoft SQL Server용 RDS DB 인스턴스의 최대 저장 용량은 16TiB이므로 SQL Server용 항목이 없습니다.

인스턴스 클래스	Db2	MariaDB	MySQL	Oracle	PostgreSQL
db.m3 – 표준 인스턴스 클래스					
db.t4g – 버스트 가능 성능 인스턴스 클래스					
db.t4g.medium	N/A	16	16	N/A	32

인스턴스 클래스	Db2	MariaDB	MySQL	Oracle	PostgreSQL
db.t4g.small	N/A	16	16	N/A	16
db.t4g.micro	N/A	6	6	N/A	6
db.t3 – 버스트 가능 성능 인스턴스 클래스					
db.t3.medium	32	16	16	32	32
db.t3.small	32	16	16	32	16
db.t3.micro	N/A	6	6	32	6
db.t2 – 버스트 가능 성능 인스턴스 클래스					

지원되는 모든 인스턴스 클래스에 대한 자세한 내용은 이전 세대 DB 인스턴스를 참조하세요.

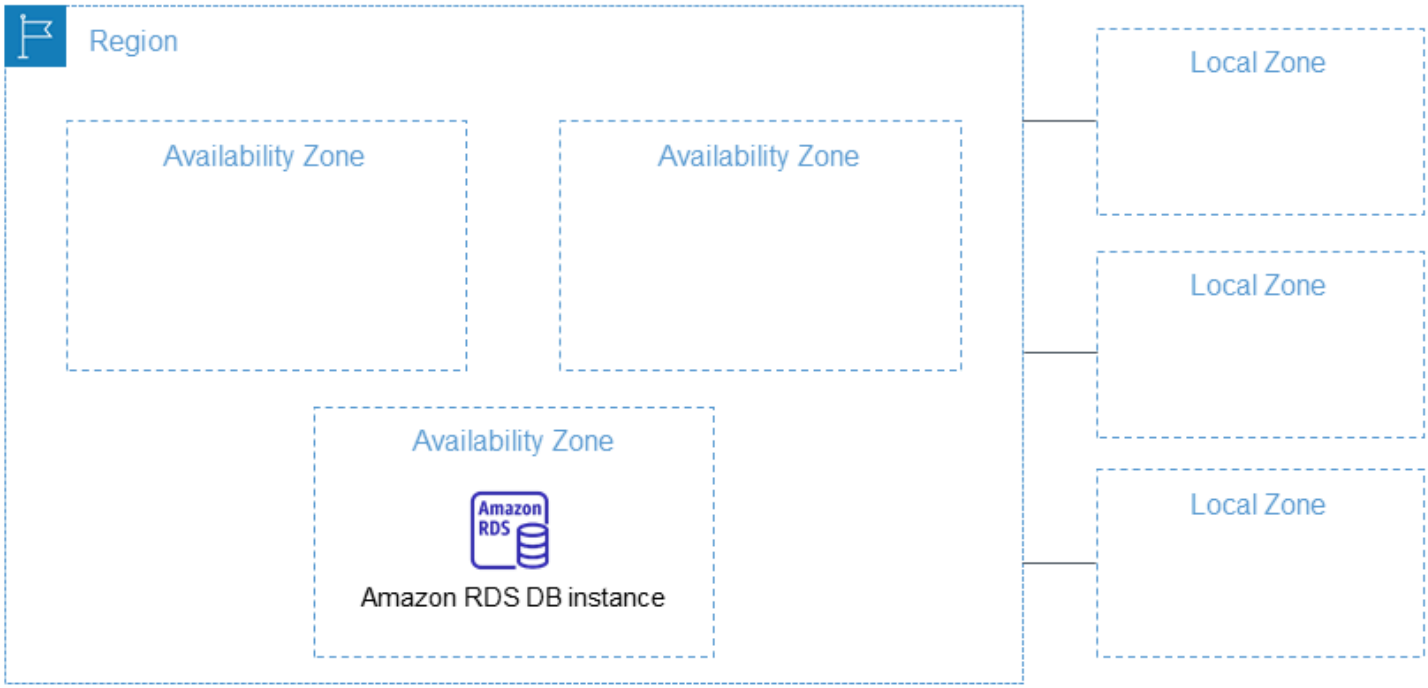
리전, 가용 영역 및 로컬 영역

Amazon 클라우드 컴퓨팅 리소스는 세계 각지의 여러 곳에서 호스팅됩니다. 이러한 위치는 AWS 리전, 가용 영역 및 Local Zones로 구성됩니다. 각 AWS 리전은 개별 지리 영역입니다. 각 AWS 리전은 가용 영역이라고 알려진 격리된 위치를 여러 개 가지고 있습니다.

> **Note**
>
> AWS 리전의 가용 영역을 찾는 방법에 대한 자세한 내용은 Amazon EC2 설명서의 가용 영역 설명을 참조하세요.

Local Zones를 통해 사용자에게 가까운 여러 위치에서 컴퓨팅, 스토리지 등의 리소스를 배치할 수 있습니다. Amazon RDS를 사용하면 DB 인스턴스와 같은 리소스와 데이터를 여러 위치에 배치할 수 있습니다. 특별히 지정하지 않을 경우 리소스는 여러 AWS 리전에 복제되지 않습니다.

Amazon은 최신 기술을 탑재한 고가용성 데이터 센터를 운영하고 있습니다. 드물기는 하지만 동일한 위치에 있는 DB 인스턴스의 가용성에 영향을 미치는 장애가 발생할 수도 있습니다. 그런 장애의 영향을 받는 하나의 위치에서 모든 DB 인스턴스를 호스팅하는 경우에는 모든 DB 인스턴스가 사용이 불가능해질 수 있습니다.



각 AWS 리전이 서로 완전히 독립적이라는 점에 유의하세요. 사용자의 모든 Amazon RDS 활동(예: 데이터베이스 인스턴스 또는 사용할 수 있는 데이터베이스 인스턴스 목록 생성 등)은 현재 기본 AWS 리

전에서만 실행됩니다. 기본 AWS 리전은 콘솔에서 변경하거나 `AWS_DEFAULT_REGION` 환경 변수를 설정하여 변경할 수 있습니다. 또는 AWS Command Line Interface(AWS CLI)에서 `--region` 파라미터를 사용하여 재정의할 수 있습니다. 자세한 내용은 AWS Command Line Interface 구성, 특히 환경 변수 및 명령줄 옵션에 대한 섹션을 참조하십시오.

Amazon RDS는 AWS라는 특수 AWS GovCloud (US) 리전을 지원합니다. 이러한 리전은 미국 정부 기관 및 고객이 더욱 민감한 워크로드를 클라우드로 이전하도록 설계되었습니다. AWS GovCloud (US) 리전은 미국 정부의 규제 및 규정 준수 요구 사항을 충족합니다. 자세한 내용은 AWS GovCloud (US)란 무엇입니까?를 참조하십시오.

특정 AWS 리전에서 Amazon RDS DB 인스턴스를 생성하거나 사용하려면 해당 리전 서비스 엔드포인트를 사용하세요.

AWS 리전

각 AWS 리전은 다른 AWS 리전에서 격리되도록 설계되었습니다. 이를 통해 가장 강력한 내결함성 및 안정성을 달성할 수 있습니다.

리소스를 볼 때 지정한 AWS 리전에 연결된 리소스만 표시됩니다. AWS 리전이 서로 격리되어 있고 여러 AWS 리전에 리소스가 자동으로 복제되지 않기 때문입니다.

리전 가용성

다음 표에는 현재 Amazon RDS를 사용할 수 있는 AWS 리전 및 각 리전의 엔드포인트가 나와 있습니다.

리전 이름	지역	엔드포인트	프로토콜	
미국 동부(오하이오)	us-east-2	rds.us-east-2.amazonaws.com	HTTPS	
		rds-fips.us-east-2.api.aws	HTTPS	
		rds.us-east-2.api.aws	HTTPS	
		rds-fips.us-east-2.amazonaws.com	HTTPS	
미국 동부(버지니아 북부)	us-east-1	rds.us-east-1.amazonaws.com	HTTPS	
		rds-fips.us-east-1.api.aws	HTTPS	
		rds-fips.us-east-1.amazonaws.com	HTTPS	

리전 이름	지역	엔드포인트	프로토콜
		rds.us-east-1.api.aws	HTTPS
미국 서부(캘리포니아 북부)	us-west-1	rds.us-west-1.amazonaws.com	HTTPS
		rds.us-west-1.api.aws	HTTPS
		rds-fips.us-west-1.amazonaws.com	HTTPS
		rds-fips.us-west-1.api.aws	HTTPS
미국 서부(오레곤)	us-west-2	rds.us-west-2.amazonaws.com	HTTPS
		rds-fips.us-west-2.amazonaws.com	HTTPS
		rds.us-west-2.api.aws	HTTPS
		rds-fips.us-west-2.api.aws	HTTPS
아프리카(케이프타운)	af-south-1	rds.af-south-1.amazonaws.com	HTTPS
		rds.af-south-1.api.aws	HTTPS
아시아 태평양(홍콩)	ap-east-1	rds.ap-east-1.amazonaws.com	HTTPS
		rds.ap-east-1.api.aws	HTTPS
아시아 태평양(하이데라바드)	ap-south-2	rds.ap-south-2.amazonaws.com	HTTPS
		rds.ap-south-2.api.aws	HTTPS
아시아 태평양(자카르타)	ap-southeast-3	rds.ap-southeast-3.amazonaws.com	HTTPS
		rds.ap-southeast-3.api.aws	HTTPS
아시아 태평양(멜버른)	ap-southeast-4	rds.ap-southeast-4.amazonaws.com	HTTPS
		rds.ap-southeast-4.api.aws	HTTPS

리전 이름	지역	엔드포인트	프로토콜
아시아 태평양(뭄바이)	ap-south-1	rds.ap-south-1.amazonaws.com	HTTPS
		rds.ap-south-1.api.aws	HTTPS
아시아 태평양(오사카)	ap-northeast-3	rds.ap-northeast-3.amazonaws.com	HTTPS
		rds.ap-northeast-3.api.aws	HTTPS
아시아 태평양(서울)	ap-northeast-2	rds.ap-northeast-2.amazonaws.com	HTTPS
		rds.ap-northeast-2.api.aws	HTTPS
아시아 태평양(싱가포르)	ap-southeast-1	rds.ap-southeast-1.amazonaws.com	HTTPS
		rds.ap-southeast-1.api.aws	HTTPS
아시아 태평양(시드니)	ap-southeast-2	rds.ap-southeast-2.amazonaws.com	HTTPS
		rds.ap-southeast-2.api.aws	HTTPS
아시아 태평양(도쿄)	ap-northeast-1	rds.ap-northeast-1.amazonaws.com	HTTPS
		rds.ap-northeast-1.api.aws	HTTPS
캐나다(중부)	ca-central-1	rds.ca-central-1.amazonaws.com	HTTPS
		rds.ca-central-1.api.aws	HTTPS
		rds-fips.ca-central-1.api.aws	HTTPS
		rds-fips.ca-central-1.amazonaws.com	HTTPS
캐나다 서부(캘거리)	ca-west-1	rds.ca-west-1.amazonaws.com	HTTPS
		rds-fips.ca-west-1.amazonaws.com	HTTPS

리전 이름	지역	엔드포인트	프로토콜
유럽(프랑크푸르트)	eu-central-1	rds.eu-central-1.amazonaws.com	HTTPS
		rds.eu-central-1.api.aws	HTTPS
유럽(아일랜드)	eu-west-1	rds.eu-west-1.amazonaws.com	HTTPS
		rds.eu-west-1.api.aws	HTTPS
유럽(런던)	eu-west-2	rds.eu-west-2.amazonaws.com	HTTPS
		rds.eu-west-2.api.aws	HTTPS
유럽(밀라노)	eu-south-1	rds.eu-south-1.amazonaws.com	HTTPS
		rds.eu-south-1.api.aws	HTTPS
유럽(파리)	eu-west-3	rds.eu-west-3.amazonaws.com	HTTPS
		rds.eu-west-3.api.aws	HTTPS
유럽(스페인)	eu-south-2	rds.eu-south-2.amazonaws.com	HTTPS
		rds.eu-south-2.api.aws	HTTPS
유럽(스톡홀름)	eu-north-1	rds.eu-north-1.amazonaws.com	HTTPS
		rds.eu-north-1.api.aws	HTTPS
유럽(취리히)	eu-central-2	rds.eu-central-2.amazonaws.com	HTTPS
		rds.eu-central-2.api.aws	HTTPS
이스라엘(텔아비브)	il-central-1	rds.il-central-1.amazonaws.com	HTTPS
		rds.il-central-1.api.aws	HTTPS
중동(바레인)	me-south-1	rds.me-south-1.amazonaws.com	HTTPS
		rds.me-south-1.api.aws	HTTPS

리전 이름	지역	엔드포인트	프로토콜
중동(UAE)	me-central-1	rds.me-central-1.amazonaws.com rds.me-central-1.api.aws	HTTPS HTTPS
남아메리카(상파울루)	sa-east-1	rds.sa-east-1.amazonaws.com rds.sa-east-1.api.aws	HTTPS HTTPS
AWS GovCloud(미국 동부)	us-gov-east-1	rds.us-gov-east-1.amazonaws.com rds.us-gov-east-1.api.aws	HTTPS HTTPS
AWS GovCloud(미국 서부)	us-gov-west-1	rds.us-gov-west-1.amazonaws.com rds.us-gov-west-1.api.aws	HTTPS HTTPS

엔드포인트를 명시적으로 지정하지 않는 경우 기본 엔드포인트는 미국 서부(오레곤)입니다.

AWS CLI 또는 API 작업을 사용하여 DB 인스턴스로 작업할 때는 리전 엔드포인트를 지정해야 합니다.

가용 영역

DB 인스턴스를 생성할 때 가용 영역을 선택하거나 Amazon RDS가 무작위로 가용 영역을 선택하도록 설정할 수 있습니다. 가용 영역은 AWS 리전 코드와 식별 문자의 조합으로 표시됩니다(예: `us-east-1a`).

다음과 같이 describe-availability-zones Amazon EC2 명령을 사용하여 계정에 대해 활성화된 지정된 리전 내의 가용 영역을 설명합니다.

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

예를 들어 계정에 활성화된 미국 동부(버지니아 북부) 리전(us-east-1) 내의 가용 영역을 설명하려면 다음 명령을 실행합니다.

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

다중 AZ DB 배포에서는 기본 및 세컨더리 DB 인스턴스의 가용 영역을 선택할 수 없습니다. Amazon RDS에서 가용 영역을 무작위로 선택합니다. 다중 AZ 배포에 대한 자세한 정보는 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 단원을 참조하십시오.

> **Note**
>
> RDS가 가용 영역을 무작위로 선택한다고 해서 단일 계정 또는 DB 서브넷 그룹 내의 가용 영역 간에 DB 인스턴스가 균등하게 분산되는 것은 아닙니다. 단일 AZ 인스턴스를 생성하거나 수정할 때 특정 AZ를 요청할 수 있으며, 다중 AZ 인스턴스에 더 구체적인 DB 서브넷 그룹을 사용할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 및 Amazon RDS DB 인스턴스 수정 단원을 참조하세요.

Local Zones

로컬 영역은 사용자와 지리적으로 가까운 AWS 리전의 확장입니다. 상위 AWS 리전에서 로컬 영역으로 어떤 VPC든 확장할 수 있습니다. 이렇게 하려면 새 서브넷을 생성하고 이 서브넷을 AWS 로컬 영역에 할당해야 합니다. 로컬 영역에 서브넷을 생성하면 VPC도 해당 로컬 영역으로 확장됩니다. 로컬 영역의 서브넷은 VPC의 다른 서브넷과 동일하게 작동합니다.

DB 인스턴스를 생성할 때 로컬 영역에서 서브넷을 선택할 수 있습니다. Local Zones는 인터넷에 대한 자체 연결을 가지고 있으며 AWS Direct Connect를 지원합니다. 따라서 로컬 영역에서 생성된 리소스는 지연 시간이 매우 짧은 통신으로 로컬 사용자에게 서비스를 제공할 수 있습니다. 자세한 내용은 AWS Local Zones를 참조하세요.

로컬 영역은 AWS 리전 코드 뒤에 위치를 나타내는 식별자를 붙여 표시됩니다(예: `us-west-2-lax-1a`).

> **Note**
>
> 로컬 영역은 다중 AZ 배포에 포함될 수 없습니다.

로컬 영역을 사용하려면

1. Amazon EC2 콘솔에서 로컬 영역을 활성화합니다.

   자세한 내용은 Amazon EC2 사용 설명서의 Local Zones 활성화를 참조하세요.
2. 로컬 영역에서 서브넷을 만듭니다.

자세한 내용은 Amazon VPC 사용 설명서의 VPC에서 서브넷 만들기를 참조하십시오.

3. 로컬 영역에서 DB 서브넷 그룹을 만듭니다.

   DB 서브넷 그룹을 생성할 때 로컬 영역에 대한 가용 영역 그룹을 선택합니다.

   자세한 내용은 VPC에 DB 인스턴스 만들기 섹션을 참조하세요.

4. 로컬 영역에서 DB 서브넷 그룹을 사용하는 DB 인스턴스를 만듭니다.

   자세한 내용은 Amazon RDS DB 인스턴스 생성 단원을 참조하세요.

> **Important**
>
> 현재 Amazon RDS를 사용할 수 있는 AWS 로컬 영역은 미국 서부(오레곤) 리전의 로스앤젤레스뿐입니다.

AWS 리전 및 DB 엔진별 Amazon RDS에서 지원되는 기능

Amazon RDS 기능 및 옵션에 대한 지원은 AWS 리전 및 각 DB 엔진의 특정 버전에 따라 다릅니다. 지정된 AWS 리전에서 RDS DB 엔진 버전 지원 및 가용성을 식별하기 위해 다음 섹션을 사용할 수 있습니다.

Amazon RDS 기능은 엔진 네이티브 기능 및 옵션과 다릅니다. 엔진 기본 기능 및 옵션에 대한 자세한 내용은 엔진 기본 기능을 참조하세요.

지원되는 리전 및 DB 엔진

- 테이블 규칙
- 기능 빠른 참조
- Amazon RDS 블루/그린 배포를 지원하는 리전 및 DB 엔진
- Amazon RDS에서 크로스 리전 자동 백업을 지원하는 리전 및 DB 엔진
- Amazon RDS에서 크로스 리전 읽기 전용 복제본을 지원하는 리전 및 DB 엔진
- Amazon RDS에서 데이터베이스 활동 스트림을 지원하는 리전 및 DB 엔진
- Amazon RDS의 이중 스택 모드를 지원하는 리전 및 DB 엔진
- Amazon RDS에서 S3로 스냅샷 내보내기가 지원되는 리전 및 DB 엔진
- Amazon RDS에서 IAM 데이터베이스 인증을 지원하는 리전 및 DB 엔진
- Amazon RDS에서 Kerberos 인증을 지원하는 리전 및 DB 엔진
- Amazon RDS에서 다중 AZ DB 클러스터를 지원하는 리전 및 DB 엔진
- Amazon RDS에서 성능 개선 도우미를 지원하는 리전 및 DB 엔진
- RDS Custom을 지원하는 리전 및 DB 엔진
- Amazon RDS 프록시를 지원하는 리전 및 DB 엔진
- Secrets Manager와 Amazon RDS 통합을 지원하는 리전 및 DB 엔진
- Amazon RDS 제로 ETL 통합을 지원하는 리전 및 DB 엔진
- Amazon RDS의 엔진 네이티브 기능

테이블 규칙

이 기능 섹션의 테이블은 다음 패턴을 사용하여 버전 번호 및 가용성 수준을 지정합니다:

- 버전 x.y – 특정 버전을 단독으로 사용할 수 있습니다.

- 버전 x.y 이상 – 지정된 버전 및 해당 버전의 메이저 버전에 속하며 지정된 버전보다 높은 모든 마이너 버전도 지원합니다. 예를 들어 '버전 10.11 이상'은 버전 10.11, 10.11.1은 물론 10.12도 사용할 수 있음을 의미합니다.

기능 빠른 참조

다음 빠른 참조 표에는 각 기능 및 사용 가능한 RDS DB 엔진이 나열되어 있습니다. 리전 및 특정 버전 사용 가능 여부는 이후 기능 섹션에 나와 있습니다.

기능	RDS for Db2	RDS for MariaDB	RDS for MySQL	RDS for Oracle	RDS for PostgreSQL	RDS for SQL Server
블루/그린 배포	사용할 수 없음	사용 가능	사용 가능	사용할 수 없음	사용 가능	사용할 수 없음
교차 리전 자동 백업	사용 가능	사용 가능	사용 가능	사용 가능	사용 가능	사용 가능
리전 간 읽기 전용 복제본	사용할 수 없음	사용 가능	사용 가능	사용 가능	사용 가능	사용 가능
데이터베이스 활동 스트림	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용 가능	사용할 수 없음	사용 가능

기능	RDS for Db2	RDS for MariaDB	RDS for MySQL	RDS for Oracle	RDS for PostgreSQL	RDS for SQL Server
듀얼 스택 모드	사용할 수 없음	사용 가능	사용 가능	사용 가능	사용 가능	사용 가능
Amazo S3 S3 로 스냅샷 내보내기	사용할 수 없음	사용 가능	사용 가능	사용할 수 없음	사용 가능	사용할 수 없음
AWS Identity and Access Manag t(IAM) 데이터베이스 인증	사용할 수 없음	사용 가능	사용 가능	사용할 수 없음	사용 가능	사용할 수 없음
Kerbe 인증	사용 가능	사용할 수 없음	사용 가능	사용 가능	사용 가능	사용 가능
다중 AZ DB 클러스터	사용할 수 없음	사용할 수 없음	사용 가능	사용할 수 없음	사용 가능	사용할 수 없음

기능	RDS for Db2	RDS for MariaDB	RDS for MySQL	RDS for Oracle	RDS for PostgreSQL	RDS for SQL Server
성능 개선 도우미	사용할 수 없음	사용 가능	사용 가능	사용 가능	사용 가능	사용 가능
RDS Custor	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용 가능	사용할 수 없음	사용 가능
RDS 프록시	사용할 수 없음	사용 가능	사용 가능	사용할 수 없음	사용 가능	사용 가능
Secret Manag 통합	사용 가능	사용 가능	사용 가능	사용 가능	사용 가능	사용 가능

Amazon RDS 블루/그린 배포를 지원하는 리전 및 DB 엔진

블루/그린 배포는 프로덕션 데이터베이스 환경을 별도의 동기화된 스테이징 환경에 복사합니다. Amazon RDS 블루/그린 배포를 사용하면 프로덕션 환경에 영향을 주지 않고 스테이징 환경에서 데이터베이스를 변경할 수 있습니다. 예를 들어 메이저 또는 마이너 DB 엔진 버전을 업그레이드하거나, 데이터베이스 파라미터를 변경하거나, 스테이징 환경 내 스키마를 변경할 수 있습니다. 준비가 되면 스테이징 환경을 새 프로덕션 데이터베이스 환경으로 승격할 수 있습니다. 자세한 내용은 데이터베이스 업데이트에 Amazon RDS 블루/그린 배포 사용 단원을 참조하십시오.

블루/그린 배포는 모든 AWS 리전에서 지원됩니다.

블루/그린 배포는 다음 엔진에서는 지원되지 않습니다.

- RDS for Db2
- RDS for SQL Server
- RDS for Oracle

주제

- RDS for MariaDB를 사용한 블루/그린 배포
- RDS for MySQL을 사용한 블루/그린 배포
- RDS for PostgreSQL을 사용한 블루/그린 배포

RDS for MariaDB를 사용한 블루/그린 배포

RDS for MariaDB의 경우 블루/그린 배포는 다음 버전에서 지원됩니다.

- RDS for MariaDB 11.4(사용 가능한 모든 버전)
- RDS for MariaDB 버전 10.2 이상 10 버전

RDS for MySQL을 사용한 블루/그린 배포

RDS for MySQL의 경우 블루/그린 배포는 다음 버전에서 지원됩니다.

- RDS for MySQL 8.4(사용 가능한 모든 버전)
- RDS for MySQL 8.0(사용 가능한 모든 버전)
- RDS for MySQL 5.7(사용 가능한 모든 버전)

RDS for PostgreSQL을 사용한 블루/그린 배포

RDS for PostgreSQL의 경우 블루/그린 배포는 버전 11.1 이상 메이저 및 마이너 버전에서 지원됩니다.

> **Note**
>
> 특정 조건에서 RDS for PostgreSQL은 물리적 복제 대신 논리적 복제를 사용하여 그린 환경을 블루 환경과 동기화된 상태로 유지합니다. 자세한 내용은 ??? 단원을 참조하십시오.

Amazon RDS에서 크로스 리전 자동 백업을 지원하는 리전 및 DB 엔진

Amazon RDS에서 백업 복제를 사용하면 스냅샷과 트랜잭션 로그를 대상 리전에 복제하도록 RDS DB 인스턴스를 구성할 수 있습니다. DB 인스턴스에 대해 백업 복제가 구성된 경우 RDS는 모든 스냅샷 및 트랜잭션 로그가 준비되면 교차 리전 복사를 시작합니다. 자세한 내용은 다른 AWS 리전에 자동 백업 복제 섹션을 참조하세요.

백업 복제를 사용할 수 있는 AWS 리전에 대한 정보는 다른 AWS 리전에 자동 백업 복제 섹션을 참조하세요.

주제

- RDS for Db2를 사용한 백업 복제
- RDS for MariaDB를 사용한 백업 복제
- RDS for MySQL을 사용한 백업 복제
- RDS for Oracle를 사용한 백업 복제
- RDS for PostgreSQL을 사용한 백업 복제
- RDS for SQL Server를 사용한 백업 복제

RDS for Db2를 사용한 백업 복제

Amazon RDS는 현재 사용 가능한 모든 RDS for Db2 버전에 대한 백업 복제를 지원합니다.

RDS for MariaDB를 사용한 백업 복제

Amazon RDS는 현재 사용 가능한 모든 RDS for MariaDB 버전에 대한 백업 복제를 지원합니다.

RDS for MySQL을 사용한 백업 복제

Amazon RDS는 현재 사용 가능한 모든 RDS for MySQL 버전에 대한 백업 복제를 지원합니다.

RDS for Oracle를 사용한 백업 복제

Amazon RDS는 현재 사용 가능한 모든 RDS for Oracle 버전에 대한 백업 복제를 지원합니다.

RDS for PostgreSQL을 사용한 백업 복제

Amazon RDS는 현재 사용 가능한 모든 RDS for PostgreSQL 버전에 대한 백업 복제를 지원합니다.

RDS for SQL Server를 사용한 백업 복제

Amazon RDS는 현재 사용 가능한 모든 RDS for SQL Server 버전에 대한 백업 복제를 지원합니다.

Amazon RDS에서 크로스 리전 읽기 전용 복제본을 지원하는 리전 및 DB 엔진

Amazon RDS에서 리전 간 읽기 전용 복제본을 사용하면 소스 DB 인스턴스와 다른 리전에서 MariaDB, MySQL, Oracle, PostgreSQL 또는 SQL Server 읽기 전용 복제본을 생성할 수 있습니다. 소스 및 대상

리전 고려 사항을 포함한 리전 간 읽기 전용 복제본에 대한 자세한 내용은 다른 AWS 리전에서 읽기 전용 복제본 생성 단원을 참조하세요.

리전 간 읽기 전용 복제본은 다음 엔진에서 사용할 수 없습니다.

- RDS for Db2

주제

- RDS for MariaDB를 사용하는 리전 간 읽기 전용 복제본
- RDS for MySQL을 사용하는 리전 간 읽기 전용 복제본
- RDS for Oracle을 사용하는 리전 간 읽기 전용 복제본
- RDS for PostgreSQL을 사용하는 리전 간 읽기 전용 복제본
- RDS for SQL Server를 사용하는 리전 간 읽기 전용 복제본

RDS for MariaDB를 사용하는 리전 간 읽기 전용 복제본

RDS for MariaDB를 사용하는 리전 간 읽기 전용 복제본은 다음 버전의 경우 모든 리전에서 사용할 수 있습니다.

- RDS for MariaDB 11.4(사용 가능한 모든 버전)
- RDS for MariaDB 10.11(사용 가능한 모든 버전)
- RDS for MariaDB 10.6(사용 가능한 모든 버전)
- RDS for MariaDB 10.5(사용 가능한 모든 버전)
- RDS for MariaDB 10.4(사용 가능한 모든 버전)

RDS for MySQL을 사용하는 리전 간 읽기 전용 복제본

RDS for MySQL을 사용하는 리전 간 읽기 전용 복제본은 다음 버전의 경우 모든 리전에서 사용할 수 있습니다.

- RDS for MySQL 8.4(사용 가능한 모든 버전)
- RDS for MySQL 8.0(사용 가능한 모든 버전)
- RDS for MySQL 5.7(사용 가능한 모든 버전)

RDS for Oracle을 사용하는 리전 간 읽기 전용 복제본

RDS for Oracle용 교차 리전 읽기 복제본은 Enterprise Edition을 사용하는 모든 지원되는 데이터베이스 버전에 대해 모든 AWS 리전에서 사용할 수 있습니다. 복제본은 비 CDB 및 CDB 아키텍처의 단일 테넌트 구성에서만 지원됩니다. CDB 아키텍처의 멀티테넌트 구성에서는 교차 리전 읽기 복제본이 지원되지 않습니다.

RDS for Oracle을 사용하는 리전 간 읽기 전용 복제본에 대한 추가 요구 사항에 대한 자세한 내용은 RDS for Oracle 복제본에 대한 요구 사항 및 고려 사항 단원을 참조하세요.

RDS for PostgreSQL을 사용하는 리전 간 읽기 전용 복제본

RDS for PostgreSQL을 사용하는 리전 간 읽기 전용 복제본은 다음 버전의 경우 모든 리전에서 사용할 수 있습니다.

- RDS for PostgreSQL 17(사용 가능한 모든 버전)
- RDS for PostgreSQL 16(사용 가능한 모든 버전)
- RDS for PostgreSQL 15(사용 가능한 모든 버전)
- RDS for PostgreSQL 14(사용 가능한 모든 버전)
- RDS for PostgreSQL 13(사용 가능한 모든 버전)
- RDS for PostgreSQL 12(사용 가능한 모든 버전)
- RDS for PostgreSQL 11(사용 가능한 모든 버전)
- RDS for PostgreSQL 10(사용 가능한 모든 버전)

RDS for SQL Server를 사용하는 리전 간 읽기 전용 복제본

RDS for SQL Server를 사용하는 리전 간 읽기 전용 복제본은 다음을 제외한 모든 리전에서 사용할 수 있습니다.

- 아프리카(케이프타운)
- 아시아 태평양(홍콩)
- 아시아 태평양(하이데라바드)
- 아시아 태평양(자카르타)
- 아시아 태평양(멜버른)
- 캐나다 서부(캘거리)

- 유럽(밀라노)
- 유럽(스페인)
- 유럽(취리히)
- 이스라엘(텔아비브)
- 중동(바레인)
- 중동(UAE)

RDS for SQL Server가 포함된 리전 간 읽기 전용 복제본은 Microsoft SQL Server 엔터프라이즈 에디션을 사용하는 다음 버전에서 사용할 수 있습니다.

- RDS for SQL Server 2022
- RDS for SQL Server 2019(버전 15.00.4073.23 이상)
- RDS for SQL Server 2017(버전 14.00.3281.6 이상)
- RDS for SQL Server 2016(버전 13.00.6300.2 이상)

Amazon RDS에서 데이터베이스 활동 스트림을 지원하는 리전 및 DB 엔진

Amazon RDS에서 데이터베이스 활동 스트림을 사용하면 Oracle 데이터베이스 및 SQL Server 데이터베이스에서의 감사 활동에 대한 경보를 모니터링하고 설정할 수 있습니다. 자세한 내용은 데이터베이스 활동 스트림 개요 단원을 참조하십시오.

다음 엔진에서는 데이터베이스 활동 스트림을 사용할 수 없습니다.

- RDS for Db2
- RDS for MariaDB
- RDS for MySQL
- RDS for PostgreSQL

주제

- RDS for Oracle을 사용하는 데이터베이스 활동 스트림
- 데이터베이스 활동 스트림과 RDS for SQL Server 함께 사용

RDS for Oracle을 사용하는 데이터베이스 활동 스트림

다음 리전 및 엔진 버전을 RDS for Oracle을 사용하는 데이터베이스 활동 스트림에 사용할 수 있습니다.

RDS for Oracle을 사용하는 데이터베이스 활동 스트림에 대한 추가 요구 사항에 대한 자세한 내용은 데이터베이스 활동 스트림 개요 단원을 참조하세요.

리전	RDS for Oracle 21c	RDS for Oracle 19c
미국 동부(버지니아 북부)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용
미국 동부(오하이오)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용
미국 서부(캘리포니아 북부)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용
미국 서부(오레곤)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용
아프리카(케이프타운)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용
아시아 태평양(홍콩)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용

리전	RDS for Oracle 21c	RDS for Oracle 19c
아시아 태평양(하이데라바드)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용
아시아 태평양(자카르타)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용
아시아 태평양(말레이시아)	사용할 수 없음	사용할 수 없음
아시아 태평양(멜버른)	사용할 수 없음	사용할 수 없음
아시아 태평양(뭄바이)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용
아시아 태평양(오사카)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용
아시아 태평양(서울)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용
아시아 태평양(싱가포르)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용

리전	RDS for Oracle 21c	RDS for Oracle 19c
아시아 태평양(시드니)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용
아시아 태평양(도쿄)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용
캐나다(중부)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용
캐나다 서부(캘거리)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용
중국(베이징)	사용할 수 없음	사용할 수 없음
중국(닝샤)	사용할 수 없음	사용할 수 없음
유럽(프랑크푸르트)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용
유럽(아일랜드)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용

리전	RDS for Oracle 21c	RDS for Oracle 19c
유럽(런던)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용
유럽(밀라노)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용
유럽(파리)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용
유럽(스페인)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용
유럽(스톡홀름)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용
유럽(취리히)	사용할 수 없음	사용할 수 없음
아시아 태평양(멜버른)	사용할 수 없음	사용할 수 없음
중동(바레인)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용

리전	RDS for Oracle 21c	RDS for Oracle 19c
중동(UAE)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용
남아메리카(상파울루)	사용할 수 없음	Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 이상, Enterprise Edition(EE) 또는 Standard Edition 2(SE2) 중 하나 사용
AWS GovCloud(미국 동부)	사용할 수 없음	사용할 수 없음
AWS GovCloud(미국 서부)	사용할 수 없음	사용할 수 없음

데이터베이스 활동 스트림과 RDS for SQL Server 함께 사용

다음 리전 및 엔진 버전을 RDS for SQL Server를 사용하는 데이터베이스 활동 스트림에 사용할 수 있습니다.

RDS for SQL Server를 사용하는 데이터베이스 활동 스트림에 대한 추가 요구 사항에 대한 자세한 내용은 데이터베이스 활동 스트림 개요 섹션을 참조하세요.

리전	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
미국 동부(버지니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 동부(오하이오)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(캘리포니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(오레곤)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
아프리카(케이프타운)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(홍콩)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(하이데라바드)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(자카르타)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(말레이시아)	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(멜버른)	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(뭄바이)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(오사카)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(서울)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(싱가포르)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(시드니)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(도쿄)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다(중부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다 서부(캘거리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중국(베이징)	사용할 수 없음	사용할 수 없음	사용할 수 없음
중국(닝샤)	사용할 수 없음	사용할 수 없음	사용할 수 없음
유럽(프랑크푸르트)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
유럽(아일랜드)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(런던)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(밀라노)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(파리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(스페인)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(스톡홀름)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(취리히)	사용할 수 없음	사용할 수 없음	사용할 수 없음
이스라엘(텔아비브)	사용할 수 없음	사용할 수 없음	사용할 수 없음
중동(바레인)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중동(UAE)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
남아메리카(상파울루)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
AWS GovCloud(미국 동부)	사용할 수 없음	사용할 수 없음	사용할 수 없음
AWS GovCloud(미국 서부)	사용할 수 없음	사용할 수 없음	사용할 수 없음

Amazon RDS의 이중 스택 모드를 지원하는 리전 및 DB 엔진

RDS에서 듀얼 스택 모드를 사용하면 리소스가 인터넷 프로토콜 버전 4(IPv4), 인터넷 프로토콜 버전 6(IPv6) 또는 둘 다를 통해 DB 인스턴스와 통신할 수 있습니다. 자세한 내용은 듀얼 스택 모드 섹션을 참조하세요.

주제

- RDS for Db2를 사용하는 듀얼 스택 모드

- RDS for MariaDB가 포함된 듀얼 스택 모드
- RDS for MySQL을 사용하는 듀얼 스택 모드
- RDS for Oracle을 사용하는 듀얼 스택 모드
- RDS for PostgreSQL을 사용하는 듀얼 스택 모드
- RDS for SQL Server를 사용하는 듀얼 스택 모드

IPv6는 다음 AWS 리전의 Amazon RDS DB 엔진에서 지원되지 않습니다.

- 아시아 태평양(하이데라바드)
- 중동(UAE)
- 아시아 태평양(말레이시아)
- 아시아 태평양(멜버른)
- 캐나다 서부(캘거리)
- 유럽(취리히)
- 이스라엘(텔아비브)

RDS for Db2를 사용하는 듀얼 스택 모드

다음 리전 및 엔진 버전을 RDS for Db2를 사용하는 듀얼 스택 모드에 사용할 수 있습니다.

리전	RDS for Db2 11.5				
미국 동부(버지니아 북부)	사용 가능한 모든 버전				
미국 동부(오하이오)	사용 가능한 모든 버전				
미국 서부(캘리포니아 북부)	사용 가능한 모든 버전				
미국 서부(오리건)	사용 가능한 모든 버전				

리전	RDS for Db2 11.5				
아프리카(케이프타운)	사용 가능한 모든 버전				
아시아 태평양(홍콩)	사용 가능한 모든 버전				
아시아 태평양(하이데라바드)	사용할 수 없음				
아시아 태평양(자카르타)	사용 가능한 모든 버전				
아시아 태평양(말레이시아)	사용할 수 없음				
아시아 태평양(멜버른)	사용할 수 없음				
아시아 태평양(뭄바이)	사용 가능한 모든 버전				
아시아 태평양(오사카)	사용 가능한 모든 버전				
아시아 태평양(서울)	사용 가능한 모든 버전				
아시아 태평양(싱가포르)	사용 가능한 모든 버전				
아시아 태평양(시드니)	사용 가능한 모든 버전				

리전	RDS for Db2 11.5				
아시아 태평양(도쿄)	사용 가능한 모든 버전				
캐나다(중부)	사용 가능한 모든 버전				
캐나다 서부(캘거리)	사용할 수 없음				
중국(베이징)	사용할 수 없음				
중국(닝샤)	사용할 수 없음				
유럽(프랑크푸르트)	사용 가능한 모든 버전				
유럽(아일랜드)	사용 가능한 모든 버전				
유럽(런던)	사용 가능한 모든 버전				
유럽(밀라노)	사용 가능한 모든 버전				
유럽(파리)	사용 가능한 모든 버전				
유럽(스페인)	사용할 수 없음				
유럽(스톡홀름)	사용 가능한 모든 버전				

리전	RDS for Db2 11.5				
유럽(취리히)	사용할 수 없음				
이스라엘(텔아비브)	사용할 수 없음				
중동(바레인)	사용 가능한 모든 버전				
중동(UAE)	사용할 수 없음				
남아메리카(상파울루)	사용 가능한 모든 버전				
AWS GovCloud(미국 동부)	사용할 수 없음				
AWS GovCloud(미국 서부)	사용할 수 없음				

RDS for MariaDB가 포함된 듀얼 스택 모드

다음 리전 및 엔진 버전을 RDS for MariaDB를 사용하는 듀얼 스택 모드에 사용할 수 있습니다.

리전	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
미국 동부(버지니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
미국 동부(오하이오)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(캘리포니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(오리건)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아프리카(케이프타운)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(홍콩)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(하이데라바드)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(자카르타)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(말레이시아)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(멜버른)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(뭄바이)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(오사카)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
아시아 태평양(서울)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(싱가포르)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(시드니)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(도쿄)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다(중부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다 서부(캘거리)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
중국(베이징)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중국(닝샤)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(프랑크푸르트)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(아일랜드)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(런던)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(밀라노)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
유럽(파리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(스페인)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
유럽(스톡홀름)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(취리히)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
이스라엘(텔아비브)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
중동(바레인)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중동(UAE)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
남아메리카(상파울루)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
AWS GovCloud(미국 동부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
AWS GovCloud(미국 서부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

RDS for MySQL을 사용하는 듀얼 스택 모드

다음 리전 및 엔진 버전을 RDS for MySQL을 사용하는 듀얼 스택 모드에 사용할 수 있습니다.

리전	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7(RDS 추가 지원 적용)
미국 동부(버지니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 동부(오하이오)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(캘리포니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(오리건)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아프리카(케이프타운)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(홍콩)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(하이데라바드)	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(자카르타)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(말레이시아)	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(멜버른)	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(뭄바이)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(오사카)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(서울)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(싱가포르)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(시드니)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7(RDS 추가 지원 적용)
아시아 태평양(도쿄)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다(중부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다 서부(캘거리)	사용할 수 없음	사용할 수 없음	사용할 수 없음
중국(베이징)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중국(닝샤)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(프랑크푸르트)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(아일랜드)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(런던)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(밀라노)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(파리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(스페인)	사용할 수 없음	사용할 수 없음	사용할 수 없음
유럽(스톡홀름)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(취리히)	사용할 수 없음	사용할 수 없음	사용할 수 없음
이스라엘(텔아비브)	사용할 수 없음	사용할 수 없음	사용할 수 없음
중동(바레인)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중동(UAE)	사용할 수 없음	사용할 수 없음	사용할 수 없음
남아메리카(상파울루)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
AWS GovCloud(미국 동부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7(RDS 추가 지원 적용)
AWS GovCloud(미국 서부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

RDS for Oracle을 사용하는 듀얼 스택 모드

다음 리전 및 엔진 버전을 RDS for Oracle을 사용하는 듀얼 스택 모드에 사용할 수 있습니다.

리전	RDS for Oracle 21c	RDS for Oracle 19c
미국 동부(버지니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 동부(오하이오)	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(캘리포니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(오리건)	사용 가능한 모든 버전	사용 가능한 모든 버전
아프리카(케이프타운)	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(홍콩)	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(하이데라바드)	사용할 수 없음	사용할 수 없음
아시아 태평양(자카르타)	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(말레이시아)	사용할 수 없음	사용할 수 없음
아시아 태평양(멜버른)	사용할 수 없음	사용할 수 없음
아시아 태평양(뭄바이)	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(오사카)	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(서울)	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(싱가포르)	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for Oracle 21c	RDS for Oracle 19c
아시아 태평양(시드니)	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(도쿄)	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다(중부)	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다 서부(캘거리)	사용할 수 없음	사용할 수 없음
중국(베이징)	사용 가능한 모든 버전	사용 가능한 모든 버전
중국(닝샤)	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(프랑크푸르트)	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(아일랜드)	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(런던)	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(밀라노)	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(파리)	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(스페인)	사용할 수 없음	사용할 수 없음
유럽(스톡홀름)	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(취리히)	사용할 수 없음	사용할 수 없음
이스라엘(텔아비브)	사용할 수 없음	사용할 수 없음
중동(바레인)	사용 가능한 모든 버전	사용 가능한 모든 버전
중동(UAE)	사용할 수 없음	사용할 수 없음
남아메리카(상파울루)	사용 가능한 모든 버전	사용 가능한 모든 버전
AWS GovCloud(미국 동부)	사용 가능한 모든 버전	사용 가능한 모든 버전
AWS GovCloud(미국 서부)	사용 가능한 모든 버전	사용 가능한 모든 버전

RDS for PostgreSQL을 사용하는 듀얼 스택 모드

다음 리전 및 엔진 버전을 RDS for PostgreSQL을 사용하는 듀얼 스택 모드에 사용할 수 있습니다.

리전	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
미국 동부(버지니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 동부(오하이오)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(캘리포니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(오리건)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아프리카(케이프타운)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(홍콩)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(하이데라바드)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음

리전	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
아시아 태평양 (자카르타)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양 (말레이시아)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양 (멜버른)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양 (뭄바이)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양 (오사카)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양 (서울)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양 (싱가포르)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양 (시드니)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
아시아 태평양(도쿄)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다(중부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다 서부(캘거리)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
중국(베이징)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중국(닝샤)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(프랑크푸르트)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(아일랜드)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(런던)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(밀라노)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
유럽(파리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(스페인)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
유럽(스톡홀름)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(취리히)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
이스라엘(텔아비브)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
중동(바레인)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중동(UAE)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
남아메리카(상파울루)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
AWS GovCloud 미국 동부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
AWS GovCloud(미국 서부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

RDS for SQL Server를 사용하는 듀얼 스택 모드

다음 리전 및 엔진 버전을 RDS for SQL Server를 사용하는 듀얼 스택 모드에 사용할 수 있습니다.

리전	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
미국 동부(버지니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 동부(오하이오)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(캘리포니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(오리건)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아프리카(케이프타운)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(홍콩)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(하이데라바드)	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(자카르타)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(말레이시아)	사용할 수 없음	사용할 수 없음	사용할 수 없음

리전	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
아시아 태평양(멜버른)	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(뭄바이)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(오사카)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(서울)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(싱가포르)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(시드니)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(도쿄)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다(중부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다 서부(캘거리)	사용할 수 없음	사용할 수 없음	사용할 수 없음
중국(베이징)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중국(닝샤)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(프랑크푸르트)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(아일랜드)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(런던)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(밀라노)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(파리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(스페인)	사용할 수 없음	사용할 수 없음	사용할 수 없음
유럽(스톡홀름)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(취리히)	사용할 수 없음	사용할 수 없음	사용할 수 없음

리전	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
이스라엘(텔아비브)	사용할 수 없음	사용할 수 없음	사용할 수 없음
중동(바레인)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중동(UAE)	사용할 수 없음	사용할 수 없음	사용할 수 없음
남아메리카(상파울루)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
AWS GovCloud(미국 동부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
AWS GovCloud(미국 서부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

Amazon RDS에서 S3로 스냅샷 내보내기가 지원되는 리전 및 DB 엔진

RBS DB 스냅샷 데이터를 Amazon S3 버킷으로 내보낼 수 있습니다. 수동 스냅샷, 자동 시스템 스냅샷 및 AWS Backup에서 생성한 스냅샷을 포함하여 모든 유형의 DB 스냅샷을 내보낼 수 있습니다. 데이터를 내보낸 후에는 Amazon Athena 또는 Amazon Redshift Spectrum 같은 도구를 통해 직접 내보낸 데이터를 분석할 수 있습니다. 자세한 내용은 Amazon S3 for Amazon RDS로 DB 스냅샷 데이터 내보내기 단원을 참조하십시오.

다음 엔진에서는 스냅샷을 S3로 내보낼 수 없습니다.

- RDS for Db2
- RDS for Oracle
- RDS for SQL Server

주제

- RDS for MariaDB를 사용하여 스냅샷을 S3로 내보내기
- RDS for MySQL을 사용하여 스냅샷을 S3로 내보내기
- RDS for PostgreSQL을 사용하여 스냅샷을 S3로 내보내기

RDS for MariaDB를 사용하여 스냅샷을 S3로 내보내기

다음 리전 및 엔진 버전을 RDS for MariaDB를 사용하는 S3로 스냅샷 내보내기에 사용할 수 있습니다.

리전	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
미국 동부(버지니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 동부(오하이오)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(캘리포니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(오리건)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아프리카(케이프타운)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(홍콩)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(하이데라바드)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(자카르타)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(말레이시아)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음

리전	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
아시아 태평양(멜버른)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(뭄바이)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(오사카)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(서울)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(싱가포르)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(시드니)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(도쿄)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다(중부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다 서부(캘거리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중국(베이징)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중국(닝샤)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(프랑크푸르트)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
유럽(아일랜드)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(런던)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(밀라노)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(파리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(스페인)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(스톡홀름)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(취리히)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
이스라엘(텔아비브)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중동(바레인)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중동(UAE)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
남아메리카(상파울루)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
AWS GovCloud(미국 동부)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
AWS GovCloud(미국 서부)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음

RDS for MySQL을 사용하여 스냅샷을 S3로 내보내기

RDS for MySQL를 사용하여 스냅샷을 S3로 내보내는 데 사용할 수 있는 리전 및 엔진 버전은 다음과 같습니다.

리전	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7(RDS 추가 지원 적용)
미국 동부(오하이오)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 동부(버지니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(캘리포니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(오리건)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아프리카(케이프타운)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(홍콩)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(하이데라바드)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7(RDS 추가 지원 적용)
아시아 태평양(자카르타)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(말레이시아)	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(멜버른)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(뭄바이)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(오사카)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(서울)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(싱가포르)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(시드니)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(도쿄)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다(중부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다 서부(캘거리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중국(베이징)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중국(닝샤)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(프랑크푸르트)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(아일랜드)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(런던)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(밀라노)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7(RDS 추가 지원 적용)
유럽(파리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(스페인)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(스톡홀름)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(취리히)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
이스라엘(텔아비브)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중동(바레인)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중동(UAE)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
남아메리카(상파울루)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
AWS GovCloud(미국 동부)	사용할 수 없음	사용할 수 없음	사용할 수 없음
AWS GovCloud(미국 서부)	사용할 수 없음	사용할 수 없음	사용할 수 없음

RDS for PostgreSQL을 사용하여 스냅샷을 S3로 내보내기

다음 리전 및 엔진 버전을 RDS for PostgreSQL을 사용하는 S3로 스냅샷 내보내기에 사용할 수 있습니다.

리전	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
미국 동부(오하이오)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
미국 동부(버지니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(캘리포니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(오리건)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아프리카(케이프타운)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(홍콩)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(하이데라바드)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(자카르타)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
아시아 태평양 (말레이시아)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양 (멜버른)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양 (뭄바이)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양 (오사카)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양 (서울)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양 (싱가포르)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양 (시드니)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양 (도쿄)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
캐나다(중부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다 서부(캘거리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중국(베이징)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중국(닝샤)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(프랑크푸르트)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(아일랜드)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(런던)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(밀라노)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(파리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
유럽(스페인)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(스톡홀름)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(취리히)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
이스라엘(텔아비브)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중동(바레인)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중동(UAE)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
남아메리카(상파울루)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
AWS GovCloud(미국 동부)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음

리전	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
AWS GovCloud (미국 서부)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음

Amazon RDS에서 IAM 데이터베이스 인증을 지원하는 리전 및 DB 엔진

Amazon RDS에서 IAM 데이터베이스 인증을 사용하면 DB 인스턴스에 연결할 때 암호 없이 인증할 수 있습니다. 대신에 인증 토큰을 사용합니다. 자세한 내용은 MariaDB, MySQL 및 PostgreSQL IAM 데이터베이스 인증을 참조하세요.

IAM 데이터베이스 인증은 다음 엔진에서 사용할 수 없습니다.

- RDS for Db2
- RDS for Oracle
- RDS for SQL Server

주제

- RDS for MariaDB를 사용한 IAM 데이터베이스 인증
- RDS for MariaDB를 사용한 IAM 데이터베이스 인증
- RDS for PostgreSQL을 위한 IAM 데이터베이스 인증

RDS for MariaDB를 사용한 IAM 데이터베이스 인증

다음 리전 및 엔진 버전을 RDS for MariaDB를 사용한 IAM 데이터베이스 인증에 사용할 수 있습니다.

리전	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
미국 동부(버지니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
미국 동부(오하이오)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
미국 서부(캘리포니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
미국 서부(오리건)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
아프리카(케이프타운)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
아시아 태평양(홍콩)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
아시아 태평양(하이데라바드)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(자카르타)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
아시아 태평양(말레이시아)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(멜버른)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(뭄바이)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음

리전	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
아시아 태평양(오사카)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
아시아 태평양(서울)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
아시아 태평양(싱가포르)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
아시아 태평양(시드니)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
아시아 태평양(도쿄)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
캐나다(중부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
캐나다 서부(캘거리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
중국(베이징)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
중국(닝샤)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
유럽(프랑크푸르트)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
유럽(아일랜드)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
유럽(런던)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음

리전	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
유럽(밀라노)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
유럽(파리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
유럽(스페인)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
유럽(스톡홀름)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
유럽(취리히)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
이스라엘(텔아비브)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
중동(바레인)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
중동(UAE)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
남아메리카(상파울루)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
AWS GovCloud(미국 동부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
AWS GovCloud(미국 서부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음

RDS for MariaDB를 사용한 IAM 데이터베이스 인증

RDS for MySQL을 사용하는 IAM 데이터베이스 인증은 다음 버전의 경우 모든 리전에서 사용할 수 있습니다.

- RDS for MySQL 8.4 - 사용 가능한 모든 버전
- RDS for MySQL 8.0 - 사용 가능한 모든 버전
- RDS for MySQL 5.7 - 사용 가능한 모든 버전

RDS for PostgreSQL을 위한 IAM 데이터베이스 인증

RDS for PostgreSQL을 사용하는 IAM 데이터베이스 인증은 다음 버전의 경우 모든 리전에서 사용할 수 있습니다.

- RDS for PostgreSQL 17 – 사용 가능한 모든 버전
- RDS for PostgreSQL 16 – 사용 가능한 모든 버전
- RDS for PostgreSQL 15 – 사용 가능한 모든 버전
- RDS for PostgreSQL 14 – 사용 가능한 모든 버전
- RDS for PostgreSQL 13 – 사용 가능한 모든 버전
- RDS for PostgreSQL 12 – 사용 가능한 모든 버전
- RDS for PostgreSQL 11 – 사용 가능한 모든 버전
- RDS for PostgreSQL 10 – 사용 가능한 모든 버전

Amazon RDS에서 Kerberos 인증을 지원하는 리전 및 DB 엔진

Amazon RDS의 Kerberos 인증을 사용하면 Kerberos 및 Microsoft Active Directory를 통해 데이터베이스 사용자의 외부 인증을 사용할 수 있습니다. Kerberos 및 Active Directory는 데이터베이스 사용자에게 SSO(Single Sign-On) 및 중앙 집중식 인증의 이점을 제공합니다.

Kerberos 인증은 다음 엔진에서 사용할 수 없습니다.

- RDS for MariaDB

대부분의 AWS 리전은 AWS 계정에 기본적으로 활성화되어 있지만 특정 리전은 수동으로 선택한 경우에만 활성화됩니다. 이러한 리전을 옵트인 리전이라고 합니다. 반대로 AWS 계정이 생성되자마자 기본적으로 활성화되는 리전은 상업 리전 또는 줄여서 리전이라고 합니다. 옵트인 리전의 경

우 `directoryservice.rds.region_name.amazonaws.com` 양식의 리전화된 서비스 보안 주체를 사용해야 합니다. 예를 들어 아프리카(케이프타운)의 경우 신뢰 정책에 서비스 보안 주체(`directoryservice.rds.region-af-south-1.amazonaws.com`)를 추가해야 합니다. 자세한 내용은 Kerberos 인증 단원을 참조하십시오.

주제

- RDS for Db2를 사용하는 Kerberos 인증
- RDS for MySQL를 사용하는 Kerberos 인증
- RDS for Oracle을 사용하는 Kerberos 인증
- RDS for PostgreSQL를 사용하는 Kerberos 인증 사용
- RDS for MySQL를 사용하는 Kerberos 인증

RDS for Db2를 사용하는 Kerberos 인증

다음 리전 및 엔진 버전을 RDS for Db2를 사용한 Kerberos 인증에 사용할 수 있습니다.

리전	RDS for Db2 11.5
미국 동부(오하이오)	모든 버전
미국 동부(버지니아 북부)	모든 버전
미국 서부(캘리포니아 북부)	모든 버전
미국 서부(오레곤)	모든 버전
아프리카(케이프타운)	사용할 수 없음
아시아 태평양(홍콩)	사용할 수 없음
아시아 태평양(하이데라바드)	사용할 수 없음
아시아 태평양(자카르타)	사용할 수 없음
아시아 태평양(멜버른)	사용할 수 없음
아시아 태평양(뭄바이)	모든 버전
아시아 태평양(오사카)	사용할 수 없음

리전	RDS for Db2 11.5
아시아 태평양(서울)	모든 버전
아시아 태평양(싱가포르)	모든 버전
아시아 태평양(시드니)	모든 버전
아시아 태평양(도쿄)	모든 버전
캐나다(중부)	모든 버전
캐나다 서부(캘거리)	사용할 수 없음
중국(베이징)	모든 버전
중국(닝샤)	모든 버전
유럽(프랑크푸르트)	모든 버전
유럽(아일랜드)	모든 버전
유럽(런던)	모든 버전
유럽(밀라노)	사용할 수 없음
유럽(파리)	사용할 수 없음
유럽(스페인)	사용할 수 없음
유럽(스톡홀름)	모든 버전
유럽(취리히)	사용할 수 없음
이스라엘(텔아비브)	사용할 수 없음
중동(바레인)	사용할 수 없음
중동(UAE)	사용할 수 없음
남아메리카(상파울루)	모든 버전

리전	RDS for Db2 11.5
AWS GovCloud(미국 동부)	사용할 수 없음
AWS GovCloud(미국 서부)	사용할 수 없음

RDS for MySQL를 사용하는 Kerberos 인증

다음 리전 및 엔진 버전을 RDS for MySQL을 사용한 Kerberos 인증에 사용할 수 있습니다.

리전	RDS for MySQL 8.0	RDS for MySQL 5.7	RDS for MySQL 5.6
미국 동부(오하이오)	모든 버전	모든 버전	모든 버전
미국 동부(버지니아 북부)	모든 버전	모든 버전	모든 버전
미국 서부(캘리포니아 북부)	모든 버전	모든 버전	모든 버전
미국 서부(오레곤)	모든 버전	모든 버전	모든 버전
아프리카(케이프타운)	모든 버전	모든 버전	모든 버전
아시아 태평양(홍콩)	모든 버전	모든 버전	모든 버전
아시아 태평양(하이데라바드)	모든 버전	모든 버전	모든 버전
아시아 태평양(자카르타)	모든 버전	모든 버전	모든 버전
아시아 태평양(멜버른)	모든 버전	모든 버전	모든 버전
아시아 태평양(뭄바이)	모든 버전	모든 버전	모든 버전
아시아 태평양(오사카)	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(서울)	모든 버전	모든 버전	모든 버전

리전	RDS for MySQL 8.0	RDS for MySQL 5.7	RDS for MySQL 5.6
아시아 태평양(싱가포르)	모든 버전	모든 버전	모든 버전
아시아 태평양(시드니)	모든 버전	모든 버전	모든 버전
아시아 태평양(도쿄)	모든 버전	모든 버전	모든 버전
캐나다(중부)	모든 버전	모든 버전	모든 버전
캐나다 서부(캘거리)	사용할 수 없음	사용할 수 없음	사용할 수 없음
중국(베이징)	모든 버전	모든 버전	모든 버전
중국(닝샤)	모든 버전	모든 버전	모든 버전
유럽(프랑크푸르트)	모든 버전	모든 버전	모든 버전
유럽(아일랜드)	모든 버전	모든 버전	모든 버전
유럽(런던)	모든 버전	모든 버전	모든 버전
유럽(밀라노)	모든 버전	모든 버전	모든 버전
유럽(파리)	모든 버전	모든 버전	모든 버전
유럽(스페인)	모든 버전	모든 버전	모든 버전
유럽(스톡홀름)	모든 버전	모든 버전	모든 버전
유럽(취리히)	모든 버전	모든 버전	모든 버전
이스라엘(텔아비브)	모든 버전	모든 버전	모든 버전
중동(바레인)	모든 버전	모든 버전	모든 버전
중동(UAE)	모든 버전	모든 버전	모든 버전
남아메리카(상파울루)	모든 버전	모든 버전	모든 버전

리전	RDS for MySQL 8.0	RDS for MySQL 5.7	RDS for MySQL 5.6
AWS GovCloud(미국 동부)	사용할 수 없음	사용할 수 없음	사용할 수 없음
AWS GovCloud(미국 서부)	사용할 수 없음	사용할 수 없음	사용할 수 없음

RDS for Oracle을 사용하는 Kerberos 인증

다음 리전 및 엔진 버전을 RDS for Oracle을 사용한 Kerberos 인증에 사용할 수 있습니다.

리전	RDS for Oracle 21c	RDS for Oracle 19c
미국 동부(오하이오)	모든 버전	모든 버전
미국 동부(버지니아 북부)	모든 버전	모든 버전
미국 서부(캘리포니아 북부)	모든 버전	모든 버전
미국 서부(오레곤)	모든 버전	모든 버전
아프리카(케이프타운) (옵트인 리전)	모든 버전	모든 버전
아시아 태평양(홍콩) (옵트인 리전)	모든 버전	모든 버전
아시아 태평양(하이데라바드) (옵트인 리전)	모든 버전	모든 버전
아시아 태평양(자카르타) (옵트인 리전)	모든 버전	모든 버전
아시아 태평양(멜버른) (옵트인 리전)	모든 버전	모든 버전
아시아 태평양(뭄바이)	모든 버전	모든 버전

리전	RDS for Oracle 21c	RDS for Oracle 19c
아시아 태평양(오사카)	사용할 수 없음	사용할 수 없음
아시아 태평양(서울)	모든 버전	모든 버전
아시아 태평양(싱가포르)	모든 버전	모든 버전
아시아 태평양(시드니)	모든 버전	모든 버전
아시아 태평양(도쿄)	모든 버전	모든 버전
캐나다(중부)	모든 버전	모든 버전
캐나다 서부(캘거리)	사용할 수 없음	사용할 수 없음
중국(베이징)	사용할 수 없음	사용할 수 없음
중국(닝샤)	사용할 수 없음	사용할 수 없음
유럽(프랑크푸르트)	모든 버전	모든 버전
유럽(아일랜드)	모든 버전	모든 버전
유럽(런던)	모든 버전	모든 버전
유럽(밀라노) (옵트인 리전)	모든 버전	모든 버전
유럽(파리)	사용할 수 없음	사용할 수 없음
유럽(스페인) (옵트인 리전)	모든 버전	모든 버전
유럽(스톡홀름)	모든 버전	모든 버전
유럽(취리히) (옵트인 리전)	모든 버전	모든 버전
이스라엘(텔아비브) (옵트인 리전)	모든 버전	모든 버전
중동(바레인) (옵트인 리전)	모든 버전	모든 버전
중동(UAE) (옵트인 리전)	모든 버전	모든 버전

리전	RDS for Oracle 21c	RDS for Oracle 19c
남아메리카(상파울루)	모든 버전	모든 버전
AWS GovCloud(미국 동부)	모든 버전	모든 버전
AWS GovCloud(미국 서부)	모든 버전	모든 버전

RDS for PostgreSQL를 사용하는 Kerberos 인증 사용

다음 리전 및 엔진 버전을 RDS for PostgreSQL을 사용한 Kerberos 인증에 사용할 수 있습니다.

리전	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
미국 동부(오하이오)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
미국 동부(버지니아 북부)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
미국 서부(캘리포니아 북부)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
미국 서부(오레곤)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
아프리카(케이프타운)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
아시아 태평양(홍콩)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전

리전	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
아시아 태평양(하이데라바드)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
아시아 태평양(자카르타)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
아시아 태평양(멜버른)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
아시아 태평양(뭄바이)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
아시아 태평양(오사카)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(서울)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
아시아 태평양(싱가포르)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
아시아 태평양(시드니)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
아시아 태평양(도쿄)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전

리전	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
캐나다(중부)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
캐나다 서부(캘거리)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
중국(베이징)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
중국(닝샤)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
유럽(프랑크푸르트)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
유럽(아일랜드)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
유럽(런던)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
유럽(밀라노)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
유럽(파리)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
유럽(스페인)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
유럽(스톡홀름)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전

리전	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
유럽(취리히)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
이스라엘(텔아비브)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
중동(바레인)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
중동(UAE)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
남아메리카(상파울루)	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전	모든 버전
AWS GovCloud(미국 동부)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
AWS GovCloud(미국 서부)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음

RDS for MySQL를 사용하는 Kerberos 인증

다음 리전 및 엔진 버전을 RDS for SQL Server를 사용한 Kerberos 인증에 사용할 수 있습니다.

리전	RDS for SQL Server 2022	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
미국 동부(오하이오)	모든 버전	모든 버전	모든 버전	모든 버전
미국 동부(버지니아 북부)	모든 버전	모든 버전	모든 버전	모든 버전
미국 서부(캘리포니아 북부)	모든 버전	모든 버전	모든 버전	모든 버전
미국 서부(오레곤)	모든 버전	모든 버전	모든 버전	모든 버전
아프리카(케이프타운)	모든 버전	모든 버전	모든 버전	모든 버전
아시아 태평양(홍콩)	모든 버전	모든 버전	모든 버전	모든 버전
아시아 태평양(하이데라바드)	모든 버전	모든 버전	모든 버전	모든 버전
아시아 태평양(자카르타)	모든 버전	모든 버전	모든 버전	모든 버전
아시아 태평양(멜버른)	모든 버전	모든 버전	모든 버전	모든 버전
아시아 태평양(뭄바이)	모든 버전	모든 버전	모든 버전	모든 버전
아시아 태평양(오사카)	모든 버전	모든 버전	모든 버전	모든 버전
아시아 태평양(서울)	모든 버전	모든 버전	모든 버전	모든 버전

리전	RDS for SQL Server 2022	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
아시아 태평양(싱가포르)	모든 버전	모든 버전	모든 버전	모든 버전
아시아 태평양(시드니)	모든 버전	모든 버전	모든 버전	모든 버전
아시아 태평양(도쿄)	모든 버전	모든 버전	모든 버전	모든 버전
캐나다(중부)	모든 버전	모든 버전	모든 버전	모든 버전
캐나다 서부(캘거리)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
중국(베이징)	모든 버전	모든 버전	모든 버전	모든 버전
중국(닝샤)	모든 버전	모든 버전	모든 버전	모든 버전
유럽(프랑크푸르트)	모든 버전	모든 버전	모든 버전	모든 버전
유럽(아일랜드)	모든 버전	모든 버전	모든 버전	모든 버전
유럽(런던)	모든 버전	모든 버전	모든 버전	모든 버전
유럽(밀라노)	모든 버전	모든 버전	모든 버전	모든 버전
유럽(파리)	모든 버전	모든 버전	모든 버전	모든 버전
유럽(스페인)	모든 버전	모든 버전	모든 버전	모든 버전
유럽(스톡홀름)	모든 버전	모든 버전	모든 버전	모든 버전
유럽(취리히)	모든 버전	모든 버전	모든 버전	모든 버전
이스라엘(텔아비브)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음

리전	RDS for SQL Server 2022	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
중동(바레인)	모든 버전	모든 버전	모든 버전	모든 버전
중동(UAE)	모든 버전	모든 버전	모든 버전	모든 버전
남아메리카(상파울루)	모든 버전	모든 버전	모든 버전	모든 버전
AWS GovCloud(미국 동부)	모든 버전	모든 버전	모든 버전	모든 버전
AWS GovCloud(미국 서부)	모든 버전	모든 버전	모든 버전	모든 버전

Amazon RDS에서 다중 AZ DB 클러스터를 지원하는 리전 및 DB 엔진

Amazon RDS의 다중 AZ DB 클러스터 배포는 2개의 읽기 가능한 대기 DB 인스턴스가 있는 Amazon RDS의 고가용성 배포 모드를 제공합니다. 다중 AZ DB 클러스터에는 동일한 리전에 있는 세 개의 개별 가용 영역에 라이터 DB 인스턴스와 두 개의 리더 DB 인스턴스가 있습니다. 다중 AZ DB 클러스터는 다중 AZ DB 인스턴스 배포에 비해 고가용성, 높은 읽기 워크로드 용량, 낮은 쓰기 대기 시간을 지원합니다. 자세한 내용은 Amazon RDS용 다중 AZ DB 클러스터 배포을 참조하세요.

다중 AZ DB 클러스터는 다음 엔진에서 사용할 수 없습니다.

- RDS for Db2
- RDS for MariaDB
- RDS for Oracle
- RDS for SQL Server

주제

- RDS for MySQL를 사용하는 다중 AZ DB 클러스터
- RDS for MySQL를 사용하는 다중 AZ DB 클러스터

RDS for MySQL를 사용하는 다중 AZ DB 클러스터

다음 리전 및 엔진 버전을 RDS for MySQL을 사용하는 다중 AZ DB 클러스터에 사용할 수 있습니다.

리전	RDS for MySQL 8.4	RDS for MySQL 8.0
미국 동부(버지니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 동부(오하이오)	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(캘리포니아 북부)	사용할 수 없음	사용할 수 없음
미국 서부(오리건)	사용 가능한 모든 버전	사용 가능한 모든 버전
아프리카(케이프타운)	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(홍콩)	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(하이데라바드)	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(자카르타)	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(말레이시아)	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(멜버른)	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(뭄바이)	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(오사카)	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(서울)	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(싱가포르)	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(시드니)	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for MySQL 8.4	RDS for MySQL 8.0
아시아 태평양(도쿄)	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다(중부)	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다(중부)	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다 서부(캘거리)	사용 가능한 모든 버전	사용 가능한 모든 버전
중국(베이징)	사용 가능한 모든 버전	사용 가능한 모든 버전
중국(닝샤)	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(프랑크푸르트)	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(아일랜드)	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(런던)	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(밀라노)	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(파리)	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(스페인)	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(스톡홀름)	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(취리히)	사용 가능한 모든 버전	사용 가능한 모든 버전
이스라엘(텔아비브)	사용 가능한 모든 버전	사용 가능한 모든 버전
중동(바레인)	사용 가능한 모든 버전	사용 가능한 모든 버전
중동(UAE)	사용 가능한 모든 버전	사용 가능한 모든 버전
남아메리카(상파울루)	사용 가능한 모든 버전	사용 가능한 모든 버전
AWS GovCloud(미국 동부)	사용할 수 없음	사용할 수 없음

리전	RDS for MySQL 8.4	RDS for MySQL 8.0
AWS GovCloud(미국 서부)	사용할 수 없음	사용할 수 없음

AWS CLI를 사용하여 주어진 DB 인스턴스 클래스의 리전에 사용 가능한 버전을 나열할 수 있습니다. DB 인스턴스 클래스를 변경하여 사용 가능한 엔진 버전을 표시합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-orderable-db-instance-options \
--engine mysql \
--db-instance-class db.r5d.large \
--query '*[][?SupportsClusters == `true`].[EngineVersion]'  \
--output text
```

Windows의 경우:

```
aws rds describe-orderable-db-instance-options ^
--engine mysql ^
--db-instance-class db.r5d.large ^
--query "*[][?SupportsClusters == `true`].[EngineVersion]"  ^
--output text
```

RDS for MySQL를 사용하는 다중 AZ DB 클러스터

다음 리전 및 엔진 버전을 RDS for PostgreSQL을 사용하는 다중 AZ DB 클러스터에 사용할 수 있습니다.

리전	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13
미국 동부(버지니아 북부)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상

리전	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13
미국 동부(오하이오)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
미국 서부(캘리포니아 북부)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
미국 서부(오리건)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
아프리카(케이프타운)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
아시아 태평양(홍콩)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
아시아 태평양(하이데라바드)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
아시아 태평양(자카르타)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
아시아 태평양(말레이시아)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
아시아 태평양(멜버른)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상

리전	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13
아시아 태평양(뭄바이)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
아시아 태평양(오사카)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
아시아 태평양(서울)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
아시아 태평양(싱가포르)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
아시아 태평양(시드니)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
아시아 태평양(도쿄)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
캐나다(중부)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
캐나다 서부(캘거리)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
중국(베이징)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상

리전	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13
중국(닝샤)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
유럽(프랑크푸르트)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
유럽(아일랜드)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
유럽(런던)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
유럽(밀라노)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
유럽(파리)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
유럽(스페인)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
유럽(스톡홀름)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
유럽(취리히)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상

리전	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13
이스라엘(텔아비브)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
중동(바레인)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
중동(UAE)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
남아메리카(상파울루)	모든 PostgreSQL 17 버전	모든 PostgreSQL 16 버전	모든 PostgreSQL 15 버전	버전 14.5 이상	버전 13.4 및 버전 13.7 이상
AWS GovCloud(미국 동부)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
AWS GovCloud(미국 서부)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음

AWS CLI를 사용하여 주어진 DB 인스턴스 클래스의 리전에 사용 가능한 버전을 나열할 수 있습니다. DB 인스턴스 클래스를 변경하여 사용 가능한 엔진 버전을 표시합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-orderable-db-instance-options \
--engine postgres \
--db-instance-class db.r5d.large \
--query '*[]|[?SupportsClusters == `true`].[EngineVersion]'  \
--output text
```

Windows의 경우:

```
aws rds describe-orderable-db-instance-options ^
--engine postgres ^
--db-instance-class db.r5d.large ^
--query "*[]|[?SupportsClusters == `true`].[EngineVersion]"  ^
--output text
```

Amazon RDS에서 성능 개선 도우미를 지원하는 리전 및 DB 엔진

> **Important**
>
> AWS는 Performance Insights의 수명 종료일을 2025년 11월 30일로 발표했습니다. 이 날짜 이후에는 Amazon RDS가 더 이상 Performance Insights 콘솔 환경, 유연한 보존 기간(1~24개월) 및 연결 요금을 지원하지 않습니다. Performance Insights API는 요금 변경 없이 계속 제공됩니다. Performance Insights API 비용은 CloudWatch Database Insights 비용과 함께 AWS 청구서에 표시됩니다.
>
> Performance Insights의 유료 티어를 사용하는 DB 인스턴스를 2025년 11월 30일 이전에 Database Insights의 고급 모드로 업그레이드하는 것이 좋습니다. Database Insights의 고급 모드로 업그레이드하는 방법에 대한 자세한 내용은 Amazon RDS에 Database Insights의 고급 모드 켜기 섹션을 참조하세요.
>
> 조치를 취하지 않으면 Performance Insights를 사용하는 DB 인스턴스가 기본적으로 Database Insights의 표준 모드를 사용하게 됩니다. Database Insights의 표준 모드를 사용하는 경우 7일이 지난 성능 데이터 기록에 액세스할 수 없으며 Amazon RDS 콘솔에서 실행 계획 및 온디맨드 분석 기능을 사용하지 못할 수 있습니다. 2025년 11월 30일 이후에는 Database Insights의 고급 모드만 실행 계획과 온디맨드 분석을 지원합니다.
>
> CloudWatch Database Insights를 사용하면 데이터베이스 플릿의 데이터베이스 로드를 모니터링하며 대규모로 성능을 분석하고 문제를 해결할 수 있습니다. Database Insights에 대한 자세한 내용은 CloudWatch Database Insights를 사용하여 Amazon RDS 데이터베이스 모니터링 섹션을 참조하세요. 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

Amazon RDS의 성능 개선 도우미는 기존 Amazon RDS 모니터링 기능을 확장하여 데이터베이스 성능을 설명하고 분석하는 데 도움을 줍니다. 성능 개선 도우미 대시보드를 사용하여 Amazon RDS DB 인스턴스의 데이터베이스 로드를 시각화할 수 있습니다. 대기, SQL 문, 호스트 또는 사용자별로 로드를 필터링할 수도 있습니다. 자세한 내용은 성능 개선 도우미를 통한 Amazon RDS 모니터링 섹션을 참조하세요.

성능 개선 도우미는 RDS for Db2를 제외한 모든 RDS DB 엔진에서 사용할 수 있습니다.

지원되는 DB 엔진의 경우 성능 개선 도우미는 사용 가능한 모든 엔진 버전과 함께, 그리고 모든 AWS 리전에서 사용할 수 있습니다.

성능 개선 도우미에 대한 리전, DB 엔진 및 인스턴스 클래스 지원 정보는 성능 개선 도우미 기능에 대한 Amazon RDS DB 엔진, 리전 및 인스턴스 클래스 지원 섹션을 참조하세요.

RDS Custom을 지원하는 리전 및 DB 엔진

Amazon RDS Custom은 데이터베이스 관리 작업 및 운영을 자동화합니다. RDS Custom을 사용하면 데이터베이스 관리자로서 데이터베이스 환경 및 운영 체제에 액세스하고 사용자 지정할 수 있습니다. RDS Custom을 사용하면 레거시, 커스텀 및 패키지 애플리케이션의 요구 사항에 맞게 커스터마이징할 수 있습니다. 자세한 내용은 Amazon RDS Custom 섹션을 참조하세요.

RDS Custom은 다음 DB 엔진에서만 지원됩니다.

주제

- RDS Custom for Oracle을 지원하는 리전 및 DB 엔진
- RDS Custom for SQL Server를 지원하는 리전 및 DB 엔진

RDS Custom for Oracle을 지원하는 리전 및 DB 엔진

다음 리전 및 엔진 버전을 RDS for SQL Oracle에 사용할 수 있습니다.

리전	Oracle Database 19c	Oracle Database 18c	Oracle Database 12c
미국 동부(버지니아 북부)	19c(2021년 1월 출시) 이상의 RU/RUR	18c(2021년 1월 출시) 이상의 RU/RUR	12.1 및 12.2(2021년 1월 출시) 이상의 RU/RUR
미국 동부(오하이오)	19c(2021년 1월 출시) 이상의 RU/RUR	18c(2021년 1월 출시) 이상의 RU/RUR	12.1 및 12.2(2021년 1월 출시) 이상의 RU/RUR
미국 서부(캘리포니아 북부)	사용할 수 없음	사용할 수 없음	사용할 수 없음

리전	Oracle Database 19c	Oracle Database 18c	Oracle Database 12c
미국 서부(오리건)	19c(2021년 1월 출시) 이상의 RU/RUR	18c(2021년 1월 출시) 이상의 RU/RUR	12.1 및 12.2(2021년 1월 출시) 이상의 RU/RUR
아프리카(케이프타운)	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(홍콩)	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(자카르타)	19c(2021년 1월 출시) 이상의 RU/RUR	18c(2021년 1월 출시) 이상의 RU/RUR	12.1 및 12.2(2021년 1월 출시) 이상의 RU/RUR
아시아 태평양(말레이시아)	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(멜버른)	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(뭄바이)	19c(2021년 1월 출시) 이상의 RU/RUR	18c(2021년 1월 출시) 이상의 RU/RUR	12.1 및 12.2(2021년 1월 출시) 이상의 RU/RUR
아시아 태평양(오사카)	19c(2021년 1월 출시) 이상의 RU/RUR	18c(2021년 1월 출시) 이상의 RU/RUR	12.1 및 12.2(2021년 1월 출시) 이상의 RU/RUR
아시아 태평양(서울)	19c(2021년 1월 출시) 이상의 RU/RUR	18c(2021년 1월 출시) 이상의 RU/RUR	12.1 및 12.2(2021년 1월 출시) 이상의 RU/RUR
아시아 태평양(싱가포르)	19c(2021년 1월 출시) 이상의 RU/RUR	18c(2021년 1월 출시) 이상의 RU/RUR	12.1 및 12.2(2021년 1월 출시) 이상의 RU/RUR

리전	Oracle Database 19c	Oracle Database 18c	Oracle Database 12c
아시아 태평양(시드니)	19c(2021년 1월 출시) 이상의 RU/RUR	18c(2021년 1월 출시) 이상의 RU/RUR	12.1 및 12.2(2021년 1월 출시) 이상의 RU/RUR
아시아 태평양(도쿄)	19c(2021년 1월 출시) 이상의 RU/RUR	18c(2021년 1월 출시) 이상의 RU/RUR	12.1 및 12.2(2021년 1월 출시) 이상의 RU/RUR
캐나다(중부)	19c(2021년 1월 출시) 이상의 RU/RUR	18c(2021년 1월 출시) 이상의 RU/RUR	12.1 및 12.2(2021년 1월 출시) 이상의 RU/RUR
캐나다 서부(캘거리)	사용할 수 없음	사용할 수 없음	사용할 수 없음
중국(베이징)	사용할 수 없음	사용할 수 없음	사용할 수 없음
중국(닝샤)	사용할 수 없음	사용할 수 없음	사용할 수 없음
유럽(프랑크푸르트)	19c(2021년 1월 출시) 이상의 RU/RUR	18c(2021년 1월 출시) 이상의 RU/RUR	12.1 및 12.2(2021년 1월 출시) 이상의 RU/RUR
유럽(아일랜드)	19c(2021년 1월 출시) 이상의 RU/RUR	18c(2021년 1월 출시) 이상의 RU/RUR	12.1 및 12.2(2021년 1월 출시) 이상의 RU/RUR
유럽(런던)	19c(2021년 1월 출시) 이상의 RU/RUR	18c(2021년 1월 출시) 이상의 RU/RUR	12.1 및 12.2(2021년 1월 출시) 이상의 RU/RUR
유럽(밀라노)	19c(2021년 1월 출시) 이상의 RU/RUR	18c(2021년 1월 출시) 이상의 RU/RUR	12.1 및 12.2(2021년 1월 출시) 이상의 RU/RUR

리전	Oracle Database 19c	Oracle Database 18c	Oracle Database 12c
유럽(파리)	19c(2021년 1월 출시) 이상의 RU/RUR	18c(2021년 1월 출시) 이상의 RU/RUR	12.1 및 12.2(2021년 1월 출시) 이상의 RU/RUR
유럽(스톡홀름)	19c(2021년 1월 출시) 이상의 RU/RUR	18c(2021년 1월 출시) 이상의 RU/RUR	12.1 및 12.2(2021년 1월 출시) 이상의 RU/RUR
이스라엘(텔아비브)	사용할 수 없음	사용할 수 없음	사용할 수 없음
중동(바레인)	사용할 수 없음	사용할 수 없음	사용할 수 없음
중동(UAE)	19c(2021년 1월 출시) 이상의 RU/RUR	18c(2021년 1월 출시) 이상의 RU/RUR	12.1 및 12.2(2021년 1월 출시) 이상의 RU/RUR
남아메리카(상파울루)	19c(2021년 1월 출시) 이상의 RU/RUR	18c(2021년 1월 출시) 이상의 RU/RUR	12.1 및 12.2(2021년 1월 출시) 이상의 RU/RUR
AWS GovCloud(미국 동부)	19c(2021년 1월 출시) 이상의 RU/RUR	18c(2021년 1월 출시) 이상의 RU/RUR	12.1 및 12.2(2021년 1월 출시) 이상의 RU/RUR
AWS GovCloud(미국 서부)	19c(2021년 1월 출시) 이상의 RU/RUR	18c(2021년 1월 출시) 이상의 RU/RUR	12.1 및 12.2(2021년 1월 출시) 이상의 RU/RUR

RDS Custom for SQL Server를 지원하는 리전 및 DB 엔진

RDS 제공 엔진 버전(RPEV) 또는 사용자 지정 엔진 버전(CEV) 중 하나를 사용하여 RDS Custom for SQL Server를 배포할 수 있습니다.

- RPEV를 사용하는 경우 기본 Amazon Machine Image(AMI) 와 SQL Server 설치가 포함됩니다. 운영 체제(OS)를 사용자 지정하거나 수정하는 경우 패치 적용, 스냅샷 복원 또는 자동 복구 중에는 변경 사항이 계속 반영되지 않을 수 있습니다.

- CEV를 사용하는 경우 사전 설치된 Microsoft SQL Server 또는 자체 미디어를 사용하여 설치한 SQL Server가 포함된 자체 AMI를 선택할 수 있습니다. AWS 제공 CEV를 사용하는 경우 RDS Custom for SQL Server에서 지원하는 누적 업데이트(CU)가 있는 AWS 제공 최신 Amazon EC2 이미지(AMI)를 선택합니다. CEV의 경우 기업 요구 사항에 맞게 OS 및 SQL Server의 구성을 사용자 지정할 수 있습니다.

다음 AWS 리전 및 DB 엔진 버전을 RDS Custom for SQL Server에 사용할 수 있습니다. 엔진 버전 지원은 RPEV, AWS 제공 CEV 또는 고객 제공 CEV 중 무엇과 함께 RDS Custom for SQL Server를 사용하는지에 따라 달라집니다.

리전	RPEV	AWS 제공 CEV	고객 제공 CEV
미국 동부(버지니아 북부)	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32
미국 동부(오하이오)	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-

리전	RPEV	AWS 제공 CEV	고객 제공 CEV
	GDR, CU29-GDR, CU30 및 CU32	GDR, CU29-GDR, CU30 및 CU32	GDR, CU29-GDR, CU30 및 CU32
미국 서부(캘리포니아 북부)	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32
미국 서부(오리건)	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32

리전	RPEV	AWS 제공 CEV	고객 제공 CEV
아프리카(케이프타운)	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32
아시아 태평양(홍콩)	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(하이데라바드)	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(자카르타)	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(말레이시아)	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(멜버른)	사용할 수 없음	사용할 수 없음	사용할 수 없음

리전	RPEV	AWS 제공 CEV	고객 제공 CEV
아시아 태평양(뭄바이)	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32
아시아 태평양(오사카)	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32

리전	RPEV	AWS 제공 CEV	고객 제공 CEV
아시아 태평양(서울)	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32
아시아 태평양(싱가포르)	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32

리전	RPEV	AWS 제공 CEV	고객 제공 CEV
아시아 태평양(시드니)	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32
아시아 태평양(도쿄)	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32

리전	RPEV	AWS 제공 CEV	고객 제공 CEV
캐나다(중부)	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32
캐나다 서부(캘거리)	사용할 수 없음	사용할 수 없음	사용할 수 없음
중국(베이징)	사용할 수 없음	사용할 수 없음	사용할 수 없음
중국(닝샤)	사용할 수 없음	사용할 수 없음	사용할 수 없음
유럽(프랑크푸르트)	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32

리전	RPEV	AWS 제공 CEV	고객 제공 CEV
유럽(아일랜드)	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32
유럽(런던)	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32
유럽(밀라노)	사용할 수 없음	사용할 수 없음	사용할 수 없음

리전	RPEV	AWS 제공 CEV	고객 제공 CEV
유럽(파리)	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32
유럽(스페인)	사용할 수 없음	사용할 수 없음	사용할 수 없음
유럽(스톡홀름)	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32
유럽(취리히)	사용할 수 없음	사용할 수 없음	사용할 수 없음
이스라엘(텔아비브)	사용할 수 없음	사용할 수 없음	사용할 수 없음
중동(바레인)	사용할 수 없음	사용할 수 없음	사용할 수 없음

리전	RPEV	AWS 제공 CEV	고객 제공 CEV
중동(UAE)	사용할 수 없음	사용할 수 없음	사용할 수 없음
남아메리카(상파울루)	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32	SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19. SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32
AWS GovCloud(미국 동부)	사용할 수 없음	사용할 수 없음	사용할 수 없음
AWS GovCloud(미국 서부)	사용할 수 없음	사용할 수 없음	사용할 수 없음

Amazon RDS 프록시를 지원하는 리전 및 DB 엔진

Amazon RDS 프록시는 설정된 데이터베이스 연결을 풀링하고 공유하여 응용 프로그램의 확장성을 높여주는 완전히 관리되는 고가용성 데이터베이스 프록시입니다. 자세한 내용은 Amazon RDS Proxy 섹션을 참조하세요.

RDS 프록시는 다음 엔진에서는 사용할 수 없습니다.

- RDS for Db2
- RDS for Oracle

주제

- RDS for MariaDB를 지원하는 RDS 프록시

- RDS for MySQL을 지원하는 RDS 프록시
- RDS for PostgreSQL 지원하는 RDS 프록시
- RDS for SQL Server를 사용하는 RDS 프록시

RDS for MariaDB를 지원하는 RDS 프록시

다음 리전 및 엔진 버전을 RDS for MariaDB를 사용하는 RDS 프록시에 사용할 수 있습니다.

리전	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
미국 동부(버지니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 동부(오하이오)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(캘리포니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(오리건)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아프리카(케이프타운)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(홍콩)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(하이데라바드)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(자카르타)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
아시아 태평양(말레이시아)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(멜버른)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(뭄바이)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(오사카)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(서울)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(싱가포르)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(시드니)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(태국)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(도쿄)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다(중부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다 서부(캘거리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
중국(베이징)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중국(닝샤)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(프랑크푸르트)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(아일랜드)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(런던)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(밀라노)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(파리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(스페인)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(스톡홀름)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(취리히)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
이스라엘(텔아비브)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중동(바레인)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
중동(UAE)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
멕시코(중부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
남아메리카(상파울루)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
AWS GovCloud(미국 동부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
AWS GovCloud(미국 서부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

RDS for MySQL을 지원하는 RDS 프록시

다음 리전 및 엔진 버전을 RDS for MySQL을 사용하는 RDS 프록시에 사용할 수 있습니다.

리전	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7
미국 동부(버지니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 동부(오하이오)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(캘리포니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(오리건)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아프리카(케이프타운)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7
아시아 태평양(홍콩)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(하이데라바드)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(자카르타)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(말레이시아)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(멜버른)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(뭄바이)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(오사카)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(서울)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(싱가포르)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(시드니)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(태국)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(도쿄)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다(중부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다 서부(캘거리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중국(베이징)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중국(닝샤)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(프랑크푸르트)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(아일랜드)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7
유럽(런던)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(밀라노)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(파리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(스페인)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(스톡홀름)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(취리히)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
이스라엘(텔아비브)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중동(바레인)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중동(UAE)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
멕시코(중부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
남아메리카(상파울루)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
AWS GovCloud(미국 동부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
AWS GovCloud(미국 서부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

RDS for PostgreSQL 지원하는 RDS 프록시

다음 리전 및 엔진 버전을 RDS for PostgreSQL을 사용하는 RDS 프록시에 사용할 수 있습니다.

리전	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
미국 동부(버지니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 동부(오하이오)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(캘리포니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(오리건)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아프리카(케이프타운)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(홍콩)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(하이데라바드)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
(자카르타)								
아시아 태평양 (말레이시아)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양 (멜버른)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양 (뭄바이)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양 (오사카)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양 (서울)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양 (싱가포르)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양 (시드니)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
아시아 태평양(태국)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(도쿄)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다(중부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다 서부(캘거리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중국(베이징)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중국(닝샤)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(프랑크푸르트)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(아일랜드)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(런던)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
유럽(밀라노)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(파리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(스페인)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(스톡홀름)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(취리히)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
이스라엘(텔아비브)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중동(바레인)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중동(UAE)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
멕시코(중부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
남아메리카(상파울루)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
AWS GovCloud 미국 동부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
AWS GovCloud 미국 서부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

RDS for SQL Server를 사용하는 RDS 프록시

다음 리전 및 엔진 버전을 RDS for SQL Server을 사용하는 RDS 프록시에 사용할 수 있습니다.

리전	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
미국 동부(버지니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 동부(오하이오)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(캘리포니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부(오리건)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아프리카(케이프타운)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(홍콩)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
아시아 태평양(하이데라바드)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(자카르타)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(말레이시아)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(멜버른)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(뭄바이)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(오사카)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(서울)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(싱가포르)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(시드니)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(태국)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(도쿄)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다(중부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다 서부(캘거리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중국(베이징)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중국(닝샤)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(프랑크푸르트)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(아일랜드)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

리전	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
유럽(런던)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(밀라노)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(파리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(스페인)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(스톡홀름)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(취리히)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
이스라엘(텔아비브)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중동(바레인)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
중동(UAE)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
멕시코(중부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
남아메리카(상파울루)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
AWS GovCloud(미국 동부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
AWS GovCloud(미국 서부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전

Secrets Manager와 Amazon RDS 통합을 지원하는 리전 및 DB 엔진

AWS Secrets Manager를 사용하면 데이터베이스 암호를 포함한 하드 코딩된 보안 인증 정보를 Secrets Manager에서 프로그래밍 방식으로 비밀을 검색하게 하는 API 호출로 바꿀 수 있습니다. Secrets Manager에 대한 자세한 내용은 AWS Secrets Manager 사용 설명서를 참조하세요.

Amazon RDS가 Amazon RDS DB 인스턴스 또는 다중 AZ DB 클러스터용 Secrets Manager에서 마스터 사용자 암호를 관리하도록 지정할 수 있습니다. RDS는 암호를 생성하여 Secrets Manager에 저장

한 다음 정기적으로 교체합니다. 자세한 내용은 Amazon RDS 및 AWS Secrets Manager를 통한 암호 관리 단원을 참조하십시오.

Secrets Manager 통합은 모든 AWS 리전에서 사용할 수 있습니다.

Amazon RDS 제로 ETL 통합을 지원하는 리전 및 DB 엔진

RDS 제로 ETL 통합은 트랜잭션 데이터가 Amazon RDS DB 인스턴스에 기록된 후 Amazon Redshift 또는 Amazon SageMaker에서 사용할 수 있도록 하기 위한 완전관리형 솔루션입니다. 자세한 내용은 제로 ETL 통합 섹션을 참조하세요.

주제

- RDS for MySQL과의 제로 ETL 통합

RDS for MySQL과의 제로 ETL 통합

Amazon Redshift 및 Amazon SageMaker와 RDS for MySQL의 제로 ETL 통합을 사용할 수 있는 리전 및 엔진 버전은 다음과 같습니다.

리전	RDS for MySQL 8.4와 Amazon Redshift의 제로 ETL 통합	RDS for MySQL 8.0과 Amazon Redshift의 제로 ETL 통합	RDS for MySQL 8.4와 Amazon SageMaker의 제로 ETL 통합	RDS for MySQL 8.0과 Amazon SageMaker의 제로 ETL 통합
미국 동부(버지니아 북부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 동부(오하이오)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
미국 서부	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음

리전	RDS for MySQL 8.4와 Amazon Redshift의 제로 ETL 통합	RDS for MySQL 8.0과 Amazon Redshift의 제로 ETL 통합	RDS for MySQL 8.4와 Amazon SageMaker의 제로 ETL 통합	RDS for MySQL 8.0과 Amazon SageMaker의 제로 ETL 통합
(캘리포니아 북부)				
미국 서부(오리건)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아프리카(케이프타운)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
아시아 태평양(홍콩)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(하이데라바드)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음

리전	RDS for MySQL 8.4와 Amazon Redshift의 제로 ETL 통합	RDS for MySQL 8.0과 Amazon Redshift의 제로 ETL 통합	RDS for MySQL 8.4와 Amazon SageMaker의 제로 ETL 통합	RDS for MySQL 8.0과 Amazon SageMaker의 제로 ETL 통합
아시아 태평양(자카르타)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(말레이시아)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(멜버른)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
아시아 태평양(뭄바이)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
아시아 태평양(오사카)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음

리전	RDS for MySQL 8.4와 Amazon Redshift의 제로 ETL 통합	RDS for MySQL 8.0과 Amazon Redshift의 제로 ETL 통합	RDS for MySQL 8.4와 Amazon SageMaker의 제로 ETL 통합	RDS for MySQL 8.0과 Amazon SageMaker의 제로 ETL 통합
아시아 태평양(서울)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(싱가포르)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(시드니)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
아시아 태평양(도쿄)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다(중부)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
캐나다 서부(캘거리)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음

리전	RDS for MySQL 8.4와 Amazon Redshift의 제로 ETL 통합	RDS for MySQL 8.0과 Amazon Redshift의 제로 ETL 통합	RDS for MySQL 8.4와 Amazon SageMaker의 제로 ETL 통합	RDS for MySQL 8.0과 Amazon SageMaker의 제로 ETL 통합
중국(베이징)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
중국(닝샤)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
유럽(프랑크푸르트)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(아일랜드)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(런던)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(밀라노)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
유럽(파리)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
유럽(스페인)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음

리전	RDS for MySQL 8.4와 Amazon Redshift의 제로 ETL 통합	RDS for MySQL 8.0과 Amazon Redshift의 제로 ETL 통합	RDS for MySQL 8.4와 Amazon SageMaker의 제로 ETL 통합	RDS for MySQL 8.0과 Amazon SageMaker의 제로 ETL 통합
유럽(스톡홀름)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
유럽(취리히)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
이스라엘(텔아비브)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
중동(바레인)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용할 수 없음	사용할 수 없음
중동(UAE)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음
남아메리카(상파울루)	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전	사용 가능한 모든 버전
AWS GovCl 미국 동부)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음

리전	RDS for MySQL 8.4와 Amazon Redshift의 제로 ETL 통합	RDS for MySQL 8.0과 Amazon Redshift의 제로 ETL 통합	RDS for MySQL 8.4와 Amazon SageMaker의 제로 ETL 통합	RDS for MySQL 8.0과 Amazon SageMaker의 제로 ETL 통합
AWS GovCloud (미국 서부)	사용할 수 없음	사용할 수 없음	사용할 수 없음	사용할 수 없음

Amazon RDS의 엔진 네이티브 기능

Amazon RDS 데이터베이스 엔진은 가장 일반적인 엔진 기본 기능도 많이 지원합니다. 이러한 기능은 이 페이지에 나와 있는 Amazon RDS 네이티브 기능과는 다릅니다. 일부 엔진 기반 기능에는 제한된 지원 또는 제한된 권한이 있을 수 있습니다.

엔진 기반 기능에 대한 자세한 내용은 다음을 참조하세요.

- Amazon RDS for Db2 기능
- Amazon RDS에서의 MariaDB 기능 지원
- Amazon RDS에서 지원되는 MySQL 기능
- RDS for Oracle 기능
- Amazon RDS for PostgreSQL에서 지원되는 PostgreSQL 기능 작업
- Amazon RDS의 Microsoft SQL Server 기능

Amazon RDS에 대한 DB 인스턴스 결제

Amazon RDS 인스턴스는 다음 구성 요소를 기준으로 청구됩니다.

- DB 인스턴스 시간(시간별) – DB 인스턴스의 DB 인스턴스 클래스를 기준으로 합니다(예: db.t2.small 또는 db.m4.large). 요금은 시간 단위로 고시되지만, 청구서는 초 단위로 계산되고 시간을 10진수 형식으로 표시합니다. RDS 사용량은 1초 단위로 청구되며 최소 청구 시간은 10분입니다. 자세한 내용은 DB 인스턴스 클래스 단원을 참조하십시오.
- 스토리지(월별 GiB별) – DB 인스턴스에 프로비저닝한 스토리지 용량입니다. 해당 월에 프로비저닝된 스토리지 용량의 크기를 조정하는 경우 청구서 금액은 비례 할당으로 계산됩니다. 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 섹션을 참조하세요.
- 입출력(I/O) 요청(1백만 요청별) – 결제 주기에 요청한 총 스토리지 I/O 요청 수입니다. Amazon RDS 마그네틱 스토리지에만 해당됩니다.
- 프로비저닝된 IOPS(월별 IOPS별) – 사용된 IOPS에 상관없이 프로비저닝된 IOPS 비율입니다. Amazon RDS 프로비저닝된 IOPS(SSD) 및 범용 (SSD) gp3 스토리지만 해당됩니다. EBS 볼륨에 대해 프로비저닝된 스토리지는 1초 단위로 청구되며 최소 청구 시간은 10분입니다.
- 백업 스토리지(월별 GiB별) – 백업 스토리지는 자동화된 데이터베이스 백업 및 생성한 활성 데이터베이스 스냅샷과 연결된 스토리지입니다. 백업 보존 기간을 연장하거나 추가 데이터베이스 스냅샷을 찍으면 데이터베이스가 사용하는 백업 스토리지가 증가합니다. 초 단위 결제는 백업 스토리지(GB-월 단위로 측정됨)에는 적용되지 않습니다.

  자세한 내용은 데이터 백업, 복원 및 내보내기 섹션을 참조하세요.
- 데이터 전송(GB별) - DB 인스턴스와 인터넷 및 기타 AWS 리전 간의 양방향 데이터 전송입니다. 유용한 예는 AWS 관리형 데이터베이스의 데이터 전송 비용 살펴보기 AWS 블로그 게시물을 참조하세요.

Amazon RDS는 사용자가 요구 사항에 따라 비용을 최적화할 수 있도록 다음과 같은 구입 옵션을 제공합니다.

- 온디맨드 인스턴스 – 사용하는 DB 인스턴스 시간에 대해 시간별로 지불합니다. 요금은 시간 단위로 고시되지만, 청구서는 초 단위로 계산되고 시간을 10진수 형식으로 표시합니다. RDS 사용량은 이제 1초 단위로 청구되며 최소 청구 시간은 10분입니다.
- 예약 인스턴스 – 1년 또는 3년 기간으로 DB 인스턴스를 예약하고 온디맨드 DB 인스턴스 요금에 비해 상당한 할인을 받습니다. 예약 인스턴스 사용 시 여러 인스턴스를 1시간 내에 시작, 삭제, 사용 또는 종료하고 모든 인스턴스에 대해 예약 인스턴스 혜택을 적용받을 수 있습니다.

Amazon RDS 요금에 대한 자세한 정보는 Amazon RDS 요금 페이지를 참조하세요.

주제

- Amazon RDS용 온디맨드 DB 인스턴스
- Amazon RDS용 예약 DB 인스턴스

Amazon RDS용 온디맨드 DB 인스턴스

Amazon RDS 온디맨드 DB 인스턴스는 DB 인스턴스 클래스를 기반으로 청구됩니다(예: db.t3.small 또는 db.m5.large). Amazon RDS 요금에 대한 자세한 정보는 Amazon RDS 제품 페이지를 참조하십시오.

DB 인스턴스가 사용 가능하면 즉시 DB 인스턴스에 대한 청구가 시작됩니다. 요금은 시간 단위로 고시되지만, 청구서는 초 단위로 계산되고 시간을 10진수 형식으로 표시합니다. Amazon RDS 사용량은 1초 단위로 청구되며 최소 청구 시간은 10분입니다. 컴퓨팅 또는 스토리지 용량 조정과 같은 청구 대상 구성 변경의 경우 최소 시간인 10분에 대해 요금이 부과됩니다. DB 인스턴스를 삭제하거나 DB 인스턴스에 장애가 발생하여 DB 인스턴스가 종료될 때까지 청구가 계속됩니다.

DB 인스턴스 요금이 더 이상 부과되지 않도록 하려면 추가 DB 인스턴스 시간에 대해 청구되지 않도록 인스턴스를 중지하거나 삭제해야 합니다. 청구되는 DB 인스턴스 상태에 대한 자세한 정보는 Amazon RDS DB 인스턴스 상태 보기 단원을 참조하십시오.

중지된 DB 인스턴스

DB 인스턴스가 중지되는 동안 프로비저닝된 IOPS를 포함하여 프로비저닝된 스토리지에 대해 요금이 부과됩니다. 지정된 보존 기간 내의 수동 스냅샷 및 자동 백업용 스토리지를 포함하여 백업 스토리지에 대한 요금도 부과됩니다. DB 인스턴스 시간에 대해서는 요금이 부과되지 않습니다

다중 AZ DB 인스턴스

다중 AZ 설정은 다른 가용 영역에서 동기식 대기 복제본을 자동으로 프로비저닝하고 유지 관리하여 데이터 내구성과 가용성을 향상시킵니다. 추가 리소스와 가용성 증가로 인해 다중 AZ 배포는 단일 AZ 배포보다 요금이 높으며 추가 대기 인스턴스 및 관련 리소스 때문에 약 2배의 비용이 발생할 수 있습니다.

다중 AZ 요금에 대한 다음과 같은 중요한 세부 정보를 고려하세요.

- 컴퓨팅 비용: 기본 인스턴스와 대기 인스턴스 모두에 대해 DB 인스턴스 시간당 요금이 청구됩니다.
- 스토리지 비용: 기본 인스턴스와 대기 인스턴스 모두에 프로비저닝된 스토리지에 대해 월별 GB당 요금이 부과됩니다.
- 데이터 전송 비용: 기본 인스턴스와 대기 인스턴스 간의 복제는 비용에 포함되지만 사용량에 따라 기타 데이터 전송 요금이 적용될 수 있습니다.

특정 사용 사례 및 AWS 리전을 기반으로 월별 비용을 정확하게 추정하려면 AWS Pricing Calculator를 사용하세요. 이 도구를 사용하면 구성 세부 정보를 입력하고 포괄적인 비용 분석을 제공받을 수 있습니다.

> **Note**
>
> 요금은 변경될 수 있습니다. 최신 정보는 Amazon RDS 요금 페이지를 참조하세요.

Amazon RDS용 예약 DB 인스턴스

예약 DB 인스턴스를 사용하면 1년 또는 3년 단위로 DB 인스턴스를 예약할 수 있습니다. 예약 DB 인스턴스는 온디맨드 DB 인스턴스 요금과 비교하여 대폭 할인된 요금을 제공합니다. 예약 DB 인스턴스는 물리적 인스턴스가 아니고 오히려 계정에서 온디맨드 DB 인스턴스를 사용할 때 적용되는 결제 할인에 가깝습니다. 예약 DB 인스턴스의 할인 요금은 인스턴스 유형 및 AWS 리전에 따라 결정됩니다.

DB 인스턴스를 예약하기 위한 프로세스는 다음과 같습니다. 먼저 구매할 수 있는 DB 인스턴스 예약 상품에 대한 정보를 확인합니다. 그런 다음 DB 인스턴스 예약 상품을 구매하고 마지막으로 기존에 예약되어 있는 DB 인스턴스에 대한 정보를 확인합니다.

예약 DB 인스턴스 구매 및 예약 DB 인스턴스 요금 청구서 보기에 대한 자세한 내용은 다음 섹션을 참조하세요.

- Amazon RDS용 예약 DB 인스턴스 구매
- Amazon RDS의 예약 DB 인스턴스에 대한 청구서 보기

예약 DB 인스턴스 개요

Amazon RDS에서 예약 DB 인스턴스를 구매할 때는 예약 DB 인스턴스의 기간 동안 특정 DB 인스턴스 유형에 대해 할인 요금을 이용하는 약정을 구매하는 것입니다. Amazon RDS 예약 DB 인스턴스를 사용하려면 온디맨드 인스턴스와 똑같은 방법으로 새로운 DB 인스턴스를 생성해야 합니다.

새롭게 생성한 DB 인스턴스는 다음에 대해 예약 DB 인스턴스의 사양과 일치해야 합니다.

- AWS 리전
- DB 엔진(DB 엔진의 버전 번호가 일치할 필요는 없습니다.)
- DB 인스턴스 유형
- DB 인스턴스 크기(RDS for Microsoft SQL Server 및 Amazon RDS for Oracle License Included)
- 에디션(RDS for SQL Server 및 RDS for Oracle)
- 라이선스 유형(라이선스 포함 기존 보유 라이선스 사용)

새로운 DB 인스턴스의 사양이 계정의 기존 DB 예약 인스턴스와 일치하면 예약 DB 인스턴스에 제공되는 할인 요금이 청구됩니다. 그렇지 않으면 DB 인스턴스에 대해 온디맨드 요금이 청구됩니다.

예약형 DB 인스턴스로 사용 중인 DB 인스턴스를 수정할 수 있습니다. 변경 사항이 예약 DB 인스턴스의 사양 내에 있는 경우 수정된 DB 인스턴스에도 할인의 일부 또는 전부가 적용됩니다. 인스턴스 클래

스를 변경하는 경우와 같이 변경 사항이 사양을 벗어나면 할인이 더 이상 적용되지 않습니다. 자세한 내용은 유연한 크기의 예약 DB 인스턴스 단원을 참조하십시오.

주제

- 제공 유형
- 유연한 크기의 예약 DB 인스턴스
- 예약 DB 인스턴스 결제 예제
- 다중 AZ DB 클러스터에 대한 예약 DB 인스턴스
- 예약 DB 인스턴스 삭제

요금을 포함하여 예약 DB 인스턴스에 대한 자세한 내용은 Amazon RDS 예약 인스턴스를 참조하십시오.

제공 유형

예약 DB 인스턴스는 세 가지 유형(No Upfront, Partial Upfront 및 All Upfront)으로 제공되며 예상되는 사용률에 따라 Amazon RDS 비용을 최적화할 수 있습니다.

> **Note**
>
> 모든 RDS 인스턴스 클래스가 모든 예약 인스턴스 오퍼링 유형을 지원하는 것은 아닙니다. 예를 들어 일부 인스턴스 클래스는 선결제 없음 옵션을 제공하지 않을 수 있습니다. 이용 가능 여부를 확인하려면 AWS Management Console에서 예약 인스턴스 오퍼링을 검토하거나 `describe-reserved-db-instances-offerings` AWS CLI 명령을 사용합니다.

선수금 없음

선결제 없이 예약 DB 인스턴스에 액세스할 수 있는 옵션입니다. 비선결제 예약 DB 인스턴스는 사용 기간 동안 사용량에 상관없이 할인된 시간당 요금이 청구되며, 선결제가 필요하지 않습니다. 이 옵션은 1년 예약만 가능합니다.

부분 선결제

예약 DB 인스턴스 사용비의 일부를 먼저 결제해야 하는 옵션입니다. 결제하지 않은 시간에 대해서는 사용 기간 동안 사용량에 상관없이 할인된 시간당 요금이 청구됩니다. 이 옵션은 이전 Heavy 사용률 옵션을 대신합니다.

전체 선결제

> 약관이 시작되는 시점에서 모든 금액을 결제하고 사용 기간 동안 추가 비용 없이 무제한으로 사용할 수 있습니다.

통합 결제를 사용하는 경우, 결제의 편의를 위해 조직 내 모든 계정은 하나의 계정으로 취급됩니다. 즉 조직 내 모든 계정은 다른 계정에서 구입한 예약 DB 인스턴스에 대해 시간당 비용 혜택을 받을 수 있습니다. 통합 결제에 대한 자세한 내용은 AWS Billing and Cost Management 사용 설명서에서 Amazon RDS 예약 DB 인스턴스를 참조하세요.

유연한 크기의 예약 DB 인스턴스

예약 DB 인스턴스를 구매할 때 지정해야 하는 것 중 하나가 인스턴스 클래스(db.r5.large 등)입니다. DB 인스턴스 클래스에 대한 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요.

이미 DB 인스턴스가 있지만 용량을 확장해야 하는 경우에는 예약 DB 인스턴스가 확장된 DB 인스턴스에 자동으로 적용됩니다. 다시 말해서 예약 DB 인스턴스는 모든 DB 인스턴스 클래스 크기에 자동으로 적용됩니다. 동일한 AWS 리전 및 데이터베이스 엔진에서 유연한 크기의 예약 DB 인스턴스를 DB 인스턴스에 사용할 수 있습니다. 유연한 크기의 예약 DB 인스턴스는 해당 인스턴스 클래스 유형에서만 확장할 수 있습니다. 예를 들어 db.r5.large의 예약 DB 인스턴스는 db.r5.large에 적용할 수 있지만, db.r6g.large에는 적용할 수 없습니다. db.r5와 db.r6g는 다른 인스턴스 클래스 유형이기 때문입니다.

이러한 예약 DB 인스턴스의 이점은 다중 AZ와 단일 AZ 구성 모두에 적용됩니다. 이는 동일한 DB 인스턴스 클래스 유형 내에서 구성 간에 자유롭게 이동할 수 있음을 의미합니다. 예를 들어, 하나의 대형 DB 인스턴스(시간당 4개의 정규화된 유닛)에서 실행 중인 단일 AZ 배포로부터 2개의 중형 DB 인스턴스(시간당 2+2 = 4개의 정규화된 유닛)에서 실행 중인 다중 AZ 배포로 이동할 수 있습니다.

유연한 크기의 예약 DB 인스턴스는 다음 Amazon RDS 데이터베이스 엔진에서 제공됩니다.

- RDS for MariaDB
- RDS for MySQL
- RDS for Oracle, Bring Your Own License
- RDS for PostgreSQL

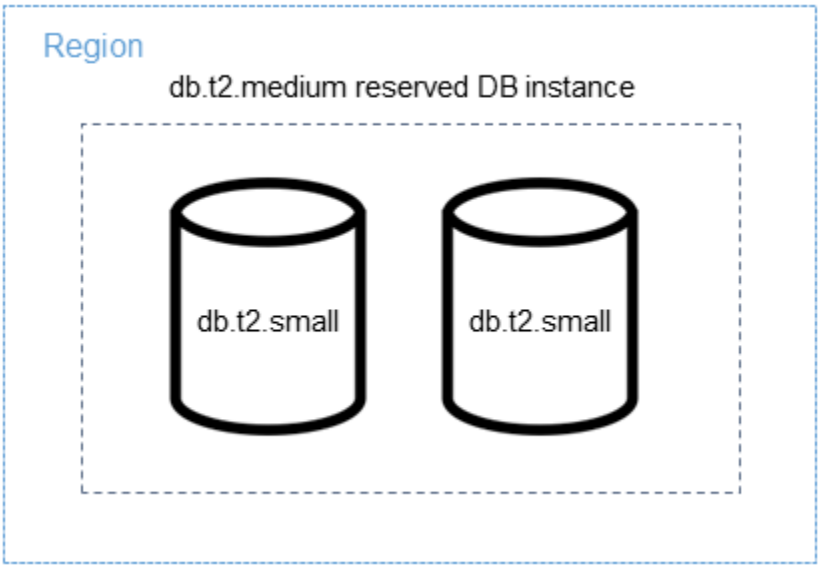
RDS for SQL Server와 RDS for Oracle License Included에는 크기 유연성이 적용되지 않습니다.

Aurora에서 유연한 크기의 예약 인스턴스를 사용하는 자세한 내용은 Aurora용 예약 DB 인스턴스를 참조하십시오.

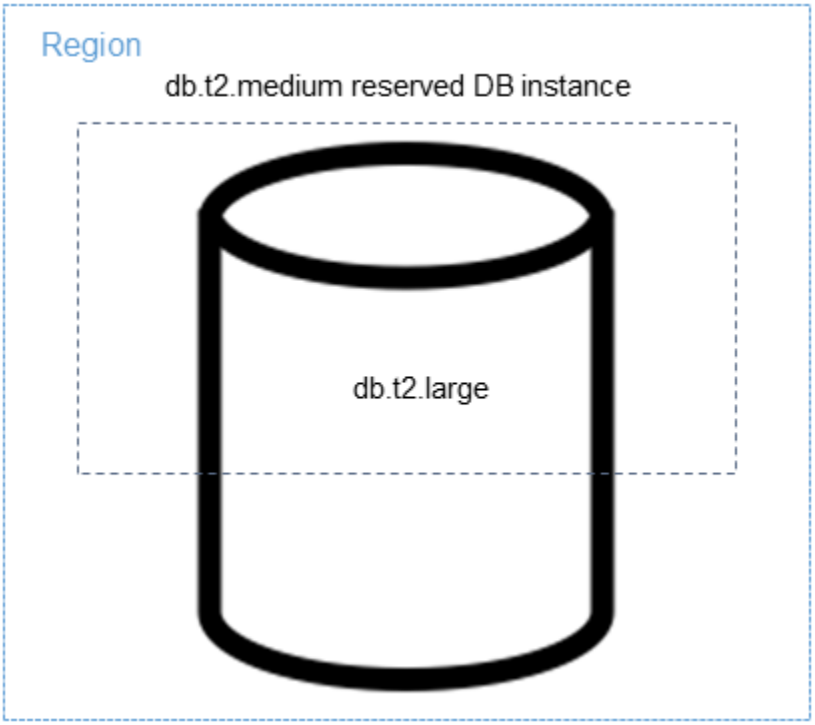
예약 DB 인스턴스의 크기에 따른 사용량은 시간당 정규화된 유닛을 사용하여 비교할 수 있습니다. 예를 들어 db.r3.large DB 인스턴스 2개일 때 사용량의 유닛 1개는 db.r3.small 1개일 때 사용량의 시간당 정규화된 유닛 8개와 같습니다. 다음 표는 각 DB 인스턴스 크기에 따른 시간당 정규화된 유닛의 수를 나타낸 것입니다.

인스턴스 크기	단일 AZ 시간당 정규화된 유닛(하나의 DB 인스턴스를 사용하여 배포)	다중 AZ DB 인스턴스 시간당 정규화된 유닛(하나의 DB 인스턴스 및 하나의 대기 인스턴스를 사용하여 배포)	다중 AZ DB 클러스터 시간당 정규화된 유닛(하나의 DB 인스턴스 및 두 개의 대기 인스턴스를 사용하여 배포)
micro	0.5	1	1.5
small	1	2	3
medium	2	4	6
large	4	8	12
xlarge	8	16	24
2xlarge	16	32	48
4xlarge	32	64	96
6xlarge	48	96	144
8xlarge	64	128	192
10xlarge	80	160	240
12xlarge	96	192	288
16xlarge	128	256	384
24xlarge	192	384	576
32xlarge	256	512	768

예약 DB 인스턴스로 db.t2.medium을 1개 구매하고, 동일한 AWS 리전의 계정에서 db.t2.small DB 인스턴스를 2개 실행하는 경우를 예로 들어 보겠습니다. 이 경우 결제 혜택은 두 인스턴스에 100% 적용됩니다.



또는 동일한 AWS 리전의 계정에서 실행 중인 db.t2.large 인스턴스 1개가 있는 경우 결제 혜택은 DB 인스턴스 사용량의 50%에 적용됩니다.



예약 DB 인스턴스 결제 예제

예약 DB 인스턴스의 요금은 스토리지, 백업 및 I/O와 관련된 비용의 할인을 제공하지 않으며, 시간당 온디맨드 인스턴스 사용량에 대한 할인만 제공합니다. 다음 예는 예약 DB 인스턴스의 월 총 요금을 보여 줍니다.

- 미국 동부(버지니아 북부)의 RDS for MySQL 예약 단일 AZ db.r5.large DB 인스턴스 클래스, 선결제 없음 옵션 포함, 인스턴스 요금 0.12 USD(월 90 USD)
- 월 기준 GiB당 0.115 USD(월 45.60 USD)의 400GiB of 범용 SSD(gp2) 스토리지
- 0.095 USD(월 19 USD)의 600GiB 백업 스토리지(400GiB 무료)

예약 DB 인스턴스에 이러한 요금을 모두 추가할 경우(90 USD + 45.60 USD + 19 USD), 월 총요금은 154.60 USD입니다.

예약 DB 인스턴스 대신 온디맨드 DB 인스턴스를 사용하기로 선택하는 경우, 미국 동부(버지니아 북부)의 RDS for MySQL 단일 AZ db.r5.large DB 인스턴스 클래스 요금은 시간당 0.1386 USD(월 101.18 USD)입니다. 따라서 온디맨드 DB 인스턴스의 경우, 이러한 옵션을 모두 추가하면(101.18 USD + 45.60 USD + 19 USD), 월 총 요금은 165.78 USD입니다. 예약 DB 인스턴스를 사용하면 매달 11USD가 조금 넘는 비용을 절감할 수 있습니다.

> **Note**
>
> 이 예의 요금은 예제 요금이며 실제 요금과 다를 수 있습니다. Amazon RDS 요금에 대한 자세한 정보는 Amazon RDS 요금 페이지를 참조하십시오.

다중 AZ DB 클러스터에 대한 예약 DB 인스턴스

다음 중 하나를 수행하여 다중 AZ DB 클러스터에서 대한 동일한 예약 DB 인스턴스를 구매할 수 있습니다.

- 클러스터에 있는 인스턴스와 크기가 동일한 단일 AZ DB 인스턴스 3개를 예약합니다.
- 클러스터에 있는 DB 인스턴스와 크기가 동일한 하나의 다중 AZ DB 인스턴스 및 하나의 단일 AZ DB 인스턴스를 예약합니다.

예를 들어 하나의 클러스터가 3개의 db.m6gd.large DB 인스턴스로 구성된 경우를 가정해 보겠습니다. 이 경우 db.m6gd.large 단일 AZ 예약 DB 인스턴스 3개를 구매하거나, db.m6gd.large 다중 AZ 예약 DB

인스턴스 1개와 db.m6gd.large 단일 AZ 예약 DB 인스턴스 1개를 구매할 수 있습니다. 두 옵션 모두 다중 AZ DB 클러스터에 대한 최대 예약 인스턴스 할인을 예약합니다.

또는 유연한 크기의 DB 인스턴스를 사용하고 더 큰 DB 인스턴스를 구매하여 하나 이상의 클러스터에서 더 작은 DB 인스턴스를 지원할 수 있습니다. 예를 들어 총 6개의 db.m6gd.large DB 인스턴스가 포함된 두 개의 클러스터를 보유하고 있을 경우, db.m6gd.xl 단일 AZ 예약 DB 인스턴스 3개를 구매할 수 있습니다. 이렇게 하면 두 클러스터에 있는 DB 인스턴스 6개가 모두 예약됩니다. 자세한 내용은 유연한 크기의 예약 DB 인스턴스 단원을 참조하십시오.

클러스터에 있는 DB 인스턴스와 크기가 동일하지만 클러스터에 있는 DB 인스턴스의 총 개수보다 적은 수의 DB 인스턴스를 예약할 수도 있습니다. 그러나 이렇게 하면 클러스터가 일부만 예약됩니다. 예를 들어 클러스터에 db.m6gd.large DB 인스턴스 3개가 있고 db.m6gd.large 다중 AZ 예약 DB 인스턴스를 1개 구매한다고 가정해 보겠습니다. 이 경우 클러스터의 인스턴스 3개 중 2개만 예약 DB 인스턴스에 포함되므로 클러스터는 부분적으로만 예약됩니다. 나머지 DB 인스턴스에는 온디맨드 db.m6gd.large 시간당 요금이 청구됩니다.

다중 AZ DB 클러스터에 대한 자세한 내용은 Amazon RDS용 다중 AZ DB 클러스터 배포 섹션을 참조하세요.

예약 DB 인스턴스 삭제

예약 DB 인스턴스에 대한 약정 기간은 1년 또는 3년입니다. 예약 DB 인스턴스는 취소할 수 없습니다. 하지만 예약 DB 인스턴스 할인이 적용되는 DB 인스턴스를 삭제할 수는 있습니다. 예약 DB 인스턴스 할인이 적용되는 DB 인스턴스의 삭제 프로세스는 다른 DB 인스턴스를 삭제할 때와 동일합니다.

리소스 사용 여부에 관계없이 선결제 비용이 청구됩니다.

예약 DB 인스턴스 할인이 적용되는 DB 인스턴스를 삭제할 경우에는 다르지만 서로 사양이 호환되는 DB 인스턴스를 시작할 수 있습니다. 이 경우 예약 기간(1년 또는 3년)에 요금 할인을 계속 받을 수 있습니다.

Amazon RDS용 예약 DB 인스턴스 구매

AWS Management Console, AWS CLI 및 RDS API를 사용하여 예약 DB 인스턴스 작업을 수행할 수 있습니다.

콘솔

예약 DB 인스턴스에 대한 작업은 AWS Management Console에서 다음 절차에 따라 진행할 수 있습니다.

사용 가능한 예약 DB 인스턴스 상품에 대한 요금과 정보를 가져오려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 예약 인스턴스를 선택합니다.
3. [Purchase Reserved DB Instance]를 선택합니다.
4. 제품 설명에서 DB 엔진과 라이선스 유형을 선택합니다.
5. DB 인스턴스 클래스에서 DB 인스턴스 클래스를 선택합니다.
6. 배포 옵션에서 단일 AZ 배포 또는 다중 AZ 배포를 사용할지 선택합니다.

   > **Note**
   >
   > 다중 AZ DB 클러스터 배포에 대해 동일한 예약 DB 인스턴스를 구매하려면 단일 AZ 예약 DB 인스턴스 3개를 구매하거나, 다중 AZ 예약 DB 인스턴스 1개와 단일 AZ 예약 DB 인스턴스 1개를 구입하세요. 자세한 내용은 다중 AZ DB 클러스터에 대한 예약 DB 인스턴스 단원을 참조하십시오.

7. 기간에서 DB 인스턴스를 예약할 기간을 선택합니다.
8. 제공 유형에서 해당 제공 유형을 선택합니다.

   상품 유형을 선택하면 요금 정보가 표시됩니다.

   > **Important**
   >
   > 취소를 선택하면 예약 DB 인스턴스를 구입하지 않으며 요금이 발생하지 않습니다.

구매할 수 있는 DB 인스턴스 예약 상품에 대한 정보를 확인하였으면 이제 정보를 사용하여 다음 절차에 따라 상품을 구매할 수 있습니다.

예약 DB 인스턴스를 구입하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 예약 인스턴스를 선택합니다.
3. Purchase reserved DB instance(예약 DB 인스턴스 구매)를 선택합니다.
4. 제품 설명에서 DB 엔진과 라이선스 유형을 선택합니다.

5. DB 인스턴스 클래스에서 DB 인스턴스 클래스를 선택합니다.
6. 다중 AZ 배포에서 단일 AZ 또는 다중 AZ DB 인스턴스 배포를 사용할지 여부를 선택합니다.

> **Note**
>
> 다중 AZ DB 클러스터 배포에 대해 동일한 예약 DB 인스턴스를 구매하려면 단일 AZ 예약 DB 인스턴스 3개를 구매하거나, 다중 AZ 예약 DB 인스턴스 1개와 단일 AZ 예약 DB 인스턴스 1개를 구입하세요. 자세한 내용은 다중 AZ DB 클러스터에 대한 예약 DB 인스턴스 단원을 참조하십시오.

7. [Term]에서 DB 인스턴스를 예약할 기간을 선택합니다.
8. 제공 유형에서 해당 제공 유형을 선택합니다.

   오퍼링 유형을 선택하면 요금 정보가 표시됩니다.
9. (선택 사항) - 예약 DB 인스턴스를 조회할 수 있도록 구매하는 예약 인스턴스에 자체 식별자를 할당할 수 있습니다. [Reserved Id]에 자신이 예약한 DB 인스턴스 식별자를 입력하면 됩니다.
10. 제출을 선택합니다.

    예약 DB 인스턴스를 구매하면 예약 인스턴스 목록에 표시됩니다.

예약한 DB 인스턴스를 구매한 후에는 다음 절차에 따라 예약한 DB 인스턴스에 대한 정보를 가져올 수 있습니다.

AWS 계정에 대한 예약 DB 인스턴스 관련 정보를 가져오려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 예약 인스턴스를 선택합니다.

   현재 계정에서 예약한 DB 인스턴스가 나타납니다. 특정 예약 DB 인스턴스의 세부 정보를 보려면 목록에서 해당 인스턴스를 선택합니다. 그러면 콘솔 아래쪽의 세부 정보 창에 인스턴스에 대한 세부 정보가 표시됩니다.

AWS CLI

예약 DB 인스턴스에 대한 작업은 다음 예제와 같이 AWS CLI를 사용하여 진행할 수 있습니다.

Example 사용 가능한 예약 DB 인스턴스 오퍼링 가져오기

구매 가능한 DB 인스턴스 상품에 대한 정보를 가져오려면 AWS CLI 명령 describe-reserved-db-instances-offerings를 호출합니다.

```
aws rds describe-reserved-db-instances-offerings
```

이 호출은 다음과 비슷한 출력을 반환합니다.

```
OFFERING  OfferingId                            Class         Multi-AZ  Duration  Fixed
 Price  Usage Price  Description  Offering Type
OFFERING  438012d3-4052-4cc7-b2e3-8d3372e0e706  db.r3.large   y         1y
 1820.00 USD  0.368 USD    mysql        Partial  Upfront
OFFERING  649fd0c8-cf6d-47a0-bfa6-060f8e75e95f  db.r3.small   n         1y
 227.50 USD  0.046 USD    mysql        Partial  Upfront
OFFERING  123456cd-ab1c-47a0-bfa6-12345667232f  db.r3.small   n         1y
 162.00 USD   0.00 USD    mysql        All      Upfront
    Recurring Charges:   Amount  Currency  Frequency
    Recurring Charges:   0.123   USD       Hourly
OFFERING  123456cd-ab1c-37a0-bfa6-12345667232d  db.r3.large   y         1y
 700.00 USD   0.00 USD    mysql        All      Upfront
    Recurring Charges:   Amount  Currency  Frequency
    Recurring Charges:   1.25    USD       Hourly
OFFERING  123456cd-ab1c-17d0-bfa6-12345667234e  db.r3.xlarge  n         1y
 4242.00 USD   2.42 USD    mysql        No       Upfront
```

구매할 수 있는 DB 인스턴스 예약 상품에 대한 정보를 확인하였으면 이제 정보를 사용하여 오퍼링을 구매할 수 있습니다.

예약 DB 인스턴스를 구매하려면 다음 파라미터와 함께 AWS CLI 명령 purchase-reserved-db-instances-offering을 사용합니다.

- `--reserved-db-instances-offering-id` – 구매하려는 오퍼링의 ID입니다. 위의 예제를 참조하여 상품 ID를 가져옵니다.
- `--reserved-db-instance-id` – 구매하는 예약 DB 인스턴스에 자체 식별자를 할당하여 관리할 수 있습니다.

Example 예약 DB 인스턴스 구매

다음은 ID가 *649fd0c8-cf6d-47a0-bfa6-060f8e75e95f*인 DB 인스턴스 예약 상품을 구매하고 식별자로 *MyReservation*을 할당하는 예제입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds purchase-reserved-db-instances-offering \
    --reserved-db-instances-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f \
    --reserved-db-instance-id MyReservation
```

Windows의 경우:

```
aws rds purchase-reserved-db-instances-offering ^
    --reserved-db-instances-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f ^
    --reserved-db-instance-id MyReservation
```

이 명령은 다음과 비슷한 출력을 반환합니다.

```
RESERVATION  ReservationId      Class        Multi-AZ  Start Time
 Duration  Fixed Price  Usage Price  Count  State           Description  Offering Type
RESERVATION  MyReservation      db.r3.small  y         2011-12-19T00:30:23.247Z  1y
    455.00 USD   0.092 USD    1      payment-pending  mysql        Partial  Upfront
```

예약 DB 인스턴스를 구매한 후에는 예약 DB 인스턴스에 대한 정보를 가져올 수 있습니다.

AWS 계정에서 예약 DB 인스턴스에 대한 정보를 가져오려면 다음 예제와 같이 AWS CLI 명령 describe-reserved-db-instances를 호출합니다.

Example 예약 DB 인스턴스 가져오기

```
aws rds describe-reserved-db-instances
```

이 명령은 다음과 비슷한 출력을 반환합니다.

```
RESERVATION  ReservationId     Class        Multi-AZ  Start Time
 Duration  Fixed Price  Usage Price  Count  State    Description  Offering Type
RESERVATION  MyReservation     db.r3.small  y         2011-12-09T23:37:44.720Z  1y
   455.00 USD   0.092 USD    1      retired  mysql        Partial  Upfront
```

RDS API

RDS API를 사용하여 예약 DB 인스턴스 작업을 수행할 수 있습니다.

- 구매할 수 있는 DB 인스턴스 예약 상품에 대한 정보를 가져오려면 Amazon RDS API 작업 DescribeReservedDBInstancesOfferings를 호출합니다.
- 구매할 수 있는 DB 인스턴스 예약 상품에 대한 정보를 확인하였으면 이제 정보를 사용하여 오퍼링을 구매할 수 있습니다. 다음 파라미터로 PurchaseReservedDBInstancesOffering RDS API 작업을 호출합니다.
  - `--reserved-db-instances-offering-id` – 구매하려는 오퍼링의 ID입니다.
  - `--reserved-db-instance-id` – 구매하는 예약 DB 인스턴스에 자체 식별자를 할당하여 관리할 수 있습니다.
- 예약 DB 인스턴스를 구매한 후에는 예약 DB 인스턴스에 대한 정보를 가져올 수 있습니다. RDS API 작업 DescribeReservedDBInstances를 호출합니다.

Amazon RDS의 예약 DB 인스턴스에 대한 청구서 보기

예약 DB 인스턴스에 대한 결제는 AWS Management Console의 결제 대시보드(Billing Dashboard)에서 확인할 수 있습니다.

예약 DB 인스턴스 결제 확인

1. AWS Management Console에 로그인합니다.
2. 오른쪽 상단의 계정 메뉴(account menu)에서 결제 대시보드(Billing Dashboard)를 선택합니다.
3. 대시보드 오른쪽 상단의 청구 세부 정보(Service Charge)를 선택합니다.
4. AWS 서비스 요금(Service Charges)에서 관계형 데이터베이스 서비스(Relational Database Service)를 확장합니다.
5. 미국 서부(오레곤)과 같이 예약 DB 인스턴스가 있는 AWS 리전을 확장합니다.

   예약 DB 인스턴스와 현재 월의 시간당 요금은 ****** ##용 Amazon Relational Database Service 예약 인스턴스에서 볼 수 있습니다.

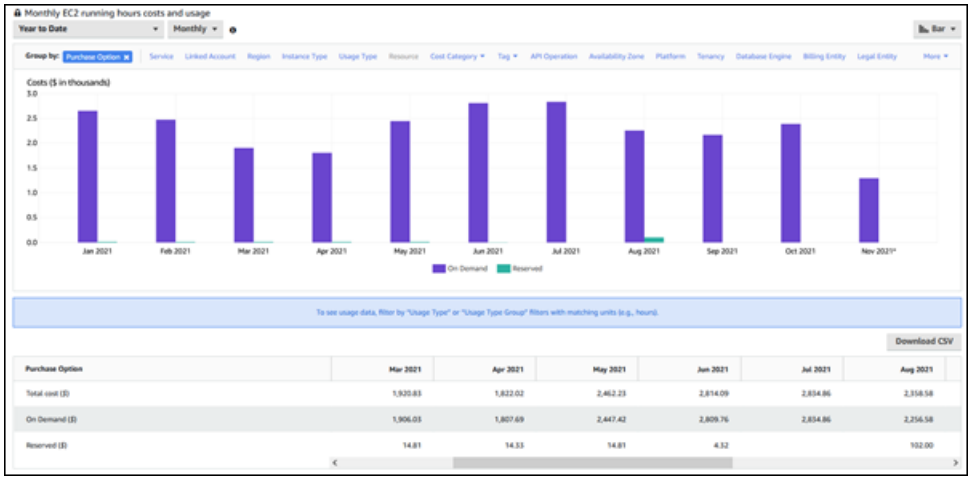
   

   이 예시의 예약 DB 인스턴스는 전체 선결제 구매되었으므로 시간당 요금이 부과되지 않습니다.
6. 예약 인스턴스(Reserved Instances) 제목 옆의 비용 탐색기(Cost Explorer)(막대 그래프) 아이콘을 선택합니다.

Cost Explorer는 월별 EC2 운영 시간 비용 및 사용량(Monthly EC2 running hours costs and usage) 그래프를 보여줍니다.

7. 그래프 오른쪽에 있는 사용 유형 그룹(Usage Type Group) 필터를 해제합니다.
8. 사용 비용을 검사할 기간 및 시간 단위를 선택합니다.

다음 예에서는 해당 연도의 온디맨드 및 예약 DB 인스턴스의 사용 비용을 월별로 보여줍니다.



2021년 1월부터 6월까지 예약 DB 인스턴스 비용은 부분 선결제 인스턴스에 대한 월별 요금이며, 2021년 8월 요금은 전체 선결제 인스턴스에 대한 일회성 요금입니다.

부분 선결제 인스턴스에 대한 예약 인스턴스 할인은 2021년 6월에 만료되었지만 DB 인스턴스는 삭제되지 않았습니다. 만료일 이후에는 온디맨드 요금으로 간단히 청구되었습니다.

Amazon RDS 환경 설정

이 페이지는 계정 구성, 보안 및 리소스 관리를 포함하여 Amazon Relational Database Service를 설정하기 위한 포괄적인 안내를 제공합니다. 데이터베이스 환경을 효율적으로 생성, 관리 및 보호하는 데 필요한 필수 단계를 안내합니다. Amazon RDS를 처음 사용하든, 특정 요구 사항에 맞게 설정하든 관계없이 이 섹션을 통해 설정을 최적화하고 모범 사례를 준수할 수 있습니다.

주제

- AWS 계정에 등록
- 관리자 액세스 권한이 있는 사용자 생성
- 프로그래밍 방식 액세스 권한 부여
- 요구 사항 결정
- 보안 그룹을 생성하여 VPC 내부의 DB 인스턴스에 대한 액세스를 제공

AWS 계정가 이미 있고 Amazon RDS 요구 사항을 알고 있으며 IAM 및 VPC 보안 그룹에 모두 기본값을 사용하려는 경우에는 Amazon RDS 시작하기 섹션으로 건너뛰세요.

AWS 계정에 등록

AWS 계정이 없는 경우 다음 절차에 따라 계정을 생성합니다.

AWS 계정에 가입

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지시 사항을 따릅니다.

   등록 절차 중 전화 또는 텍스트 메시지를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

   AWS 계정 루트 사용자에 가입하면 AWS 계정 루트 사용자가 만들어집니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS는 가입 절차 완료된 후 사용자에게 확인 이메일을 전송합니다. 언제든지 https://aws.amazon.com/으로 이동하고 내 계정을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

AWS 계정에 가입하고 AWS 계정 루트 사용자에 보안 조치를 한 다음, AWS IAM Identity Center을 활성화하고 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 생성합니다.

귀하의 AWS 계정 루트 사용자 보호

1. 루트 사용자를 선택하고 AWS 계정이메일 주소를 입력하여 AWS Management Console에 계정 소유자로 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 User Guide의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자용 가상 MFA 디바이스 활성화(콘솔)를 참조하세요.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center 설정을 참조하세요.
2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   IAM Identity Center 디렉터리를 ID 소스로 사용하는 방법에 대한 자습서는 AWS IAM Identity Center 사용 설명서의 기본 IAM Identity Center 디렉터리로 사용자 액세스 구성을 참조하세요.

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM Identity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자로 로그인하는 데 도움이 필요한 경우 AWS 로그인 사용 설명서의 AWS 액세스 포털에 로그인을 참조하세요.

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.
2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

프로그래밍 방식 액세스 권한 부여

사용자가 AWS Management Console 외부에서 AWS와 상호 작용하려면 프로그래밍 방식의 액세스가 필요합니다. 프로그래밍 방식으로 액세스를 부여하는 방법은 AWS에 액세스하는 사용자 유형에 따라 다릅니다.

사용자에게 프로그래밍 방식 액세스 권한을 부여하려면 다음 옵션 중 하나를 선택합니다.

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
작업 인력 ID (IAM Identity Center가 관리하는 사용자)	임시 자격 증명을 사용하여 AWS CLI, AWS SDK 또는 AWS API에 대한 프로그래밍 요청에 서명합니다.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • AWS CLI에 대해서는 AWS Command Line Interface 사용 설명서에서 AWS IAM Identity Center을 사용하도록 AWS CLI 구성을 참조하세요. • AWS SDK, 도구,AWS API에 대해서는 AWS SDK 및 도구 참조 가이드에서 IAM Identity Center 인증을 참조하세요.
IAM	임시 자격 증명을 사용하여 AWS CLI, AWS SDK 또는	IAM 사용자 설명서의 AWS 리소스와 함께 임시 자격 증명 사

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
	AWS API에 대한 프로그래밍 요청에 서명합니다.	용에 나와 있는 지침을 따르세요.
IAM	(권장되지 않음) 장기 자격 증명을 사용하여 AWS CLI, AWS SDK 또는 AWS API에 대한 프로그래밍 요청에 서명합니다.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • AWS CLI에 대해서는 AWS Command Line Interface 사용 설명서에서 IAM 사용자 자격 증명을 사용한 인증을 참조하세요. • AWS SDK와 도구는 AWS SDK 및 도구 참조 가이드에서 장기 자격 증명을 사용한 인증을 참조하세요. • AWS API는 IAM 사용자 설명서에서 IAM 사용자의 액세스 키 관리를 참조하세요.

요구 사항 결정

Amazon RDS의 기본 빌딩 블록은 DB 인스턴스입니다. DB 인스턴스에서는 데이터베이스를 생성합니다. DB 인스턴스는 엔드포인트라고 하는 네트워크 주소를 할당합니다. 애플리케이션은 이 엔드포인트를 사용하여 DB 인스턴스에 연결합니다. DB 인스턴스를 생성할 때 스토리지, 메모리, 데이터베이스 엔진 및 버전, 네트워크 구성, 보안, 유지 관리 기간 등의 세부 정보를 지정합니다. 보안 그룹을 통해 DB 인스턴스에 대한 네트워크 액세스를 제어할 수 있습니다.

DB 인스턴스와 보안 그룹을 생성하기 전에 DB 인스턴스 및 네트워크 필요를 알아야 합니다. 고려해야 할 몇 가지 중요 사항은 다음과 같습니다:

- 리소스 요구 사항 – 애플리케이션 또는 서비스의 메모리 및 프로세서 요구 사항은 무엇입니까? 이러한 설정을 사용하면 어떤 DB 인스턴스 클래스를 사용할지를 결정하는 데 도움이 됩니다. DB 인스턴스 클래스에 대한 사양은 DB 인스턴스 클래스 단원을 참조하십시오.

- VPC, 서브넷 및 보안 그룹 – DB 인스턴스는 대부분의 경우 Virtual Private Cloud(VPC) 안에 있습니다. DB 인스턴스에 연결하려면 보안 그룹 규칙을 설정해야 합니다. 이러한 규칙은 사용하는 VPC 종류와 사용 방식에 따라 다르게 설정됩니다. 예를 들어 기본 VPC 또는 사용자 정의 VPC를 사용할 수 있습니다.

  다음은 각 VPC 옵션의 규칙을 설명한 목록입니다.

  - 기본 VPC - AWS 계정에 최신 AWS 리전의 기본 VPC가 있는 경우 DB 인스턴스를 지원하도록 해당 VPC를 구성할 수 있습니다. DB 인스턴스 생성 시 기본 VPC를 지정할 경우 다음을 수행합니다.
    - 애플리케이션 또는 서비스에서 Amazon RDS DB 인스턴스로의 연결 권한을 부여하는 VPC 보안 그룹을 생성해야 합니다. VPC 콘솔의 [보안 그룹(Security Group)] 옵션 또는 AWS CLI를 사용하여 VPC 보안 그룹을 생성합니다. 자세한 정보는 3단계: VPC 보안 그룹 만들기 섹션을 참조하세요.
    - 기본 DB 서브넷 그룹을 지정합니다. 이 AWS 리전에서 처음 DB 인스턴스를 생성하는 경우에는 Amazon RDS가 DB 인스턴스를 생성할 때 기본 DB 서브넷 그룹을 생성합니다.
  - 사용자 정의 VPC – DB 인스턴스 생성 시 사용자 정의 VPC를 지정할 경우 다음 사항에 유의해야 합니다.
    - 애플리케이션 또는 서비스에서 Amazon RDS DB 인스턴스로의 연결 권한을 부여하는 VPC 보안 그룹을 생성해야 합니다. VPC 콘솔의 [보안 그룹(Security Group)] 옵션 또는 AWS CLI를 사용하여 VPC 보안 그룹을 생성합니다. 자세한 정보는 3단계: VPC 보안 그룹 만들기 섹션을 참조하세요.
    - VPC가 DB 인스턴스를 호스팅하려면 별도의 가용 영역에서 각각 최소 2개 이상씩 서브넷을 구성하는 등 특정 요구 사항을 충족해야 합니다. 자세한 정보는 Amazon VPC 및 Amazon RDS 섹션을 참조하세요.
    - DB 서브넷 그룹을 지정하여 DB 인스턴스에서 VPC를 사용할 서브넷을 정의해야 합니다. 자세한 내용은 VPC에서 DB 인스턴스를 사용한 작업에서 DB 서브넷 그룹을 참조하세요.
- 높은 가용성 - 장애 조치 지원이 필요합니까 Amazon RDS에서 다중 AZ 배포는 기본 DB 인스턴스를 생성하고 장애 조치 지원을 위해 다른 가용 영역에 보조 예비 DB 인스턴스를 생성합니다. 고가용성을 유지하기 위해 프로덕션 워크로드에는 다중 AZ 배포를 권장합니다. 개발 및 테스트 목적으로는 비 다중 AZ 배포를 사용할 수 있습니다. 자세한 내용은 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 섹션을 참조하세요.
- IAM 정책 - AWS 계정에 Amazon RDS 작업을 수행하는 데 필요한 권한을 부여하는 정책이 있습니까? IAM 자격 증명을 사용하여 AWS에 연결하는 경우 IAM 계정에는 Amazon RDS 작업을 수행하는 데 필요한 권한을 부여하는 IAM 정책이 있어야 합니다. 자세한 내용은 Amazon RDS의 자격 증명 및 액세스 관리 섹션을 참조하세요.

- 개방 포트 - 데이터베이스가 어떤 TCP/IP 포트에서 수신 대기합니까? 일부 기업에서는 방화벽이 데이터베이스 엔진의 기본 포트 연결을 차단합니다. 이처럼 기업 방화벽이 기본 포트를 차단할 경우 새로운 DB 인스턴스를 생성할 때 다른 포트를 선택해야 합니다. 단, 지정 포트에서 수신 대기할 DB 인스턴스를 생성하는 경우 해당 DB 인스턴스를 수정하여 포트를 변경할 수 있습니다.
- AWS 리전 - 데이터베이스를 구성하려고 하는 AWS 리전은 어디입니까? 애플리케이션이나 웹 서비스에 가깝게 데이터베이스를 구성하면 네트워크 지연 시간을 줄일 수 있습니다. 자세한 내용은 리전, 가용 영역 및 로컬 영역 섹션을 참조하세요.
- DB 디스크 하위 시스템 - 스토리지 요구 사항은 무엇입니까? Amazon RDS는 세 가지 스토리지 유형을 제공합니다.
  - 범용(SSD)
  - 프로비저닝된 IOPS(PIOPS)
  - 마그네틱(표준 스토리지라고도 함)

  Amazon RDS 스토리지에 대한 자세한 정보는 Amazon RDS DB 인스턴스 스토리지 단원을 참조하십시오.

보안 그룹과 DB 인스턴스 생성에 필요한 정보를 확인하였으면 다음 단계로 진행합니다.

보안 그룹을 생성하여 VPC 내부의 DB 인스턴스에 대한 액세스를 제공

VPC 보안 그룹은 VPC에서 실행되는 DB 인스턴스에 대한 액세스를 제공합니다. 이들은 연결된 DB 인스턴스에 대한 방화벽 역할을 하여 DB 인스턴스 수준에서 인바운드 트래픽과 아웃바운드 트래픽을 모두 제어합니다. 기본적으로 DB 인스턴스는 DB 인스턴스를 보호하는 방화벽 및 기본 보안 그룹과 함께 생성됩니다.

DB 인스턴스에 연결하려면 먼저 연결하는 데 사용할 수 있는 규칙을 보안 그룹에 추가해야 합니다. 네트워크 및 구성 정보를 사용하여 DB 인스턴스에 액세스하는 데 사용할 수 있는 규칙을 생성합니다.

예를 들어 VPC의 DB 인스턴스에서 데이터베이스에 액세스하는 애플리케이션이 있다고 가정해 보겠습니다. 이 경우, 애플리케이션이 데이터베이스에 액세스하는 데 사용할 포트 범위와 IP 주소를 지정하는 사용자 지정 TCP 규칙을 추가해야 합니다. Amazon EC2 인스턴스에 애플리케이션이 있는 경우 Amazon EC2 인스턴스에 대해 설정한 보안 그룹을 사용할 수 있습니다.

DB 인스턴스를 만들 때 Amazon EC2 인스턴스와 DB 인스턴스 간의 연결을 구성할 수 있습니다. 자세한 내용은 EC2 인스턴스와의 자동 네트워크 연결 구성 단원을 참조하세요.

> **Tip**
>
> DB 인스턴스를 생성할 때 Amazon EC2 인스턴스와 DB 클러스터 간의 네트워크 연결을 자동으로 설정할 수 있습니다. 자세한 내용은 EC2 인스턴스와의 자동 네트워크 연결 구성 섹션을 참조하세요.

Amazon Lightsail의 리소스를 DB 인스턴스에 연결하는 방법에 대한 자세한 내용은 Connect Lightsail resources to AWS 서비스 using VPC peering을 참조하세요.

DB 인스턴스 액세스의 일반적인 시나리오에 대한 자세한 내용은 VPC에서 DB 인스턴스에 액세스하는 시나리오 섹션을 참조하세요.

VPC 보안 그룹의 생성 방법

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/vpc에서 Amazon VPC 콘솔을 엽니다.

   > **Note**
   >
   > RDS 콘솔이 아니라 VPC 콘솔을 사용해야 합니다.

2. AWS Management Console의 오른쪽 상단에서 VPC 보안 그룹 및 DB 인스턴스를 만들 AWS 리전을 선택합니다. 해당 AWS 리전의 Amazon VPC 리소스 목록에 1개 이상의 VPC와 몇 개의 서브넷이 표시되어야 합니다. 그렇지 않으면 해당 AWS 리전에 기본 VPC가 없는 것입니다.
3. 탐색 창에서 보안 그룹을 선택합니다.
4. 보안 그룹 생성을 선택합니다.

   [보안 그룹 생성(Create security group)] 페이지가 나타납니다.
5. [기본 세부 정보(Basic details)]에서 [보안 그룹 이름(Security group name)]과 [설명(Description)]을 입력합니다. [VPC]에서 DB 인스턴스를 생성할 VPC를 선택합니다.
6. [인바운드 규칙(Inbound rules)]에서 [규칙 추가(Add rule)]를 선택합니다.

   a. 유형에 대해 사용자 지정 TCP를 선택합니다.

   b. [포트 범위(Port range)]에 DB 인스턴스에 사용할 포트 값을 입력합니다.

   c. [소스(Source)]에서 DB 인스턴스에 액세스할 IP 주소 범위(CIDR 값)를 입력하거나 보안 그룹 이름을 선택합니다. 내 IP를 선택하면 브라우저에서 감지된 IP 주소에서 DB 인스턴스에 액세스할 수 있습니다.

7. IP 주소 또는 다른 포트 범위를 추가해야 하는 경우 [규칙 추가(Add rule)]를 선택하고 규칙에 대한 정보를 입력합니다.
8. (선택 사항) [아웃바운드 규칙(Outbound rules)]에서 아웃바운드 트래픽에 대한 규칙을 추가합니다. 기본적으로 모든 아웃바운드 트래픽이 허용됩니다.
9. 보안 그룹 생성을 선택합니다.

방금 생성한 VPC 보안 그룹을 DB 인스턴스 생성 시 보안 그룹으로 사용할 수 있습니다.

> **Note**
>
> 기본 VPC를 사용하는 경우 VPC의 모든 서브넷을 포괄하는 기본 서브넷 그룹이 자동으로 생성됩니다. DB 인스턴스를 생성할 때 기본 VPC를 선택하고 DB 서브넷 그룹의 기본값을 사용할 수 있습니다.

설정 요구 사항을 완료한 후에는 요구 사항과 보안 그룹을 사용하여 DB 인스턴스를 생성할 수 있습니다. 그러려면 Amazon RDS DB 인스턴스 생성의 지침을 따르세요. 특정 DB 엔진을 사용하는 DB 인스턴스를 생성하는 방법에 대한 자세한 내용은 다음 표의 관련 설명서를 참조하세요.

데이터베이스 엔진	설명서
MariaDB	MariaDB DB 인스턴스 생성 및 해당 인스턴스에 연결
Microsoft SQL Server	Microsoft SQL Server DB 인스턴스 생성 및 해당 인스턴스에 연결
MySQL	MySQL DB 인스턴스 생성 및 해당 인스턴스에 연결
Oracle	Oracle DB 인스턴스 생성 및 해당 인스턴스에 연결
PostgreSQL	PostgreSQL DB 인스턴스 생성 및 해당 인스턴스에 연결

> **Note**
>
> DB 인스턴스를 생성한 후 연결할 수 없는 경우 Amazon RDS DB 인스턴스에 연결할 수 없음의 문제 해결 정보를 참조하십시오.

Amazon RDS 시작하기

다음 예제에서는 Amazon Relational Database Service(Amazon RDS)를 사용하여 DB 인스턴스를 생성하여 연결하는 방법을 확인할 수 있습니다. Db2, MariaDB, MySQL, Microsoft SQL Server, Oracle 또는 PostgreSQL을 사용하는 DB 인스턴스를 생성할 수 있습니다.

> **Important**
>
> DB 인스턴스를 생성하거나 DB 인스턴스에 연결하려면 먼저 Amazon RDS 환경 설정 섹션의 태스크를 완료해야 합니다.

DB 인스턴스를 생성한 후 DB 인스턴스의 데이터베이스에 연결하는 작업은 DB 엔진마다 조금씩 다릅니다. 따라서 DB 인스턴스를 생성하여 연결하는 데 필요한 세부 정보를 감안하여 다음 DB 엔진 중 사용할 엔진을 선택합니다. DB 인스턴스를 생성하여 연결하고 나면 DB 인스턴스 삭제를 도와주는 지침을 이용할 수 있습니다.

주제

- MariaDB DB 인스턴스 생성 및 해당 인스턴스에 연결
- Microsoft SQL Server DB 인스턴스 생성 및 해당 인스턴스에 연결
- MySQL DB 인스턴스 생성 및 해당 인스턴스에 연결
- Oracle DB 인스턴스 생성 및 해당 인스턴스에 연결
- PostgreSQL DB 인스턴스 생성 및 해당 인스턴스에 연결
- 자습서: 웹 서버 및 Amazon RDS DB 인스턴스 생성
- 자습서: Amazon RDS에 액세스하기 위해 Lambda 함수 사용

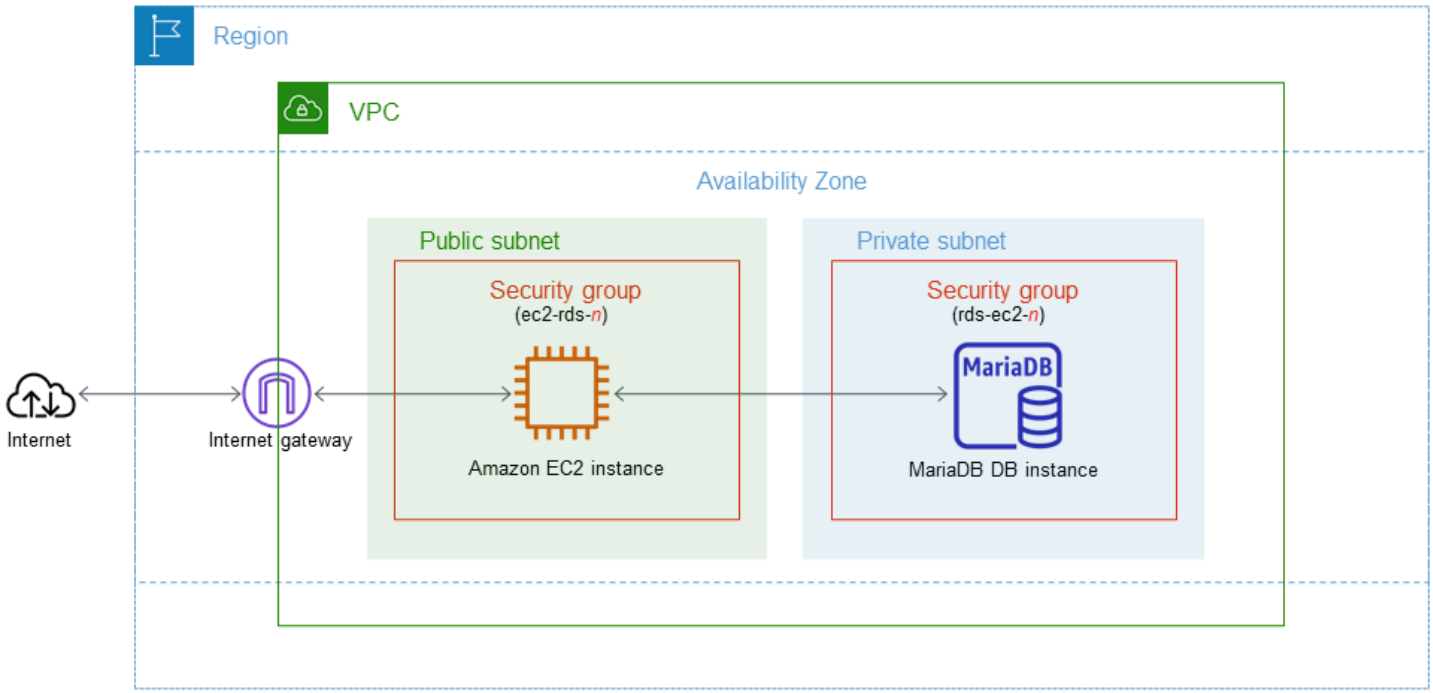
MariaDB DB 인스턴스 생성 및 해당 인스턴스에 연결

이 자습서에서는 EC2 인스턴스와 RDS for MariaDB DB 인스턴스를 생성합니다. 자습서에서는 표준 MySQL 클라이언트를 사용하여 EC2 인스턴스에서 DB 인스턴스에 액세스하는 방법을 보여줍니다. 이 자습서에서는 모범 사례를 따라 Virtual Private Cloud(VPC)에서 프라이빗 DB 인스턴스를 생성합니다. 대부분의 경우 EC2 인스턴스와 같이 동일한 VPC에 있는 다른 리소스는 DB 인스턴스에 액세스할 수 있지만 VPC 외부의 리소스는 DB 인스턴스에 액세스할 수 없습니다.

자습서를 완료하면 VPC의 각 가용 영역에 퍼블릭 서브넷과 프라이빗 서브넷이 있을 것입니다. 한 가용 영역에서 EC2 인스턴스는 퍼블릭 서브넷에 있고 DB 인스턴스는 프라이빗 서브넷에 있습니다.

> **Important**
>
> AWS 계정 생성은 무료입니다. 그러나 이 자습서를 완료하면 사용하는 리소스에 대한 비용이 발생할 수 있습니다. 자습서가 더 이상 필요하지 않은 경우 자습서를 완료한 후에 이러한 리소스를 삭제할 수 있습니다.

다음 다이어그램은 이 자습서를 완료했을 때 구성을 보여 줍니다.



이 자습서에서는 다음 방법 중 하나를 사용하여 리소스를 생성할 수 있습니다.

1. AWS Management Console 사용 - EC2 인스턴스 생성 및 MariaDB DB 인스턴스 생성

2. 데이터베이스 인스턴스 및 EC2 인스턴스를 생성하는 데 AWS CloudFormation 사용 - (선택 사항) AWS CloudFormation를 사용하여 VPC, EC2 인스턴스 및 MariaDB 인스턴스 생성

첫 번째 방법은 간편 생성을 사용하여 AWS Management Console을 통해 프라이빗 MariaDB DB 인스턴스를 생성합니다. 여기에서는 DB 엔진 유형, DB 인스턴스 크기 및 DB 인스턴스 식별자만 지정합니다. [간편 생성(Easy create)]은 다른 구성 옵션에서도 기본 설정을 사용합니다.

표준 생성을 대신 사용하는 경우에는 DB 인스턴스를 생성할 때 더 많은 구성 옵션을 지정할 수 있습니다. 이러한 옵션에는 가용성, 보안, 백업 및 유지 관리에 대한 설정이 포함됩니다. 퍼블릭 DB 인스턴스를 만들려면 표준 생성을 사용해야 합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.

주제

- 사전 조건
- EC2 인스턴스 생성
- MariaDB DB 인스턴스 생성
- (선택 사항) AWS CloudFormation를 사용하여 VPC, EC2 인스턴스 및 MariaDB 인스턴스 생성
- MariaDB DB 인스턴스에 연결
- EC2 인스턴스 및 DB 인스턴스 삭제
- (선택 사항) CloudFormation으로 생성한 EC2 인스턴스 및 DB 인스턴스 삭제
- (선택 사항) DB 인스턴스를 Lambda 함수에 연결

사전 조건

시작하기 전에 다음 섹션에서 다음 단계를 완료하세요.

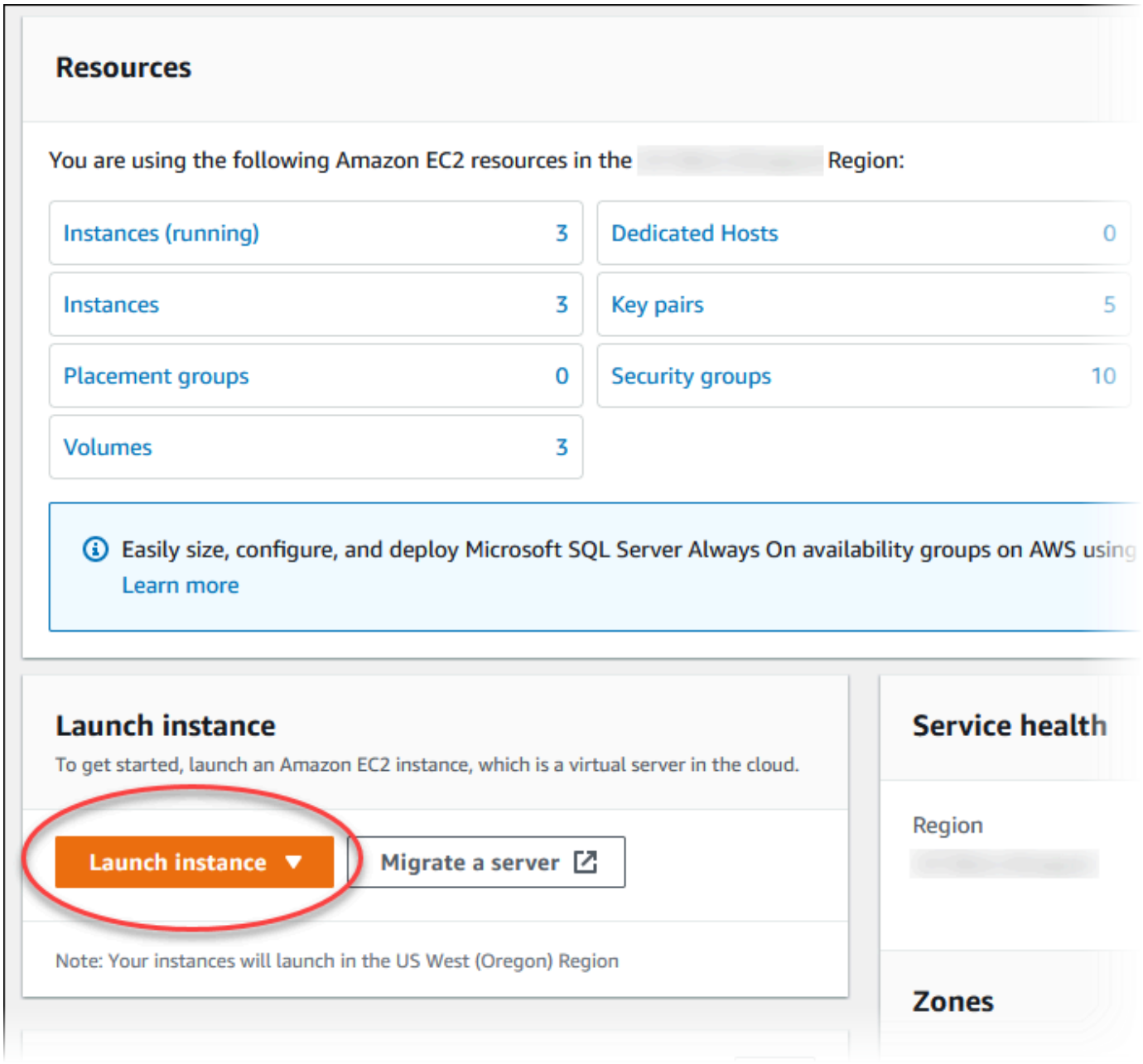
- AWS 계정에 등록
- 관리자 액세스 권한이 있는 사용자 생성

EC2 인스턴스 생성

데이터베이스에 연결하는 데 사용할 Amazon EC2 인스턴스를 생성합니다.

EC2 인스턴스를 생성하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. AWS Management Console의 오른쪽 상단에서 EC2 인스턴스를 생성하려는 AWS 리전을 선택합니다.
3. 다음 이미지에 나와 있는 것처럼 EC2 대시보드를 선택한 다음, 인스턴스 시작을 선택합니다.

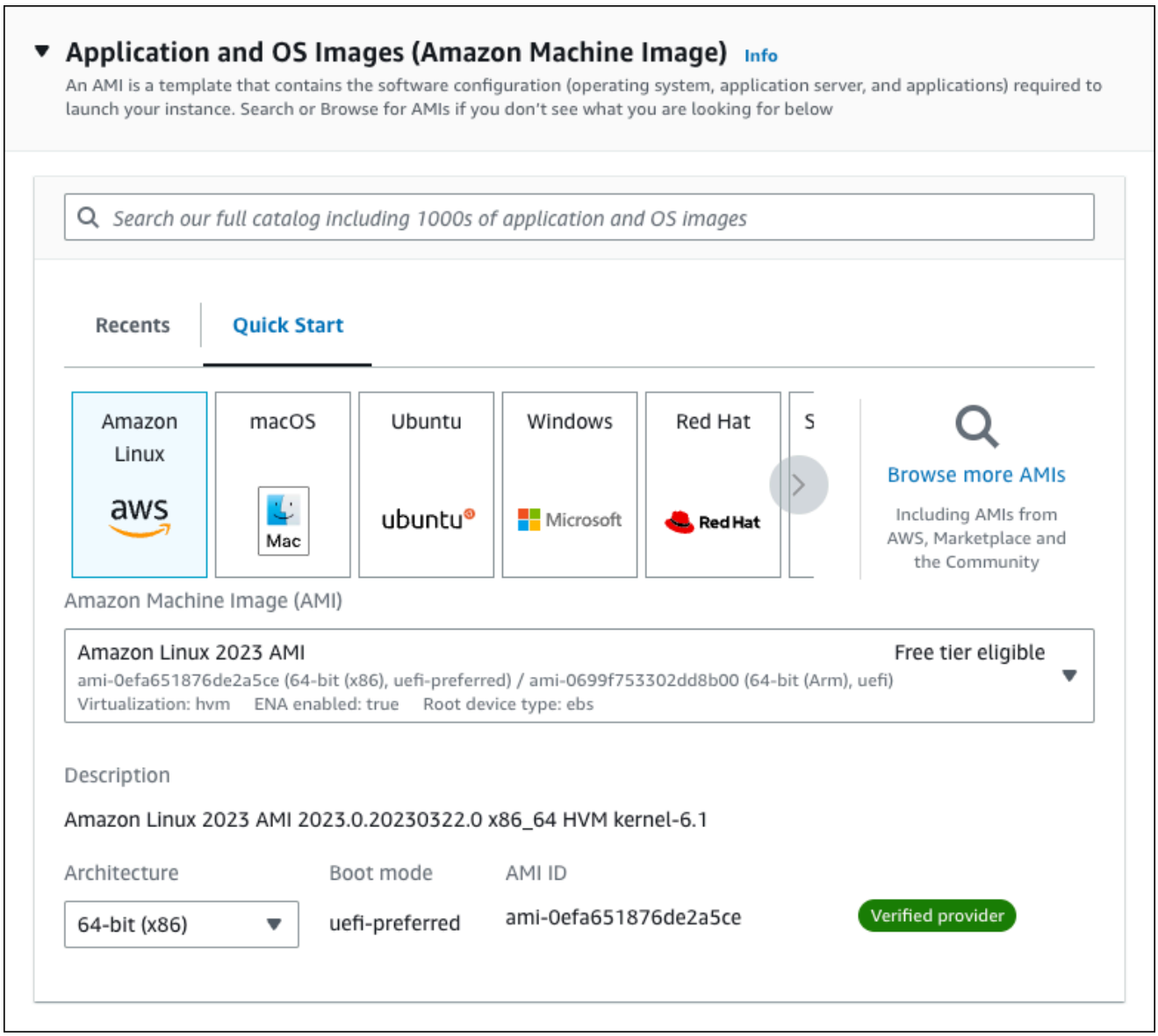


인스턴스 시작 페이지가 열립니다.

4. 인스턴스 시작 페이지에서 다음 설정을 선택합니다.

   a. Name and tags(이름 및 태그) 아래의 Name(이름)에 **ec2-database-connect**을 입력하세요.

   b. Application and OS Images (Amazon Machine Image)(애플리케이션 및 OS 이미지(Amazon Machine Image))에서 Amazon Linux를 선택한 다음 Amazon Linux 2023 AMI를 선택합니다. 다른 선택 항목에 대해서는 기본값을 그대로 유지합니다.



   c. 인스턴스 유형에서 t2.micro를 선택합니다.

   d. 키 페어(로그인)에서 기존 키 페어를 사용할 키 페어 이름을 선택합니다. Amazon EC2 인스턴스에 대한 새 키 페어를 생성하려면 새 키 페어 생성을 선택한 다음 키 페어 생성 창을 사용하여 생성합니다.

새 키 페어 생성에 대한 자세한 내용은 Amazon EC2 Linux 인스턴스용 사용 설명서의 키 페어 생성을 참조하세요.

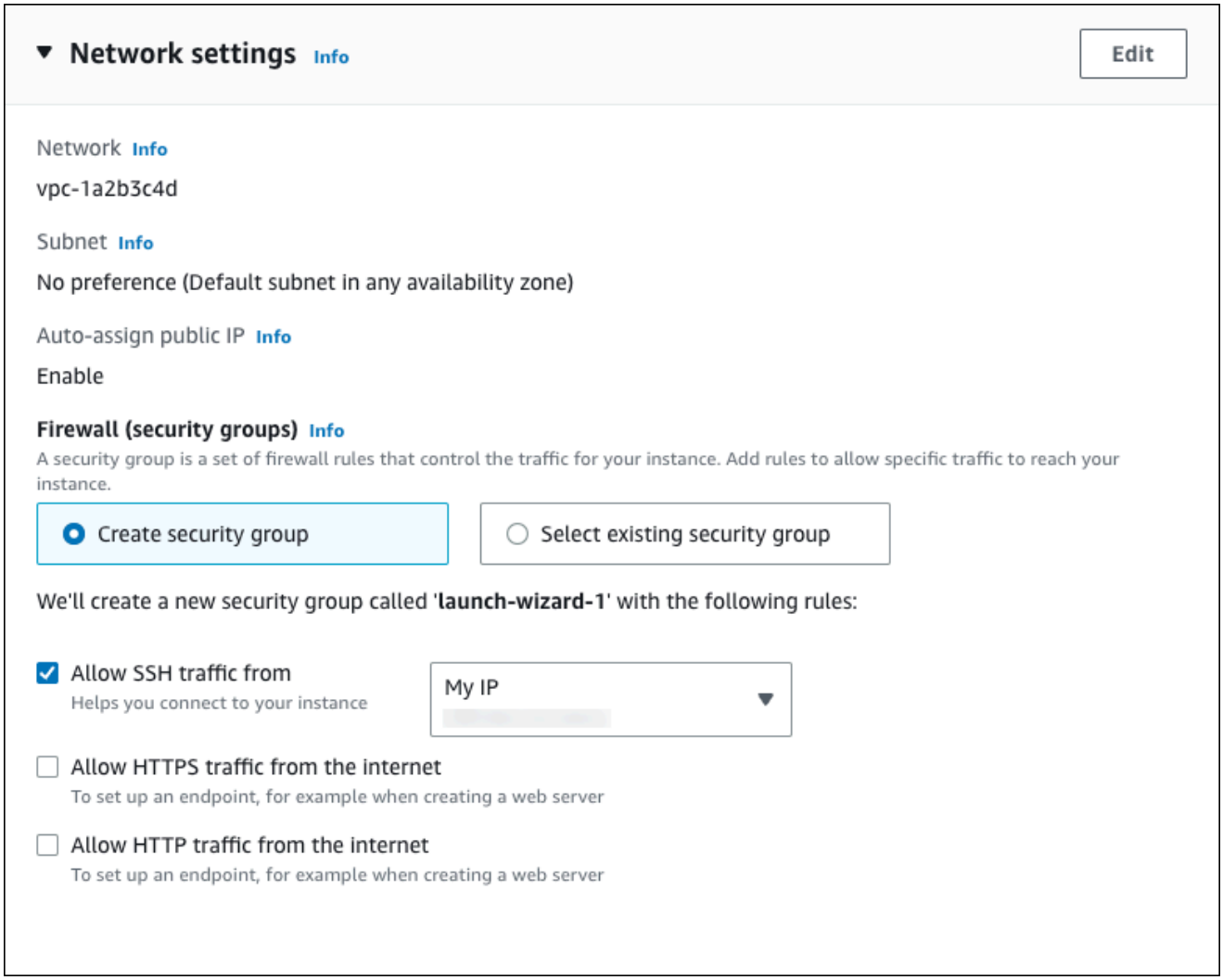
e. 네트워크 설정의 SSH 트래픽 허용에서 EC2 인스턴스에 대한 SSH 연결 소스를 선택합니다.

표시된 IP 주소가 SSH 연결에 대해 올바른 경우 내 IP를 선택할 수 있습니다. 그렇지 않으면 SSH(Secure Shell)를 사용하여 VPC의 EC2 인스턴스에 연결하는 데 사용할 IP 주소를 결정할 수 있습니다. 퍼블릭 IP 주소를 확인하려면 다른 브라우저 창 또는 탭에서 https://checkip.amazonaws.com의 서비스를 사용합니다. IP 주소의 예는 192.0.2.1/32입니다.

대부분의 경우 고정 IP 주소가 없는 방화벽 뒤나 인터넷 서비스 제공업체(ISP)를 통해 연결하는 경우가 많습니다. 그렇다면 클라이언트 컴퓨터에서 사용하는 IP 주소 범위를 결정합니다.

> **Warning**
>
> SSH 액세스에 `0.0.0.0/0`을 사용하는 경우 모든 IP 주소가 SSH를 사용하여 퍼블릭 EC2 인스턴스에 액세스할 수 있도록 활성화합니다. 이 방법은 테스트 환경에서 잠시 사용하는 것은 괜찮지만 프로덕션 환경에서는 안전하지 않습니다. 프로덕션에서는 SSH를 사용하여 EC2 인스턴스에 액세스할 수 있는 특정 IP 주소 또는 주소 범위만 인증합니다.

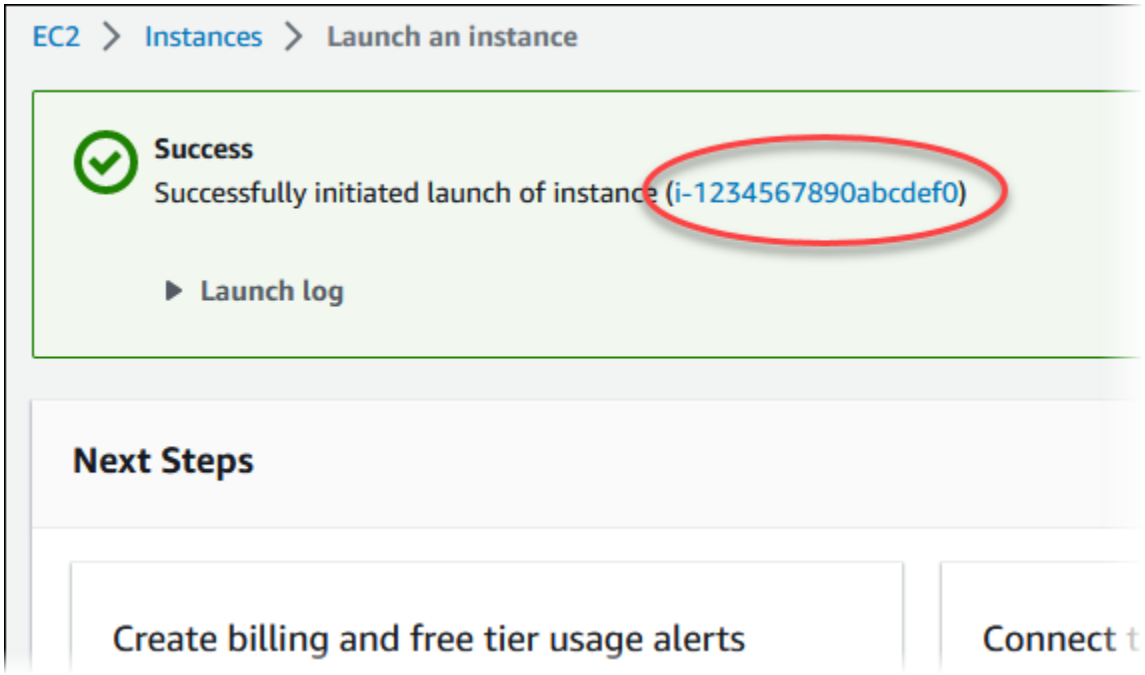
다음 이미지는 네트워크 설정 섹션의 예를 보여 줍니다.



f. 나머지 섹션에서 기본값은 그대로 둡니다.

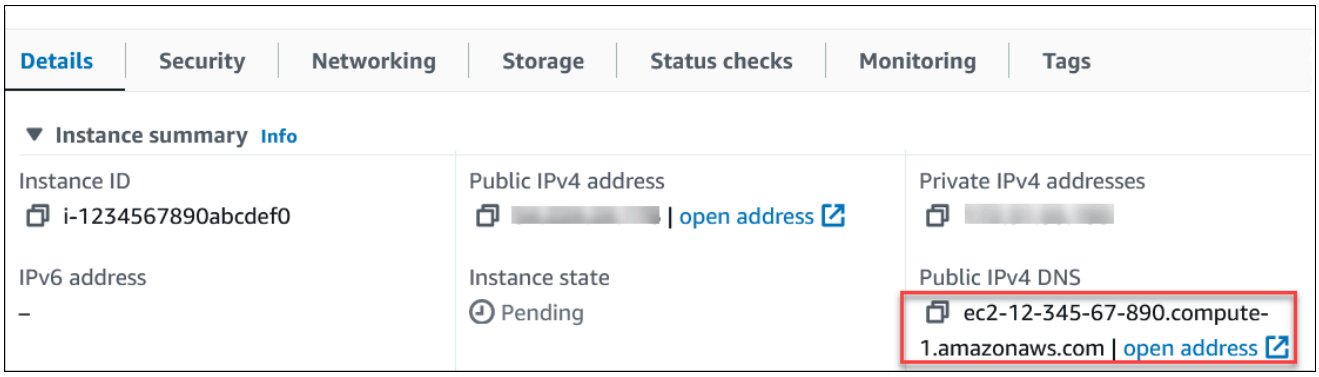
g. 요약 패널에서 EC2 인스턴스 구성 요약을 검토하고 준비가 되면 인스턴스 시작을 선택합니다.

5. 시작 상태 페이지에서, 새 EC2 인스턴스의 식별자(예: `i-1234567890abcdef0`)를 기록해 둡니다.

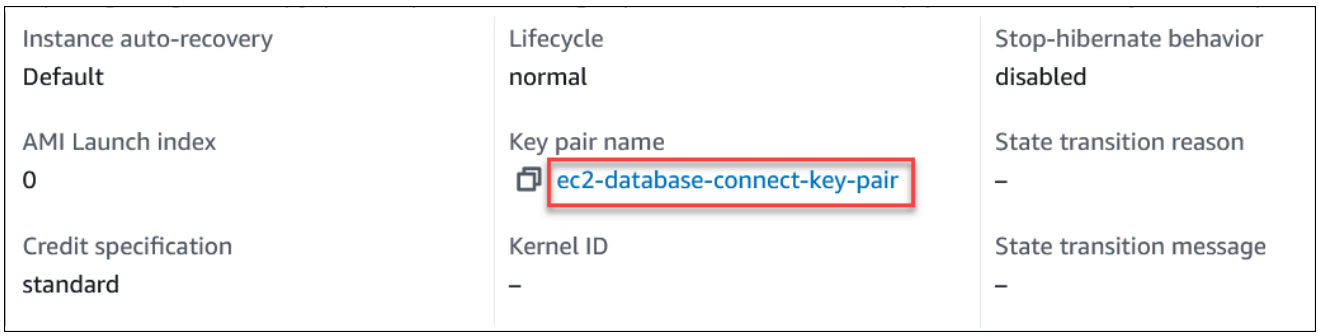


6. EC2 인스턴스 식별자를 선택하여 EC2 인스턴스 목록을 열고 EC2 인스턴스를 선택합니다.
7. 세부 정보 탭에서 SSH를 사용하여 연결할 때 필요한 다음 값을 기록해 둡니다.

   a. 인스턴스 요약에서 퍼블릭 IPv4 DNS의 값을 기록해 둡니다.



   b. 인스턴스 세부 정보에서 키 페어 이름의 값을 기록해 둡니다.



8. 계속하기 전에 EC2 인스턴스의 인스턴스 상태가 실행 중이 될 때까지 기다립니다.

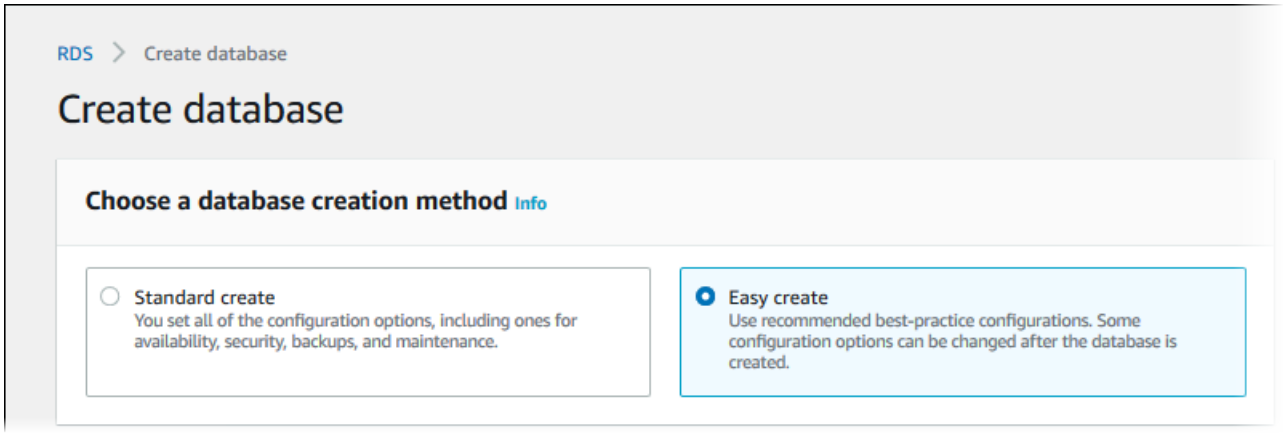
MariaDB DB 인스턴스 생성

Amazon RDS의 기본 빌딩 블록은 DB 인스턴스입니다. MariaDB 데이터베이스를 실행하는 환경입니다.

이 예시에서는 간편 생성을 사용하여 db.t4g.micro DB 인스턴스 클래스에서 MariaDB 데이터베이스 엔진을 실행하는 DB 인스턴스를 만듭니다.

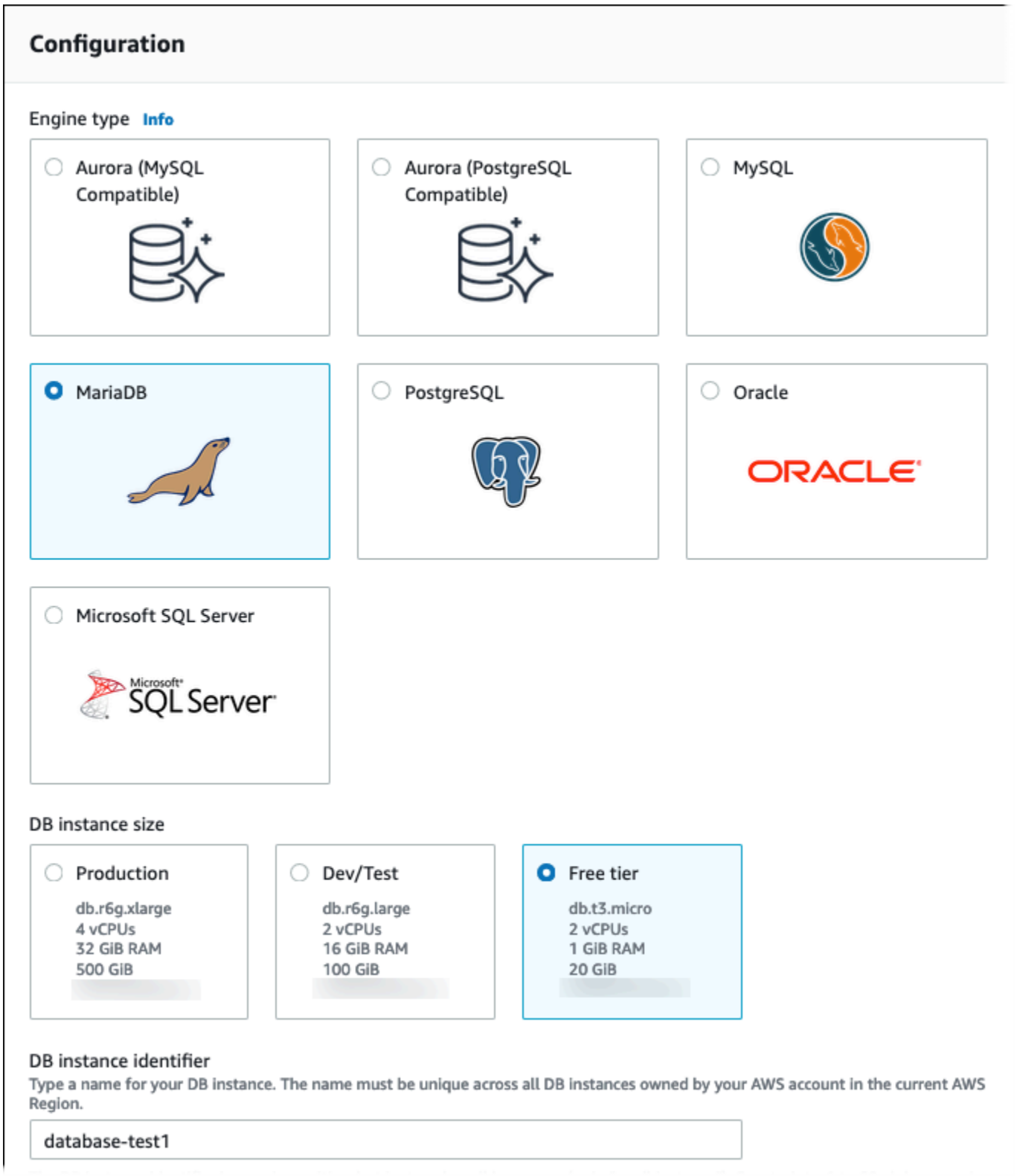
간편 생성(Easy create)을 사용하여 MariaDB DB 인스턴스를 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 콘솔의 오른쪽 상단에서 DB 인스턴스를 생성하려는 AWS 리전을 선택합니다.
3. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
4. 데이터베이스 생성에서 간편 생성을 선택합니다.



5. 구성에서 MariaDB를 선택합니다.
6. DB 인스턴스 크기에서 프리 티어 또는 샌드박스를 선택합니다. 프리 티어는 무료 요금제 계정에 표시됩니다. 샌드박스는 유료 요금제 계정에 표시됩니다.
7. DB 인스턴스 식별자에 **`database-test1`**을 입력합니다.
8. 마스터 사용자 이름에 마스터 사용자의 이름을 입력하거나 기본 이름을 그대로 유지합니다.

   데이터베이스 생성 페이지는 다음 이미지와 비슷해야 합니다. 무료 요금제 계정에는 프리 티어가 표시됩니다. 유료 요금제 계정에는 샌드박스가 표시됩니다.

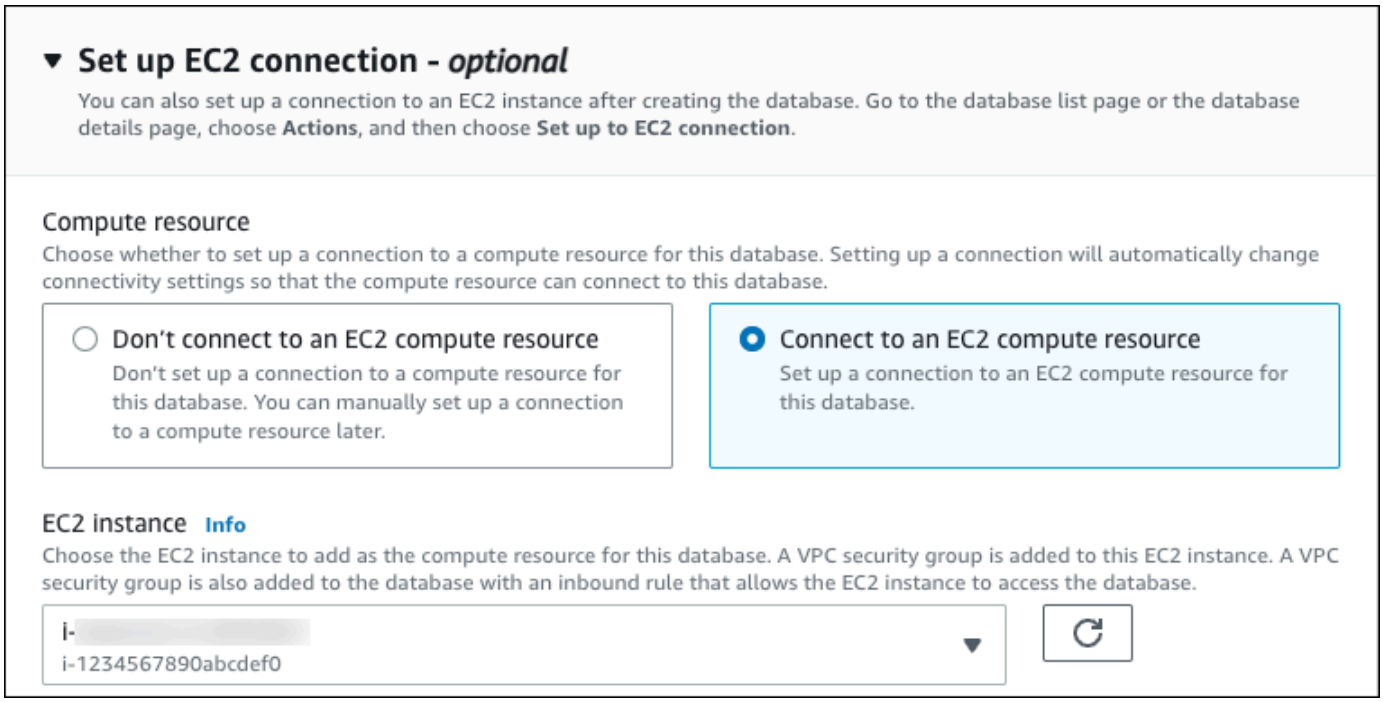


9. DB 인스턴스에 자동 생성된 마스터 암호를 사용하려면 암호 자동 생성을 선택합니다.

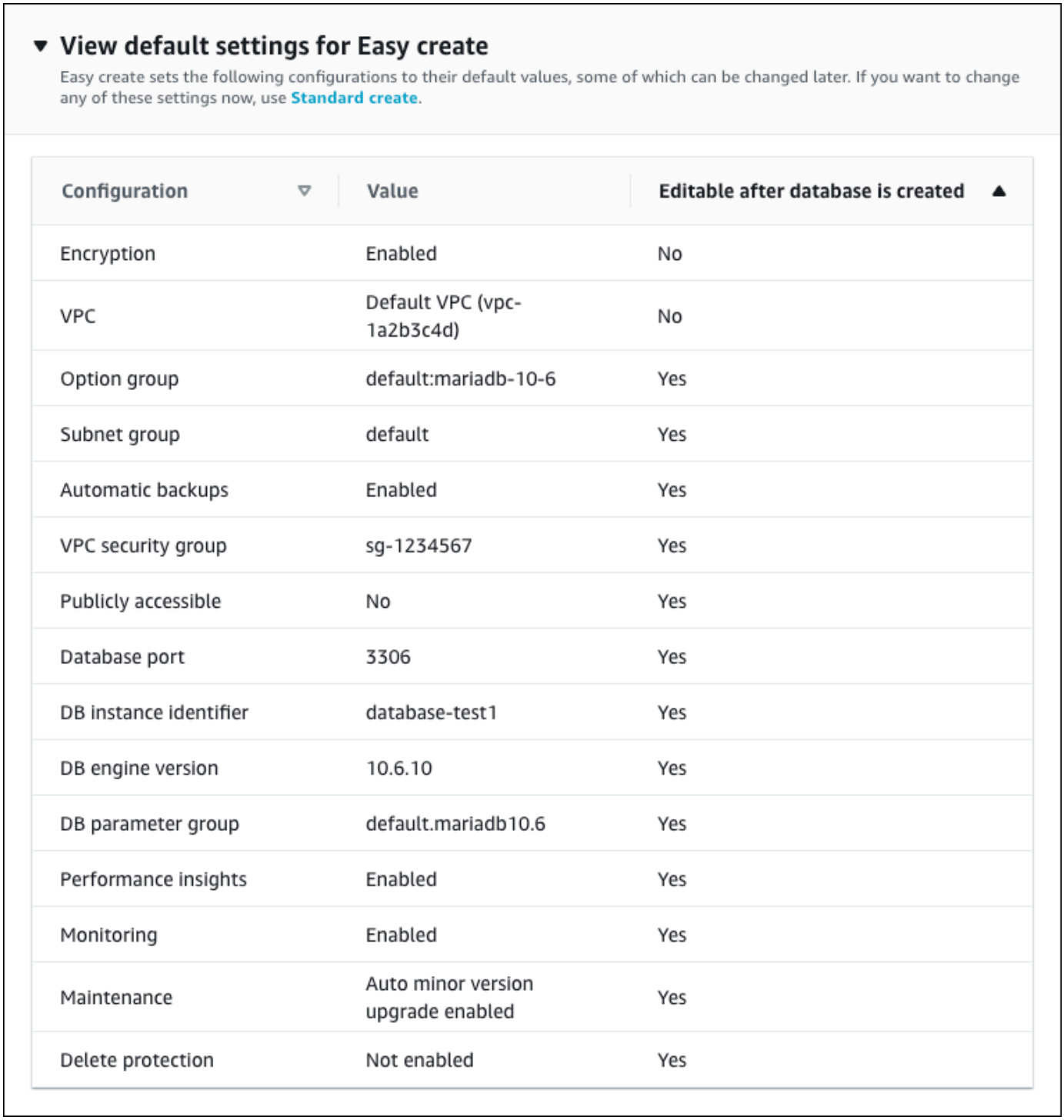
마스터 암호를 입력하려면 암호 자동 생성을 지우고 마스터 암호 및 마스터 암호 확인에 동일한 암호를 입력합니다.

10. 이전에 만든 EC2 인스턴스와의 연결을 설정하려면 EC2 연결 설정 - 선택 사항을 확장합니다.

    Connect to an EC2 compute resource(EC2 컴퓨팅 리소스에 연결)를 선택합니다. 이전에 생성한 EC2 인스턴스를 선택합니다.



11. 간편 생성 기본 설정 보기를 확장합니다.



View default settings for Easy create

Easy create sets the following configurations to their default values, some of which can be changed later. If you want to change any of these settings now, use **Standard create**.

Configuration	Value	Editable after database is created
Encryption	Enabled	No
VPC	Default VPC (vpc-1a2b3c4d)	No
Option group	default:mariadb-10-6	Yes
Subnet group	default	Yes
Automatic backups	Enabled	Yes
VPC security group	sg-1234567	Yes
Publicly accessible	No	Yes
Database port	3306	Yes
DB instance identifier	database-test1	Yes
DB engine version	10.6.10	Yes
DB parameter group	default.mariadb10.6	Yes
Performance insights	Enabled	Yes
Monitoring	Enabled	Yes
Maintenance	Auto minor version upgrade enabled	Yes
Delete protection	Not enabled	Yes

[간편 생성(Easy Create)]과 함께 사용되는 기본 설정을 검토할 수 있습니다. 데이터베이스 생성 후 편집 가능 열에는 데이터베이스 생성 후 어떤 옵션을 변경할 수 있는지 나와 있습니다.

- 설정의 해당 열에 아니요라고 되어 있지만 다른 설정을 원하는 경우 표준 생성을 사용하여 DB 인스턴스를 만들 수 있습니다.

- 설정의 해당 열에 예라고 되어 있으며 다른 설정을 원하는 경우 표준 생성을 사용하여 DB 인스턴스를 만들거나 DB 인스턴스를 생성한 후 수정하여 설정을 변경할 수 있습니다.

12. 데이터베이스 생성를 선택합니다.

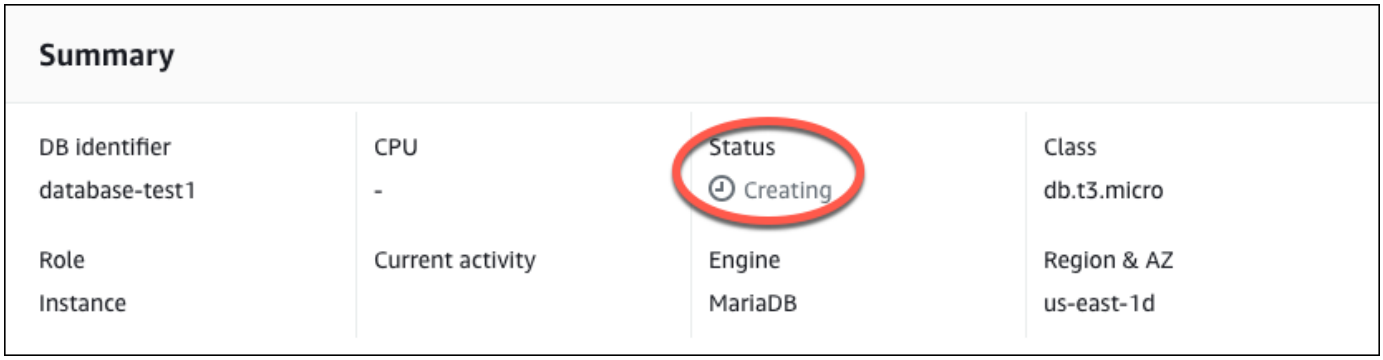
    DB 인스턴스의 마스터 사용자 이름 및 암호를 보려면 View credential details(자격 증명 세부 정보 보기)를 선택합니다.

    DB 인스턴스를 마스터 사용자로 연결하려면 화면에 나타난 사용자 이름과 암호를 사용합니다.

    > **Important**
    >
    > 마스터 사용자 암호를 다시 볼 수는 없습니다. 따라서 기록을 해두지 않으면 이를 변경해야 합니다.
    > DB 인스턴스가 사용 가능한 상태가 되고 난 후에 마스터 사용자 암호를 변경해야 하는 경우에는 다음과 같은 방법으로 DB 인스턴스를 수정할 수 있습니다. DB 인스턴스 변경에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

13. 데이터베이스 목록에서 새 MariaDB DB 인스턴스의 이름을 선택하면 세부 정보가 표시됩니다.

    DB 인스턴스를 사용할 준비가 될 때까지 DB 인스턴스의 상태는 생성 중입니다.

    상태가 Available(사용 가능)로 변경되면 DB 인스턴스에 연결할 수 있습니다. DB 인스턴스 클래스와 스토리지의 양에 따라 새 인스턴스를 사용할 수 있을 때까지 최대 20분이 걸릴 수 있습니다.

(선택 사항) AWS CloudFormation를 사용하여 VPC, EC2 인스턴스 및 MariaDB 인스턴스 생성

콘솔을 사용하여 VPC, EC2 인스턴스 및 MariaDB 인스턴스를 생성하는 대신 AWS CloudFormation을 통해 코드형 인프라로 처리하여 AWS 리소스를 프로비저닝할 수 있습니다. AWS 리소스를 더 작고 관

리하기 쉬운 단위로 구성하는 데 도움이 되도록 AWS CloudFormation 중첩 스택 기능을 사용할 수 있습니다. 자세한 내용은 AWS CloudFormation 콘솔에서 스택 생성 및 중첩된 스택 작업을 참조하세요.

> **Important**
>
> AWS CloudFormation은 무료이지만, CloudFormation에서 생성하는 리소스는 라이브입니다. 이러한 리소스를 종료하지 않으면 해당 리소스에 대한 표준 사용 요금이 발생합니다. 자세한 내용은 RDS for MariaDB 요금을 참조하세요.

- CloudFormation 템플릿 파일을 다운로드하십시오.
- CloudFormation을 사용하여 리소스 구성

CloudFormation 템플릿 파일을 다운로드하십시오.

CloudFormation 템플릿은 JSON 또는 YAML 텍스트 파일로, 스택에서 생성하려는 리소스에 대한 구성 정보가 들어 있습니다. 또한 이 템플릿은 RDS 인스턴스와 함께 VPC와 Bastion Host를 생성합니다.

템플릿 파일을 다운로드하려면 다음 링크인 MariaDB CloudFormation 템플릿을 엽니다.

Github 페이지에서 원시 파일 다운로드 버튼을 클릭하여 템플릿 YAML 파일을 저장합니다.

CloudFormation을 사용하여 리소스 구성

> **Note**
>
> 이 프로세스를 시작하기 전에 AWS 계정에 EC2 인스턴스용 키 페어가 있는지 확인합니다. 자세한 내용은 Amazon EC2 키 페어 및 Linux 인스턴스를 참조하세요.

AWS CloudFormation 템플릿을 사용할 때는 리소스가 제대로 생성되도록 올바른 파라미터를 선택해야 합니다. 다음 단계를 따릅니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/cloudformation에서 AWS CloudFormation 콘솔을 엽니다.
2. 스택 생성을 선택합니다.
3. 템플릿 지정 섹션에서 컴퓨터에서 템플릿 파일 업로드를 선택하고 다음을 선택합니다.

4. 스택 세부 정보 지정 페이지에서 다음 파라미터를 설정합니다.
   a. 스택 이름을 MariaDBTestStack으로 설정합니다.
   b. 파라미터에서 가용 영역 3개를 선택하여 가용 영역을 설정합니다.
   c. Linux Bastion Host 구성에서 키 이름에 대해 EC2 인스턴스에 로그인할 키 페어를 선택합니다.
   d. Linux Bastion Host 구성 설정에서 허용된 IP 범위를 IP 주소로 설정합니다. Secure Shell(SSH)을 사용하여 VPC의 EC2 인스턴스에 연결하려면 https://checkip.amazonaws.com의 서비스를 사용하여 퍼블릭 IP 주소를 지정합니다. IP 주소의 예는 192.0.2.1/32입니다.

      > **Warning**
      >
      > SSH 액세스에 `0.0.0.0/0`을 사용하는 경우 모든 IP 주소가 SSH를 사용하여 퍼블릭 EC2 인스턴스에 액세스할 수 있도록 활성화합니다. 이 방법은 테스트 환경에서 잠시 사용하는 것은 괜찮지만 프로덕션 환경에서는 안전하지 않습니다. 프로덕션에서는 SSH를 사용하여 EC2 인스턴스에 액세스할 수 있는 특정 IP 주소 또는 주소 범위만 인증합니다.

   e. 데이터베이스 일반 구성에서 데이터베이스 인스턴스 클래스를 db.t3.micro로 설정합니다.
   f. 데이터베이스 이름을 **`database-test1`**으로 설정합니다.
   g. 데이터베이스 마스터 사용자 이름에 마스터 사용자 이름을 입력합니다.
   h. 이 자습서에서는 Secrets Manager를 사용하여 DB 마스터 사용자 암호 관리를 `false`로 설정합니다.
   i. 데이터베이스 암호의 경우 원하는 암호를 설정합니다. 자습서의 향후 단계에 사용할 수 있도록 암호를 기억해 둡니다.
   j. 데이터베이스 스토리지 구성에서 데이터베이스 스토리지 유형을 gp2로 설정합니다.
   k. 데이터베이스 모니터링 구성에서 RDS 성능 개선 도우미 활성화를 false로 설정합니다.
   l. 다른 모든 설정은 기본값으로 둡니다. 계속하려면 다음을 클릭합니다.
5. 스택 검토 페이지에서 데이터베이스 및 Linux Bastion Host 옵션을 확인한 후 제출을 선택합니다.

스택 생성 프로세스가 완료되면 BastionStack 및 RDSNS라는 이름의 스택을 보고 데이터베이스에 연결하는 데 필요한 정보를 기록해 둡니다. 자세한 내용은 AWS Management Console에서 AWS CloudFormation 스택 데이터 및 리소스 보기를 참조하세요.

MariaDB DB 인스턴스에 연결

표준 SQL 클라이언트 애플리케이션을 사용해 DB 인스턴스에 연결할 수 있습니다. 이 예시에서는 mysql 명령줄 클라이언트를 사용해 MariaDB DB 인스턴스에 연결합니다.

MariaDB DB 인스턴스에 연결하는 방법

1. DB 인스턴스의 엔드포인트(DNS 이름)와 포트 번호를 찾습니다.

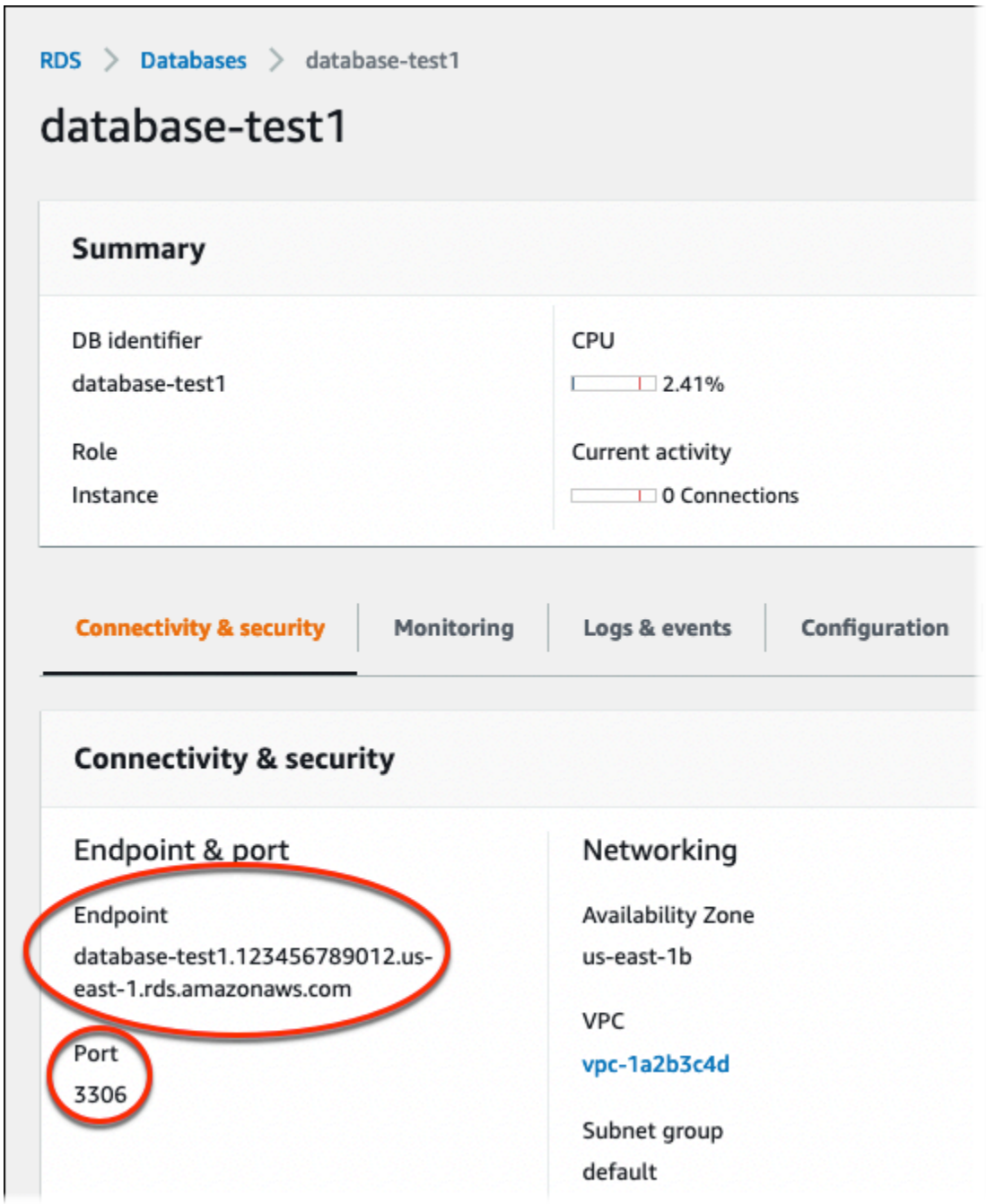
   a. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.

   b. Amazon RDS 콘솔의 오른쪽 상단에서 DB 인스턴스의 AWS 리전을 선택합니다.

   c. 탐색 창에서 Databases(데이터베이스)를 선택합니다.

   d. 세부 정보를 표시하고자 하는 MariaDB DB 인스턴스 이름을 선택합니다.

   e. Connectivity & security(연결 및 보안) 탭에서 엔드포인트를 복사합니다. 또한 포트 번호를 적어 둡니다. DB 인스턴스에 연결하려면 엔드포인트와 포트 번호가 모두 필요합니다.



2. Amazon EC2 사용 설명서에 있는 Linux 인스턴스에 연결의 단계를 따라 앞에서 만든 EC2 인스턴스에 연결합니다.

   SSH를 사용하여 EC2 인스턴스에 연결하는 것이 좋습니다. Windows, Linux 또는 Mac에 SSH 클라이언트 유틸리티가 설치된 경우 다음 명령 형식을 사용하여 인스턴스에 연결할 수 있습니다.

   ```
   ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
   ```

예를 들어 ec2-database-connect-key-pair.pem이 Linux의 /dir1에 저장되어 있고, EC2 인스턴스의 퍼블릭 IPv4 DNS가 ec2-12-345-678-90.compute-1.amazonaws.com이라고 가정해 보겠습니다. SSH 명령은 다음과 같이 표시됩니다.

```
ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
user@ec2-12-345-678-90.compute-1.amazonaws.com
```

3. EC2 인스턴스에서 소프트웨어를 업데이트하여 최신 버그 수정 및 보안 업데이트를 받습니다. 이렇게 하려면 다음 명령을 사용하십시오.

> **Note**
>
> -y 옵션을 사용하면 확인 여부를 묻지 않고 업데이트를 설치합니다. 설치 전에 업데이트 정보를 확인하려면 이 옵션을 생략합니다.

```
sudo dnf update -y
```

4. MariaDB의 mysql 명령줄 클라이언트를 설치합니다.

   Amazon Linux 2023에서 MariaDB 명령줄 클라이언트를 설치하려면 다음 명령을 실행합니다.

```
sudo dnf install mariadb105
```

5. MariaDB DB 인스턴스에 연결합니다. 예를 들어, 다음 명령을 입력합니다. 이 작업을 통해 MySQL 클라이언트를 사용하여 MariaDB DB 인스턴스에 연결할 수 있습니다.

   *endpoint*는 DB 인스턴스 엔드포인트(DNS 이름)로 대체하고, *admin*는 사용된 마스터 사용자 이름으로 대체합니다. 암호를 묻는 메시지가 표시되면 사용한 마스터 암호를 제공합니다.

```
mysql -h endpoint -P 3306 -u admin -p
```

   사용자에 대한 암호를 입력하면 다음과 유사한 출력이 나타납니다.

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 156
Server version: 10.6.10-MariaDB-log managed by https://aws.amazon.com/rds/

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

MariaDB DB 인스턴스 연결에 대한 자세한 내용은 MariaDB DB 인스턴스에 연결 단원을 참조하십시오. DB 인스턴스에 연결할 수 없는 경우 Amazon RDS DB 인스턴스에 연결할 수 없음 단원을 참조하십시오.

보안을 위해서는 암호화된 연결을 사용하는 것이 가장 좋습니다. 클라이언트와 서버가 동일한 VPC에 있고 네트워크를 신뢰할 수 있는 경우에만 암호화되지 않은 MariaDB 연결을 사용합니다. 암호화된 연결 사용에 대한 자세한 내용은 MySQL 명령줄 클라이언트(암호화됨)에서 SSL/TLS를 사용하여 Amazon RDS의 MariaDB DB 인스턴스에 연결 섹션을 참조하세요.

6. SQL 명령을 실행합니다.

   예를 들어, 다음 SQL 명령은 현재 날짜 및 시간을 보여줍니다.

   ```
   SELECT CURRENT_TIMESTAMP;
   ```

EC2 인스턴스 및 DB 인스턴스 삭제

생성한 샘플 EC2 인스턴스 및 DB 인스턴스에 연결하고 탐색한 후에는 요금이 더 이상 부과되지 않도록 삭제합니다.

AWS CloudFormation을 사용하여 리소스를 생성했다면 이 단계를 건너뛰고 다음 단계로 이동합니다.

EC2 인스턴스를 삭제하는 방법

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창에서 인스턴스를 선택합니다.
3. EC2 인스턴스를 선택하고 인스턴스 상태, 인스턴스 종료를 차례로 선택합니다.
4. 확인 메시지가 나타나면 종료를 선택합니다.

EC2 인스턴스 삭제에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 종료를 참조하세요.

최종 DB 스냅샷이 없는 DB 인스턴스를 삭제하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 삭제할 DB 인스턴스를 선택합니다.
4. [Actions]에 대해 [Delete]를 선택합니다.
5. 최종 스냅샷을 생성하시겠습니까? 및 자동 백업 보존을 선택 해제합니다.
6. 확인을 완료하고 삭제를 선택합니다.

(선택 사항) CloudFormation으로 생성한 EC2 인스턴스 및 DB 인스턴스 삭제

AWS CloudFormation을 사용하여 리소스를 생성한 경우 샘플 EC2 인스턴스 및 DB 인스턴스에 연결하고 탐색한 후 CloudFormation 스택을 삭제하여 더 이상 비용이 청구되지 않도록 합니다.

CloudFormation 리소스를 삭제하려면

1. AWS CloudFormation 콘솔을 엽니다.
2. CloudFormation 콘솔의 스택 페이지에서 루트 스택(VPCStack, BastionStack 또는 RDSNS라는 이름이 없는 스택)을 선택합니다.
3. 삭제를 선택합니다.
4. 확인 메시지가 나타나면 스택 삭제를 선택합니다.

CloudFormation에서 스택을 삭제하는 방법에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS CloudFormation 콘솔에서 스택 삭제를 참조하세요.

(선택 사항) DB 인스턴스를 Lambda 함수에 연결

RDS for MariaDB DB 인스턴스를 Lambda 서버리스 컴퓨팅 리소스에 연결할 수도 있습니다. Lambda 함수를 사용하면 인프라를 프로비저닝하거나 관리하지 않고도 코드를 실행할 수 있습니다. 또한 하루 12개에서 초당 수백 개에 이르는 모든 규모의 코드 실행 요청에 자동으로 응답할 수 있습니다. 자세한 내용은 Lambda 함수와 DB 인스턴스 자동 연결 섹션을 참조하세요.

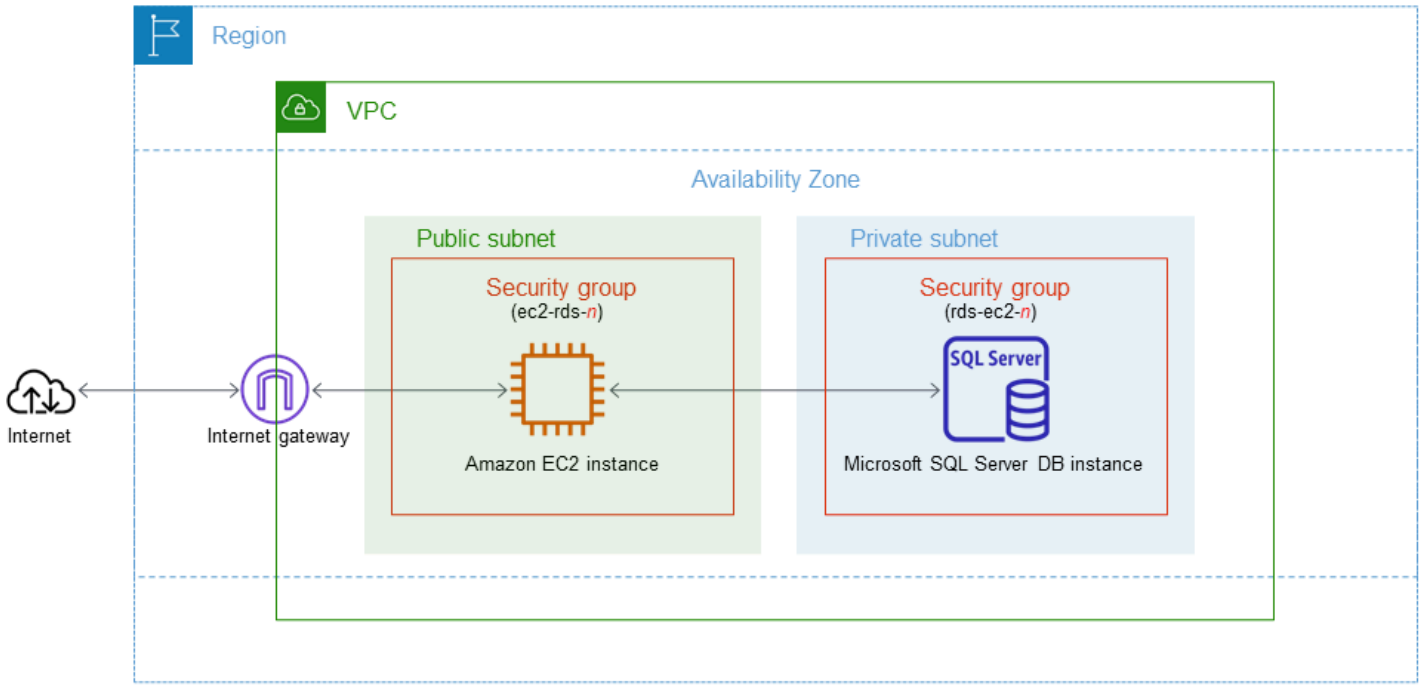
Microsoft SQL Server DB 인스턴스 생성 및 해당 인스턴스에 연결

이 자습서에서는 EC2 인스턴스와 RDS for Microsoft SQL Server DB 인스턴스를 생성합니다. 자습서에서는 Microsoft SQL Server Management Studio 클라이언트를 사용하여 EC2 인스턴스에서 DB 인스턴스에 액세스하는 방법을 보여줍니다. 이 자습서에서는 모범 사례를 따라 Virtual Private Cloud(VPC)에서 프라이빗 DB 인스턴스를 생성합니다. 대부분의 경우 EC2 인스턴스와 같이 동일한 VPC에 있는 다른 리소스는 DB 인스턴스에 액세스할 수 있지만 VPC 외부의 리소스는 DB 인스턴스에 액세스할 수 없습니다.

자습서를 완료하면 VPC의 각 가용 영역에 퍼블릭 서브넷과 프라이빗 서브넷이 있을 것입니다. 한 가용 영역에서 EC2 인스턴스는 퍼블릭 서브넷에 있고 DB 인스턴스는 프라이빗 서브넷에 있습니다.

> **Important**
>
> AWS 계정 생성은 무료입니다. 그러나 이 자습서를 완료하면 사용하는 AWS 리소스에 대한 비용이 발생할 수 있습니다. 자습서가 더 이상 필요하지 않은 경우 자습서를 완료한 후에 이러한 리소스를 삭제할 수 있습니다.

다음 다이어그램은 이 자습서를 완료했을 때 구성을 보여 줍니다.



이 자습서에서는 다음 방법 중 하나를 사용하여 리소스를 생성할 수 있습니다.

1. AWS Management Console 사용 - SQL Server DB 인스턴스 생성 및 EC2 인스턴스 생성
2. 데이터베이스 인스턴스 및 EC2 인스턴스를 생성하는 데 AWS CloudFormation 사용 - (선택 사항) AWS CloudFormation를 사용하여 VPC, EC2 인스턴스 및 SQL Server 인스턴스 생성

첫 번째 방법은 간편 생성을 사용하여 AWS Management Console을 통해 프라이빗 SQL Server DB 인스턴스를 생성합니다. 여기에서는 DB 엔진 유형, DB 인스턴스 크기 및 DB 인스턴스 식별자만 지정합니다. [간편 생성(Easy create)]은 다른 구성 옵션에서도 기본 설정을 사용합니다.

표준 생성을 대신 사용하는 경우에는 DB 인스턴스를 생성할 때 더 많은 구성 옵션을 지정할 수 있습니다. 이러한 옵션에는 가용성, 보안, 백업 및 유지 관리에 대한 설정이 포함됩니다. 퍼블릭 DB 인스턴스를 만들려면 표준 생성을 사용해야 합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.

주제

- 사전 조건
- EC2 인스턴스 생성
- SQL Server DB 인스턴스 생성
- (선택 사항) AWS CloudFormation를 사용하여 VPC, EC2 인스턴스 및 SQL Server 인스턴스 생성
- SQL Server DB 인스턴스에 연결
- 샘플 SQL Server DB 인스턴스 탐색
- EC2 인스턴스 및 DB 인스턴스 삭제
- (선택 사항) CloudFormation으로 생성한 EC2 인스턴스 및 DB 인스턴스 삭제
- (선택 사항) DB 인스턴스를 Lambda 함수에 연결

사전 조건

시작하기 전에 다음 섹션에서 다음 단계를 완료하세요.

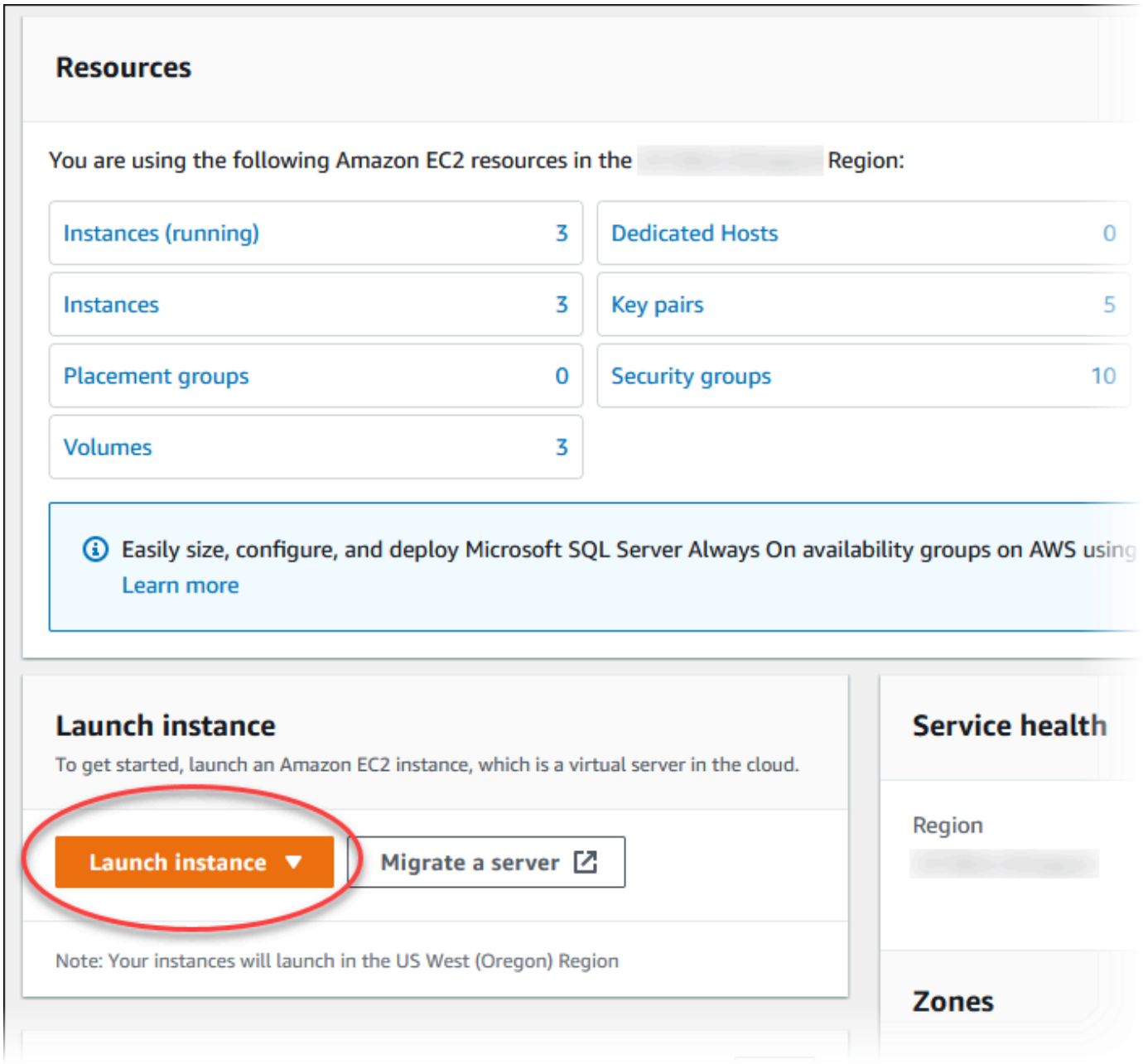
- AWS 계정에 등록
- 관리자 액세스 권한이 있는 사용자 생성

EC2 인스턴스 생성

데이터베이스에 연결하는 데 사용할 Amazon EC2 인스턴스를 생성합니다.

EC2 인스턴스를 생성하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. AWS Management Console의 오른쪽 상단에서 이전에 데이터베이스에 사용한 AWS 리전을 선택합니다.
3. 다음 이미지에 나와 있는 것처럼 EC2 대시보드를 선택한 다음, 인스턴스 시작을 선택합니다.

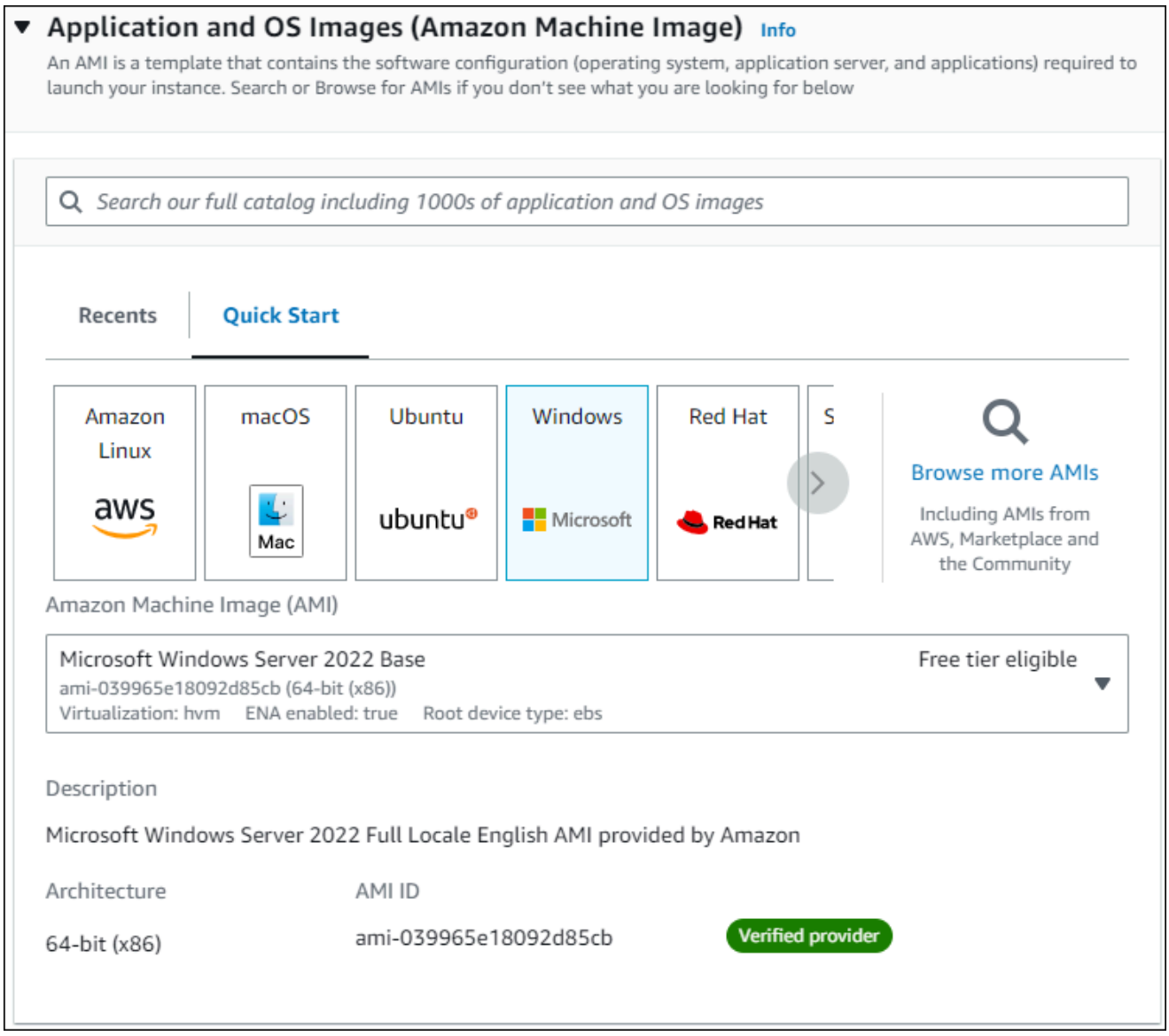


인스턴스 시작 페이지가 열립니다.

4. 인스턴스 시작 페이지에서 다음 설정을 선택합니다.

   a. Name and tags(이름 및 태그) 아래의 Name(이름)에 **ec2-database-connect**을 입력하세요.

   b. Application and OS Images (Amazon Machine Image)(애플리케이션 및 OS 이미지(Amazon Machine Image))에서 Windows를 선택한 다음 Microsoft Windows Server 2022 Base를 선택합니다. 다른 선택 항목에 대해서는 기본값을 그대로 유지합니다.



   c. 인스턴스 유형에서 t2.micro를 선택합니다.

   d. 키 페어(로그인)에서 기존 키 페어를 사용할 키 페어 이름을 선택합니다. Amazon EC2 인스턴스에 대한 새 키 페어를 생성하려면 새 키 페어 생성을 선택한 다음 키 페어 생성 창을 사용하여 생성합니다.

새로운 키 페어 생성에 대한 자세한 내용은 Amazon EC2 Windows 인스턴스용 사용 설명서의 키 페어 생성을 참조하세요.

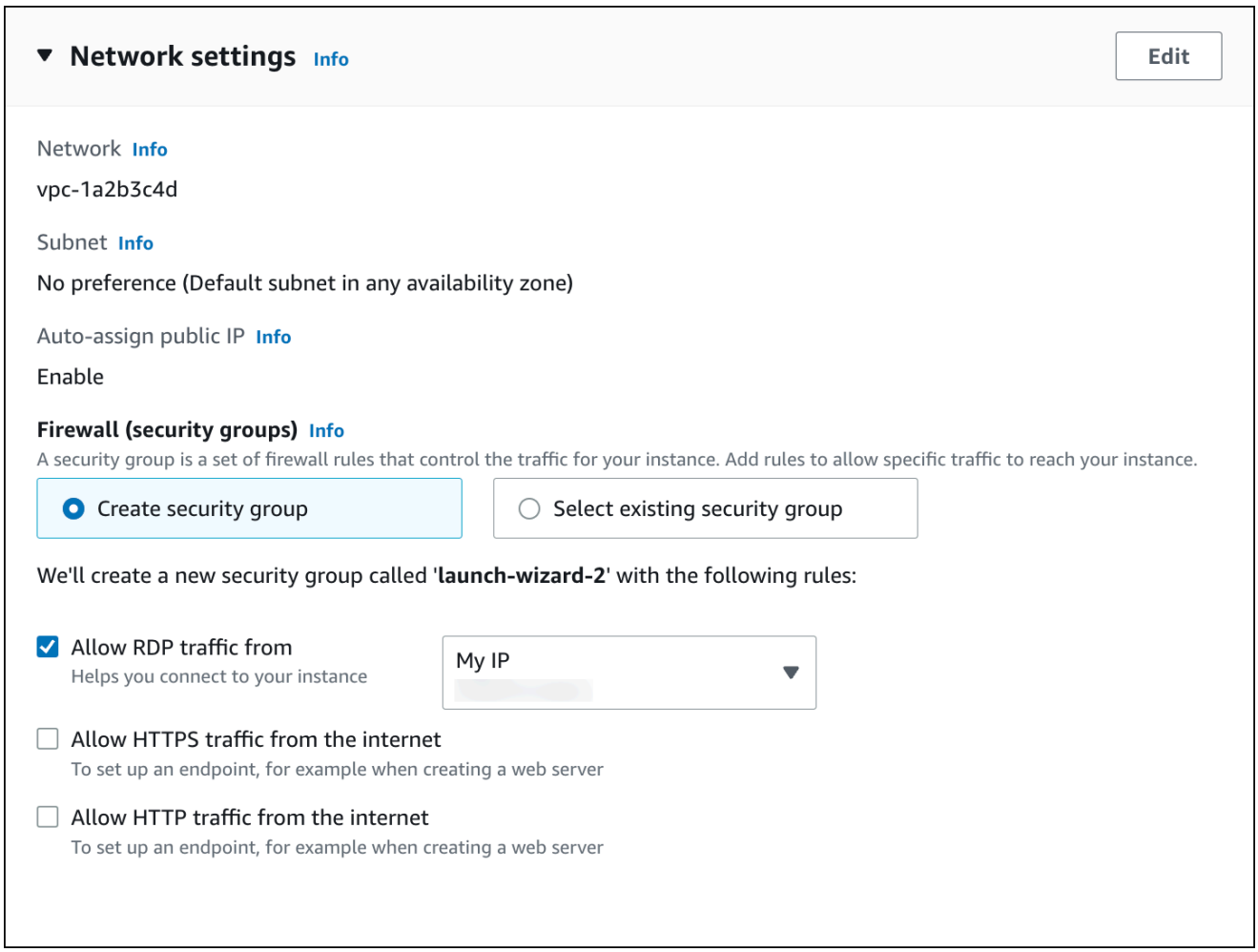
e. 네트워크 설정의 방화벽(보안 그룹)에서 다음 위치의 RDP 트래픽 허용을 선택하여 EC2 인스턴스에 연결합니다.

표시된 IP 주소가 RDP 연결에 대해 올바른 경우 내 IP를 선택할 수 있습니다. 그렇지 않으면 RDP를 사용하여 VPC의 EC2 인스턴스에 연결하는 데 사용할 IP 주소를 결정할 수 있습니다. 퍼블릭 IP 주소를 확인하려면 다른 브라우저 창 또는 탭에서 https://checkip.amazonaws.com의 서비스를 사용합니다. IP 주소의 예는 192.0.2.1/32입니다.

대부분의 경우 고정 IP 주소가 없는 방화벽 뒤나 인터넷 서비스 제공업체(ISP)를 통해 연결하는 경우가 많습니다. 그렇다면 클라이언트 컴퓨터에서 사용하는 IP 주소 범위를 결정합니다.

> **Warning**
>
> RDP 액세스에 `0.0.0.0/0`을 사용하는 경우 모든 IP 주소가 RDP를 사용하여 퍼블릭 EC2 인스턴스에 액세스하도록 설정할 수 있습니다. 이 방법은 테스트 환경에서 잠시 사용하는 것은 괜찮지만 프로덕션 환경에서는 안전하지 않습니다. 프로덕션에서는 RDP를 사용하여 EC2 인스턴스에 액세스할 수 있는 특정 IP 주소 또는 주소 범위만 인증합니다.

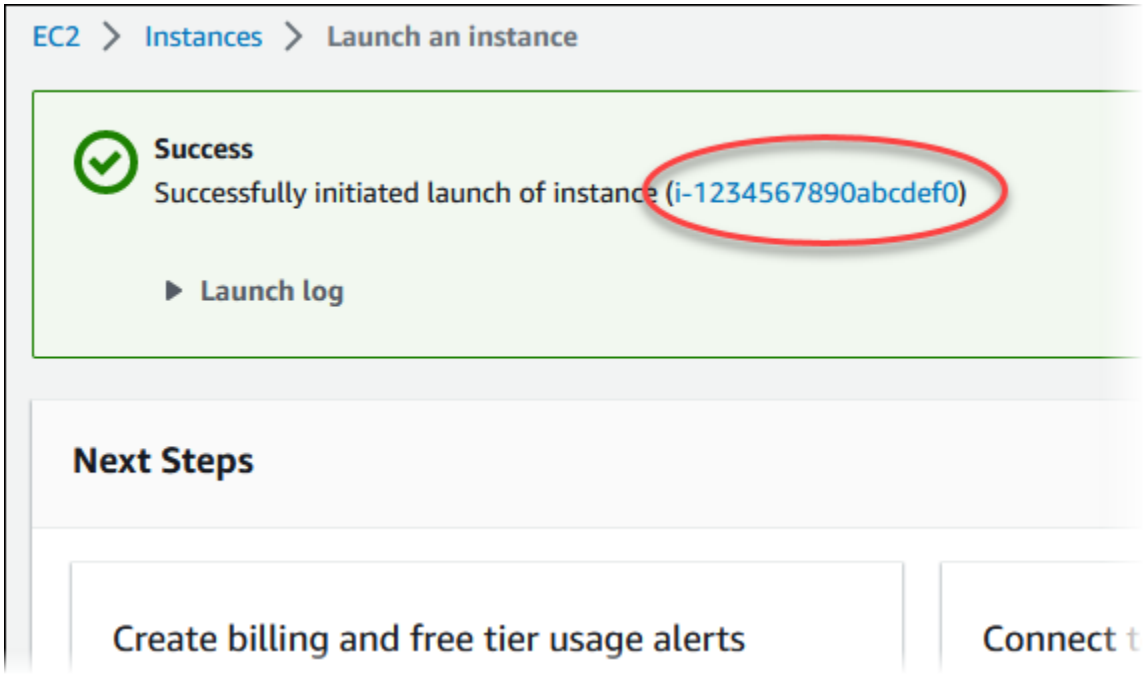
다음 이미지는 네트워크 설정 섹션의 예를 보여 줍니다.



f. 나머지 섹션에서 기본값은 그대로 둡니다.

g. 요약 패널에서 EC2 인스턴스 구성 요약을 검토하고 준비가 되면 인스턴스 시작을 선택합니다.

5. 시작 상태 페이지에서, 새 EC2 인스턴스의 식별자(예: `i-1234567890abcdef0`)를 기록해 둡니다.



6. EC2 인스턴스 식별자를 선택하여 EC2 인스턴스 목록을 엽니다.
7. 계속하기 전에 EC2 인스턴스의 인스턴스 상태가 실행 중이 될 때까지 기다립니다.

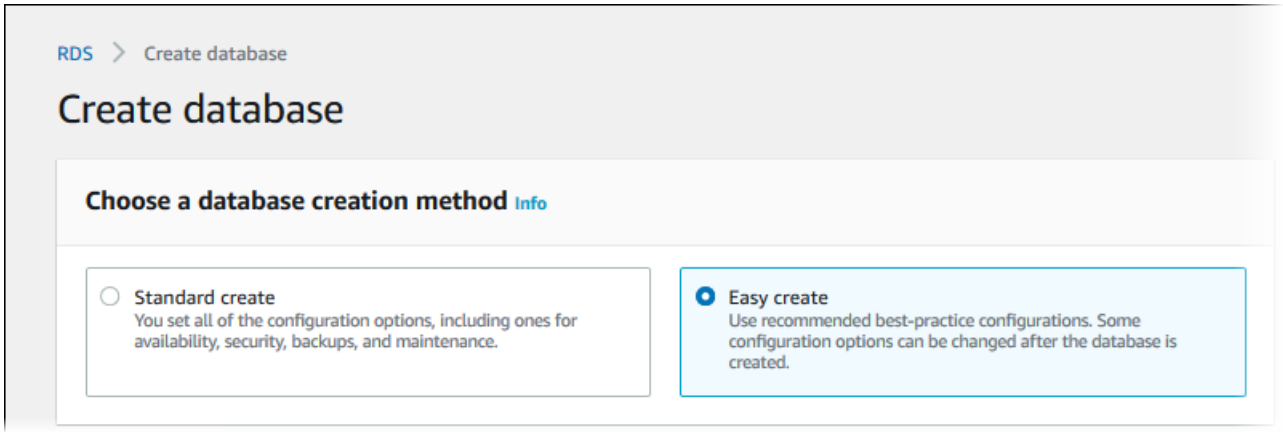
SQL Server DB 인스턴스 생성

Amazon RDS의 기본 빌딩 블록은 DB 인스턴스입니다. SQL Server 데이터베이스를 실행하는 환경입니다.

이 예시에서는 간편 생성을 사용하여 db.t2.micro DB 인스턴스 클래스에서 SQL Server 데이터베이스 엔진을 실행하는 DB 인스턴스를 생성합니다.

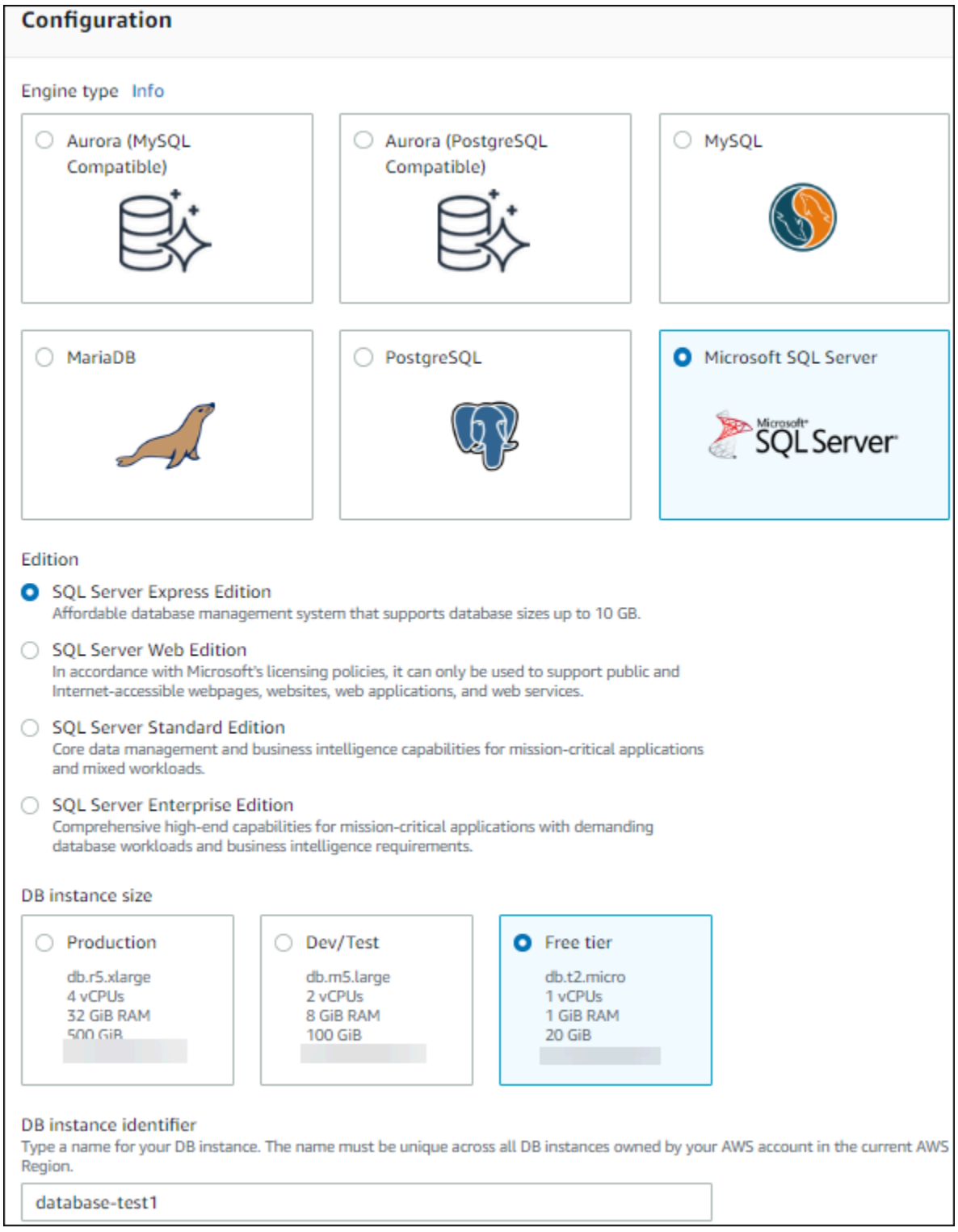
Easy Create(간편 생성)로 Microsoft SQL Server DB 인스턴스를 생성하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 콘솔의 오른쪽 상단에서 DB 인스턴스를 생성하려는 AWS 리전을 선택합니다.
3. 탐색 창에서 데이터베이스를 선택합니다.
4. [데이터베이스 생성(Create database)]을 선택하고 [간편 생성(Easy Create)]이 선택되어 있는지 확인합니다.



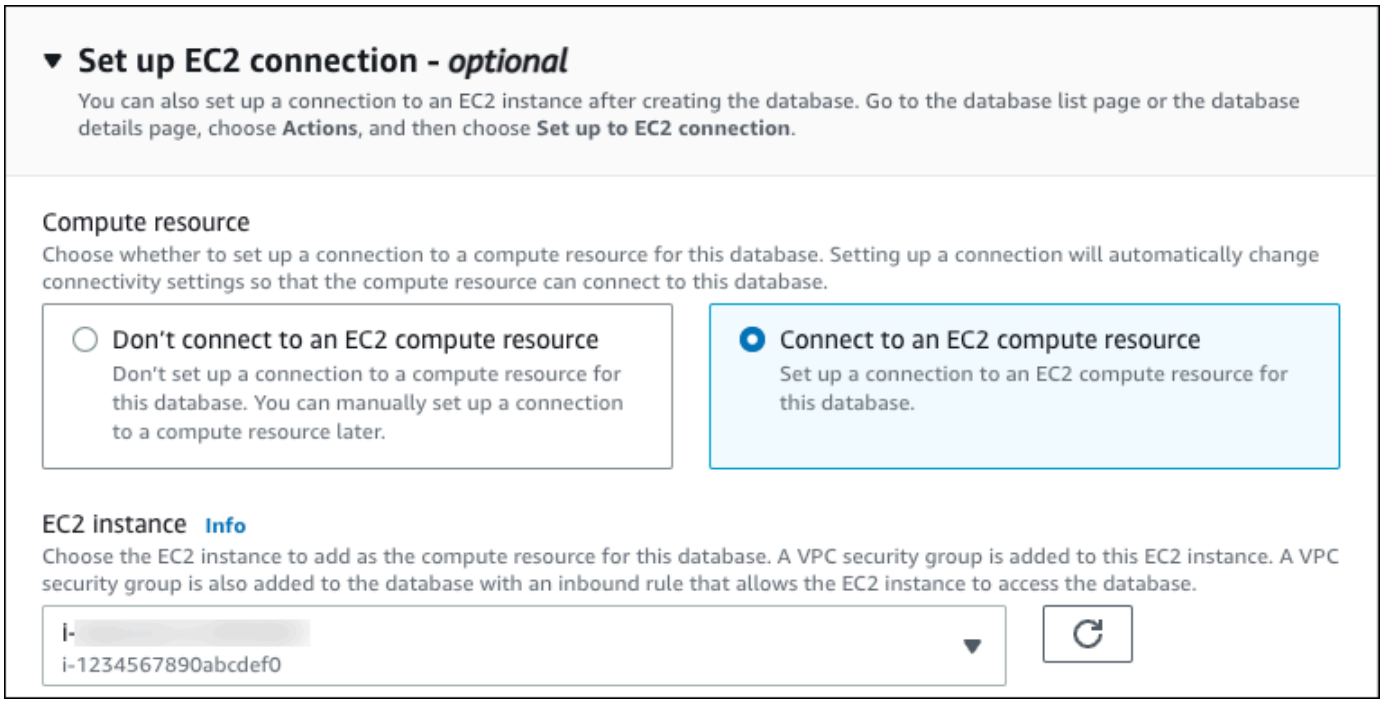
5. 구성에서 Microsoft SQL Server를 선택합니다.
6. 에디션에서 SQL Server Express Edition을 선택합니다.
7. DB instance size(DB 인스턴스 크기)에서 프리 티어를 선택합니다. 프리 티어는 무료 요금제 계정에 표시됩니다. 샌드박스는 유료 요금제 계정에 표시됩니다.
8. DB 인스턴스 식별자에 **database-test1**을 입력합니다.

   데이터베이스 생성 페이지는 다음 이미지와 비슷해야 합니다. 무료 요금제 계정에는 프리 티어가 표시됩니다. 유료 요금제 계정에는 샌드박스가 표시됩니다.



9. 마스터 사용자 이름에 마스터 사용자의 이름을 입력하거나 기본 이름을 그대로 유지합니다.
10. 이전에 생성한 EC2 인스턴스와의 연결을 설정하려면 EC2 연결 설정 - 선택 사항을 엽니다.

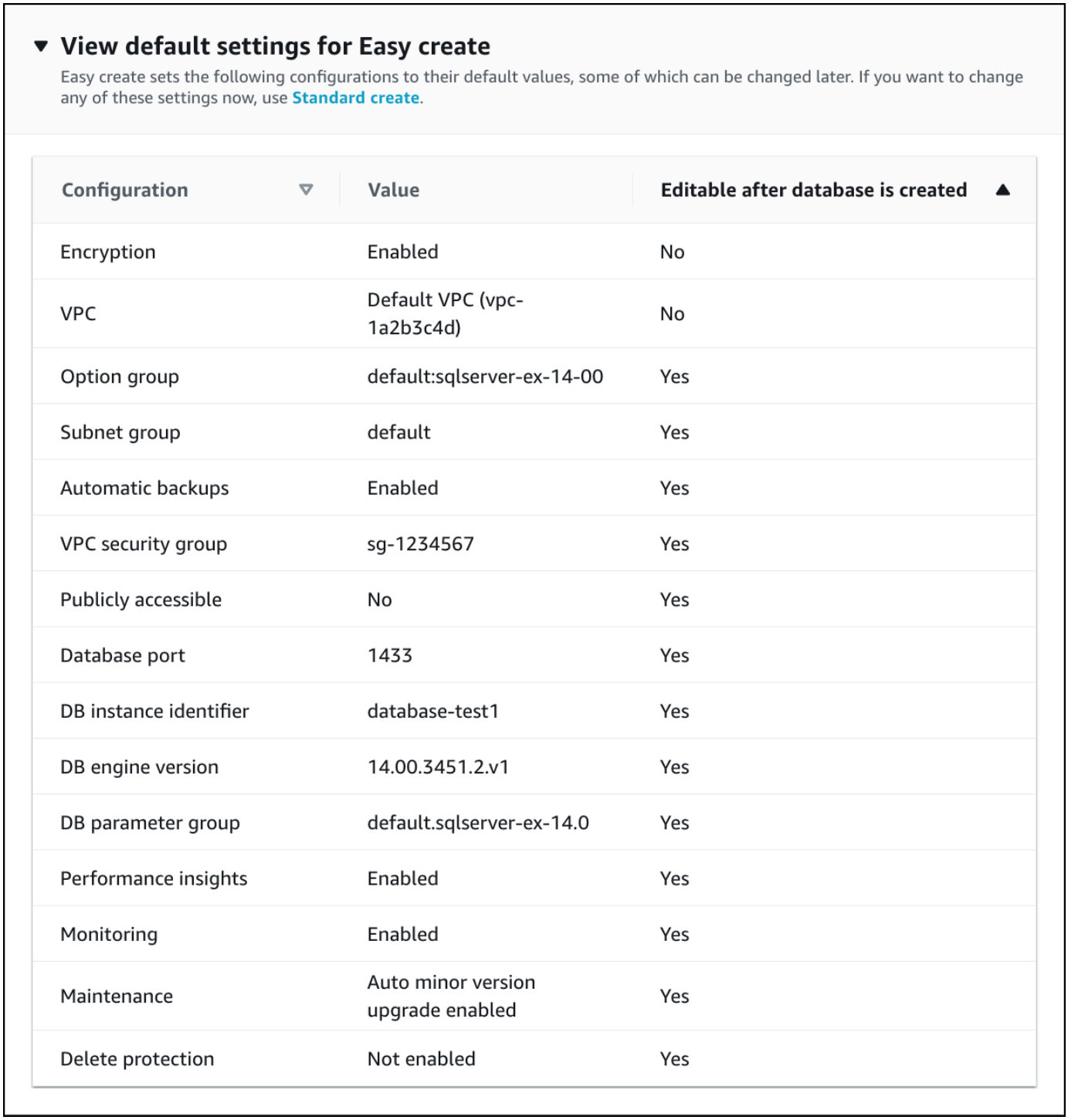
Connect to an EC2 compute resource(EC2 컴퓨팅 리소스에 연결)를 선택합니다. 이전에 생성한 EC2 인스턴스를 선택합니다.



11. DB 인스턴스에서 자동 생성된 마스터 암호를 사용하려면 [암호 자동 생성(Auto generate a password)] 확인란을 선택합니다.

    마스터 암호를 입력하려면 [암호 자동 생성(Auto generate a password)] 확인란의 선택을 해제한 다음, [마스터 암호(Master password)] 및 [암호 확인(Confirm password)]에 동일한 암호를 입력합니다.

12. 간편 생성 기본 설정 보기를 엽니다.



View default settings for Easy create

Easy create sets the following configurations to their default values, some of which can be changed later. If you want to change any of these settings now, use **Standard create**.

Configuration	Value	Editable after database is created
Encryption	Enabled	No
VPC	Default VPC (vpc-1a2b3c4d)	No
Option group	default:sqlserver-ex-14-00	Yes
Subnet group	default	Yes
Automatic backups	Enabled	Yes
VPC security group	sg-1234567	Yes
Publicly accessible	No	Yes
Database port	1433	Yes
DB instance identifier	database-test1	Yes
DB engine version	14.00.3451.2.v1	Yes
DB parameter group	default.sqlserver-ex-14.0	Yes
Performance insights	Enabled	Yes
Monitoring	Enabled	Yes
Maintenance	Auto minor version upgrade enabled	Yes
Delete protection	Not enabled	Yes

[간편 생성(Easy Create)]과 함께 사용되는 기본 설정을 검토할 수 있습니다. 데이터베이스 생성 후 편집 가능 열에는 데이터베이스 생성 후 어떤 옵션을 변경할 수 있는지 나와 있습니다.

- 설정의 해당 열에 아니요라고 되어 있지만 다른 설정을 원하는 경우 표준 생성을 사용하여 DB 인스턴스를 만들 수 있습니다.

- 설정의 해당 열에 예라고 되어 있으며 다른 설정을 원하는 경우 표준 생성을 사용하여 DB 인스턴스를 만들거나 DB 인스턴스를 생성한 후 수정하여 설정을 변경할 수 있습니다.

13. 데이터베이스 생성를 선택합니다.

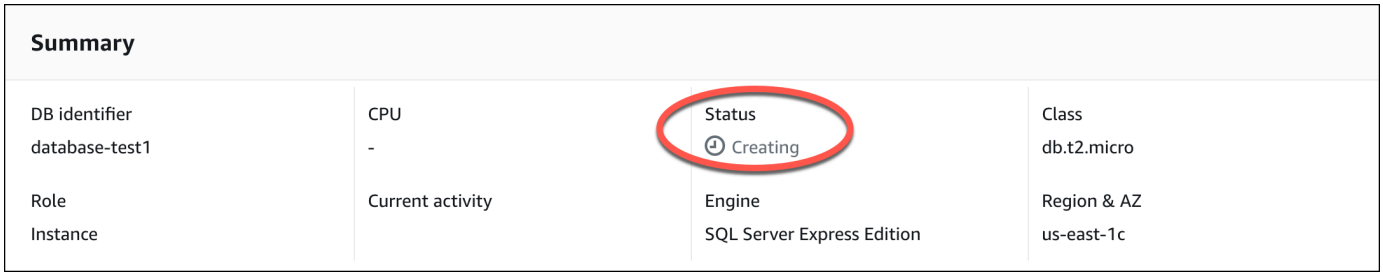
    DB 인스턴스의 마스터 사용자 이름 및 암호를 보려면 View credential details(자격 증명 세부 정보 보기)를 선택합니다.

    DB 인스턴스를 마스터 사용자로 연결하려면 화면에 나타난 사용자 이름과 암호를 사용합니다.

    > **Important**
    >
    > 마스터 사용자 암호를 다시 볼 수는 없습니다. 따라서 기록을 해두지 않으면 이를 변경해야 합니다.
    > DB 인스턴스가 사용 가능한 상태가 되고 난 후에 마스터 사용자 암호를 변경해야 하는 경우에는 다음과 같은 방법으로 DB 인스턴스를 수정할 수 있습니다. DB 인스턴스 변경에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

14. 데이터베이스 목록에서 새 SQL Server DB 인스턴스의 이름을 선택하면 세부 정보가 표시됩니다.

    DB 인스턴스를 사용할 준비가 될 때까지 DB 인스턴스의 상태는 생성 중입니다.



    상태가 Available(사용 가능)로 변경되면 DB 인스턴스에 연결할 수 있습니다. DB 인스턴스 클래스와 스토리지의 양에 따라 새 인스턴스를 사용할 수 있을 때까지 최대 20분이 걸릴 수 있습니다.

(선택 사항) AWS CloudFormation를 사용하여 VPC, EC2 인스턴스 및 SQL Server 인스턴스 생성

콘솔을 사용하여 VPC, EC2 인스턴스 및 SQL Server 인스턴스를 생성하는 대신 AWS CloudFormation을 통해 코드형 인프라로 처리하여 AWS 리소스를 프로비저닝할 수 있습니다. AWS 리소스를 더 작고 관리하기 쉬운 단위로 구성하는 데 도움이 되도록 AWS CloudFormation 중첩 스택 기능을 사용할 수

있습니다. 자세한 내용은 AWS CloudFormation 콘솔에서 스택 생성 및 중첩된 스택 작업을 참조하세요.

> **Important**
>
> AWS CloudFormation은 무료이지만, CloudFormation에서 생성하는 리소스는 라이브입니다. 이러한 리소스를 종료하지 않으면 해당 리소스에 대한 표준 사용 요금이 발생합니다. 자세한 내용은 RDS for SQL Server 요금을 참조하세요.

AWS CloudFormation 콘솔을 사용하여 리소스를 생성하려면 다음 단계를 완료합니다.

- CloudFormation 템플릿 파일을 다운로드하십시오.
- CloudFormation을 사용하여 리소스 구성

CloudFormation 템플릿 파일을 다운로드하십시오.

CloudFormation 템플릿은 JSON 또는 YAML 텍스트 파일로, 스택에서 생성하려는 리소스에 대한 구성 정보가 들어 있습니다. 또한 이 템플릿은 RDS 인스턴스와 함께 VPC와 Bastion Host를 생성합니다.

템플릿 파일을 다운로드하려면 다음 링크인 SQL Server CloudFormation 템플릿을 엽니다.

Github 페이지에서 원시 파일 다운로드 버튼을 클릭하여 템플릿 YAML 파일을 저장합니다.

CloudFormation을 사용하여 리소스 구성

> **Note**
>
> 이 프로세스를 시작하기 전에 AWS 계정에 EC2 인스턴스용 키 페어가 있는지 확인합니다. 자세한 내용은 Amazon EC2 키 페어 및 Linux 인스턴스를 참조하세요.

AWS CloudFormation 템플릿을 사용할 때는 리소스가 제대로 생성되도록 올바른 파라미터를 선택해야 합니다. 다음 단계를 따릅니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/cloudformation에서 AWS CloudFormation 콘솔을 엽니다.
2. 스택 생성을 선택합니다.

3. 템플릿 지정 섹션에서 컴퓨터에서 템플릿 파일 업로드를 선택하고 다음을 선택합니다.
4. 스택 세부 정보 지정 페이지에서 다음 파라미터를 설정합니다.
   a. 스택 이름을 SQLServerTestStack으로 설정합니다.
   b. 파라미터에서 가용 영역 3개를 선택하여 가용 영역을 설정합니다.
   c. Linux Bastion Host 구성에서 키 이름에 대해 EC2 인스턴스에 로그인할 키 페어를 선택합니다.
   d. Linux Bastion Host 구성 설정에서 허용된 IP 범위를 IP 주소로 설정합니다. Secure Shell(SSH)을 사용하여 VPC의 EC2 인스턴스에 연결하려면 https://checkip.amazonaws.com의 서비스를 사용하여 퍼블릭 IP 주소를 지정합니다. IP 주소의 예는 192.0.2.1/32입니다.

      > **Warning**
      >
      > SSH 액세스에 `0.0.0.0/0`을 사용하는 경우 모든 IP 주소가 SSH를 사용하여 퍼블릭 EC2 인스턴스에 액세스할 수 있도록 활성화합니다. 이 방법은 테스트 환경에서 잠시 사용하는 것은 괜찮지만 프로덕션 환경에서는 안전하지 않습니다. 프로덕션에서는 SSH를 사용하여 EC2 인스턴스에 액세스할 수 있는 특정 IP 주소 또는 주소 범위만 인증합니다.

   e. 데이터베이스 일반 구성에서 데이터베이스 인스턴스 클래스를 db.t3.micro로 설정합니다.
   f. 데이터베이스 이름을 **`database-test1`**으로 설정합니다.
   g. 데이터베이스 마스터 사용자 이름에 마스터 사용자 이름을 입력합니다.
   h. 이 자습서에서는 Secrets Manager를 사용하여 DB 마스터 사용자 암호 관리를 `false`로 설정합니다.
   i. 데이터베이스 암호의 경우 원하는 암호를 설정합니다. 자습서의 향후 단계에 사용할 수 있도록 암호를 기억해 둡니다.
   j. 데이터베이스 스토리지 구성에서 데이터베이스 스토리지 유형을 gp2로 설정합니다.
   k. 데이터베이스 모니터링 구성에서 RDS 성능 개선 도우미 활성화를 false로 설정합니다.
   l. 다른 모든 설정은 기본값으로 둡니다. 계속하려면 다음을 클릭합니다.
5. 스택 옵션 구성 페이지에서는 모든 기본 옵션을 그대로 둡니다. 계속하려면 다음을 클릭합니다.
6. 스택 검토 페이지에서 데이터베이스 및 Linux Bastion Host 옵션을 확인한 후 제출을 선택합니다.

스택 생성 프로세스가 완료되면 BastionStack 및 RDSNS라는 이름의 스택을 보고 데이터베이스에 연결하는 데 필요한 정보를 기록해 둡니다. 자세한 내용은 AWS Management Console에서 AWS CloudFormation 스택 데이터 및 리소스 보기를 참조하세요.

SQL Server DB 인스턴스에 연결

다음 프로시저에서는 Microsoft SQL Server Management Studio(SSMS)를 사용하여 DB 인스턴스에 연결합니다.

SSMS를 사용하여 RDS for SQL Server DB에 연결하려면

1. DB 인스턴스의 엔드포인트(DNS 이름)와 포트 번호를 찾습니다.

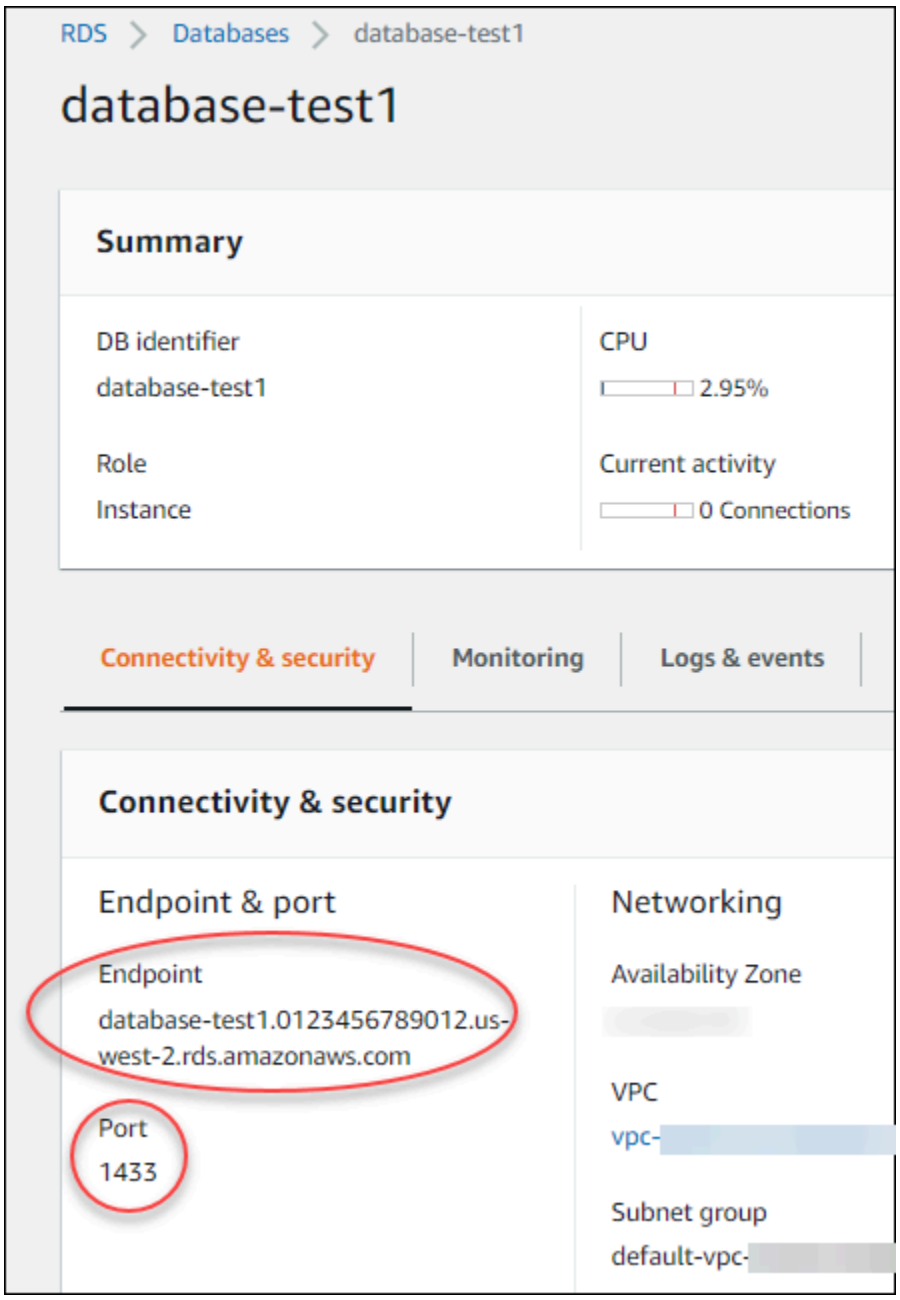
   a. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.

   b. Amazon RDS 콘솔의 오른쪽 상단에서 DB 인스턴스의 AWS 리전을 선택합니다.

   c. 탐색 창에서 Databases(데이터베이스)를 선택합니다.

   d. 세부 정보를 표시하고자 하는 SQL Server DB 인스턴스 이름을 선택합니다.

   e. 연결 탭에서 엔드포인트를 복사합니다. 또한 포트 번호를 적어 둡니다. DB 인스턴스에 연결하려면 엔드포인트와 포트 번호가 모두 필요합니다.



2. Windows 인스턴스용 Amazon EC2 사용 설명서에 있는 Microsoft Windows 인스턴스에 연결의 단계를 따라 앞에서 만든 EC2 인스턴스에 연결합니다.
3. Microsoft의 SQL Server Management Studio(SSMS) 클라이언트를 설치합니다.

   SSMS의 독립 실행형 버전을 EC2 인스턴스에 다운로드하려면 Microsoft 설명서의 SSMS(SQL Server Management Studio) 다운로드를 참조하세요.

   a. 시작 메뉴를 사용하여 Internet Explorer를 엽니다.

b. Internet Explorer를 사용하여 독립 실행형 버전의 SSMS를 다운로드하고 설치합니다. 사이트를 신뢰할 수 없다는 메시지가 표시되면 신뢰할 수 있는 사이트 목록에 해당 사이트를 추가하세요.

4. SQL Server Management Studio(SSMS)를 시작합니다.

   Connect to Server 대화 상자가 나타납니다.

5. 샘플 DB 인스턴스에 대한 다음 정보를 제공합니다.

   a. [Server type]에서 [Database Engine]을 선택합니다.

   b. 서버 이름에서 DNS 이름, 쉼표 및 포트 번호(기본 포트 1433)를 차례대로 입력합니다. 예를 들어 서버 이름은 다음과 같은 형식이어야 합니다.

   ```
   database-test1.0123456789012.us-west-2.rds.amazonaws.com,1433
   ```

   c. 인증(Authentication)]의 경우 SQL Server 인증(SQL Server Authentication)을 선택합니다.

   d. 로그인에서 샘플 DB 인스턴스에 사용하려고 선택한 사용자 이름을 입력합니다. 마스터 사용자 이름으로도 알려져 있습니다.

   e. 암호에 샘플 DB 인스턴스에 대해 이전에 선택한 암호를 입력합니다. 마스터 사용자 암호로도 알려져 있습니다.

6. [Connect]를 선택합니다.

   몇 분 정도 지나면 SSMS가 DB 인스턴스에 연결됩니다. 보안을 위해서는 암호화된 연결을 사용하는 것이 가장 좋습니다. 클라이언트와 서버가 동일한 VPC에 있고 네트워크를 신뢰할 수 있는 경우에만 암호화되지 않은 SQL Server 연결을 사용합니다. 암호화된 연결 사용에 대한 자세한 내용은 ??? 섹션을 참조하세요.

Microsoft SQL Server DB 인스턴스에 연결하는 방법에 대한 자세한 내용은 Microsoft SQL Server DB 인스턴스에 연결 섹션을 참조하세요.

연결에 대한 내용은 Amazon RDS DB 인스턴스에 연결할 수 없음 섹션을 참조하세요.

샘플 SQL Server DB 인스턴스 탐색

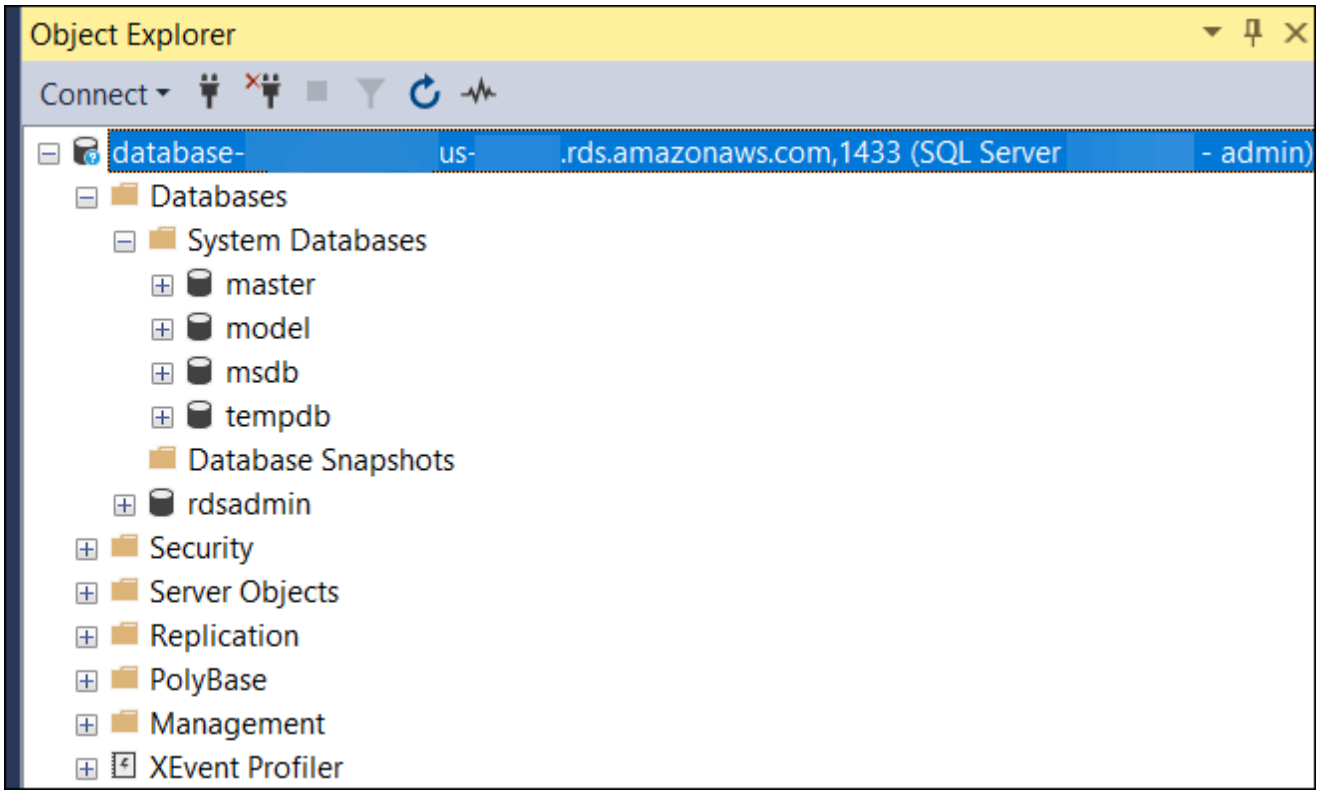
Microsoft SQL Server Management Studio(SSMS)를 사용하여 샘플 DB 인스턴스를 탐색할 수 있습니다.

SSMS를 사용하여 DB 인스턴스를 탐색하려면

1. SQL 서버 DB 인스턴스는 SQL 서버의 표준 기본 제공 시스템 데이터베이스(마스터, 모델, msdb 및 tempdb)와 함께 제공됩니다. 시스템 데이터베이스를 탐색하려면 다음을 수행하십시오.

   a. SSMS의 [View] 메뉴에서 [Object Explorer]를 선택합니다.

   b. DB 인스턴스와 데이터베이스를 확장하고, 다음과 같이 시스템 데이터베이스를 확장합니다.



   SQL Server DB 인스턴스는 `rdsadmin`이라는 이름의 데이터베이스와 함께 제공됩니다. Amazon RDS는 이 데이터베이스를 사용하여 데이터베이스를 관리하는 데 사용하는 객체를 저장합니다. `rdsadmin` 데이터베이스에도 고급 작업 수행을 위해 실행할 수 있는 저장 절차가 포함됩니다.

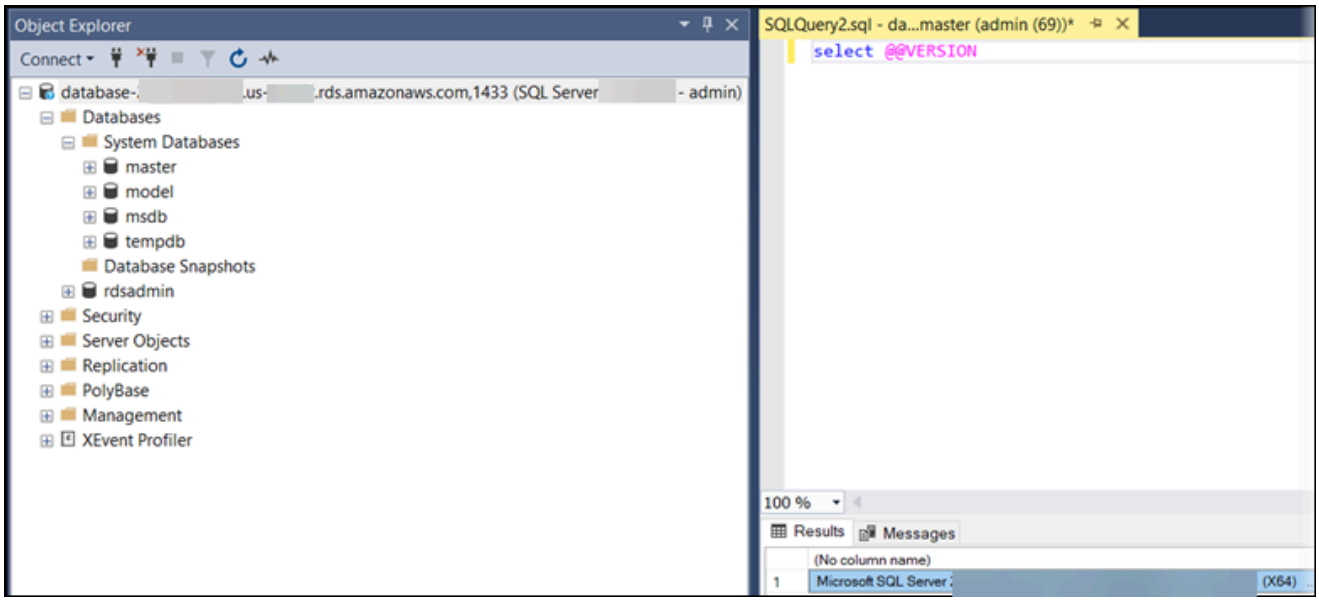
2. 자체 데이터베이스 생성을 시작하고 평소대로 DB 인스턴스와 데이터베이스에 대한 쿼리 실행을 시작합니다. 샘플 DB 인스턴스에 대한 테스트 쿼리를 실행하려면 다음을 수행합니다.

   a. SSMS의 File(파일) 메뉴에서 New(새로 만들기)를 가리킨 후 Query with Current Connection(현재 연결로 쿼리)을 선택합니다.

   b. 다음 SQL 쿼리를 입력합니다.

```
select @@VERSION
```

c. 쿼리를 실행합니다. SSMS가 Amazon RDS DB 인스턴스의 SQL Server 버전을 반환합니다.



EC2 인스턴스 및 DB 인스턴스 삭제

생성한 샘플 EC2 인스턴스 및 DB 인스턴스에 연결하고 탐색한 후에는 요금이 더 이상 부과되지 않도록 삭제합니다.

AWS CloudFormation을 사용하여 리소스를 생성했다면 이 단계를 건너뛰고 다음 단계로 이동합니다.

EC2 인스턴스를 삭제하는 방법

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창에서 인스턴스를 선택합니다.
3. EC2 인스턴스를 선택하고 인스턴스 상태, 인스턴스 종료를 차례로 선택합니다.
4. 확인 메시지가 나타나면 종료를 선택합니다.

EC2 인스턴스 삭제에 대한 자세한 내용은 Windows 인스턴스용 Amazon EC2 사용 설명서의 인스턴스 종료를 참조하세요.

최종 DB 스냅샷이 없는 DB 인스턴스를 삭제하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.

2. 탐색 창에서 데이터베이스를 선택합니다.
3. 삭제할 DB 인스턴스를 선택합니다.
4. [Actions]에 대해 [Delete]를 선택합니다.
5. 최종 스냅샷을 생성하시겠습니까? 및 자동 백업 보존을 선택 해제합니다.
6. 확인을 완료하고 삭제를 선택합니다.

(선택 사항) CloudFormation으로 생성한 EC2 인스턴스 및 DB 인스턴스 삭제

AWS CloudFormation을 사용하여 리소스를 생성한 경우 샘플 EC2 인스턴스 및 DB 인스턴스에 연결하고 탐색한 후 CloudFormation 스택을 삭제하여 더 이상 비용이 청구되지 않도록 합니다.

CloudFormation 리소스를 삭제하려면

1. AWS CloudFormation 콘솔을 엽니다.
2. CloudFormation 콘솔의 스택 페이지에서 루트 스택(VPCStack, BastionStack 또는 RDSNS라는 이름이 없는 스택)을 선택합니다.
3. 삭제를 선택합니다.
4. 확인 메시지가 나타나면 스택 삭제를 선택합니다.

CloudFormation에서 스택을 삭제하는 방법에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS CloudFormation 콘솔에서 스택 삭제를 참조하세요.

(선택 사항) DB 인스턴스를 Lambda 함수에 연결

RDS for SQL Server DB 인스턴스를 Lambda 서버리스 컴퓨팅 리소스에 연결할 수도 있습니다. Lambda 함수를 사용하면 인프라를 프로비저닝하거나 관리하지 않고도 코드를 실행할 수 있습니다. 또한 하루 12개에서 초당 수백 개에 이르는 모든 규모의 코드 실행 요청에 자동으로 응답할 수 있습니다. 자세한 내용은 Lambda 함수와 DB 인스턴스 자동 연결 섹션을 참조하세요.

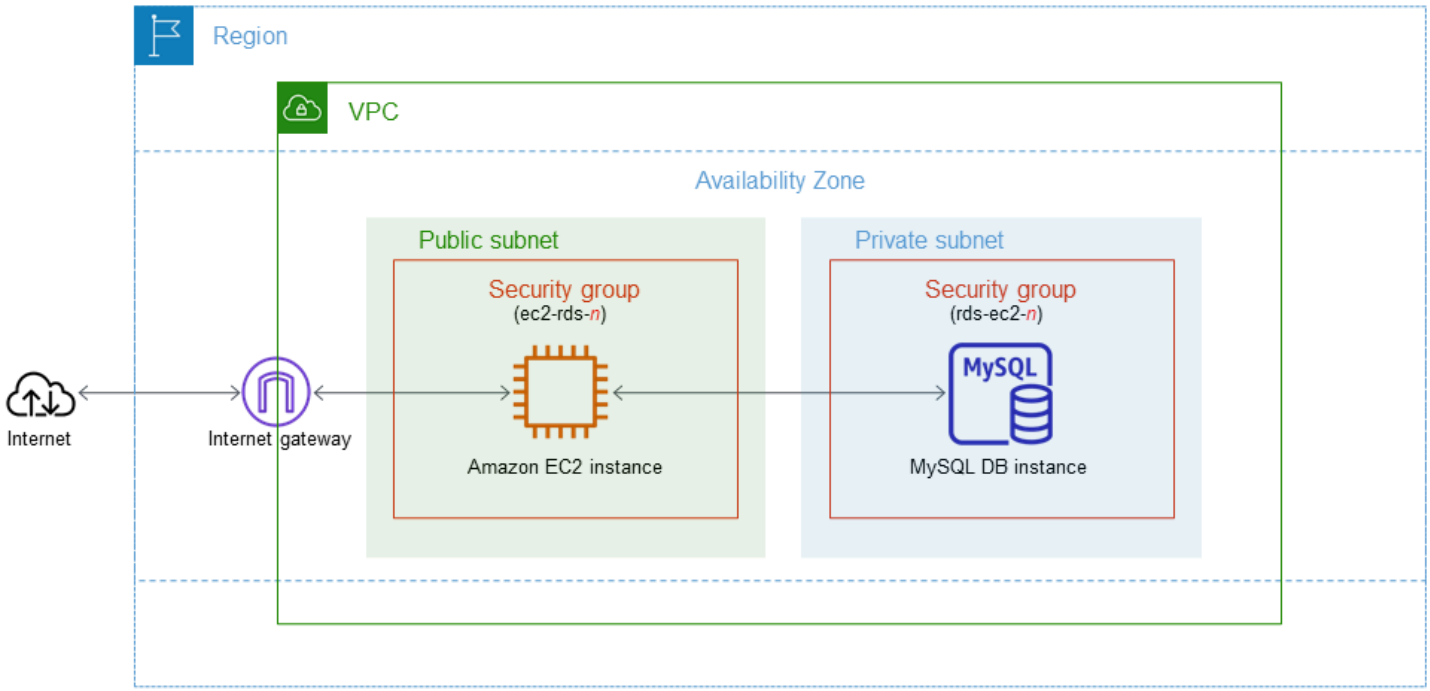
MySQL DB 인스턴스 생성 및 해당 인스턴스에 연결

이 자습서에서는 EC2 인스턴스와 RDS for MySQL DB 인스턴스를 생성합니다. 자습서에서는 표준 MySQL 클라이언트를 사용하여 EC2 인스턴스에서 DB 인스턴스에 액세스하는 방법을 보여줍니다. 이 자습서에서는 모범 사례를 따라 Virtual Private Cloud(VPC)에서 프라이빗 DB 인스턴스를 생성합니다. 대부분의 경우 EC2 인스턴스와 같이 동일한 VPC에 있는 다른 리소스는 DB 인스턴스에 액세스할 수 있지만 VPC 외부의 리소스는 DB 인스턴스에 액세스할 수 없습니다.

자습서를 완료하면 VPC의 각 가용 영역에 퍼블릭 서브넷과 프라이빗 서브넷이 있을 것입니다. 한 가용 영역에서 EC2 인스턴스는 퍼블릭 서브넷에 있고 DB 인스턴스는 프라이빗 서브넷에 있습니다.

> **Important**
>
> AWS 계정 생성은 무료입니다. 그러나 이 자습서를 완료하면 사용하는 AWS 리소스에 대한 비용이 발생할 수 있습니다. 자습서가 더 이상 필요하지 않은 경우 자습서를 완료한 후에 이러한 리소스를 삭제할 수 있습니다.

다음 다이어그램은 이 자습서를 완료했을 때 구성을 보여 줍니다.



이 자습서에서는 다음 방법 중 하나를 사용하여 리소스를 생성할 수 있습니다.

1. AWS Management Console 사용 - MySQL DB 인스턴스 생성 및 EC2 인스턴스 생성

2. 데이터베이스 인스턴스 및 EC2 인스턴스를 생성하는 데 AWS CloudFormation 사용 - (선택 사항) AWS CloudFormation를 사용하여 VPC, EC2 인스턴스 및 MySQL 인스턴스 생성

첫 번째 방법은 간편 생성을 사용하여 AWS Management Console을 통해 프라이빗 MySQL DB 인스턴스를 생성합니다. 여기에서는 DB 엔진 유형, DB 인스턴스 크기 및 DB 인스턴스 식별자만 지정합니다. [간편 생성(Easy create)]은 다른 구성 옵션에서도 기본 설정을 사용합니다.

표준 생성을 대신 사용하는 경우에는 DB 인스턴스를 생성할 때 더 많은 구성 옵션을 지정할 수 있습니다. 이러한 옵션에는 가용성, 보안, 백업 및 유지 관리에 대한 설정이 포함됩니다. 퍼블릭 DB 인스턴스를 만들려면 표준 생성을 사용해야 합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.

주제

- 사전 조건
- EC2 인스턴스 생성
- MySQL DB 인스턴스 생성
- (선택 사항) AWS CloudFormation를 사용하여 VPC, EC2 인스턴스 및 MySQL 인스턴스 생성
- MySQL DB 인스턴스에 연결
- EC2 인스턴스 및 DB 인스턴스 삭제
- (선택 사항) CloudFormation으로 생성한 EC2 인스턴스 및 DB 인스턴스 삭제
- (선택 사항) DB 인스턴스를 Lambda 함수에 연결

사전 조건

시작하기 전에 다음 섹션에서 다음 단계를 완료하세요.

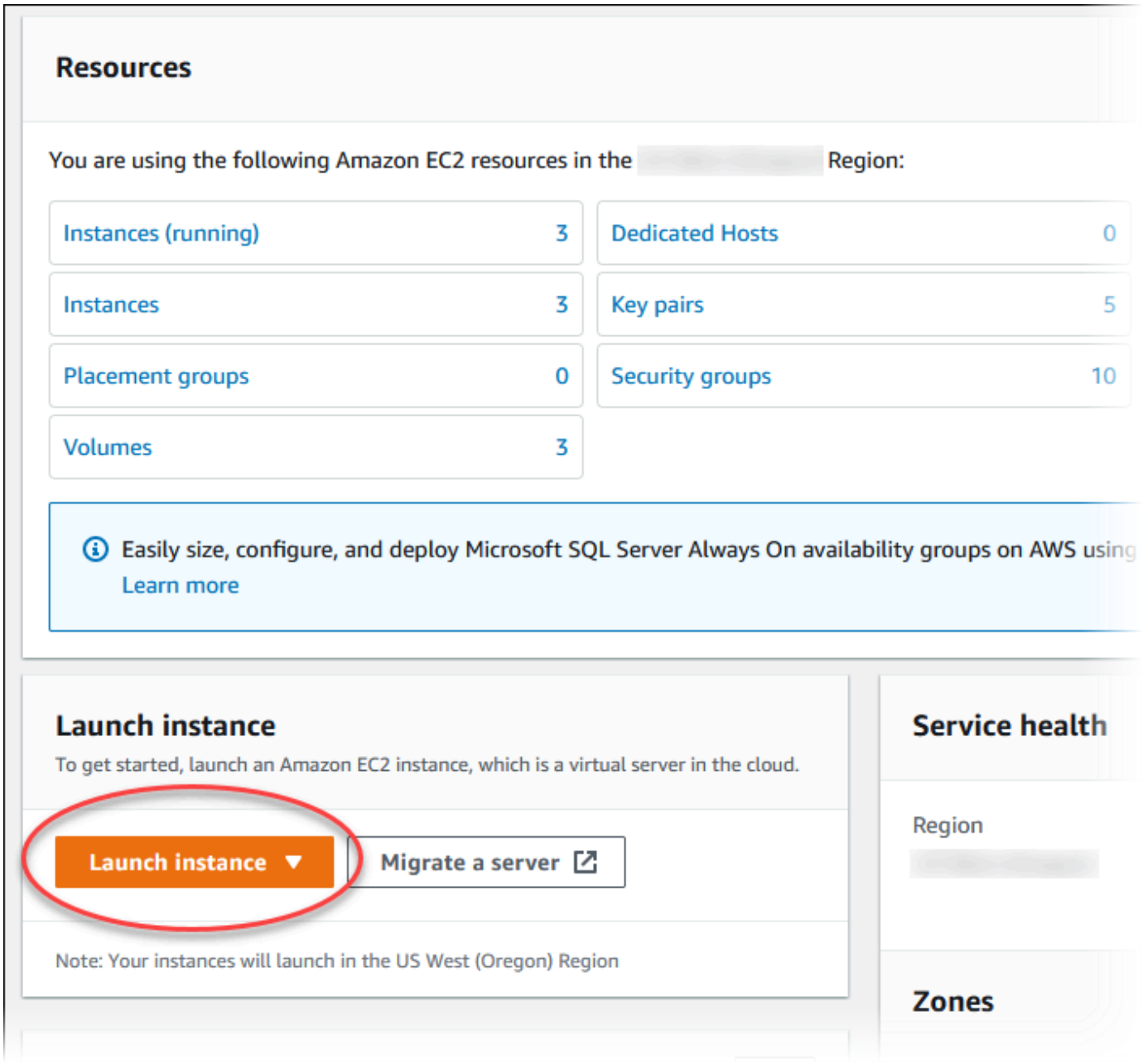
- AWS 계정에 등록
- 관리자 액세스 권한이 있는 사용자 생성

EC2 인스턴스 생성

데이터베이스에 연결하는 데 사용할 Amazon EC2 인스턴스를 생성합니다.

EC2 인스턴스를 생성하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. AWS Management Console의 오른쪽 상단에서 EC2 인스턴스를 생성하려는 AWS 리전을 선택합니다.
3. 다음 이미지에 나와 있는 것처럼 EC2 대시보드를 선택한 다음, 인스턴스 시작을 선택합니다.

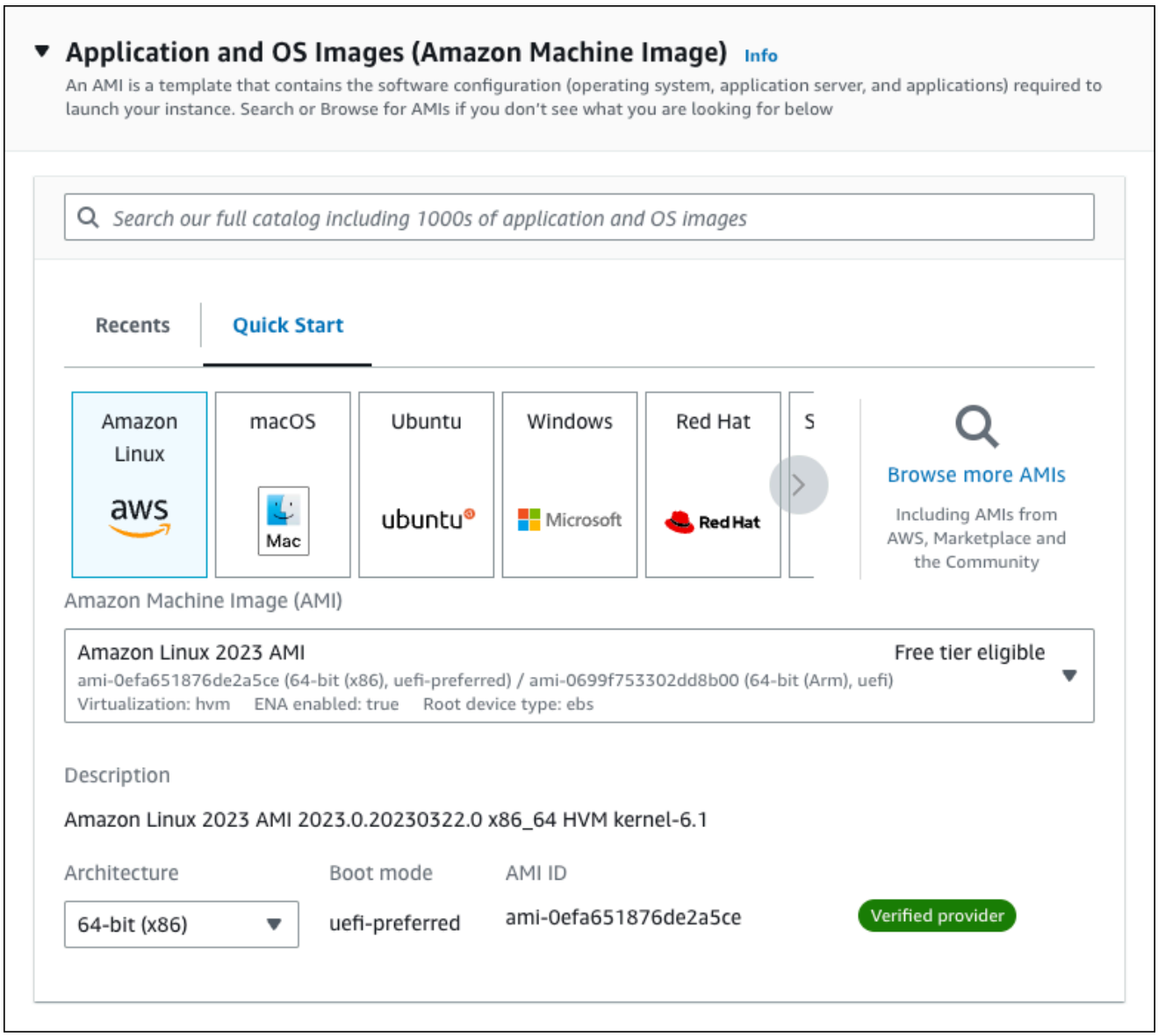


인스턴스 시작 페이지가 열립니다.

4. 인스턴스 시작 페이지에서 다음 설정을 선택합니다.

   a. Name and tags(이름 및 태그) 아래의 Name(이름)에 **ec2-database-connect**을 입력하세요.

   b. Application and OS Images (Amazon Machine Image)(애플리케이션 및 OS 이미지(Amazon Machine Image))에서 Amazon Linux를 선택한 다음 Amazon Linux 2023 AMI를 선택합니다. 다른 선택 항목에 대해서는 기본값을 그대로 유지합니다.



   c. 인스턴스 유형에서 t2.micro를 선택합니다.

   d. 키 페어(로그인)에서 기존 키 페어를 사용할 키 페어 이름을 선택합니다. Amazon EC2 인스턴스에 대한 새 키 페어를 생성하려면 새 키 페어 생성을 선택한 다음 키 페어 생성 창을 사용하여 생성합니다.

새 키 페어 생성에 대한 자세한 내용은 Amazon EC2 Linux 인스턴스용 사용 설명서의 키 페어 생성을 참조하세요.

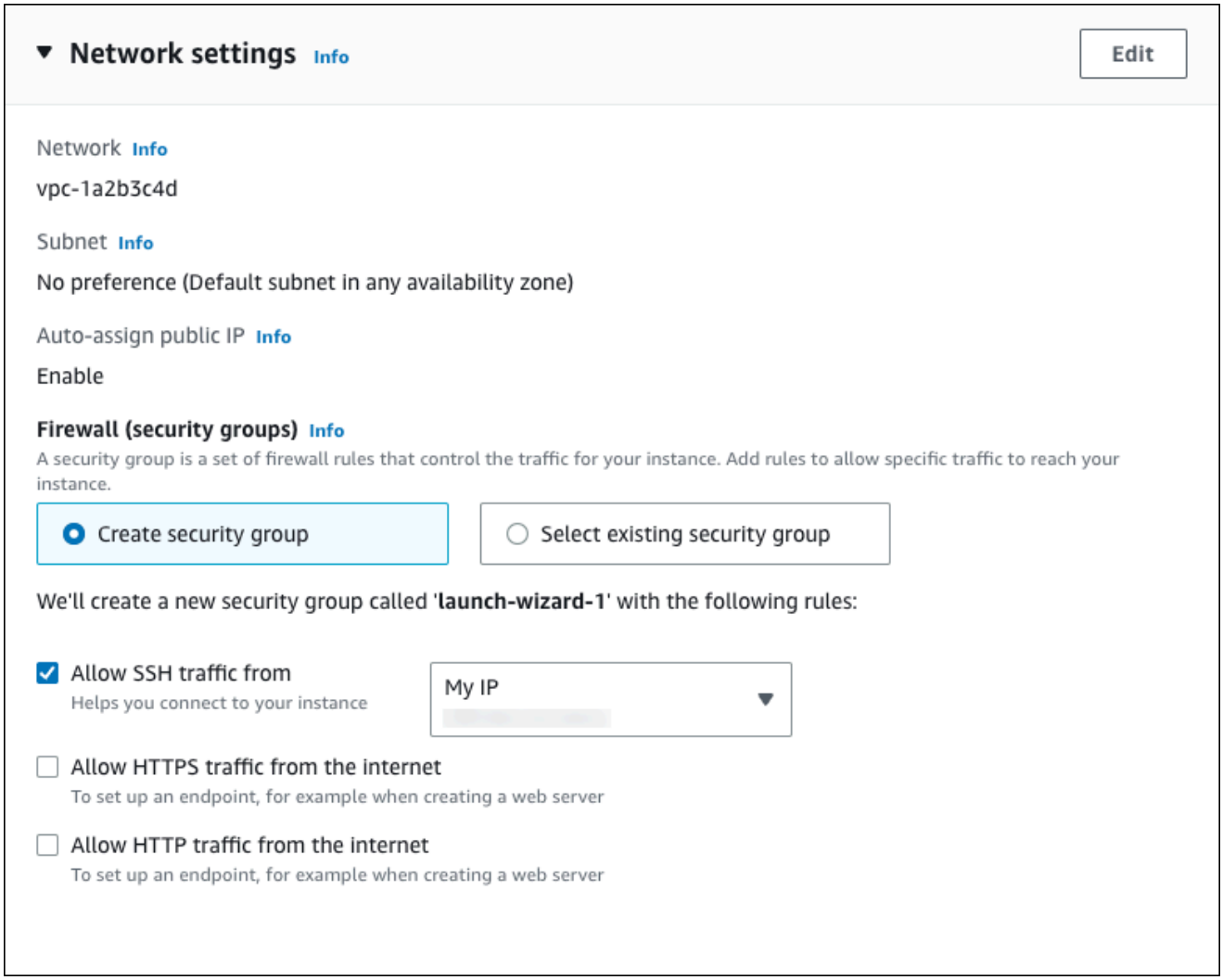
e. 네트워크 설정의 SSH 트래픽 허용에서 EC2 인스턴스에 대한 SSH 연결 소스를 선택합니다.

표시된 IP 주소가 SSH 연결에 대해 올바른 경우 내 IP를 선택할 수 있습니다. 그렇지 않으면 SSH(Secure Shell)를 사용하여 VPC의 EC2 인스턴스에 연결하는 데 사용할 IP 주소를 결정할 수 있습니다. 퍼블릭 IP 주소를 확인하려면 다른 브라우저 창 또는 탭에서 https://checkip.amazonaws.com의 서비스를 사용합니다. IP 주소의 예는 192.0.2.1/32입니다.

대부분의 경우 고정 IP 주소가 없는 방화벽 뒤나 인터넷 서비스 제공업체(ISP)를 통해 연결하는 경우가 많습니다. 그렇다면 클라이언트 컴퓨터에서 사용하는 IP 주소 범위를 결정합니다.

> **Warning**
>
> SSH 액세스에 `0.0.0.0/0`을 사용하는 경우 모든 IP 주소가 SSH를 사용하여 퍼블릭 EC2 인스턴스에 액세스할 수 있도록 활성화합니다. 이 방법은 테스트 환경에서 잠시 사용하는 것은 괜찮지만 프로덕션 환경에서는 안전하지 않습니다. 프로덕션에서는 SSH를 사용하여 EC2 인스턴스에 액세스할 수 있는 특정 IP 주소 또는 주소 범위만 인증합니다.

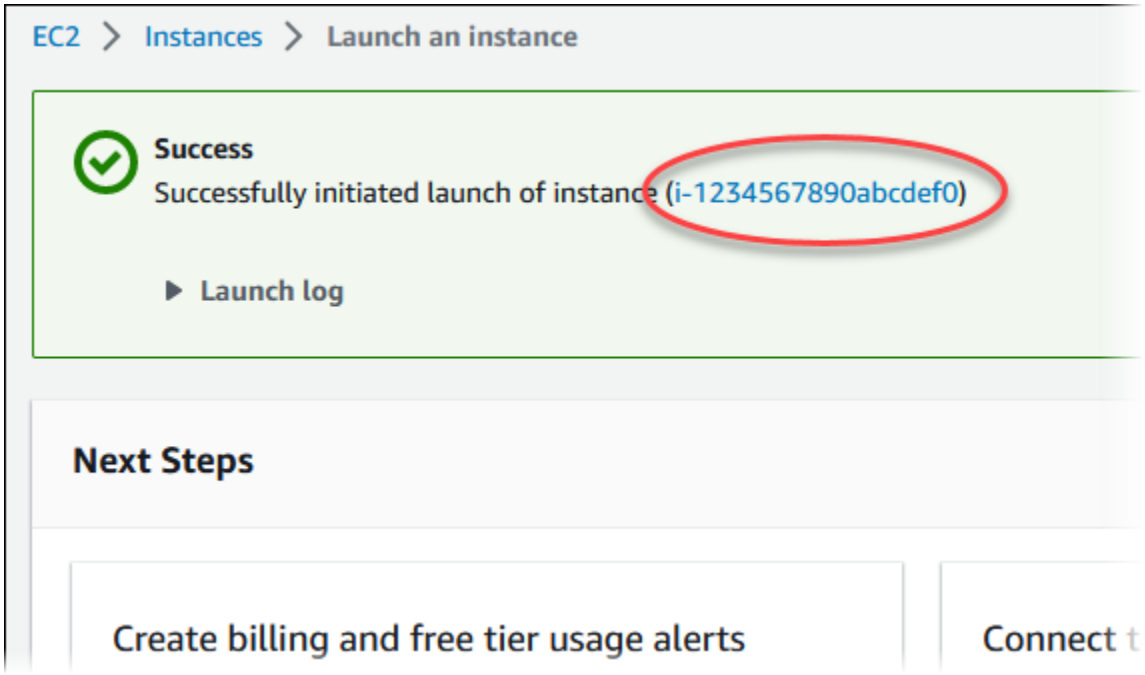
다음 이미지는 네트워크 설정 섹션의 예를 보여 줍니다.



f. 나머지 섹션에서 기본값은 그대로 둡니다.

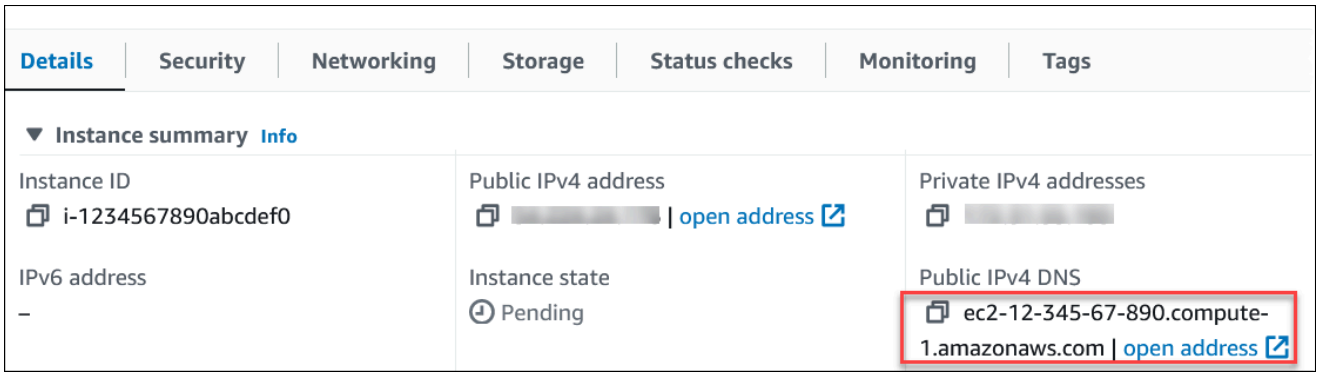
g. 요약 패널에서 EC2 인스턴스 구성 요약을 검토하고 준비가 되면 인스턴스 시작을 선택합니다.

5. 시작 상태 페이지에서, 새 EC2 인스턴스의 식별자(예: `i-1234567890abcdef0`)를 기록해 둡니다.

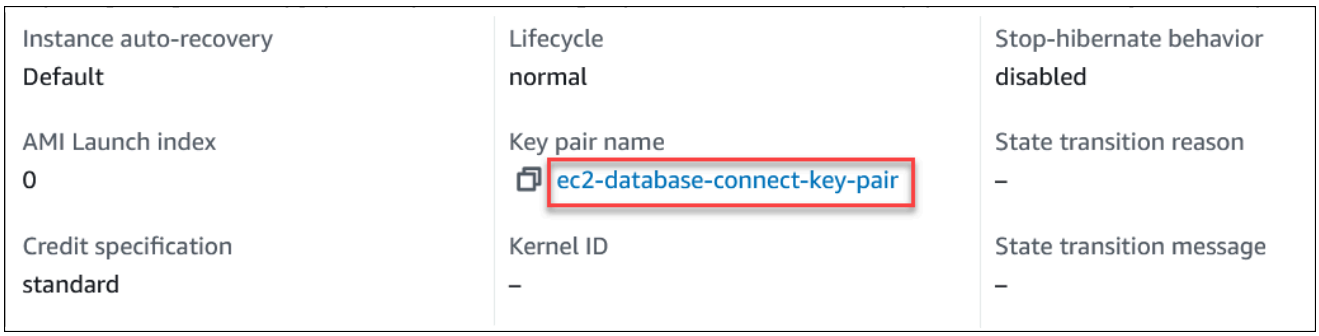


6. EC2 인스턴스 식별자를 선택하여 EC2 인스턴스 목록을 열고 EC2 인스턴스를 선택합니다.
7. 세부 정보 탭에서 SSH를 사용하여 연결할 때 필요한 다음 값을 기록해 둡니다.

   a. 인스턴스 요약에서 퍼블릭 IPv4 DNS의 값을 기록해 둡니다.



   b. 인스턴스 세부 정보에서 키 페어 이름의 값을 기록해 둡니다.



8. 계속하기 전에 EC2 인스턴스의 인스턴스 상태가 실행 중이 될 때까지 기다립니다.

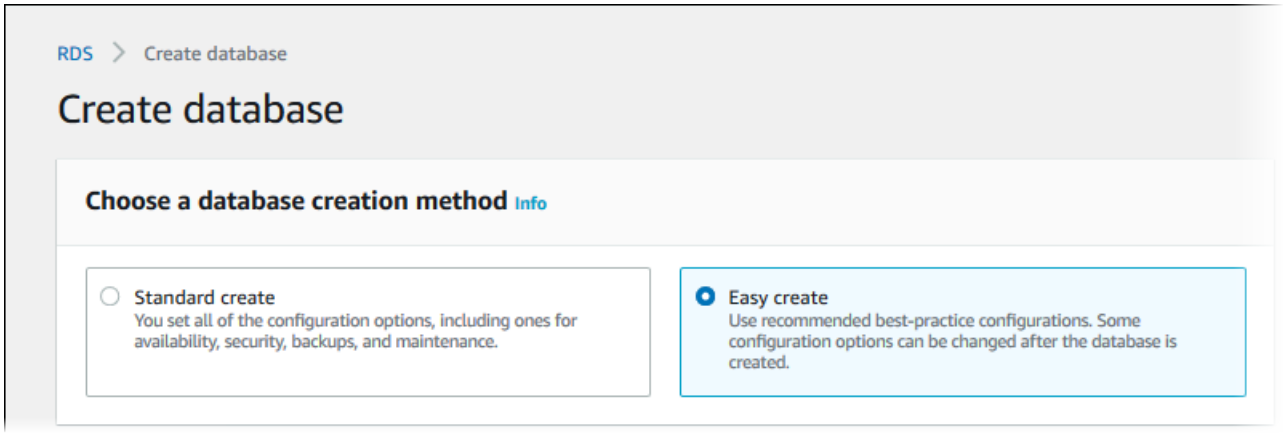
MySQL DB 인스턴스 생성

Amazon RDS의 기본 빌딩 블록은 DB 인스턴스입니다. MySQL 데이터베이스를 실행하는 환경입니다.

이 예시에서는 간편 생성을 사용하여 db.t3.micro DB 인스턴스 클래스에서 MySQL 데이터베이스 엔진을 실행하는 DB 인스턴스를 생성합니다.

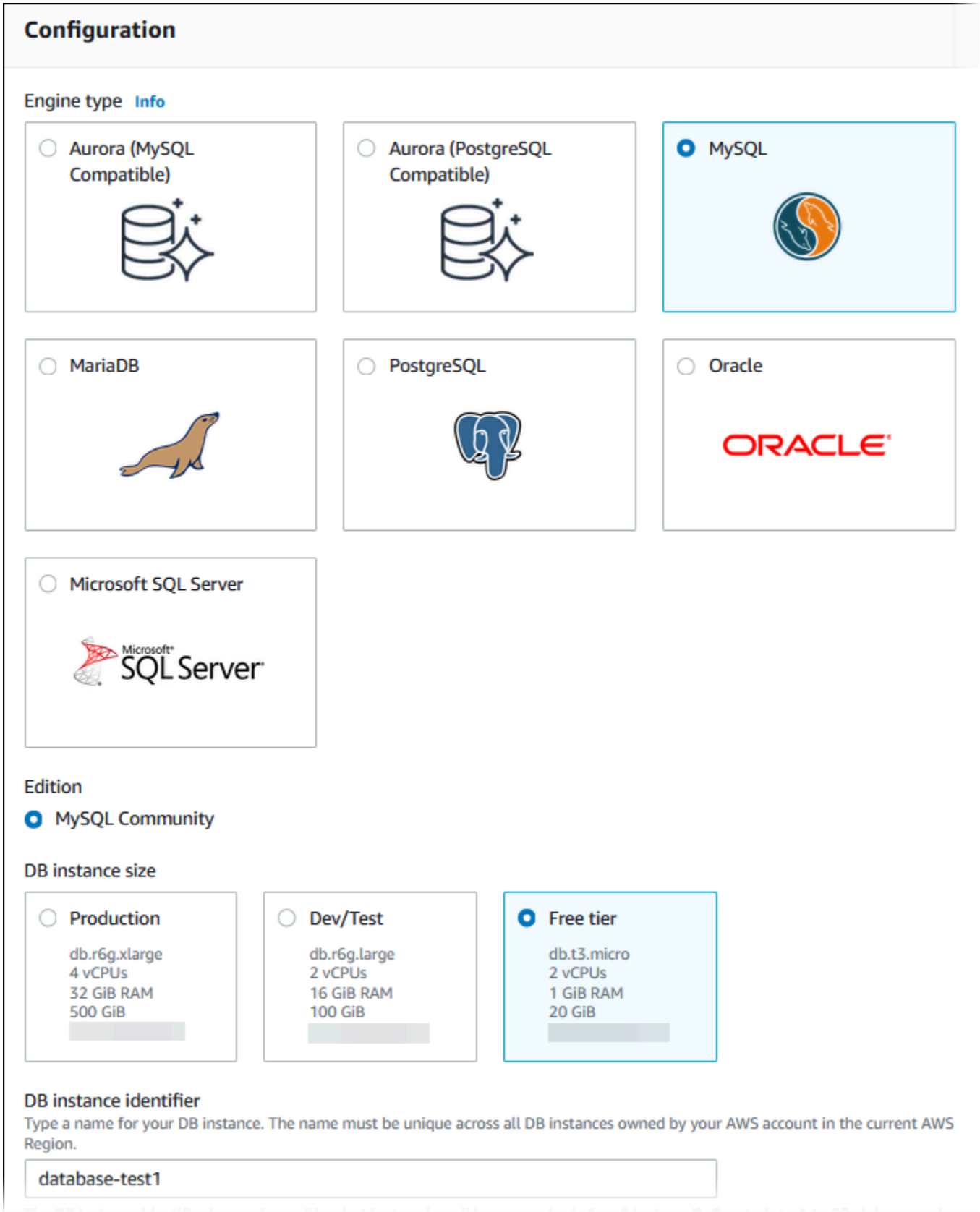
간편 생성(Easy create)을 사용하여 MySQL DB 인스턴스를 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 콘솔의 오른쪽 상단에서 이전에 EC2 인스턴스에 사용한 AWS 리전을 선택합니다.
3. 탐색 창에서 데이터베이스를 선택합니다.
4. [데이터베이스 생성(Create database)]을 선택하고 [간편 생성(Easy Create)]이 선택되어 있는지 확인합니다.



5. 구성에서 MySQL을 선택합니다.
6. DB instance size(DB 인스턴스 크기)에서 프리 티어를 선택합니다. 프리 티어는 무료 요금제 계정에 표시됩니다. 샌드박스는 유료 요금제 계정에 표시됩니다.
7. DB 인스턴스 식별자에 **`database-test1`**을 입력합니다.
8. 마스터 사용자 이름에 마스터 사용자의 이름을 입력하거나 기본 이름을 그대로 유지합니다.

   데이터베이스 생성 페이지는 다음 이미지와 비슷해야 합니다. 무료 요금제 계정에는 프리 티어가 표시됩니다. 유료 요금제 계정에는 샌드박스가 표시됩니다.

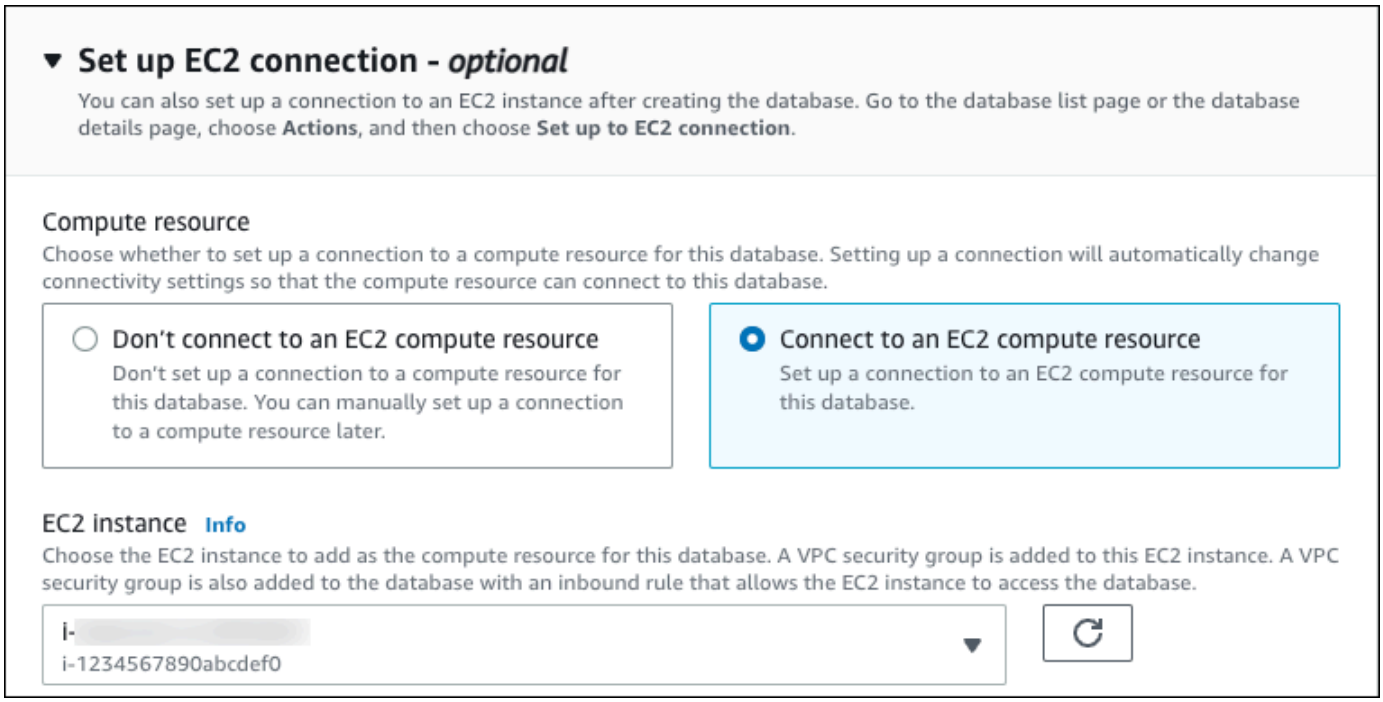


9. DB 인스턴스에 자동 생성된 마스터 암호를 사용하려면 암호 자동 생성을 선택합니다.

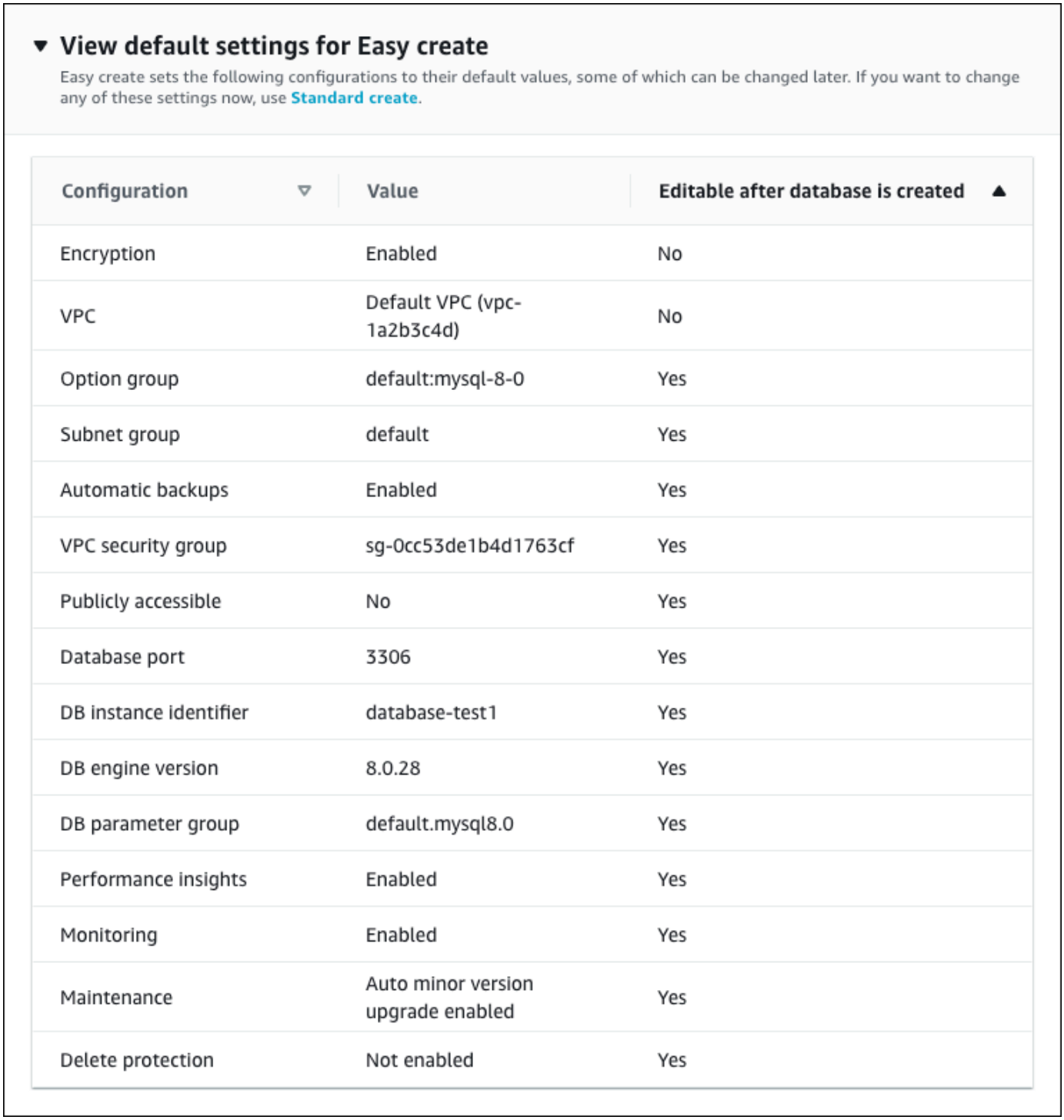
마스터 암호를 입력하려면 암호 자동 생성 선택을 해제한 다음, 마스터 암호와 암호 확인에 동일한 암호를 입력합니다.

10. 이전에 생성한 EC2 인스턴스와의 연결을 설정하려면 EC2 연결 설정 - 선택 사항을 엽니다.

    Connect to an EC2 compute resource(EC2 컴퓨팅 리소스에 연결)를 선택합니다. 이전에 생성한 EC2 인스턴스를 선택합니다.



11. (선택 사항) View default settings for Easy create(간편 생성 기본 설정 보기)를 엽니다.



View default settings for Easy create

Easy create sets the following configurations to their default values, some of which can be changed later. If you want to change any of these settings now, use **Standard create**.

Configuration	Value	Editable after database is created
Encryption	Enabled	No
VPC	Default VPC (vpc-1a2b3c4d)	No
Option group	default:mysql-8-0	Yes
Subnet group	default	Yes
Automatic backups	Enabled	Yes
VPC security group	sg-0cc53de1b4d1763cf	Yes
Publicly accessible	No	Yes
Database port	3306	Yes
DB instance identifier	database-test1	Yes
DB engine version	8.0.28	Yes
DB parameter group	default.mysql8.0	Yes
Performance insights	Enabled	Yes
Monitoring	Enabled	Yes
Maintenance	Auto minor version upgrade enabled	Yes
Delete protection	Not enabled	Yes

[간편 생성(Easy Create)]과 함께 사용되는 기본 설정을 검토할 수 있습니다. 데이터베이스 생성 후 편집 가능 열에는 데이터베이스 생성 후 어떤 옵션을 변경할 수 있는지 나와 있습니다.

- 설정의 해당 열에 아니요라고 되어 있지만 다른 설정을 원하는 경우 표준 생성을 사용하여 DB 인스턴스를 만들 수 있습니다.

- 설정의 해당 열에 예라고 되어 있으며 다른 설정을 원하는 경우 표준 생성을 사용하여 DB 인스턴스를 만들거나 DB 인스턴스를 생성한 후 수정하여 설정을 변경할 수 있습니다.

12. 데이터베이스 생성를 선택합니다.

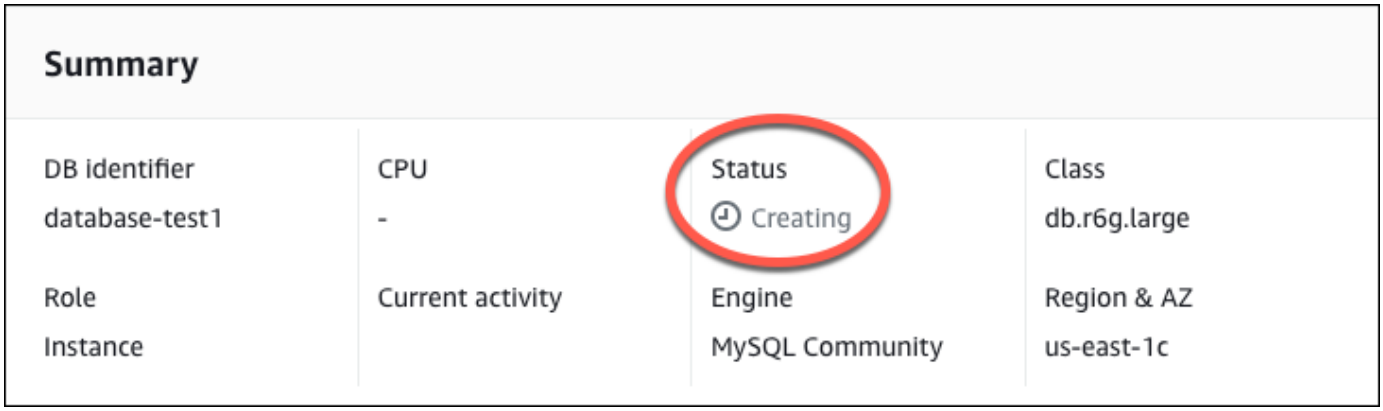
    DB 인스턴스의 마스터 사용자 이름 및 암호를 보려면 View credential details(자격 증명 세부 정보 보기)를 선택합니다.

    DB 인스턴스를 마스터 사용자로 연결하려면 화면에 나타난 사용자 이름과 암호를 사용합니다.

    > **Important**
    >
    > 마스터 사용자 암호를 다시 볼 수는 없습니다. 따라서 기록을 해두지 않으면 이를 변경해야 합니다.
    > DB 인스턴스가 사용 가능한 상태가 되고 난 후에 마스터 사용자 암호를 변경해야 하는 경우에는 다음과 같은 방법으로 DB 인스턴스를 수정할 수 있습니다. DB 인스턴스 변경에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

13. 데이터베이스 목록에서 새 MySQL DB 인스턴스의 이름을 선택하면 세부 정보가 표시됩니다.

    DB 인스턴스를 사용할 준비가 될 때까지 DB 인스턴스의 상태는 생성 중입니다.



    상태가 Available(사용 가능)로 변경되면 DB 인스턴스에 연결할 수 있습니다. DB 인스턴스 클래스와 스토리지의 양에 따라 새 인스턴스를 사용할 수 있을 때까지 최대 20분이 걸릴 수 있습니다.

(선택 사항) AWS CloudFormation를 사용하여 VPC, EC2 인스턴스 및 MySQL 인스턴스 생성

콘솔을 사용하여 VPC, EC2 인스턴스 및 MySQL 인스턴스를 생성하는 대신 AWS CloudFormation을 통해 코드형 인프라로 처리하여 AWS 리소스를 프로비저닝할 수 있습니다. AWS 리소스를 더 작고 관리하기 쉬운 단위로 구성하는 데 도움이 되도록 AWS CloudFormation 중첩 스택 기능을 사용할 수 있습니다. 자세한 내용은 AWS CloudFormation 콘솔에서 스택 생성 및 중첩된 스택 작업을 참조하세요.

> **Important**
>
> AWS CloudFormation은 무료이지만, CloudFormation에서 생성하는 리소스는 라이브입니다. 이러한 리소스를 종료하지 않으면 해당 리소스에 대한 표준 사용 요금이 발생합니다. 자세한 내용은 RDS for MySQL 요금을 참조하세요.

AWS CloudFormation 콘솔을 사용하여 리소스를 생성하려면 다음 단계를 완료합니다.

- CloudFormation 템플릿 파일을 다운로드하십시오.
- CloudFormation을 사용하여 리소스 구성

CloudFormation 템플릿 파일을 다운로드하십시오.

CloudFormation 템플릿은 JSON 또는 YAML 텍스트 파일로, 스택에서 생성하려는 리소스에 대한 구성 정보가 들어 있습니다. 또한 이 템플릿은 RDS 인스턴스와 함께 VPC와 Bastion Host를 생성합니다.

템플릿 파일을 다운로드하려면 다음 링크인 MySQL CloudFormation 템플릿을 엽니다.

Github 페이지에서 원시 파일 다운로드 버튼을 클릭하여 템플릿 YAML 파일을 저장합니다.

CloudFormation을 사용하여 리소스 구성

> **Note**
>
> 이 프로세스를 시작하기 전에 AWS 계정에 EC2 인스턴스용 키 페어가 있는지 확인합니다. 자세한 내용은 Amazon EC2 키 페어 및 Linux 인스턴스를 참조하세요.

AWS CloudFormation 템플릿을 사용할 때는 리소스가 제대로 생성되도록 올바른 파라미터를 선택해야 합니다. 다음 단계를 따릅니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/cloudformation에서 AWS CloudFormation 콘솔을 엽니다.
2. 스택 생성을 선택합니다.
3. 템플릿 지정 섹션에서 컴퓨터에서 템플릿 파일 업로드를 선택하고 다음을 선택합니다.
4. 스택 세부 정보 지정 페이지에서 다음 파라미터를 설정합니다.
   a. 스택 이름을 MySQLTestStack으로 설정합니다.
   b. 파라미터에서 가용 영역 3개를 선택하여 가용 영역을 설정합니다.
   c. Linux Bastion Host 구성에서 키 이름에 대해 EC2 인스턴스에 로그인할 키 페어를 선택합니다.
   d. Linux Bastion Host 구성 설정에서 허용된 IP 범위를 IP 주소로 설정합니다. Secure Shell(SSH)을 사용하여 VPC의 EC2 인스턴스에 연결하려면 https://checkip.amazonaws.com의 서비스를 사용하여 퍼블릭 IP 주소를 지정합니다. IP 주소의 예는 192.0.2.1/32입니다.

      > **Warning**
      >
      > SSH 액세스에 `0.0.0.0/0`을 사용하는 경우 모든 IP 주소가 SSH를 사용하여 퍼블릭 EC2 인스턴스에 액세스할 수 있도록 활성화합니다. 이 방법은 테스트 환경에서 잠시 사용하는 것은 괜찮지만 프로덕션 환경에서는 안전하지 않습니다. 프로덕션에서는 SSH를 사용하여 EC2 인스턴스에 액세스할 수 있는 특정 IP 주소 또는 주소 범위만 인증합니다.

   e. 데이터베이스 일반 구성에서 데이터베이스 인스턴스 클래스를 db.t3.micro로 설정합니다.
   f. 데이터베이스 이름을 **`database-test1`**으로 설정합니다.
   g. 데이터베이스 마스터 사용자 이름에 마스터 사용자 이름을 입력합니다.
   h. 이 자습서에서는 Secrets Manager를 사용하여 DB 마스터 사용자 암호 관리를 `false`로 설정합니다.
   i. 데이터베이스 암호의 경우 원하는 암호를 설정합니다. 자습서의 향후 단계에 사용할 수 있도록 암호를 기억해 둡니다.
   j. 데이터베이스 스토리지 구성에서 데이터베이스 스토리지 유형을 gp2로 설정합니다.
   k. 데이터베이스 모니터링 구성에서 RDS 성능 개선 도우미 활성화를 false로 설정합니다.
   l. 다른 모든 설정은 기본값으로 둡니다. 계속하려면 다음을 클릭합니다.
5. 스택 옵션 구성 페이지에서는 모든 기본 옵션을 그대로 둡니다. 계속하려면 다음을 클릭합니다.
6. 스택 검토 페이지에서 데이터베이스 및 Linux Bastion Host 옵션을 확인한 후 제출을 선택합니다.

스택 생성 프로세스가 완료되면 BastionStack 및 RDSNS라는 이름의 스택을 보고 데이터베이스에 연결하는 데 필요한 정보를 기록해 둡니다. 자세한 내용은 AWS Management Console에서 AWS CloudFormation 스택 데이터 및 리소스 보기를 참조하세요.

MySQL DB 인스턴스에 연결

표준 SQL 클라이언트 애플리케이션을 사용해 DB 인스턴스에 연결할 수 있습니다. 이 예시에서는 mysql 명령줄 클라이언트를 사용해 MySQL DB 인스턴스에 연결합니다.

MySQL DB 인스턴스에 연결하는 방법

1. DB 인스턴스의 엔드포인트(DNS 이름)와 포트 번호를 찾습니다.

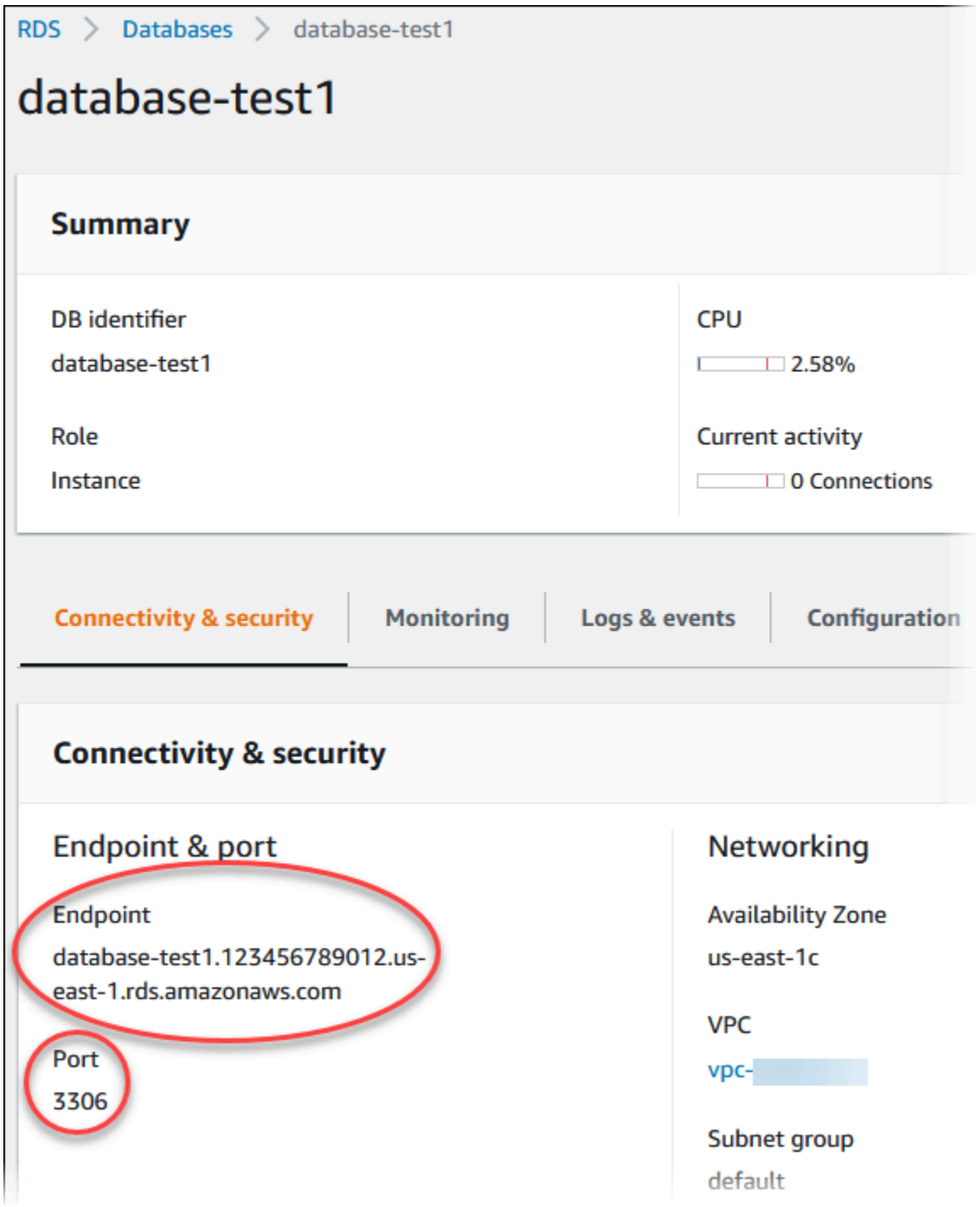
    a. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.

    b. Amazon RDS 콘솔의 오른쪽 상단에서 DB 인스턴스의 AWS 리전을 선택합니다.

    c. 탐색 창에서 Databases(데이터베이스)를 선택합니다.

    d. 세부 정보를 표시하고자 하는 MySQL DB 인스턴스 이름을 선택합니다.

    e. Connectivity & security(연결 및 보안) 탭에서 엔드포인트를 복사합니다. 또한 포트 번호를 적어 둡니다. DB 인스턴스에 연결하려면 엔드포인트와 포트 번호가 모두 필요합니다.



2. Amazon EC2 사용 설명서에 있는 Linux 인스턴스에 연결의 단계를 따라 앞에서 만든 EC2 인스턴스에 연결합니다.

   SSH를 사용하여 EC2 인스턴스에 연결하는 것이 좋습니다. Windows, Linux 또는 Mac에 SSH 클라이언트 유틸리티가 설치된 경우 다음 명령 형식을 사용하여 인스턴스에 연결할 수 있습니다.

```
ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
```

예를 들어 `ec2-database-connect-key-pair.pem`이 Linux의 `/dir1`에 저장되어 있고, EC2 인스턴스의 퍼블릭 IPv4 DNS가 `ec2-12-345-678-90.compute-1.amazonaws.com`이라고 가정해 보겠습니다. SSH 명령은 다음과 같이 표시됩니다.

```
ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
user@ec2-12-345-678-90.compute-1.amazonaws.com
```

3. EC2 인스턴스에서 소프트웨어를 업데이트하여 최신 버그 수정 및 보안 업데이트를 받습니다. 이렇게 하려면 다음 명령을 사용하십시오.

> **Note**
>
> `-y` 옵션을 사용하면 확인 여부를 묻지 않고 업데이트를 설치합니다. 설치 전에 업데이트 정보를 확인하려면 이 옵션을 생략합니다.

```
sudo dnf update -y
```

4. Amazon Linux 2023에서 MariaDB의 mysql 명령줄 클라이언트를 설치하려면 다음 명령을 실행합니다.

```
sudo dnf install mariadb105
```

5. MySQL DB 인스턴스에 연결합니다. 예를 들어, 다음 명령을 입력합니다. 이 작업을 통해 MySQL 클라이언트를 사용하여 MySQL DB 인스턴스에 연결할 수 있습니다.

   *endpoint*는 DB 인스턴스 엔드포인트(DNS 이름)로 대체하고, *admin*는 사용된 마스터 사용자 이름으로 대체합니다. 암호를 묻는 메시지가 표시되면 사용한 마스터 암호를 제공합니다.

```
mysql -h endpoint -P 3306 -u admin -p
```

   사용자에 대한 암호를 입력하면 다음과 유사한 출력이 나타납니다.

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 3082
Server version: 8.0.28 Source distribution
```

```
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

MySQL DB 인스턴스 연결에 대한 자세한 내용은 MySQL DB 인스턴스에 연결 단원을 참조하십시오. DB 인스턴스에 연결할 수 없는 경우 Amazon RDS DB 인스턴스에 연결할 수 없음 단원을 참조하십시오.

보안을 위해서는 암호화된 연결을 사용하는 것이 가장 좋습니다. 클라이언트와 서버가 동일한 VPC에 있고 네트워크를 신뢰할 수 있는 경우에만 암호화되지 않은 MySQL 연결을 사용합니다. 암호화된 연결 사용에 대한 자세한 내용은 MySQL 명령줄 클라이언트(암호화됨)에서 SSL/TLS를 사용하여 Amazon RDS의 MySQL DB 인스턴스에 연결 섹션을 참조하세요.

6. SQL 명령을 실행합니다.

   예를 들어, 다음 SQL 명령은 현재 날짜 및 시간을 보여줍니다.

```
SELECT CURRENT_TIMESTAMP;
```

EC2 인스턴스 및 DB 인스턴스 삭제

생성한 샘플 EC2 인스턴스 및 DB 인스턴스에 연결하고 탐색한 후에는 요금이 더 이상 부과되지 않도록 삭제합니다.

AWS CloudFormation을 사용하여 리소스를 생성했다면 이 단계를 건너뛰고 다음 단계로 이동합니다.

EC2 인스턴스를 삭제하는 방법

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창에서 인스턴스를 선택합니다.
3. EC2 인스턴스를 선택하고 인스턴스 상태, 인스턴스 종료를 차례로 선택합니다.
4. 확인 메시지가 나타나면 종료를 선택합니다.

EC2 인스턴스 삭제에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 종료를 참조하세요.

최종 DB 스냅샷이 없는 DB 인스턴스를 삭제하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 삭제할 DB 인스턴스를 선택합니다.
4. [Actions]에 대해 [Delete]를 선택합니다.
5. 최종 스냅샷을 생성하시겠습니까? 및 자동 백업 보존을 선택 해제합니다.
6. 확인을 완료하고 삭제를 선택합니다.

(선택 사항) CloudFormation으로 생성한 EC2 인스턴스 및 DB 인스턴스 삭제

AWS CloudFormation을 사용하여 리소스를 생성한 경우 샘플 EC2 인스턴스 및 DB 인스턴스에 연결하고 탐색한 후 CloudFormation 스택을 삭제하여 더 이상 비용이 청구되지 않도록 합니다.

CloudFormation 리소스를 삭제하려면

1. AWS CloudFormation 콘솔을 엽니다.
2. CloudFormation 콘솔의 스택 페이지에서 루트 스택(VPCStack, BastionStack 또는 RDSNS라는 이름이 없는 스택)을 선택합니다.
3. 삭제를 선택합니다.
4. 확인 메시지가 나타나면 스택 삭제를 선택합니다.

CloudFormation에서 스택을 삭제하는 방법에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS CloudFormation 콘솔에서 스택 삭제를 참조하세요.

(선택 사항) DB 인스턴스를 Lambda 함수에 연결

RDS for MySQL DB 인스턴스를 Lambda 서버리스 컴퓨팅 리소스에 연결할 수도 있습니다. Lambda 함수를 사용하면 인프라를 프로비저닝하거나 관리하지 않고도 코드를 실행할 수 있습니다. 또한 하루 12개에서 초당 수백 개에 이르는 모든 규모의 코드 실행 요청에 자동으로 응답할 수 있습니다. 자세한 내용은 Lambda 함수와 DB 인스턴스 자동 연결 섹션을 참조하세요.

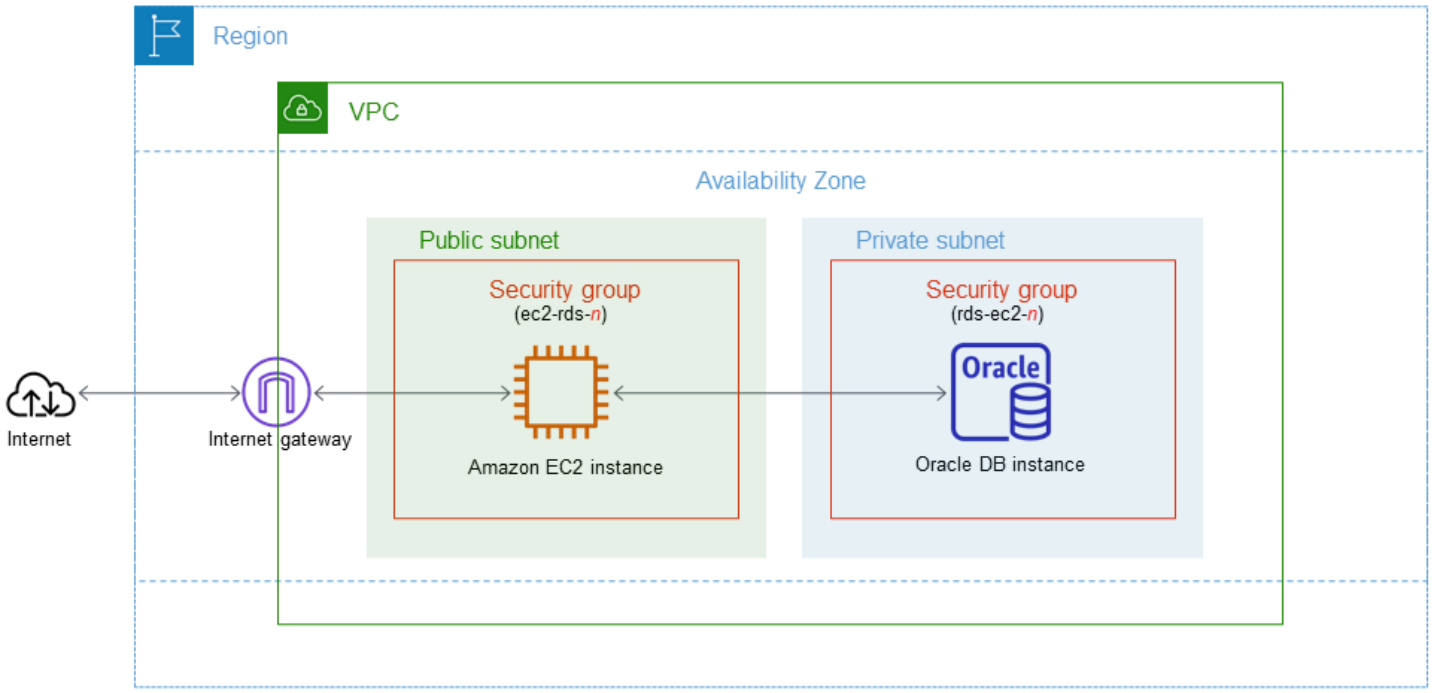
Oracle DB 인스턴스 생성 및 해당 인스턴스에 연결

이 자습서에서는 EC2 인스턴스와 RDS for Oracle DB 인스턴스를 생성합니다. 자습서에서는 표준 Oracle 클라이언트를 사용하여 EC2 인스턴스에서 DB 인스턴스에 액세스하는 방법을 보여줍니다. 이 자습서에서는 모범 사례를 따라 Virtual Private Cloud(VPC)에서 프라이빗 DB 인스턴스를 생성합니다. 대부분의 경우 EC2 인스턴스와 같이 동일한 VPC에 있는 다른 리소스는 DB 인스턴스에 액세스할 수 있지만 VPC 외부의 리소스는 DB 인스턴스에 액세스할 수 없습니다.

자습서를 완료하면 VPC의 각 가용 영역에 퍼블릭 서브넷과 프라이빗 서브넷이 있을 것입니다. 한 가용 영역에서 EC2 인스턴스는 퍼블릭 서브넷에 있고 DB 인스턴스는 프라이빗 서브넷에 있습니다.

> **Important**
>
> AWS 계정 생성은 무료입니다. 그러나 이 자습서를 완료하면 사용하는 AWS 리소스에 대한 비용이 발생할 수 있습니다. 자습서가 더 이상 필요하지 않은 경우 자습서를 완료한 후에 이러한 리소스를 삭제할 수 있습니다.

다음 다이어그램은 이 자습서를 완료했을 때 구성을 보여 줍니다.



이 자습서에서는 다음 방법 중 하나를 사용하여 리소스를 생성할 수 있습니다.

1. AWS Management Console 사용 - 2단계: Oracle DB 인스턴스 생성 및 1단계: EC2 인스턴스 생성

2. 데이터베이스 인스턴스 및 EC2 인스턴스를 생성하는 데 AWS CloudFormation 사용 - (선택 사항) AWS CloudFormation를 사용하여 VPC, EC2 인스턴스 및 Oracle DB 인스턴스 생성

첫 번째 방법은 간편 생성을 사용하여 AWS Management Console을 통해 프라이빗 Oracle DB 인스턴스를 생성합니다. 여기에서는 DB 엔진 유형, DB 인스턴스 크기 및 DB 인스턴스 식별자만 지정합니다. [간편 생성(Easy create)]은 다른 구성 옵션에서도 기본 설정을 사용합니다.

표준 생성을 대신 사용하는 경우에는 DB 인스턴스를 생성할 때 더 많은 구성 옵션을 지정할 수 있습니다. 이러한 옵션에는 가용성, 보안, 백업 및 유지 관리에 대한 설정이 포함됩니다. 퍼블릭 DB 인스턴스를 만들려면 표준 생성을 사용해야 합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.

주제

- 사전 조건
- 1단계: EC2 인스턴스 생성
- 2단계: Oracle DB 인스턴스 생성
- (선택 사항) AWS CloudFormation를 사용하여 VPC, EC2 인스턴스 및 Oracle DB 인스턴스 생성
- 3단계: SQL 클라이언트를 Oracle DB 인스턴스에 연결
- 4단계: EC2 인스턴스 및 DB 인스턴스 삭제
- (선택 사항) CloudFormation으로 생성한 EC2 인스턴스 및 DB 인스턴스 삭제
- (선택 사항) DB 인스턴스를 Lambda 함수에 연결

사전 조건

시작하기 전에 다음 섹션에서 다음 단계를 완료하세요.

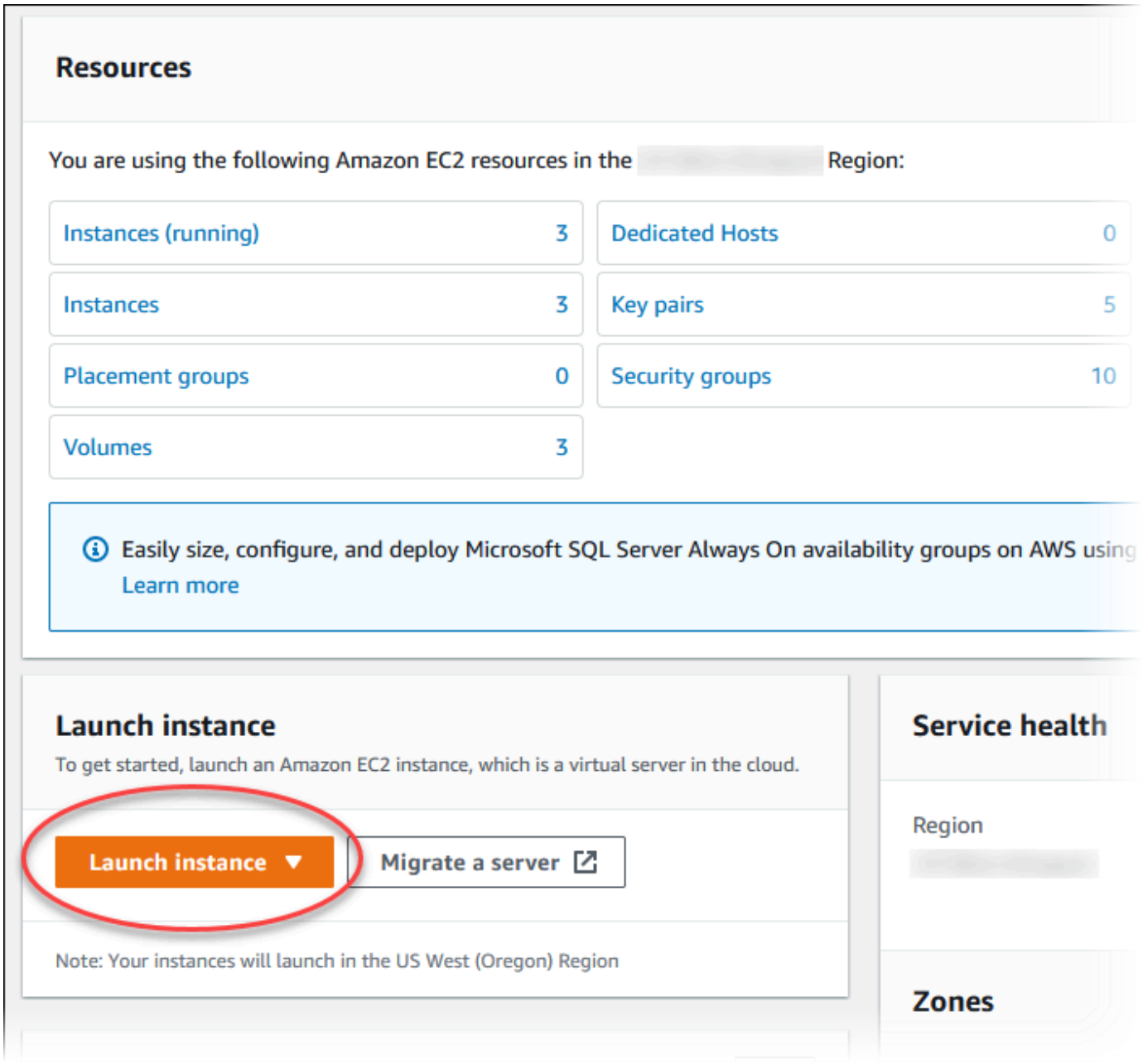
- AWS 계정에 등록
- 관리자 액세스 권한이 있는 사용자 생성

1단계: EC2 인스턴스 생성

데이터베이스에 연결하는 데 사용할 Amazon EC2 인스턴스를 생성합니다.

EC2 인스턴스를 생성하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. AWS Management Console의 오른쪽 상단에서 EC2 인스턴스를 생성하려는 AWS 리전을 선택합니다.
3. 다음 이미지에 나와 있는 것처럼 EC2 대시보드를 선택한 다음, 인스턴스 시작을 선택합니다.

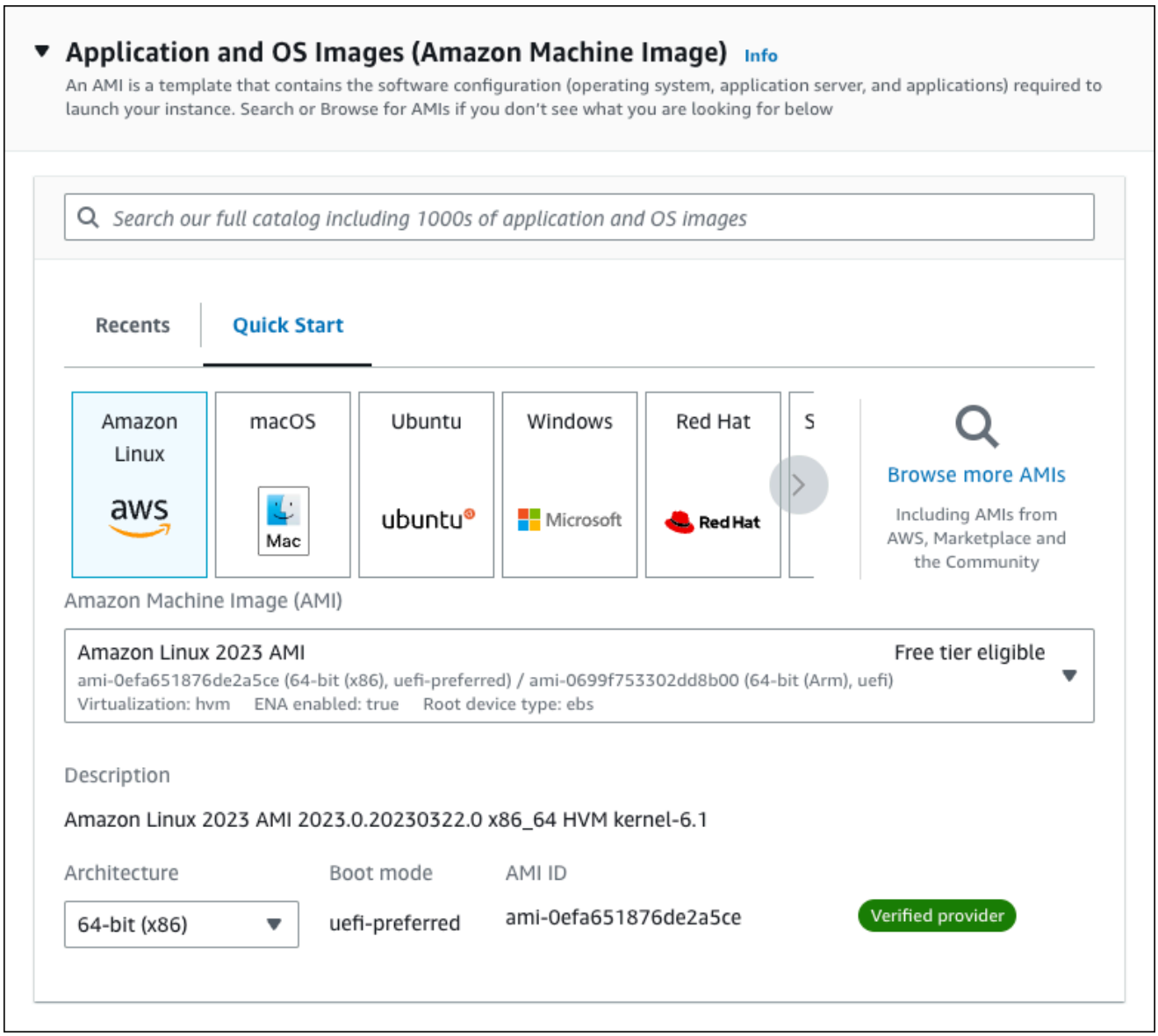


인스턴스 시작 페이지가 열립니다.

4. 인스턴스 시작 페이지에서 다음 설정을 선택합니다.

   a. Name and tags(이름 및 태그) 아래의 Name(이름)에 **ec2-database-connect**을 입력하세요.

   b. Application and OS Images (Amazon Machine Image)(애플리케이션 및 OS 이미지(Amazon Machine Image))에서 Amazon Linux를 선택한 다음 Amazon Linux 2023 AMI를 선택합니다. 다른 선택 항목에 대해서는 기본값을 그대로 유지합니다.



   c. 인스턴스 유형에서 t2.micro를 선택합니다.

   d. 키 페어(로그인)에서 기존 키 페어를 사용할 키 페어 이름을 선택합니다. Amazon EC2 인스턴스에 대한 새 키 페어를 생성하려면 새 키 페어 생성을 선택한 다음 키 페어 생성 창을 사용하여 생성합니다.

새 키 페어 생성에 대한 자세한 내용은 Amazon EC2 Linux 인스턴스용 사용 설명서의 키 페어 생성을 참조하세요.

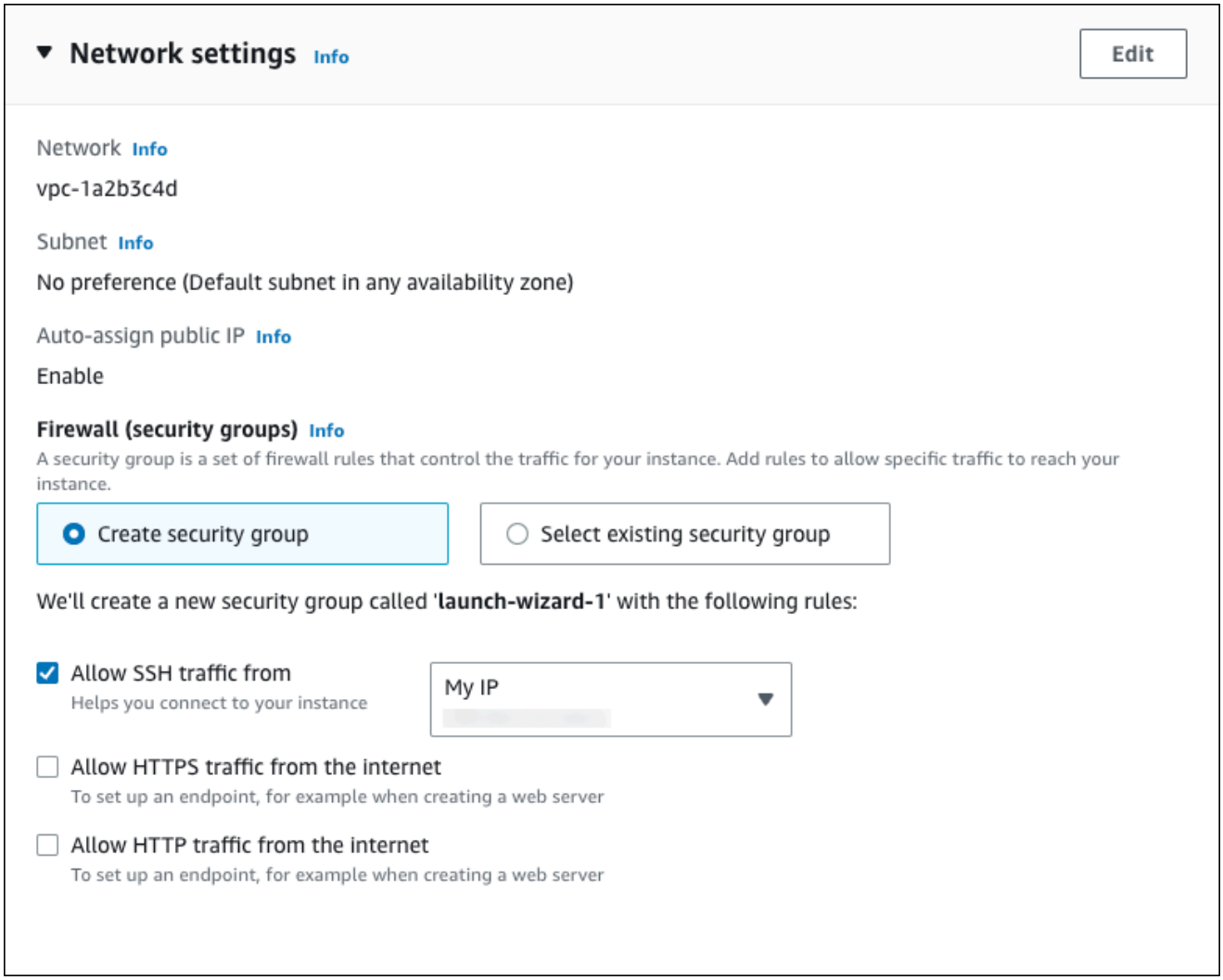
e. 네트워크 설정의 SSH 트래픽 허용에서 EC2 인스턴스에 대한 SSH 연결 소스를 선택합니다.

표시된 IP 주소가 SSH 연결에 대해 올바른 경우 내 IP를 선택할 수 있습니다. 그렇지 않으면 SSH(Secure Shell)를 사용하여 VPC의 EC2 인스턴스에 연결하는 데 사용할 IP 주소를 결정할 수 있습니다. 퍼블릭 IP 주소를 확인하려면 다른 브라우저 창 또는 탭에서 https://checkip.amazonaws.com의 서비스를 사용합니다. IP 주소의 예는 192.0.2.1/32입니다.

대부분의 경우 고정 IP 주소가 없는 방화벽 뒤나 인터넷 서비스 제공업체(ISP)를 통해 연결하는 경우가 많습니다. 그렇다면 클라이언트 컴퓨터에서 사용하는 IP 주소 범위를 결정합니다.

> **Warning**
>
> SSH 액세스에 `0.0.0.0/0`을 사용하는 경우 모든 IP 주소가 SSH를 사용하여 퍼블릭 EC2 인스턴스에 액세스할 수 있도록 활성화합니다. 이 방법은 테스트 환경에서 잠시 사용하는 것은 괜찮지만 프로덕션 환경에서는 안전하지 않습니다. 프로덕션에서는 SSH를 사용하여 EC2 인스턴스에 액세스할 수 있는 특정 IP 주소 또는 주소 범위만 인증합니다.

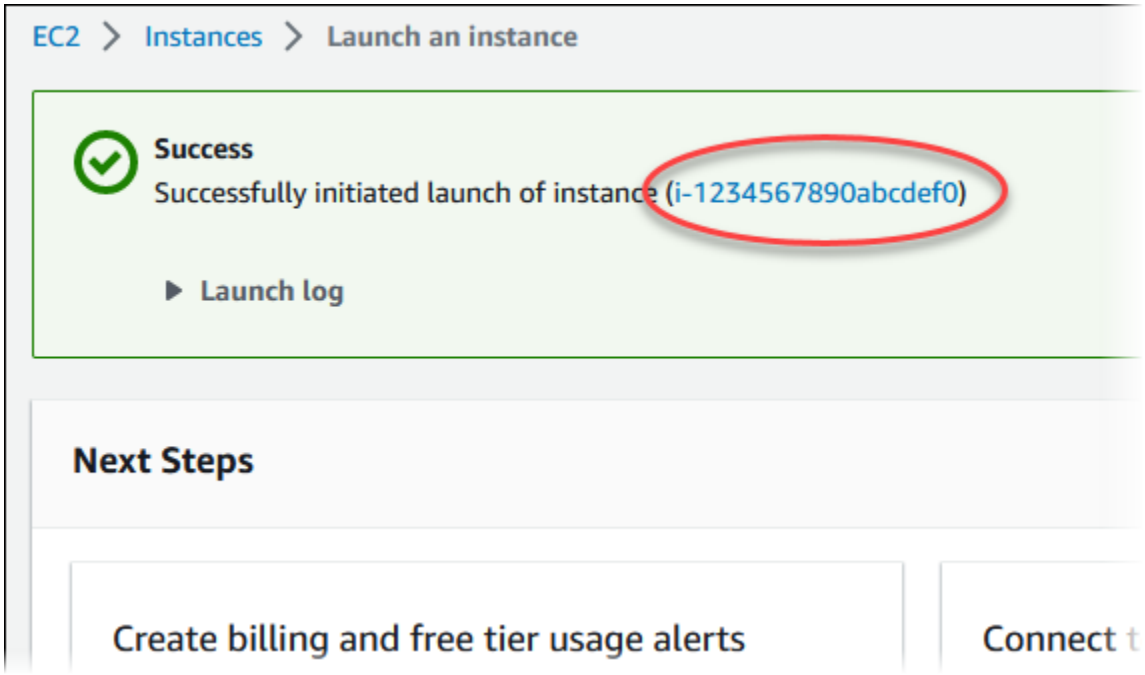
다음 이미지는 네트워크 설정 섹션의 예를 보여 줍니다.



f. 나머지 섹션에서 기본값은 그대로 둡니다.

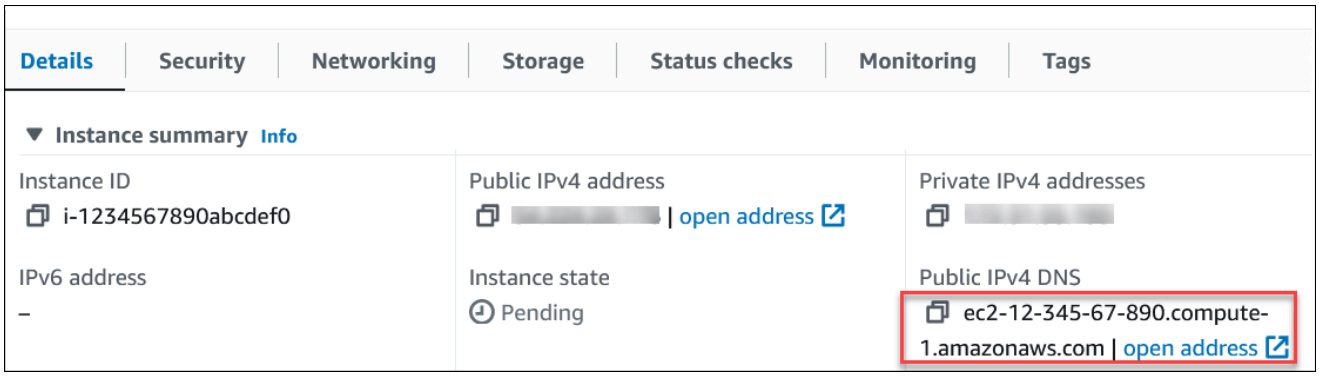
g. 요약 패널에서 EC2 인스턴스 구성 요약을 검토하고 준비가 되면 인스턴스 시작을 선택합니다.

5. 시작 상태 페이지에서, 새 EC2 인스턴스의 식별자(예: `i-1234567890abcdef0`)를 기록해 둡니다.

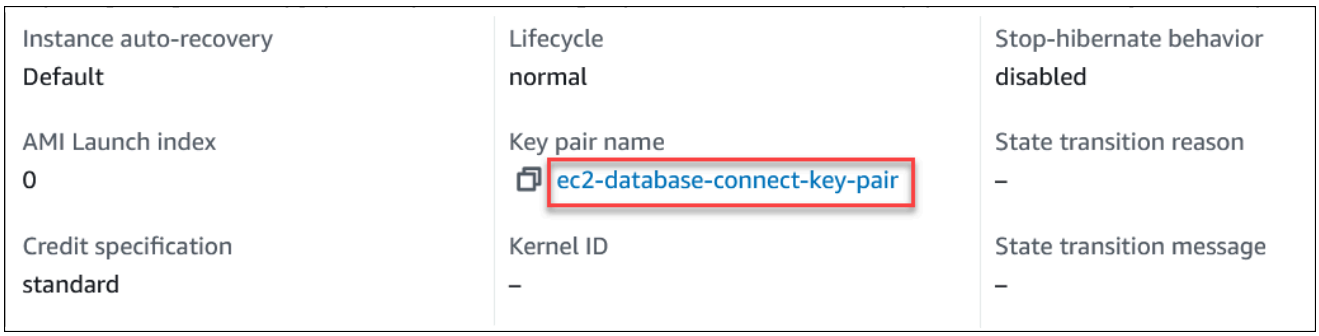


6. EC2 인스턴스 식별자를 선택하여 EC2 인스턴스 목록을 열고 EC2 인스턴스를 선택합니다.
7. 세부 정보 탭에서 SSH를 사용하여 연결할 때 필요한 다음 값을 기록해 둡니다.

   a. 인스턴스 요약에서 퍼블릭 IPv4 DNS의 값을 기록해 둡니다.



   b. 인스턴스 세부 정보에서 키 페어 이름의 값을 기록해 둡니다.



8. 계속하기 전에 EC2 인스턴스의 인스턴스 상태가 실행 중이 될 때까지 기다립니다.

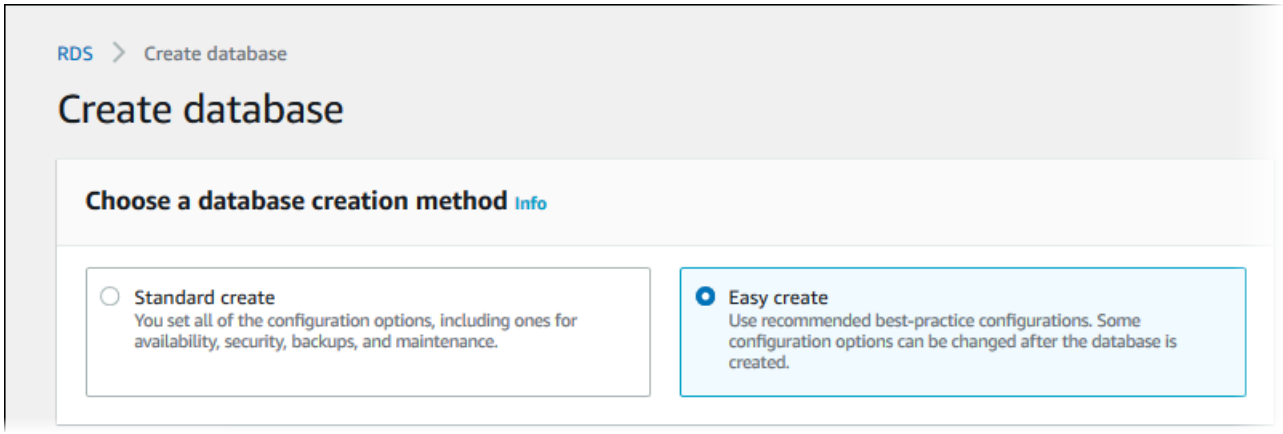
2단계: Oracle DB 인스턴스 생성

Amazon RDS의 기본 빌딩 블록은 DB 인스턴스입니다. Oracle 데이터베이스를 실행하는 환경입니다.

이 예시에서는 간편 생성을 사용하여 db.m5.large DB 인스턴스 클래스에서 Oracle 데이터베이스 엔진을 실행하는 DB 인스턴스를 생성합니다.

간편 생성을 사용하여 Oracle DB 인스턴스를 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 콘솔의 오른쪽 상단에서 DB 인스턴스를 생성하려는 AWS 리전을 선택합니다.
3. 탐색 창에서 데이터베이스를 선택합니다.
4. [데이터베이스 생성(Create database)]을 선택하고 [간편 생성(Easy Create)]이 선택되어 있는지 확인합니다.



5. 구성에서 Oracle을 선택합니다.
6. DB instance size(DB 인스턴스 크기)에서 개발/테스트를 선택합니다.
7. DB 인스턴스 식별자에 **`database-test1`**을 입력합니다.
8. 마스터 사용자 이름에 마스터 사용자의 이름을 입력하거나 기본 이름을 그대로 유지합니다.

   데이터베이스 생성 페이지는 다음 이미지와 비슷해야 합니다.

Configuration

Engine type Info

- Aurora (MySQL Compatible)
- Aurora (PostgreSQL Compatible)
- MySQL
- MariaDB
- PostgreSQL
- Oracle
- Microsoft SQL Server

Edition

- Oracle Enterprise Edition
  Affordable and full-featured database management system supporting up to 16 vCPUs.
- Oracle Standard Edition Two
  Affordable and full-featured database management system supporting up to 16 vCPUs. Oracle Database Standard Edition Two is a replacement for Standard Edition and Standard Edition One.

DB instance size

- Production
  db.r5.large
  2 vCPUs
  16 GiB RAM
  500 GiB
- Dev/Test
  db.m5.large
  2 vCPUs
  8 GiB RAM
  100 GiB

DB instance identifier

Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

database-test1

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

Master username Info

Type a login ID for the master user of your DB instance.

admin

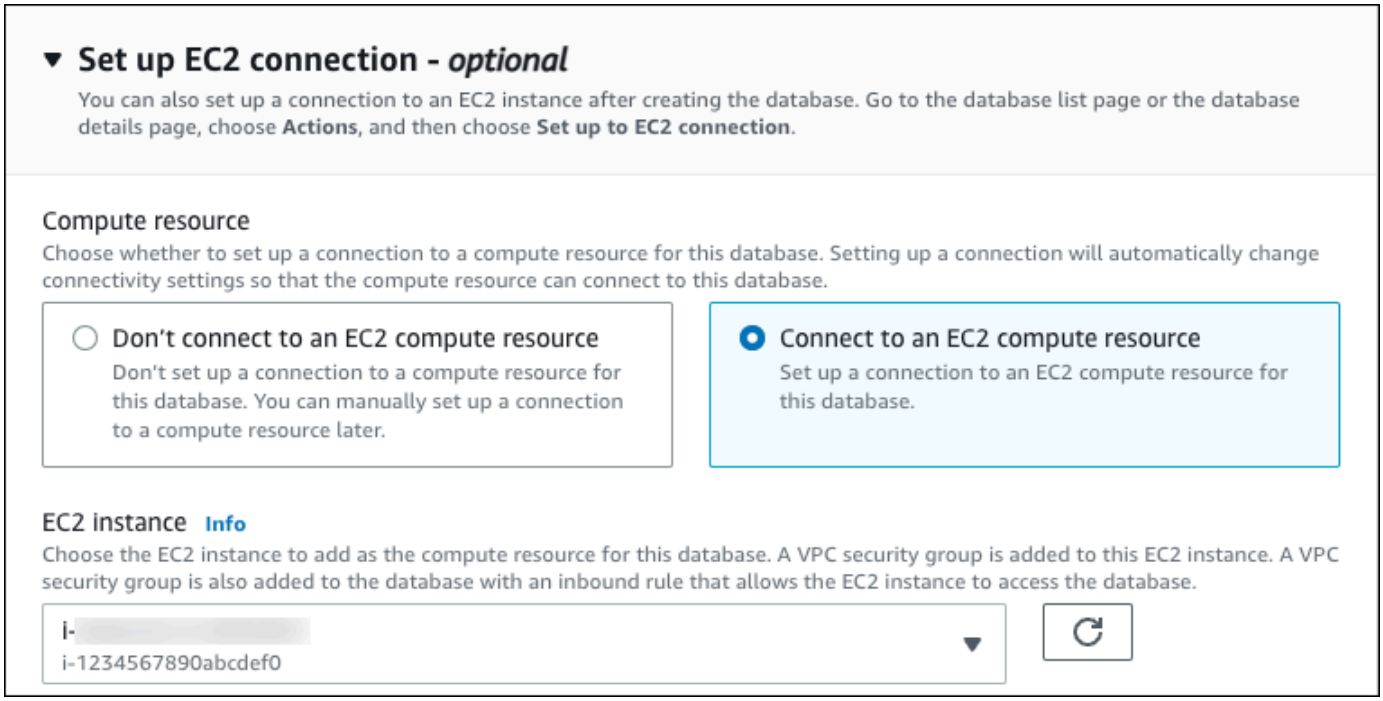
1 to 16 alphanumeric characters. First character must be a letter.

9. DB 인스턴스에 자동 생성된 마스터 암호를 사용하려면 암호 자동 생성을 선택합니다.

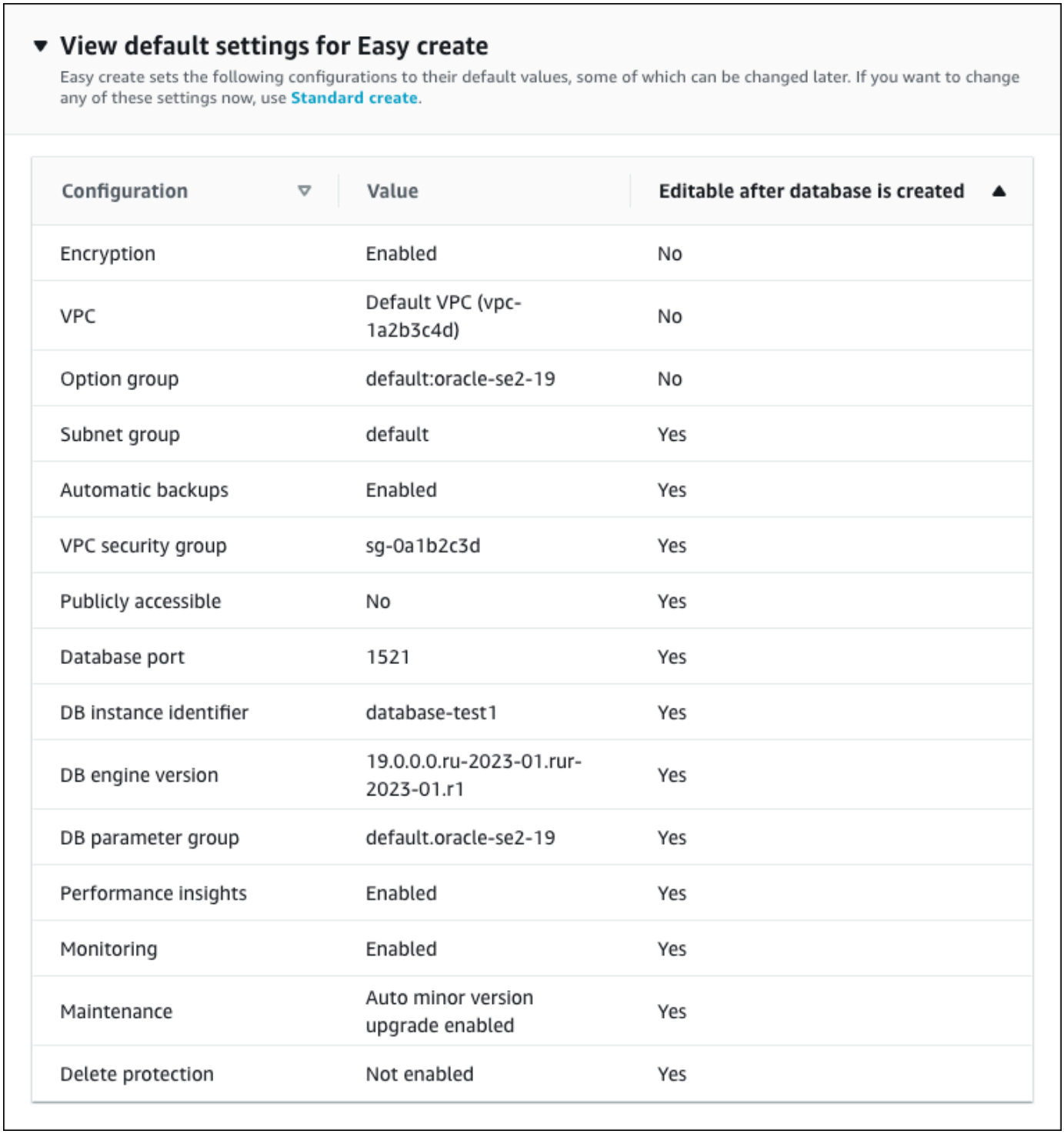
   마스터 암호를 입력하려면 암호 자동 생성 선택을 해제한 다음, 마스터 암호와 암호 확인에 동일한 암호를 입력합니다.

10. 이전에 생성한 EC2 인스턴스와의 연결을 설정하려면 EC2 연결 설정 - 선택 사항을 엽니다.

    Connect to an EC2 compute resource(EC2 컴퓨팅 리소스에 연결)를 선택합니다. 이전에 생성한 EC2 인스턴스를 선택합니다.



11. 간편 생성 기본 설정 보기를 엽니다.



View default settings for Easy create

Easy create sets the following configurations to their default values, some of which can be changed later. If you want to change any of these settings now, use **Standard create**.

Configuration	Value	Editable after database is created
Encryption	Enabled	No
VPC	Default VPC (vpc-1a2b3c4d)	No
Option group	default:oracle-se2-19	No
Subnet group	default	Yes
Automatic backups	Enabled	Yes
VPC security group	sg-0a1b2c3d	Yes
Publicly accessible	No	Yes
Database port	1521	Yes
DB instance identifier	database-test1	Yes
DB engine version	19.0.0.0.ru-2023-01.rur-2023-01.r1	Yes
DB parameter group	default.oracle-se2-19	Yes
Performance insights	Enabled	Yes
Monitoring	Enabled	Yes
Maintenance	Auto minor version upgrade enabled	Yes
Delete protection	Not enabled	Yes

[간편 생성(Easy Create)]과 함께 사용되는 기본 설정을 검토할 수 있습니다. 데이터베이스 생성 후 편집 가능 열에는 데이터베이스 생성 후 어떤 옵션을 변경할 수 있는지 나와 있습니다.

- 설정의 해당 열에 아니요라고 되어 있지만 다른 설정을 원하는 경우 표준 생성을 사용하여 DB 인스턴스를 만들 수 있습니다.

- 설정의 해당 열에 예라고 되어 있으며 다른 설정을 원하는 경우 표준 생성을 사용하여 DB 인스턴스를 만들거나 DB 인스턴스를 생성한 후 수정하여 설정을 변경할 수 있습니다.

12. 데이터베이스 생성를 선택합니다.

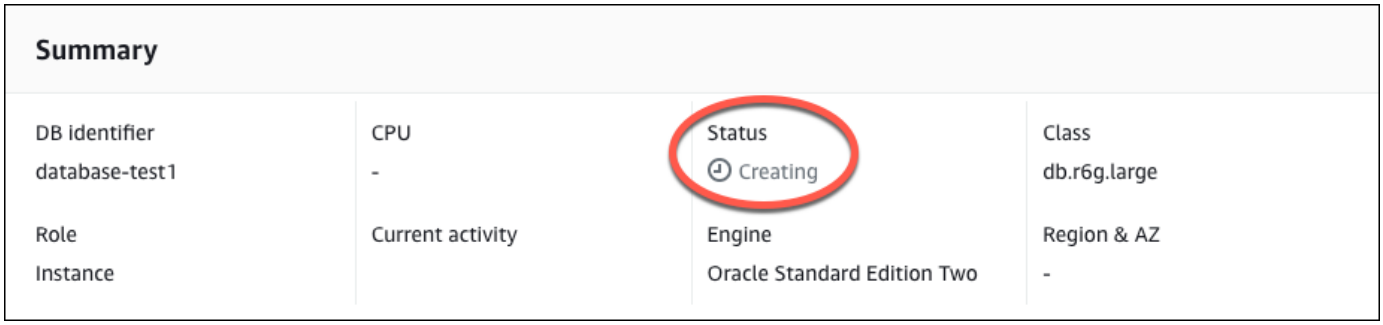
    DB 인스턴스의 마스터 사용자 이름 및 암호를 보려면 View credential details(자격 증명 세부 정보 보기)를 선택합니다.

    DB 인스턴스를 마스터 사용자로 연결하려면 화면에 나타난 사용자 이름과 암호를 사용합니다.

    > **Important**
    >
    > 마스터 사용자 암호를 다시 볼 수는 없습니다. 따라서 기록을 해두지 않으면 이를 변경해야 합니다.
    > DB 인스턴스가 사용 가능한 상태가 되고 난 후에 마스터 사용자 암호를 변경해야 하는 경우에는 다음과 같은 방법으로 DB 인스턴스를 수정할 수 있습니다. DB 인스턴스 변경에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

13. 데이터베이스 목록에서 새 Oracle DB 인스턴스의 이름을 선택하면 세부 정보가 표시됩니다.

    DB 인스턴스를 사용할 준비가 될 때까지 DB 인스턴스의 상태는 생성 중입니다.



    상태가 Available(사용 가능)로 변경되면 DB 인스턴스에 연결할 수 있습니다. DB 인스턴스 클래스와 스토리지의 양에 따라 새 인스턴스를 사용할 수 있을 때까지 최대 20분이 걸릴 수 있습니다. DB 인스턴스를 생성하는 동안 다음 단계를 진행하여 EC2 인스턴스를 생성할 수 있습니다.

(선택 사항) AWS CloudFormation를 사용하여 VPC, EC2 인스턴스 및 Oracle DB 인스턴스 생성

콘솔을 사용하여 VPC, EC2 인스턴스 및 Oracle DB 인스턴스를 생성하는 대신 AWS CloudFormation을 통해 코드형 인프라로 처리하여 AWS 리소스를 프로비저닝할 수 있습니다. AWS 리소스를 더 작고

관리하기 쉬운 단위로 구성하는 데 도움이 되도록 AWS CloudFormation 중첩 스택 기능을 사용할 수 있습니다. 자세한 내용은 AWS CloudFormation 콘솔에서 스택 생성 및 중첩된 스택 작업을 참조하세요.

> **Important**
>
> AWS CloudFormation은 무료이지만, CloudFormation에서 생성하는 리소스는 라이브입니다. 이러한 리소스를 종료하지 않으면 해당 리소스에 대한 표준 사용 요금이 발생합니다. 자세한 내용은 RDS for Oracle 요금을 참조하세요.

AWS CloudFormation 콘솔을 사용하여 리소스를 생성하려면 다음 단계를 완료합니다.

- 1단계: CloudFormation 템플릿 다운로드
- 2단계: CloudFormation을 사용하여 리소스 구성

CloudFormation 템플릿 파일을 다운로드하십시오.

CloudFormation 템플릿은 JSON 또는 YAML 텍스트 파일로, 스택에서 생성하려는 리소스에 대한 구성 정보가 들어 있습니다. 또한 이 템플릿은 RDS 인스턴스와 함께 VPC와 Bastion Host를 생성합니다.

템플릿 파일을 다운로드하려면 다음 링크인 Oracle CloudFormation 템플릿을 엽니다.

Github 페이지에서 원시 파일 다운로드 버튼을 클릭하여 템플릿 YAML 파일을 저장합니다.

CloudFormation을 사용하여 리소스 구성

> **Note**
>
> 이 프로세스를 시작하기 전에 AWS 계정에 EC2 인스턴스용 키 페어가 있는지 확인합니다. 자세한 내용은 Amazon EC2 키 페어 및 Linux 인스턴스를 참조하세요.

AWS CloudFormation 템플릿을 사용할 때는 리소스가 제대로 생성되도록 올바른 파라미터를 선택해야 합니다. 다음 단계를 따릅니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/cloudformation에서 AWS CloudFormation 콘솔을 엽니다.

2. 스택 생성을 선택합니다.
3. 템플릿 지정 섹션에서 컴퓨터에서 템플릿 파일 업로드를 선택하고 다음을 선택합니다.
4. 스택 세부 정보 지정 페이지에서 다음 파라미터를 설정합니다.
   a. 스택 이름을 OracleTestStack으로 설정합니다.
   b. 파라미터에서 가용 영역 3개를 선택하여 가용 영역을 설정합니다.
   c. Linux Bastion Host 구성에서 키 이름에 대해 EC2 인스턴스에 로그인할 키 페어를 선택합니다.
   d. Linux Bastion Host 구성 설정에서 허용된 IP 범위를 IP 주소로 설정합니다. Secure Shell(SSH)을 사용하여 VPC의 EC2 인스턴스에 연결하려면 https://checkip.amazonaws.com의 서비스를 사용하여 퍼블릭 IP 주소를 지정합니다. IP 주소의 예는 192.0.2.1/32입니다.

     > **Warning**
     >
     > SSH 액세스에 `0.0.0.0/0`을 사용하는 경우 모든 IP 주소가 SSH를 사용하여 퍼블릭 EC2 인스턴스에 액세스할 수 있도록 활성화합니다. 이 방법은 테스트 환경에서 잠시 사용하는 것은 괜찮지만 프로덕션 환경에서는 안전하지 않습니다. 프로덕션에서는 SSH를 사용하여 EC2 인스턴스에 액세스할 수 있는 특정 IP 주소 또는 주소 범위만 인증합니다.

   e. 데이터베이스 일반 구성에서 데이터베이스 인스턴스 클래스를 db.t3.micro로 설정합니다.
   f. 데이터베이스 이름을 **`database-test1`**으로 설정합니다.
   g. 데이터베이스 마스터 사용자 이름에 마스터 사용자 이름을 입력합니다.
   h. 이 자습서에서는 Secrets Manager를 사용하여 DB 마스터 사용자 암호 관리를 `false`로 설정합니다.
   i. 데이터베이스 암호의 경우 원하는 암호를 설정합니다. 자습서의 향후 단계에 사용할 수 있도록 암호를 기억해 둡니다.
   j. 데이터베이스 스토리지 구성에서 데이터베이스 스토리지 유형을 gp2로 설정합니다.
   k. 데이터베이스 모니터링 구성에서 RDS 성능 개선 도우미 활성화를 false로 설정합니다.
   l. 다른 모든 설정은 기본값으로 둡니다. 계속하려면 다음을 클릭합니다.
5. 스택 옵션 구성 페이지에서는 모든 기본 옵션을 그대로 둡니다. 계속하려면 다음을 클릭합니다.
6. 스택 검토 페이지에서 데이터베이스 및 Linux Bastion Host 옵션을 확인한 후 제출을 선택합니다.

스택 생성 프로세스가 완료되면 BastionStack 및 RDSNS라는 이름의 스택을 보고 데이터베이스에 연결하는 데 필요한 정보를 기록해 둡니다. 자세한 내용은 AWS Management Console에서 AWS CloudFormation 스택 데이터 및 리소스 보기를 참조하세요.

3단계: SQL 클라이언트를 Oracle DB 인스턴스에 연결

표준 SQL 클라이언트 애플리케이션을 사용하여 DB 인스턴스에 연결할 수 있습니다. 이 예시에서는 Oracle 명령줄 클라이언트를 사용하여 Oracle DB 인스턴스에 연결합니다.

Oracle DB 인스턴스에 연결하려면

1. DB 인스턴스의 엔드포인트(DNS 이름)와 포트 번호를 찾습니다.

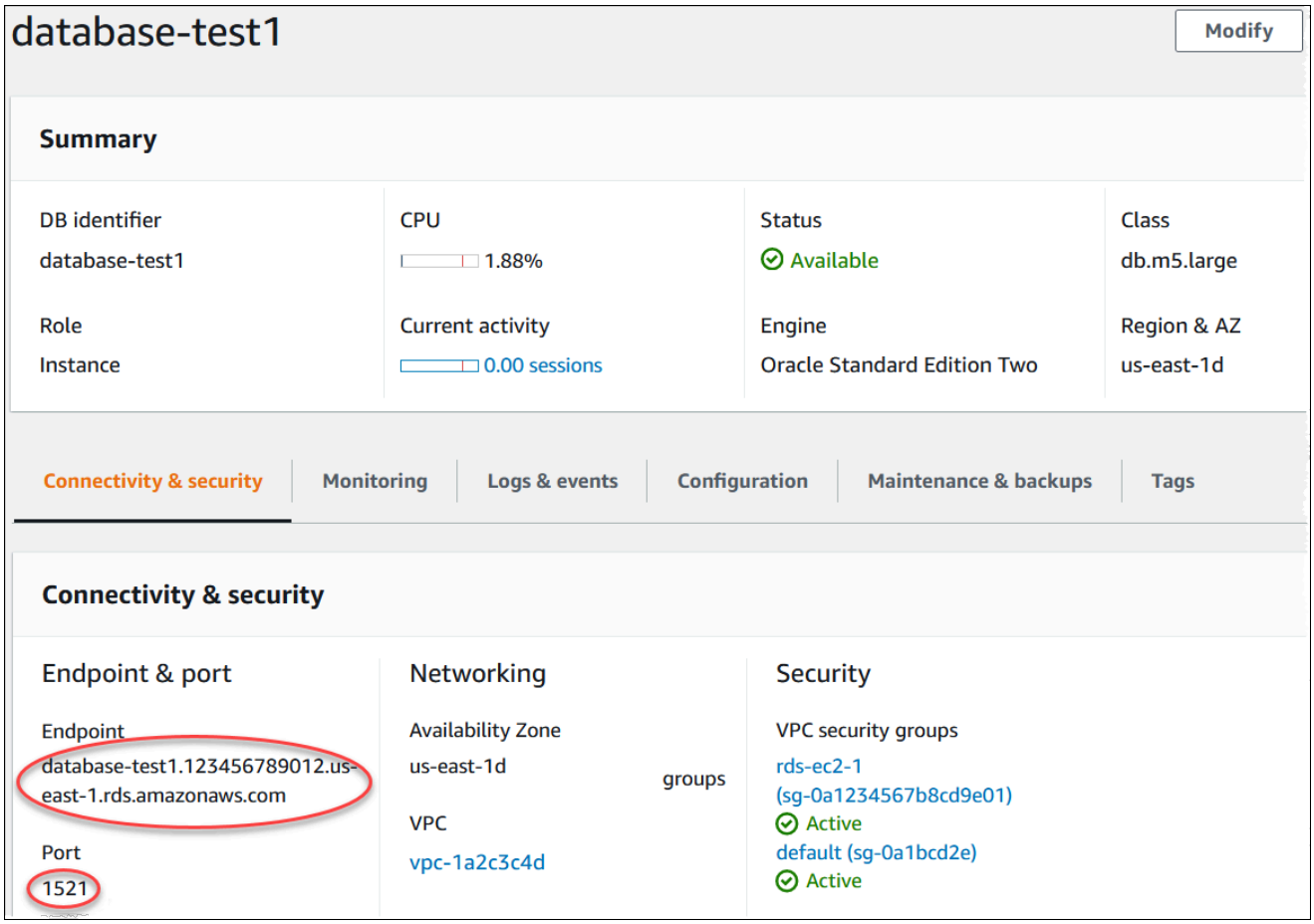
   a. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.

   b. Amazon RDS 콘솔의 오른쪽 상단에서 DB 인스턴스의 AWS 리전을 선택합니다.

   c. 탐색 창에서 Databases(데이터베이스)를 선택합니다.

   d. 세부 정보를 표시하고자 하는 Oracle DB 인스턴스 이름을 선택합니다.

   e. Connectivity & security(연결 및 보안) 탭에서 엔드포인트를 복사합니다. 또한 포트 번호를 적어 둡니다. DB 인스턴스에 연결하려면 엔드포인트와 포트 번호가 모두 필요합니다.



2. Amazon EC2 사용 설명서에 있는 Linux 인스턴스에 연결의 단계를 따라 앞에서 만든 EC2 인스턴스에 연결합니다.

   SSH를 사용하여 EC2 인스턴스에 연결하는 것이 좋습니다. Windows, Linux 또는 Mac에 SSH 클라이언트 유틸리티가 설치된 경우 다음 명령 형식을 사용하여 인스턴스에 연결할 수 있습니다.

   ```
   ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
   ```

   예를 들어 `ec2-database-connect-key-pair.pem`이 Linux의 `/dir1`에 저장되어 있고, EC2 인스턴스의 퍼블릭 IPv4 DNS가 `ec2-12-345-678-90.compute-1.amazonaws.com`이라고 가정해 보겠습니다. SSH 명령은 다음과 같이 표시됩니다.

   ```
   ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
   user@ec2-12-345-678-90.compute-1.amazonaws.com
   ```

3. EC2 인스턴스에서 소프트웨어를 업데이트하여 최신 버그 수정 및 보안 업데이트를 받습니다. 이렇게 하려면 다음 명령을 사용합니다.

   > **Note**
   >
   > `-y` 옵션을 사용하면 확인 여부를 묻지 않고 업데이트를 설치합니다. 설치 전에 업데이트 정보를 확인하려면 이 옵션을 생략합니다.

   ```
   sudo dnf update -y
   ```

4. 웹 브라우저에서 https://www.oracle.com/database/technologies/instant-client/linux-x86-64-downloads.html로 이동합니다.

5. 웹 페이지에 표시되는 최신 데이터베이스 버전을 보려면 Instant Client Basic Package 및 SQL*Plus Package의 .rpm 링크(.zip 링크 아님)를 복사하세요. 예를 들어, 다음 링크는 Oracle 데이터베이스 버전 21.9의 링크입니다.

   - https://download.oracle.com/otn_software/linux/instantclient/219000/oracle-instantclient-basic-21.9.0.0.0-1.el8.x86_64.rpm
   - https://download.oracle.com/otn_software/linux/instantclient/219000/oracle-instantclient-sqlplus-21.9.0.0.0-1.el8.x86_64.rpm

6. SSH 세션에서 `wget` 명령을 실행하여 이전 단계에서 가져온 링크를 통해 .rpm 파일을 다운로드합니다. 다음 예제에서는 Oracle 데이터베이스 버전 21.9의 .rpm 파일을 다운로드합니다.

```
wget https://download.oracle.com/otn_software/linux/instantclient/219000/oracle-
instantclient-basic-21.9.0.0.0-1.el8.x86_64.rpm
wget https://download.oracle.com/otn_software/linux/instantclient/219000/oracle-
instantclient-sqlplus-21.9.0.0.0-1.el8.x86_64.rpm
```

7. 다음과 같이 dnf 명령을 실행하여 패키지를 설치합니다.

```
sudo dnf install oracle-instantclient-*.rpm
```

8. SQL*Plus를 시작하고 Oracle DB 인스턴스에 연결합니다. 예를 들어, 다음 명령을 입력합니다.

   *oracle-db-instance-endpoint*는 DB 인스턴스 엔드포인트(DNS 이름)로 대체하고, *admin*는 사용된 마스터 사용자 이름으로 대체합니다. Oracle에서 간편 생성을 사용할 경우 데이터베이스 이름은 DATABASE입니다. 암호를 묻는 메시지가 표시되면 사용한 마스터 암호를 제공합니다.

```
sqlplus admin@oracle-db-instance-endpoint:1521/DATABASE
```

   사용자에 대한 암호를 입력하면 다음과 유사한 출력이 나타납니다.

```
SQL*Plus: Release 21.0.0.0.0 - Production on Wed Mar 1 16:41:28 2023
Version 21.9.0.0.0

Copyright (c) 1982, 2022, Oracle.  All rights reserved.

Enter password:
Last Successful login time: Wed Mar 01 2023 16:30:52 +00:00

Connected to:
Oracle Database 19c Standard Edition 2 Release 19.0.0.0.0 - Production
Version 19.18.0.0.0

SQL>
```

   RDS for Oracle DB 인스턴스 연결에 대한 자세한 내용은 Oracle DB 인스턴스에 연결 섹션을 참조하세요. DB 인스턴스에 연결할 수 없는 경우 Amazon RDS DB 인스턴스에 연결할 수 없음 단원을 참조하십시오.

보안을 위해서는 암호화된 연결을 사용하는 것이 가장 좋습니다. 클라이언트와 서버가 동일한 VPC에 있고 네트워크를 신뢰할 수 있는 경우에만 암호화되지 않은 Oracle 연결을 사용합니다. 암호화된 연결 사용에 대한 자세한 내용은 Oracle DB 인스턴스 연결 보안 섹션을 참조하세요.

9. SQL 명령을 실행합니다.

   예를 들어, 다음 SQL 명령은 현재 날짜를 보여줍니다.

```
SELECT SYSDATE FROM DUAL;
```

4단계: EC2 인스턴스 및 DB 인스턴스 삭제

생성한 샘플 EC2 인스턴스 및 DB 인스턴스에 연결하고 탐색한 후에는 요금이 더 이상 부과되지 않도록 삭제합니다.

AWS CloudFormation을 사용하여 리소스를 생성했다면 이 단계를 건너뛰고 다음 단계로 이동합니다.

EC2 인스턴스를 삭제하는 방법

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창에서 인스턴스를 선택합니다.
3. EC2 인스턴스를 선택하고 인스턴스 상태, 인스턴스 종료를 차례로 선택합니다.
4. 확인 메시지가 나타나면 종료를 선택합니다.

EC2 인스턴스 삭제에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 종료를 참조하세요.

최종 DB 스냅샷이 없는 DB 인스턴스를 삭제하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 삭제할 DB 인스턴스를 선택합니다.
4. [Actions]에 대해 [Delete]를 선택합니다.
5. 최종 스냅샷을 생성하시겠습니까? 및 자동 백업 보존을 선택 해제합니다.
6. 확인을 완료하고 삭제를 선택합니다.

(선택 사항) CloudFormation으로 생성한 EC2 인스턴스 및 DB 인스턴스 삭제

AWS CloudFormation을 사용하여 리소스를 생성한 경우 샘플 EC2 인스턴스 및 DB 인스턴스에 연결하고 탐색한 후 CloudFormation 스택을 삭제하여 더 이상 비용이 청구되지 않도록 합니다.

CloudFormation 리소스를 삭제하려면

1. AWS CloudFormation 콘솔을 엽니다.
2. CloudFormation 콘솔의 스택 페이지에서 루트 스택(VPCStack, BastionStack 또는 RDSNS라는 이름이 없는 스택)을 선택합니다.
3. 삭제를 선택합니다.
4. 확인 메시지가 나타나면 스택 삭제를 선택합니다.

CloudFormation에서 스택을 삭제하는 방법에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS CloudFormation 콘솔에서 스택 삭제를 참조하세요.

(선택 사항) DB 인스턴스를 Lambda 함수에 연결

RDS for Oracle DB 인스턴스를 Lambda 서버리스 컴퓨팅 리소스에 연결할 수도 있습니다. Lambda 함수를 사용하면 인프라를 프로비저닝하거나 관리하지 않고도 코드를 실행할 수 있습니다. 또한 하루 12개에서 초당 수백 개에 이르는 모든 규모의 코드 실행 요청에 자동으로 응답할 수 있습니다. 자세한 내용은 Lambda 함수와 DB 인스턴스 자동 연결 섹션을 참조하세요.

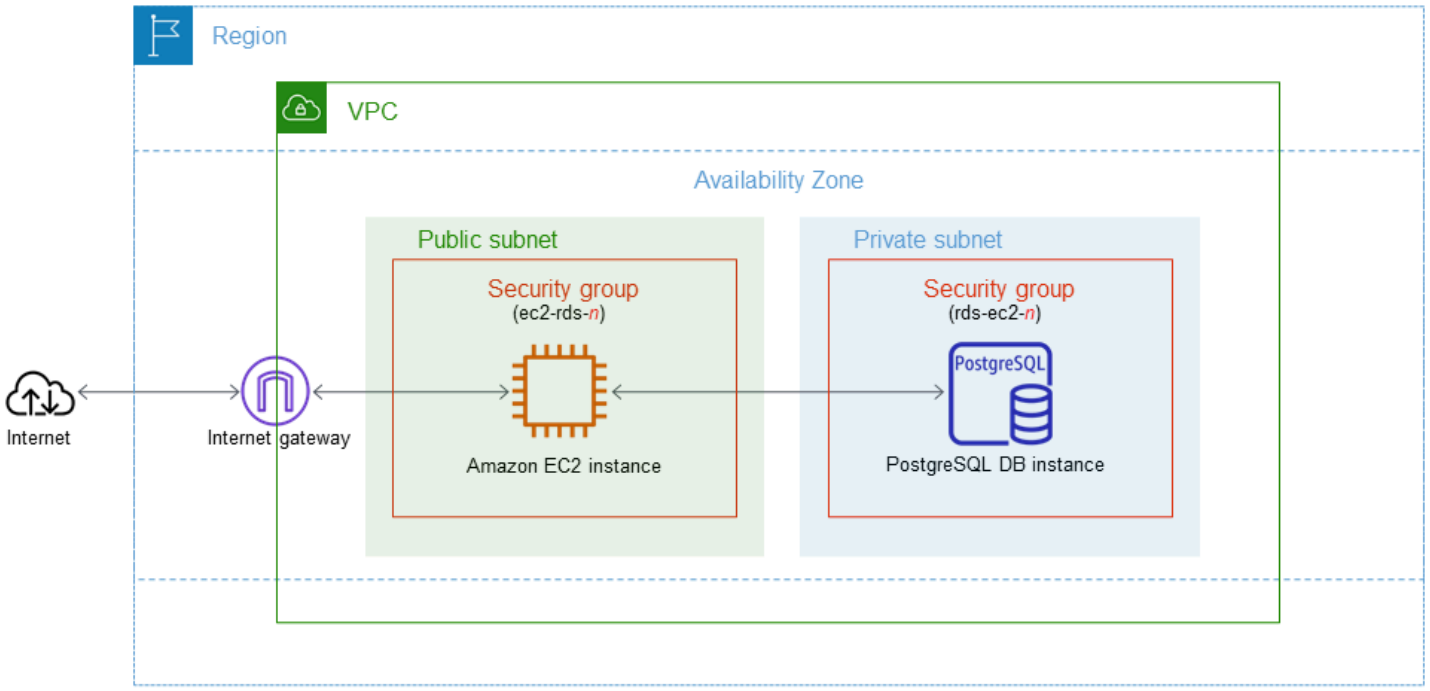
PostgreSQL DB 인스턴스 생성 및 해당 인스턴스에 연결

이 자습서에서는 EC2 인스턴스와 RDS for PostgreSQL DB 인스턴스를 생성합니다. 자습서에서는 표준 PostgreSQL 클라이언트를 사용하여 EC2 인스턴스에서 DB 인스턴스에 액세스하는 방법을 보여줍니다. 이 자습서에서는 모범 사례를 따라 Virtual Private Cloud(VPC)에서 프라이빗 DB 인스턴스를 생성합니다. 대부분의 경우 EC2 인스턴스와 같이 동일한 VPC에 있는 다른 리소스는 DB 인스턴스에 액세스할 수 있지만 VPC 외부의 리소스는 DB 인스턴스에 액세스할 수 없습니다.

자습서를 완료하면 VPC의 각 가용 영역에 퍼블릭 서브넷과 프라이빗 서브넷이 있을 것입니다. 한 가용 영역에서 EC2 인스턴스는 퍼블릭 서브넷에 있고 DB 인스턴스는 프라이빗 서브넷에 있습니다.

> **Important**
>
> AWS 계정 생성은 무료입니다. 그러나 이 자습서를 완료하면 사용하는 AWS 리소스에 대한 비용이 발생할 수 있습니다. 자습서가 더 이상 필요하지 않은 경우 자습서를 완료한 후에 이러한 리소스를 삭제할 수 있습니다.

다음 다이어그램은 이 자습서를 완료했을 때 구성을 보여 줍니다.



이 자습서에서는 다음 방법 중 하나를 사용하여 리소스를 생성할 수 있습니다.

1. AWS Management Console 사용 - EC2 인스턴스 생성 및 PostgreSQL DB 인스턴스 생성

2. 데이터베이스 인스턴스 및 EC2 인스턴스를 생성하는 데 AWS CloudFormation 사용 - (선택 사항) AWS CloudFormation를 사용하여 VPC, EC2 인스턴스 및 PostgreSQL 인스턴스 생성

첫 번째 방법은 간편 생성을 사용하여 AWS Management Console을 통해 프라이빗 PostgreSQL DB 인스턴스를 생성합니다. 여기에서는 DB 엔진 유형, DB 인스턴스 크기 및 DB 인스턴스 식별자만 지정합니다. [간편 생성(Easy create)]은 다른 구성 옵션에서도 기본 설정을 사용합니다.

표준 생성을 대신 사용하는 경우에는 DB 인스턴스를 생성할 때 더 많은 구성 옵션을 지정할 수 있습니다. 이러한 옵션에는 가용성, 보안, 백업 및 유지 관리에 대한 설정이 포함됩니다. 퍼블릭 DB 인스턴스를 만들려면 표준 생성을 사용해야 합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.

주제

- 사전 조건
- EC2 인스턴스 생성
- PostgreSQL DB 인스턴스 생성
- (선택 사항) AWS CloudFormation를 사용하여 VPC, EC2 인스턴스 및 PostgreSQL 인스턴스 생성
- PostgreSQL DB 인스턴스에 연결
- EC2 인스턴스 및 DB 인스턴스 삭제
- (선택 사항) CloudFormation으로 생성한 EC2 인스턴스 및 DB 인스턴스 삭제
- (선택 사항) DB 인스턴스를 Lambda 함수에 연결

사전 조건

시작하기 전에 다음 섹션에서 다음 단계를 완료하세요.

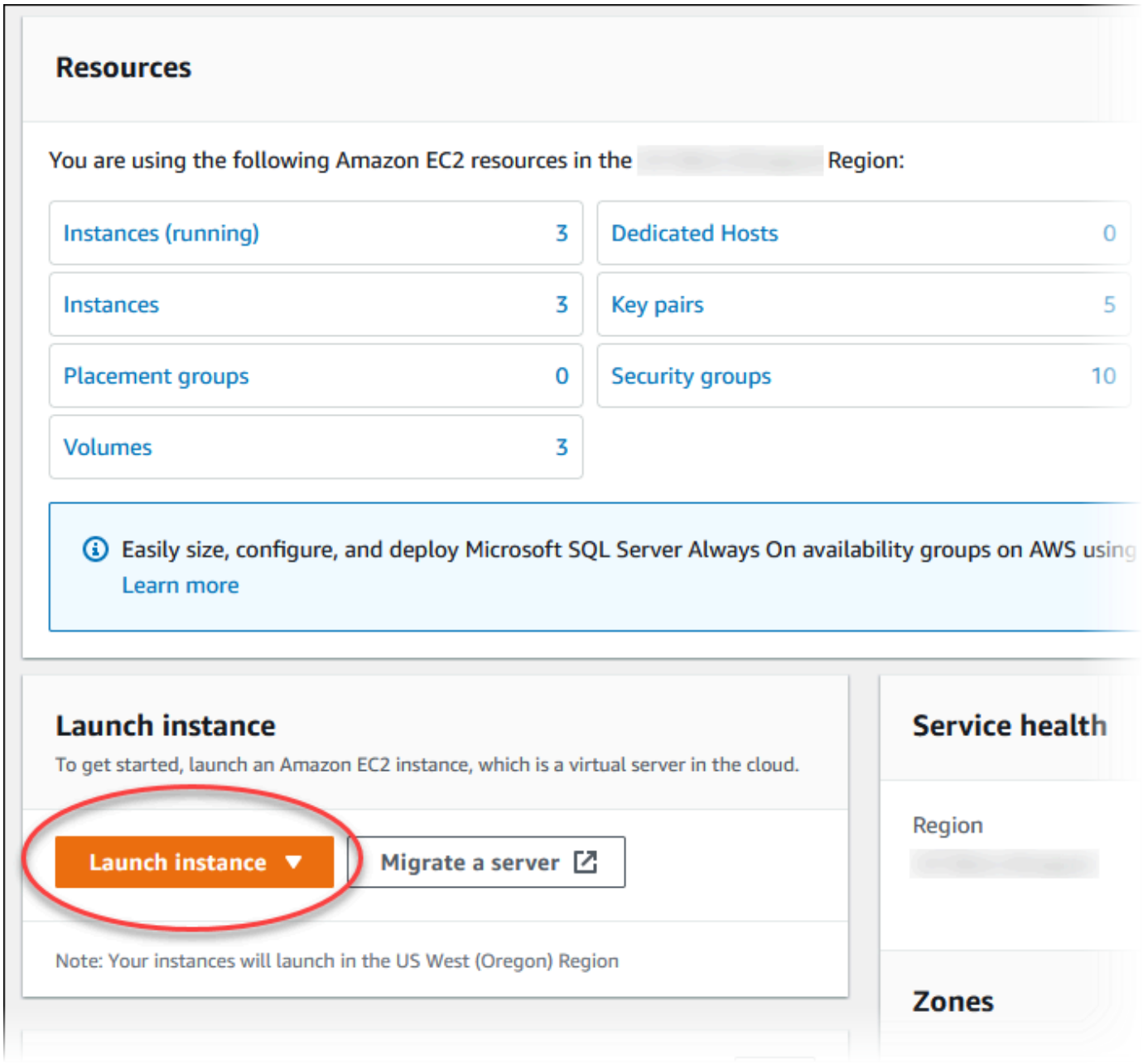
- AWS 계정에 등록
- 관리자 액세스 권한이 있는 사용자 생성

EC2 인스턴스 생성

데이터베이스에 연결하는 데 사용할 Amazon EC2 인스턴스를 생성합니다.

EC2 인스턴스를 생성하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. AWS Management Console의 오른쪽 상단에서 EC2 인스턴스를 생성하려는 AWS 리전을 선택합니다.
3. 다음 이미지에 나와 있는 것처럼 EC2 대시보드를 선택한 다음, 인스턴스 시작을 선택합니다.

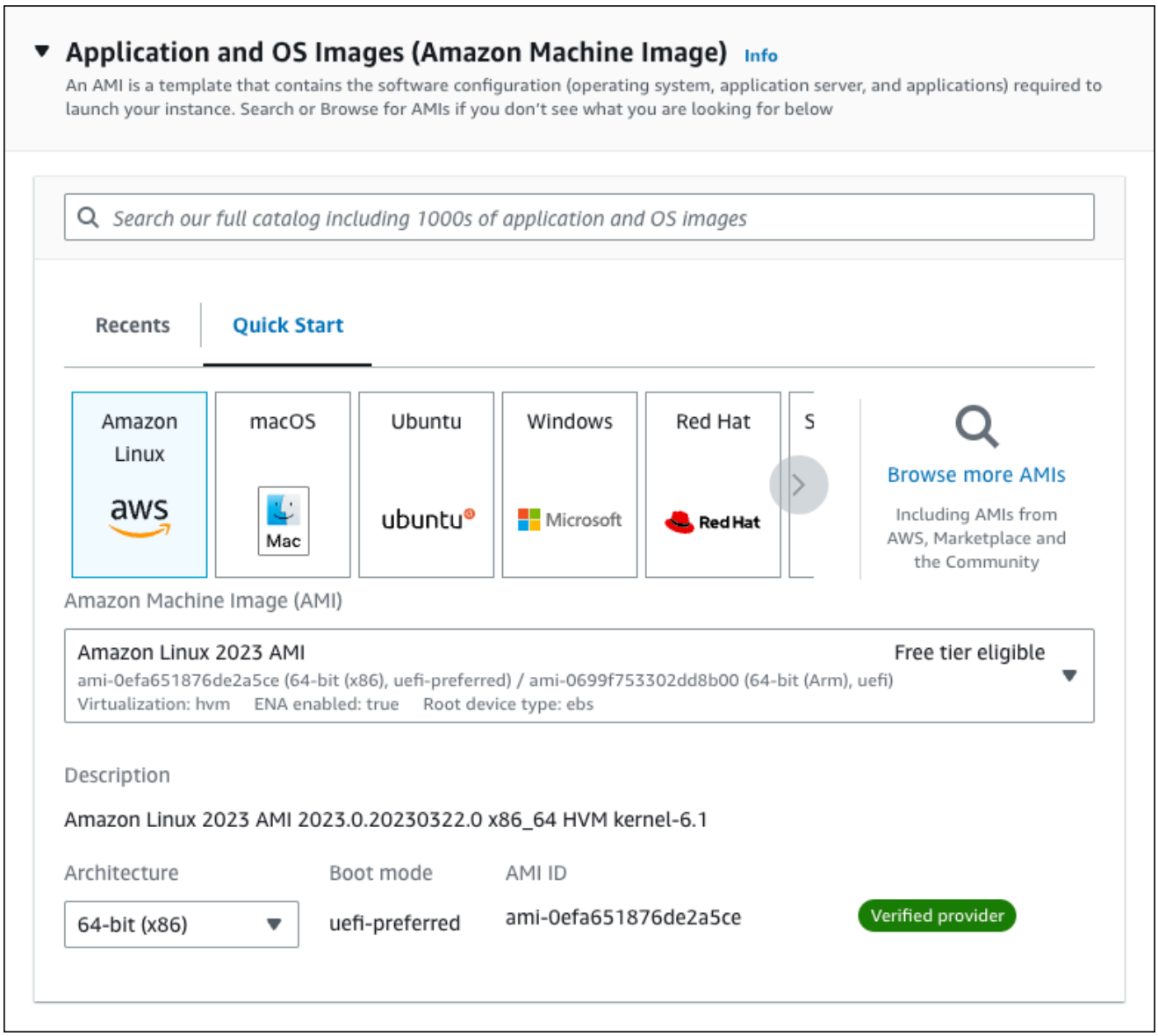


인스턴스 시작 페이지가 열립니다.

4. 인스턴스 시작 페이지에서 다음 설정을 선택합니다.

   a. Name and tags(이름 및 태그) 아래의 Name(이름)에 **ec2-database-connect**을 입력하세요.

   b. Application and OS Images (Amazon Machine Image)(애플리케이션 및 OS 이미지(Amazon Machine Image))에서 Amazon Linux를 선택한 다음 Amazon Linux 2023 AMI를 선택합니다. 다른 선택 항목에 대해서는 기본값을 그대로 유지합니다.



   c. 인스턴스 유형에서 t2.micro를 선택합니다.

   d. 키 페어(로그인)에서 기존 키 페어를 사용할 키 페어 이름을 선택합니다. Amazon EC2 인스턴스에 대한 새 키 페어를 생성하려면 새 키 페어 생성을 선택한 다음 키 페어 생성 창을 사용하여 생성합니다.

새 키 페어 생성에 대한 자세한 내용은 Amazon EC2 Linux 인스턴스용 사용 설명서의 키 페어 생성을 참조하세요.

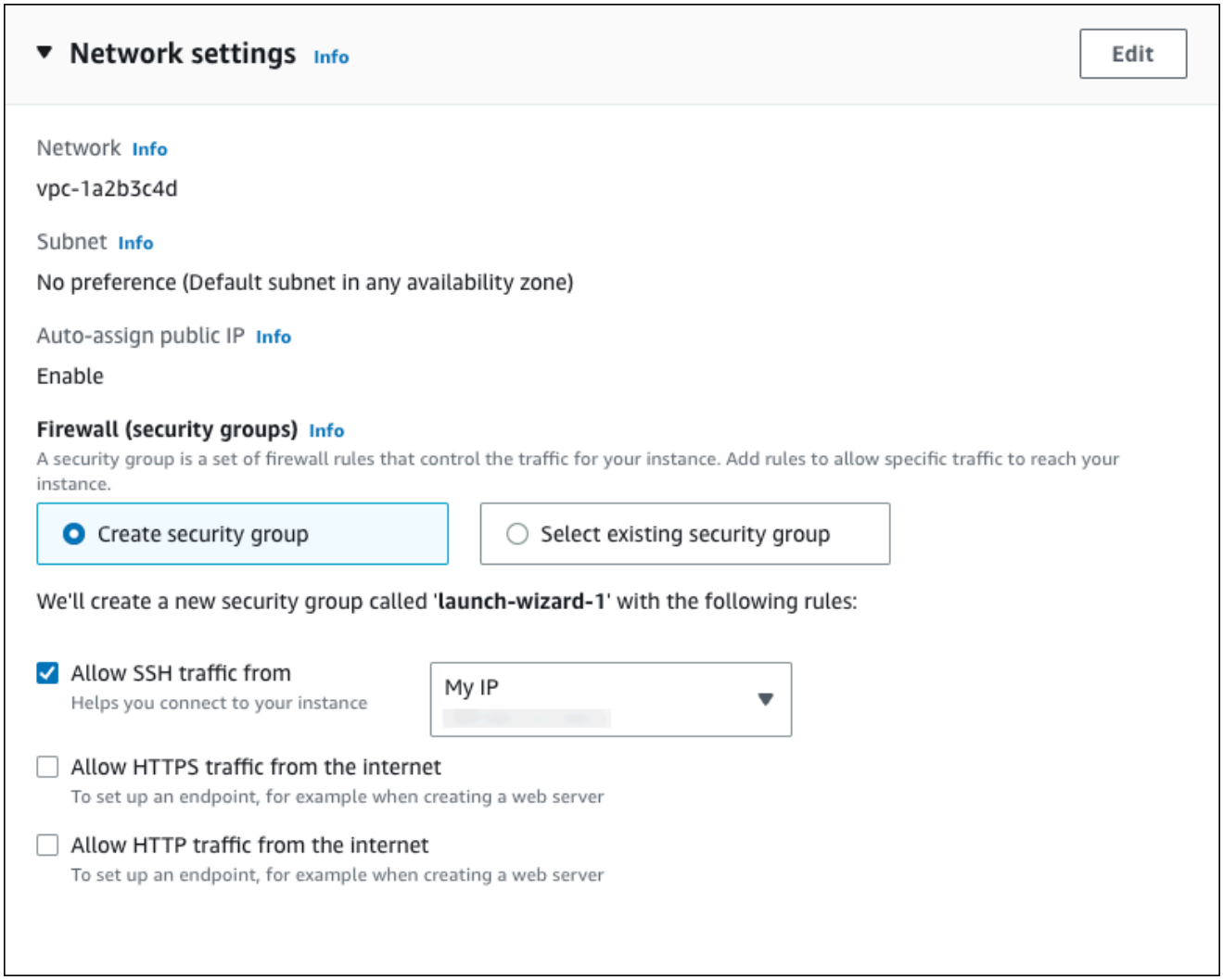
e. 네트워크 설정의 SSH 트래픽 허용에서 EC2 인스턴스에 대한 SSH 연결 소스를 선택합니다.

표시된 IP 주소가 SSH 연결에 대해 올바른 경우 내 IP를 선택할 수 있습니다. 그렇지 않으면 SSH(Secure Shell)를 사용하여 VPC의 EC2 인스턴스에 연결하는 데 사용할 IP 주소를 결정할 수 있습니다. 퍼블릭 IP 주소를 확인하려면 다른 브라우저 창 또는 탭에서 https://checkip.amazonaws.com의 서비스를 사용합니다. IP 주소의 예는 192.0.2.1/32입니다.

대부분의 경우 고정 IP 주소가 없는 방화벽 뒤나 인터넷 서비스 제공업체(ISP)를 통해 연결하는 경우가 많습니다. 그렇다면 클라이언트 컴퓨터에서 사용하는 IP 주소 범위를 결정합니다.

> **Warning**
>
> SSH 액세스에 `0.0.0.0/0`을 사용하는 경우 모든 IP 주소가 SSH를 사용하여 퍼블릭 EC2 인스턴스에 액세스할 수 있도록 활성화합니다. 이 방법은 테스트 환경에서 잠시 사용하는 것은 괜찮지만 프로덕션 환경에서는 안전하지 않습니다. 프로덕션에서는 SSH를 사용하여 EC2 인스턴스에 액세스할 수 있는 특정 IP 주소 또는 주소 범위만 인증합니다.

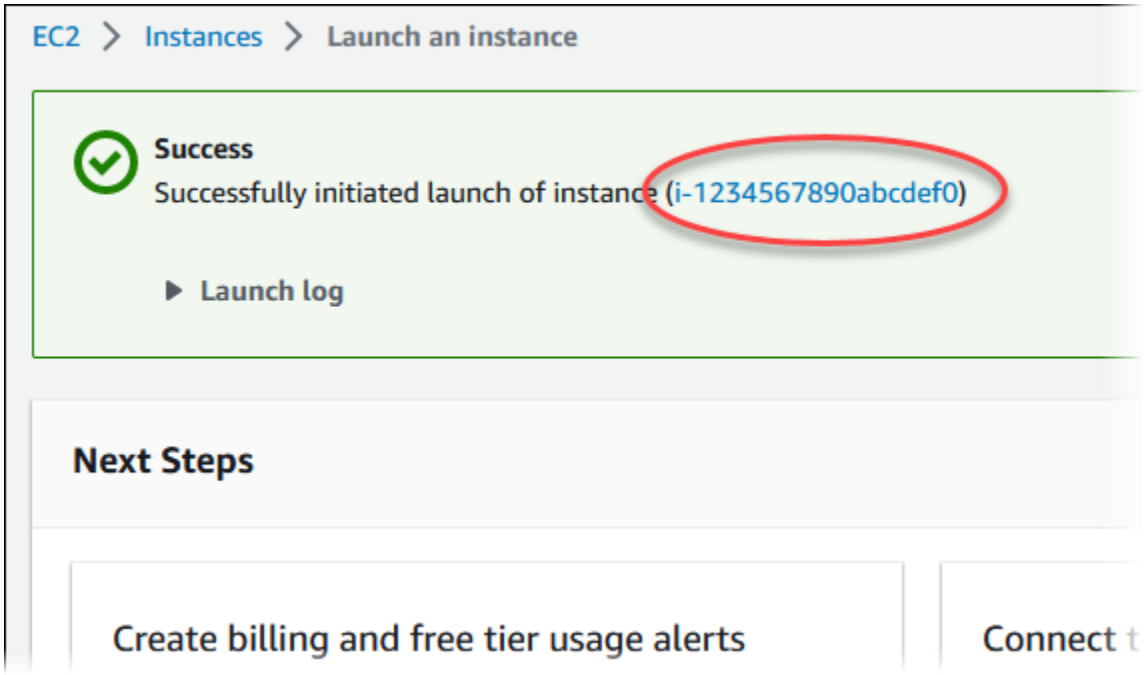
다음 이미지는 네트워크 설정 섹션의 예를 보여 줍니다.



f. 나머지 섹션에서 기본값은 그대로 둡니다.

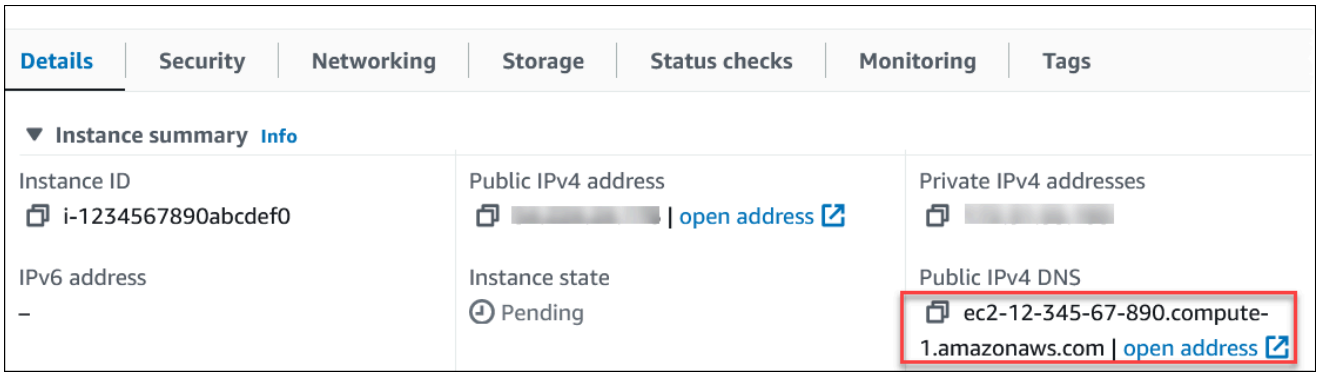
g. 요약 패널에서 EC2 인스턴스 구성 요약을 검토하고 준비가 되면 인스턴스 시작을 선택합니다.

5. 시작 상태 페이지에서, 새 EC2 인스턴스의 식별자(예: `i-1234567890abcdef0`)를 기록해 둡니다.

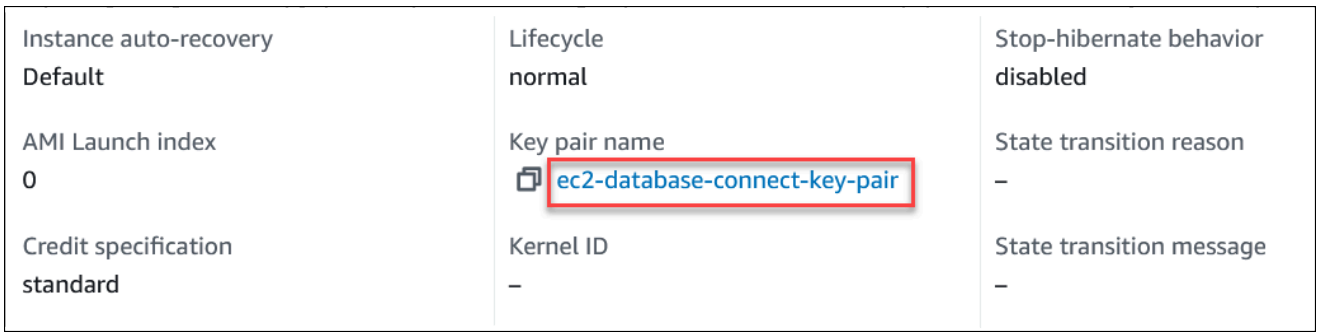


6. EC2 인스턴스 식별자를 선택하여 EC2 인스턴스 목록을 열고 EC2 인스턴스를 선택합니다.
7. 세부 정보 탭에서 SSH를 사용하여 연결할 때 필요한 다음 값을 기록해 둡니다.

   a. 인스턴스 요약에서 퍼블릭 IPv4 DNS의 값을 기록해 둡니다.



   b. 인스턴스 세부 정보에서 키 페어 이름의 값을 기록해 둡니다.



8. 계속하기 전에 EC2 인스턴스의 인스턴스 상태가 실행 중이 될 때까지 기다립니다.

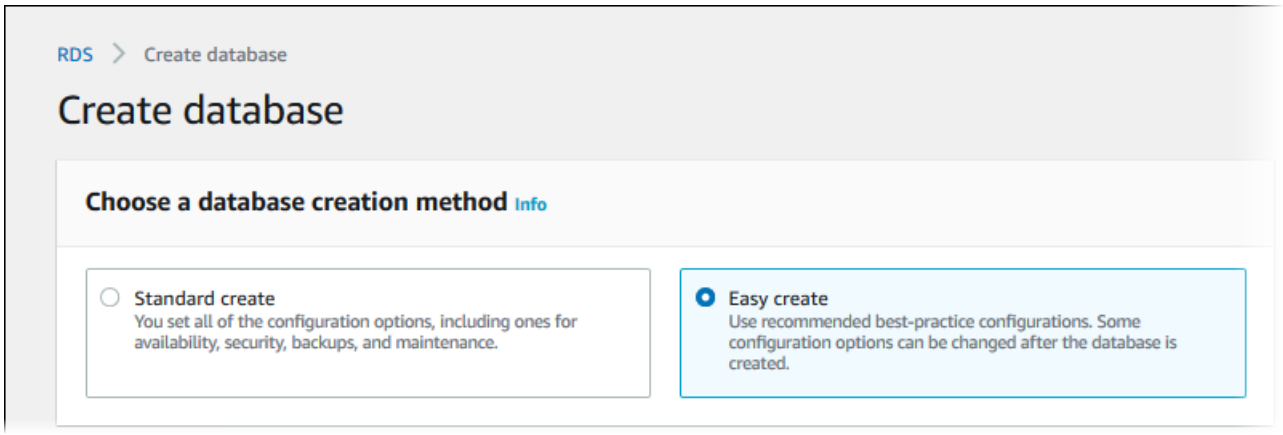
PostgreSQL DB 인스턴스 생성

Amazon RDS의 기본 빌딩 블록은 DB 인스턴스입니다. PostgreSQL 데이터베이스를 생성하는 환경입니다.

이 예시에서는 간편 생성을 사용하여 db.t3.micro DB 인스턴스 클래스에서 PostgreSQL Server 데이터베이스 엔진을 실행하는 DB 인스턴스를 생성합니다.

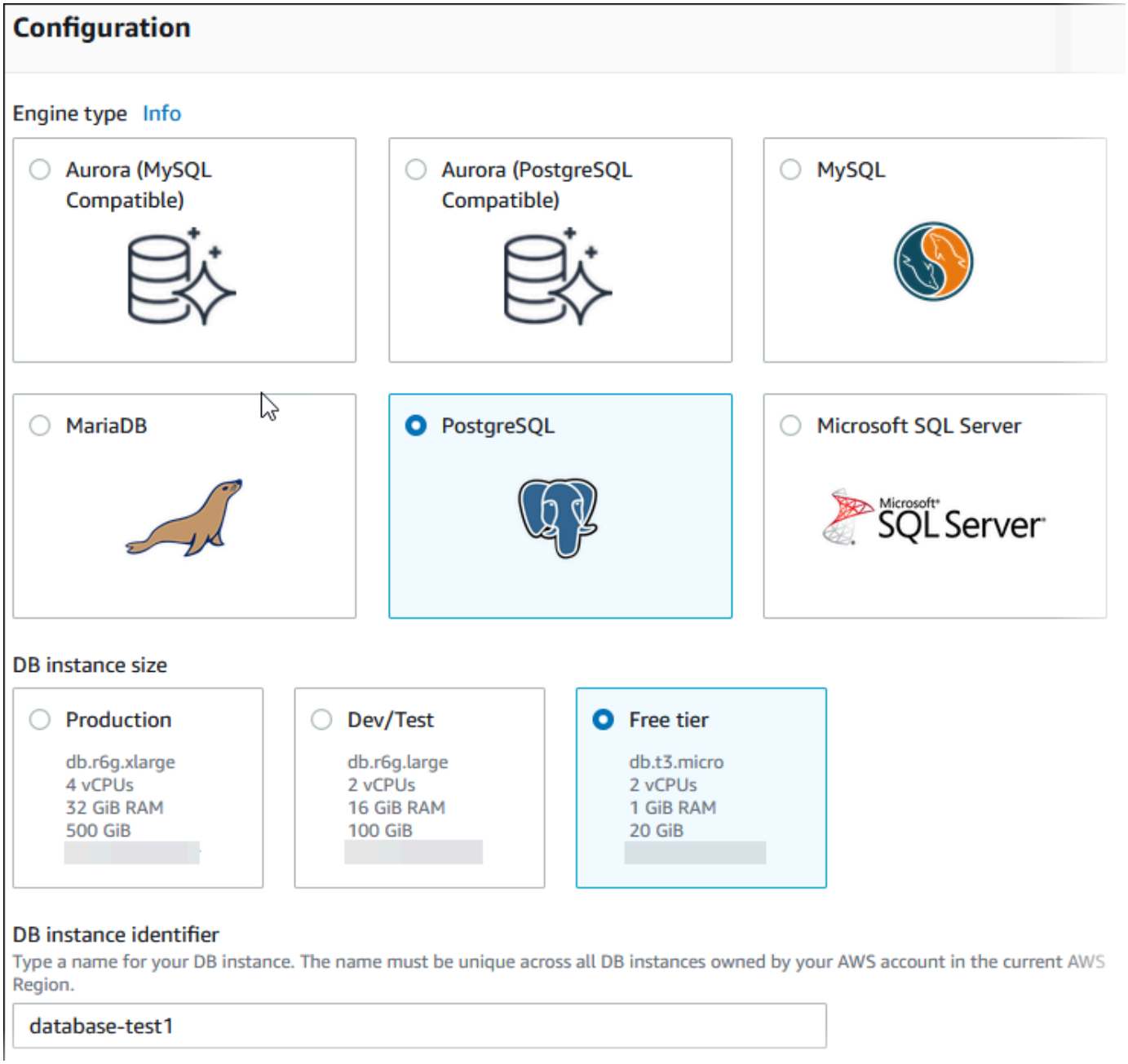
Easy create(간편 생성)을 사용하여 PostgreSQL DB 인스턴스를 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 콘솔의 오른쪽 상단에서 DB 인스턴스를 생성하려는 AWS 리전을 선택합니다.
3. 탐색 창에서 데이터베이스를 선택합니다.
4. [데이터베이스 생성(Create database)]을 선택하고 [간편 생성(Easy Create)]이 선택되어 있는지 확인합니다.



5. 구성에서 PostgreSQL을 선택합니다.
6. DB instance size(DB 인스턴스 크기)에서 프리 티어를 선택합니다. 프리 티어는 무료 요금제 계정에 표시됩니다. 샌드박스는 유료 요금제 계정에 표시됩니다.
7. DB 인스턴스 식별자에 **database-test1**을 입력합니다.
8. 마스터 사용자 이름에 마스터 사용자의 이름을 입력하거나 기본 이름(**postgres**)을 그대로 사용합니다.

   데이터베이스 생성 페이지는 다음 이미지와 비슷해야 합니다. 무료 요금제 계정에는 프리 티어가 표시됩니다. 유료 요금제 계정에는 샌드박스가 표시됩니다.

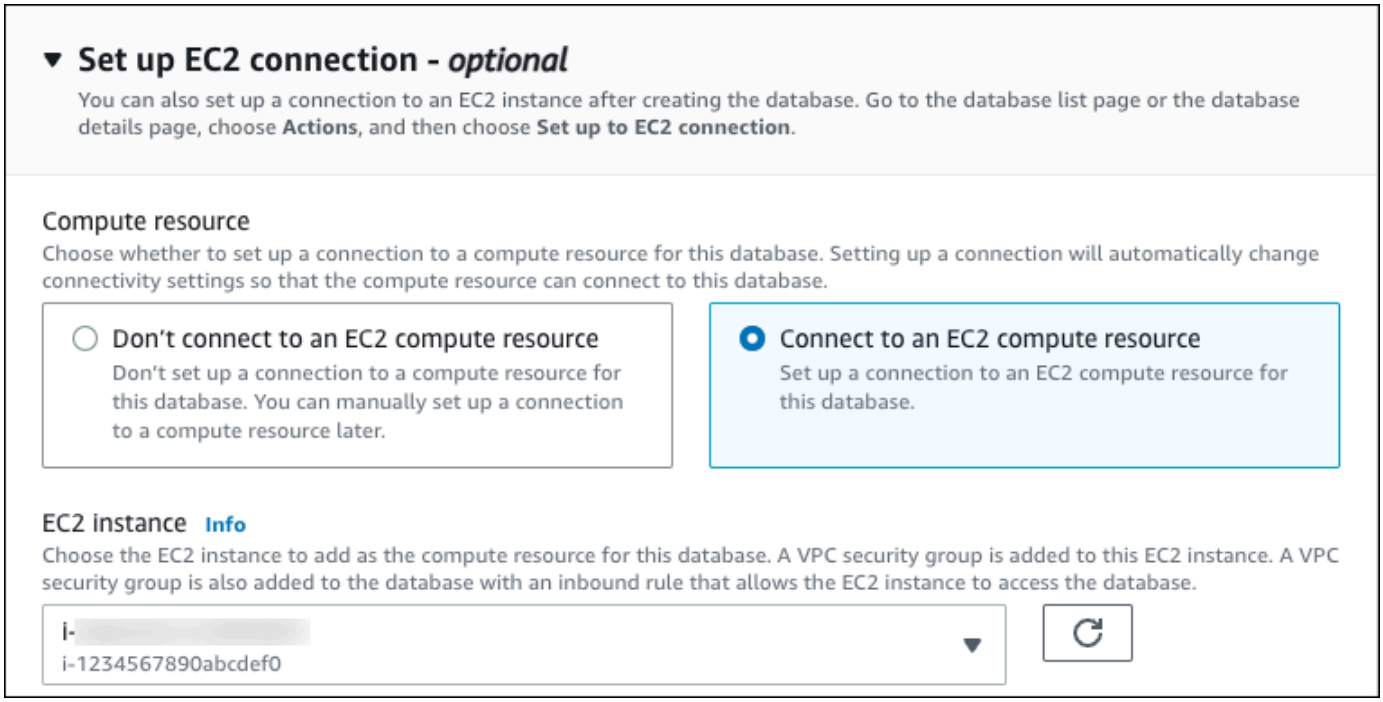


9. DB 인스턴스에 자동 생성된 마스터 암호를 사용하려면 암호 자동 생성을 선택합니다.

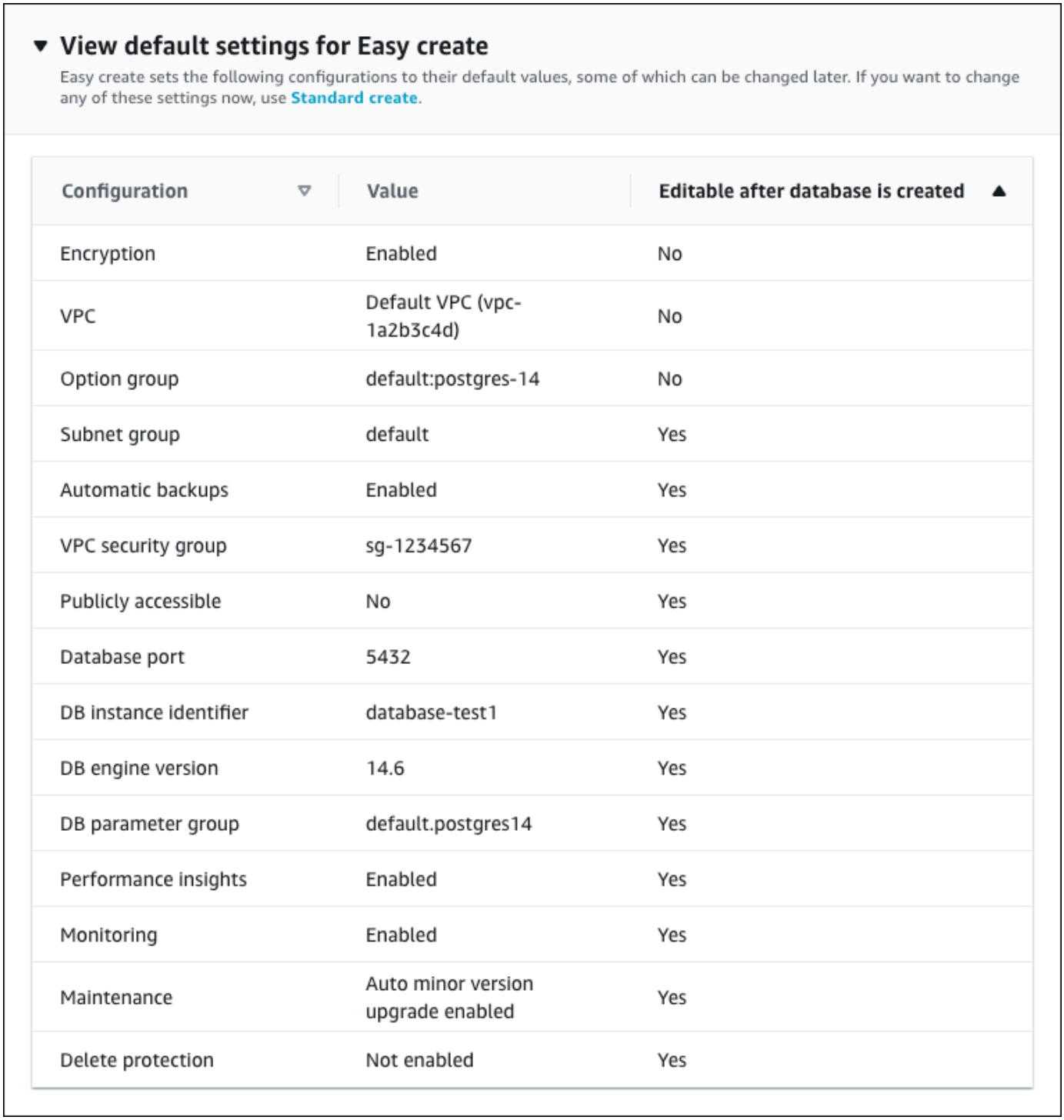
   마스터 암호를 입력하려면 암호 자동 생성 선택을 해제한 다음, 마스터 암호와 암호 확인에 동일한 암호를 입력합니다.

10. 이전에 생성한 EC2 인스턴스와의 연결을 설정하려면 EC2 연결 설정 - 선택 사항을 엽니다.

    Connect to an EC2 compute resource(EC2 컴퓨팅 리소스에 연결)를 선택합니다. 이전에 생성한 EC2 인스턴스를 선택합니다.



11. 간편 생성 기본 설정 보기를 엽니다.



▼ View default settings for Easy create

Easy create sets the following configurations to their default values, some of which can be changed later. If you want to change any of these settings now, use Standard create.

Configuration	Value	Editable after database is created
Encryption	Enabled	No
VPC	Default VPC (vpc-1a2b3c4d)	No
Option group	default:postgres-14	No
Subnet group	default	Yes
Automatic backups	Enabled	Yes
VPC security group	sg-1234567	Yes
Publicly accessible	No	Yes
Database port	5432	Yes
DB instance identifier	database-test1	Yes
DB engine version	14.6	Yes
DB parameter group	default.postgres14	Yes
Performance insights	Enabled	Yes
Monitoring	Enabled	Yes
Maintenance	Auto minor version upgrade enabled	Yes
Delete protection	Not enabled	Yes

[간편 생성(Easy Create)]과 함께 사용되는 기본 설정을 검토할 수 있습니다. 데이터베이스 생성 후 편집 가능 열에는 데이터베이스 생성 후 어떤 옵션을 변경할 수 있는지 나와 있습니다.

- 설정의 해당 열에 아니요라고 되어 있지만 다른 설정을 원하는 경우 표준 생성을 사용하여 DB 인스턴스를 만들 수 있습니다.

- 설정의 해당 열에 예라고 되어 있으며 다른 설정을 원하는 경우 표준 생성을 사용하여 DB 인스턴스를 만들거나 DB 인스턴스를 생성한 후 수정하여 설정을 변경할 수 있습니다.

12. 데이터베이스 생성를 선택합니다.

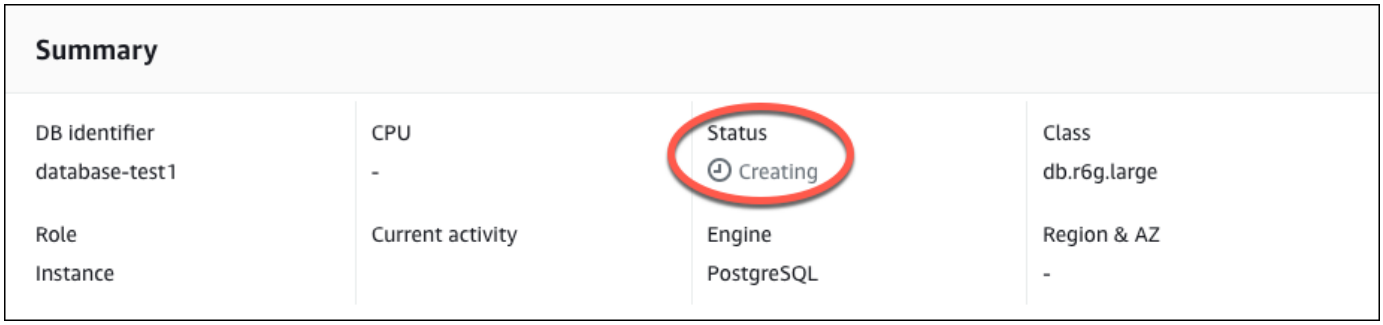
    DB 인스턴스의 마스터 사용자 이름 및 암호를 보려면 View credential details(자격 증명 세부 정보 보기)를 선택합니다.

    DB 인스턴스를 마스터 사용자로 연결하려면 화면에 나타난 사용자 이름과 암호를 사용합니다.

    > **Important**
    >
    > 마스터 사용자 암호를 다시 볼 수는 없습니다. 따라서 기록을 해두지 않으면 이를 변경해야 합니다.
    > DB 인스턴스가 사용 가능한 상태가 되고 난 후에 마스터 사용자 암호를 변경해야 하는 경우에는 다음과 같은 방법으로 DB 인스턴스를 수정할 수 있습니다. DB 인스턴스 변경에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

13. 데이터베이스 목록에서 새 PostgreSQL DB 인스턴스의 이름을 선택하면 세부 정보가 표시됩니다.

    DB 인스턴스를 사용할 준비가 될 때까지 DB 인스턴스의 상태는 생성 중입니다.



    상태가 Available(사용 가능)로 변경되면 DB 인스턴스에 연결할 수 있습니다. DB 인스턴스 클래스와 스토리지의 양에 따라 새 인스턴스를 사용할 수 있을 때까지 최대 20분이 걸릴 수 있습니다.

(선택 사항) AWS CloudFormation를 사용하여 VPC, EC2 인스턴스 및 PostgreSQL 인스턴스 생성

콘솔을 사용하여 VPC, EC2 인스턴스 및 PostgreSQL 인스턴스를 생성하는 대신 AWS CloudFormation을 통해 코드형 인프라로 처리하여 AWS 리소스를 프로비저닝할 수 있습니다. AWS 리소스를 더 작고 관리하기 쉬운 단위로 구성하는 데 도움이 되도록 AWS CloudFormation 중첩 스택 기

능을 사용할 수 있습니다. 자세한 내용은 AWS CloudFormation 콘솔에서 스택 생성 및 중첩된 스택 작업을 참조하세요.

> **Important**
>
> AWS CloudFormation은 무료이지만, CloudFormation에서 생성하는 리소스는 라이브입니다. 이러한 리소스를 종료하지 않으면 해당 리소스에 대한 표준 사용 요금이 발생합니다. 자세한 내용은 RDS for PostgreSQL 요금을 참조하세요.

AWS CloudFormation 콘솔을 사용하여 리소스를 생성하려면 다음 단계를 완료합니다.

- CloudFormation 템플릿 파일을 다운로드하십시오.
- CloudFormation을 사용하여 리소스 구성

CloudFormation 템플릿 파일을 다운로드하십시오.

CloudFormation 템플릿은 JSON 또는 YAML 텍스트 파일로, 스택에서 생성하려는 리소스에 대한 구성 정보가 들어 있습니다. 또한 이 템플릿은 RDS 인스턴스와 함께 VPC와 Bastion Host를 생성합니다.

템플릿 파일을 다운로드하려면 다음 링크인 PostgreSQL CloudFormation 템플릿을 엽니다.

Github 페이지에서 원시 파일 다운로드 버튼을 클릭하여 템플릿 YAML 파일을 저장합니다.

CloudFormation을 사용하여 리소스 구성

> **Note**
>
> 이 프로세스를 시작하기 전에 AWS 계정에 EC2 인스턴스용 키 페어가 있는지 확인합니다. 자세한 내용은 Amazon EC2 키 페어 및 Linux 인스턴스를 참조하세요.

AWS CloudFormation 템플릿을 사용할 때는 리소스가 제대로 생성되도록 올바른 파라미터를 선택해야 합니다. 다음 단계를 따릅니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/cloudformation에서 AWS CloudFormation 콘솔을 엽니다.
2. 스택 생성을 선택합니다.

3. 템플릿 지정 섹션에서 컴퓨터에서 템플릿 파일 업로드를 선택하고 다음을 선택합니다.
4. 스택 세부 정보 지정 페이지에서 다음 파라미터를 설정합니다.
   a. 스택 이름을 PostgreSQLTestStack으로 설정합니다.
   b. 파라미터에서 가용 영역 3개를 선택하여 가용 영역을 설정합니다.
   c. Linux Bastion Host 구성에서 키 이름에 대해 EC2 인스턴스에 로그인할 키 페어를 선택합니다.
   d. Linux Bastion Host 구성 설정에서 허용된 IP 범위를 IP 주소로 설정합니다. Secure Shell(SSH)을 사용하여 VPC의 EC2 인스턴스에 연결하려면 https://checkip.amazonaws.com의 서비스를 사용하여 퍼블릭 IP 주소를 지정합니다. IP 주소의 예는 192.0.2.1/32입니다.

      > **Warning**
      >
      > SSH 액세스에 `0.0.0.0/0`을 사용하는 경우 모든 IP 주소가 SSH를 사용하여 퍼블릭 EC2 인스턴스에 액세스할 수 있도록 활성화합니다. 이 방법은 테스트 환경에서 잠시 사용하는 것은 괜찮지만 프로덕션 환경에서는 안전하지 않습니다. 프로덕션에서는 SSH를 사용하여 EC2 인스턴스에 액세스할 수 있는 특정 IP 주소 또는 주소 범위만 인증합니다.

   e. 데이터베이스 일반 구성에서 데이터베이스 인스턴스 클래스를 db.t3.micro로 설정합니다.
   f. 데이터베이스 이름을 **`database-test1`**으로 설정합니다.
   g. 데이터베이스 마스터 사용자 이름에 마스터 사용자 이름을 입력합니다.
   h. 이 자습서에서는 Secrets Manager를 사용하여 DB 마스터 사용자 암호 관리를 `false`로 설정합니다.
   i. 데이터베이스 암호의 경우 원하는 암호를 설정합니다. 자습서의 향후 단계에 사용할 수 있도록 암호를 기억해 둡니다.
   j. 데이터베이스 스토리지 구성에서 데이터베이스 스토리지 유형을 gp2로 설정합니다.
   k. 데이터베이스 모니터링 구성에서 RDS 성능 개선 도우미 활성화를 false로 설정합니다.
   l. 다른 모든 설정은 기본값으로 둡니다. 계속하려면 다음을 클릭합니다.
5. 스택 옵션 구성 페이지에서는 모든 기본 옵션을 그대로 둡니다. 계속하려면 다음을 클릭합니다.
6. 스택 검토 페이지에서 데이터베이스 및 Linux Bastion Host 옵션을 확인한 후 제출을 선택합니다.

스택 생성 프로세스가 완료되면 BastionStack 및 RDSNS라는 이름의 스택을 보고 데이터베이스에 연결하는 데 필요한 정보를 기록해 둡니다. 자세한 내용은 AWS Management Console에서 AWS CloudFormation 스택 데이터 및 리소스 보기를 참조하세요.

PostgreSQL DB 인스턴스에 연결

pgadmin 또는 psql을 사용하여 DB 인스턴스에 연결할 수 있습니다. 이 예시에서는 psql 명령줄 클라이언트를 사용하여 PostgreSQL DB 인스턴스에 연결하는 방법을 설명합니다.

psql을 사용하여 PostgreSQL DB 인스턴스에 연결하려면

1. DB 인스턴스의 엔드포인트(DNS 이름)와 포트 번호를 찾습니다.

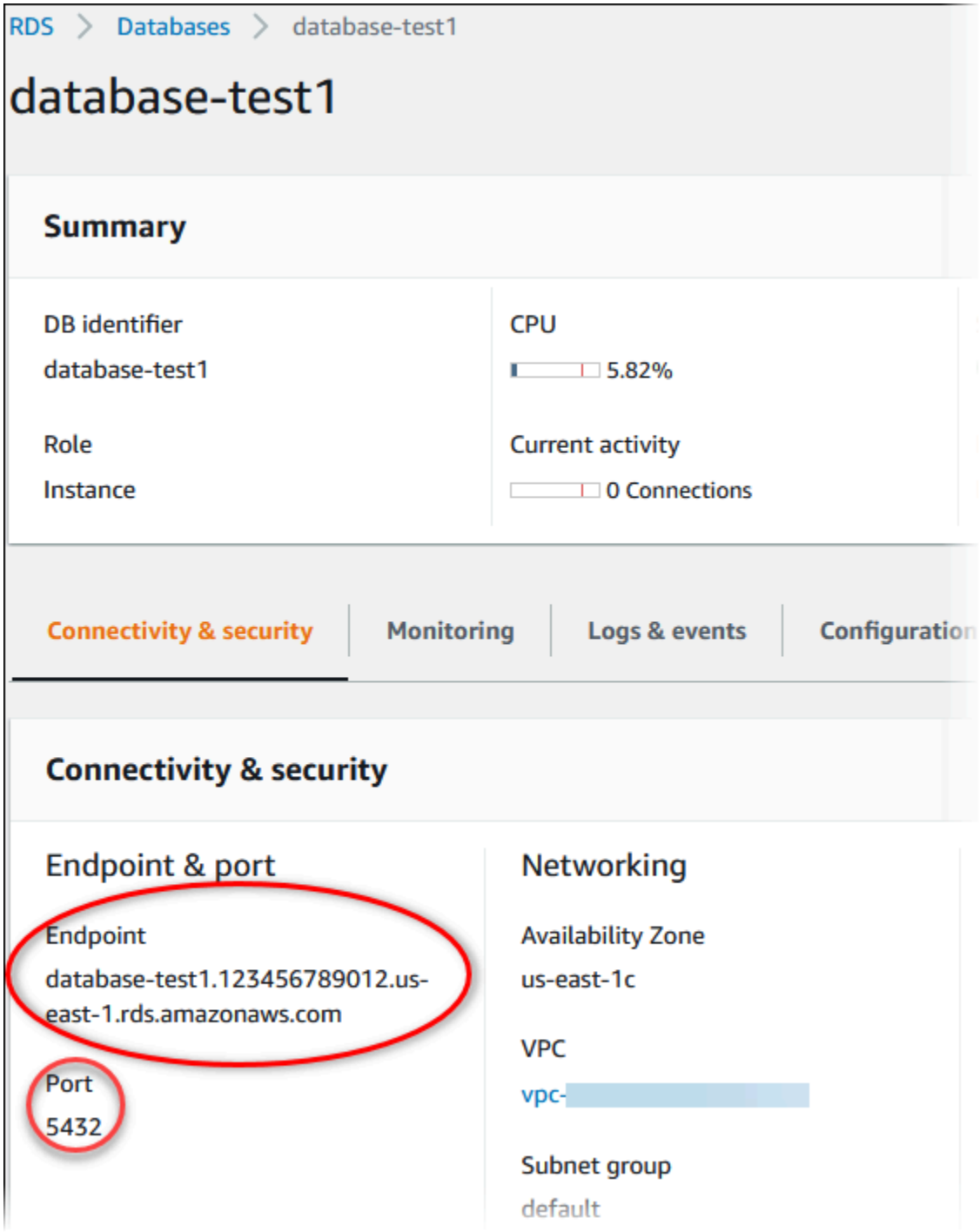
   a. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.

   b. Amazon RDS 콘솔의 오른쪽 상단에서 DB 인스턴스의 AWS 리전을 선택합니다.

   c. 탐색 창에서 Databases(데이터베이스)를 선택합니다.

   d. 세부 정보를 표시하고자 하는 PostgreSQL DB 인스턴스 이름을 선택합니다.

   e. Connectivity & security(연결 및 보안) 탭에서 엔드포인트를 복사합니다. 또한 포트 번호를 적어 둡니다. DB 인스턴스에 연결하려면 엔드포인트와 포트 번호가 모두 필요합니다.



2. Amazon EC2 사용 설명서에 있는 Linux 인스턴스에 연결의 단계를 따라 앞에서 만든 EC2 인스턴스에 연결합니다.

   SSH를 사용하여 EC2 인스턴스에 연결하는 것이 좋습니다. Windows, Linux 또는 Mac에 SSH 클라이언트 유틸리티가 설치된 경우 다음 명령 형식을 사용하여 인스턴스에 연결할 수 있습니다.

```
ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
```

예를 들어 `ec2-database-connect-key-pair.pem`이 Linux의 `/dir1`에 저장되어 있고, EC2 인스턴스의 퍼블릭 IPv4 DNS가 `ec2-12-345-678-90.compute-1.amazonaws.com`이라고 가정해 보겠습니다. SSH 명령은 다음과 같이 표시됩니다.

```
ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
user@ec2-12-345-678-90.compute-1.amazonaws.com
```

3. EC2 인스턴스에서 소프트웨어를 업데이트하여 최신 버그 수정 및 보안 업데이트를 받습니다. 이렇게 하려면 다음 명령을 사용하십시오.

> **Note**
>
> `-y` 옵션을 사용하면 확인 여부를 묻지 않고 업데이트를 설치합니다. 설치 전에 업데이트 정보를 확인하려면 이 옵션을 생략합니다.

```
sudo dnf update -y
```

4. Amazon Linux 2023의 PostgreSQL에서 psql 명령줄 클라이언트를 설치하려면 다음 명령을 실행합니다.

```
sudo dnf install postgresql15
```

5. PostgreSQL DB 인스턴스에 연결합니다. 예를 들어 클라이언트 컴퓨터의 명령 프롬프트에서 다음 명령을 입력합니다. 이 작업을 수행하면 psql 클라이언트를 사용하여 PostgreSQL DB 인스턴스에 연결할 수 있습니다.

*endpoint*는 DB 인스턴스 엔드포인트로 대체하고, *postgres*는 연결하려는 데이터베이스 이름 `--dbname`으로 대체하고, *postgres*는 사용된 마스터 사용자 이름으로 대체합니다. 암호를 묻는 메시지가 표시되면 사용한 마스터 암호를 제공합니다.

```
psql --host=endpoint --port=5432 --dbname=postgres --username=postgres
```

사용자에 대한 암호를 입력하면 다음과 유사한 출력이 나타납니다.

```
psql (14.3, server 14.6)
```

```
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256,
 compression: off)
Type "help" for help.

postgres=>
```

PostgreSQL DB 인스턴스 연결에 대한 자세한 내용은 PostgreSQL 데이터베이스 엔진을 실행하는 DB 인스턴스에 연결 섹션을 참조하세요. DB 인스턴스에 연결할 수 없는 경우 RDS for PostgreSQL 인스턴스에 대한 연결 문제 해결 단원을 참조하십시오.

보안을 위해서는 암호화된 연결을 사용하는 것이 가장 좋습니다. 클라이언트와 서버가 동일한 VPC에 있고 네트워크를 신뢰할 수 있는 경우에만 암호화되지 않은 PostgreSQL 연결을 사용합니다. 암호화된 연결 사용에 대한 자세한 내용은 SSL을 통해 PostgreSQL DB 인스턴스에 연결 섹션을 참조하세요.

6. SQL 명령을 실행합니다.

   예를 들어, 다음 SQL 명령은 현재 날짜 및 시간을 보여줍니다.

```
SELECT CURRENT_TIMESTAMP;
```

EC2 인스턴스 및 DB 인스턴스 삭제

생성한 샘플 EC2 인스턴스 및 DB 인스턴스에 연결하고 탐색한 후에는 요금이 더 이상 부과되지 않도록 삭제합니다.

AWS CloudFormation을 사용하여 리소스를 생성했다면 이 단계를 건너뛰고 다음 단계로 이동합니다.

EC2 인스턴스를 삭제하는 방법

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창에서 인스턴스를 선택합니다.
3. EC2 인스턴스를 선택하고 인스턴스 상태, 인스턴스 종료를 차례로 선택합니다.
4. 확인 메시지가 나타나면 종료를 선택합니다.

EC2 인스턴스 삭제에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 종료를 참조하세요.

최종 DB 스냅샷이 없는 DB 인스턴스를 삭제하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 삭제할 DB 인스턴스를 선택합니다.
4. [Actions]에 대해 [Delete]를 선택합니다.
5. 최종 스냅샷을 생성하시겠습니까? 및 자동 백업 보존을 선택 해제합니다.
6. 확인을 완료하고 삭제를 선택합니다.

(선택 사항) CloudFormation으로 생성한 EC2 인스턴스 및 DB 인스턴스 삭제

AWS CloudFormation을 사용하여 리소스를 생성한 경우 샘플 EC2 인스턴스 및 DB 인스턴스에 연결하고 탐색한 후 CloudFormation 스택을 삭제하여 더 이상 비용이 청구되지 않도록 합니다.

CloudFormation 리소스를 삭제하려면

1. AWS CloudFormation 콘솔을 엽니다.
2. CloudFormation 콘솔의 스택 페이지에서 루트 스택(VPCStack, BastionStack 또는 RDSNS라는 이름이 없는 스택)을 선택합니다.
3. 삭제를 선택합니다.
4. 확인 메시지가 나타나면 스택 삭제를 선택합니다.

CloudFormation에서 스택을 삭제하는 방법에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS CloudFormation 콘솔에서 스택 삭제를 참조하세요.

(선택 사항) DB 인스턴스를 Lambda 함수에 연결

RDS for PostgreSQL DB 인스턴스를 Lambda 서버리스 컴퓨팅 리소스에 연결할 수도 있습니다. Lambda 함수를 사용하면 인프라를 프로비저닝하거나 관리하지 않고도 코드를 실행할 수 있습니다. 또한 하루 12개에서 초당 수백 개에 이르는 모든 규모의 코드 실행 요청에 자동으로 응답할 수 있습니다. 자세한 내용은 Lambda 함수와 DB 인스턴스 자동 연결 섹션을 참조하세요.

자습서: 웹 서버 및 Amazon RDS DB 인스턴스 생성

이 자습서에서는 PHP가 있는 Apache 웹 서버를 설치하고 MariaDB, MySQL 또는 PostgreSQL 데이터베이스를 생성하는 방법을 보여줍니다. 이 웹 서버는 Amazon Linux 2023을 사용하여 Amazon EC2 인스턴스에서 실행되며, MySQL 또는 PostgreSQL DB 인스턴스 중에서 선택할 수 있습니다. Amazon EC2 인스턴스와 DB 인스턴스 모두 Amazon VPC 서비스를 기반으로 하는 Virtual Private Cloud(VPC)에서 실행됩니다.

Important

AWS 계정 생성은 무료입니다. 그러나 이 자습서를 완료하면 사용하는 AWS 리소스에 대한 비용이 발생할 수 있습니다. 자습서가 더 이상 필요하지 않은 경우 자습서를 완료한 후에 이러한 리소스를 삭제할 수 있습니다.

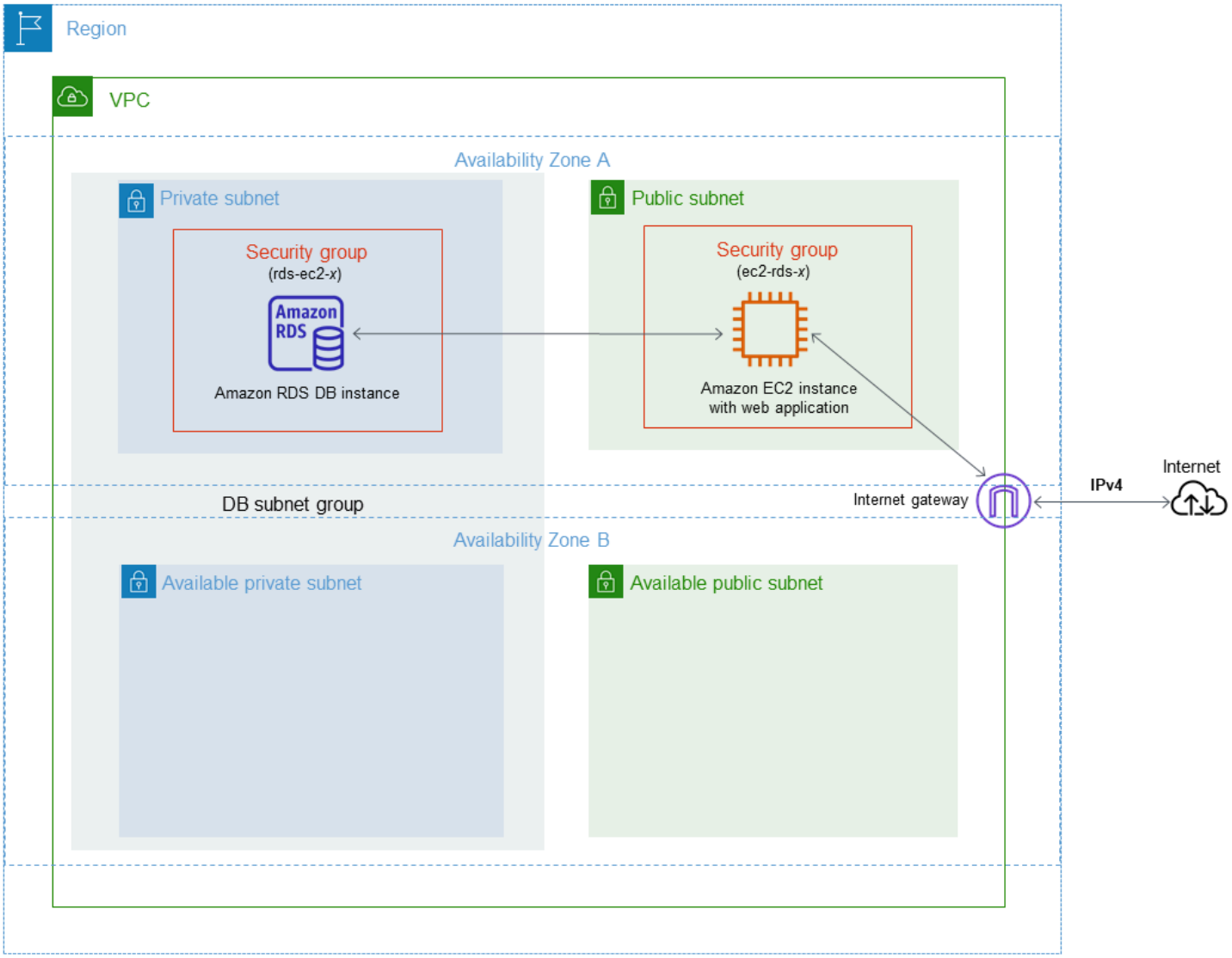
Note

이 자습서에서는 Amazon Linux 2023을 사용하여 작업하며 다른 버전의 Linux는 적용되지 않을 수 있습니다.

다음 자습서에서는 AWS 계정의 .NET용 기본 VPC, 서브넷 및 보안 그룹을 사용하는 EC2 인스턴스를 생성합니다. 이 자습서에서는 DB 인스턴스를 생성하고 생성한 EC2 인스턴스와의 연결을 자동으로 설정하는 방법을 보여줍니다. 그런 다음 자습서에서는 EC2 인스턴스에 웹 서버를 설치하는 방법을 보여줍니다. DB 인스턴스 엔드포인트를 사용하여 VPC의 DB 인스턴스에 웹 서버를 연결합니다.

1. EC2 인스턴스를 시작하여 DB 인스턴스에 연결
2. Amazon RDS DB 인스턴스 생성
3. EC2 인스턴스에 웹 서버 설치

다음 다이어그램은 이 자습서를 완료했을 때 구성을 보여 줍니다.



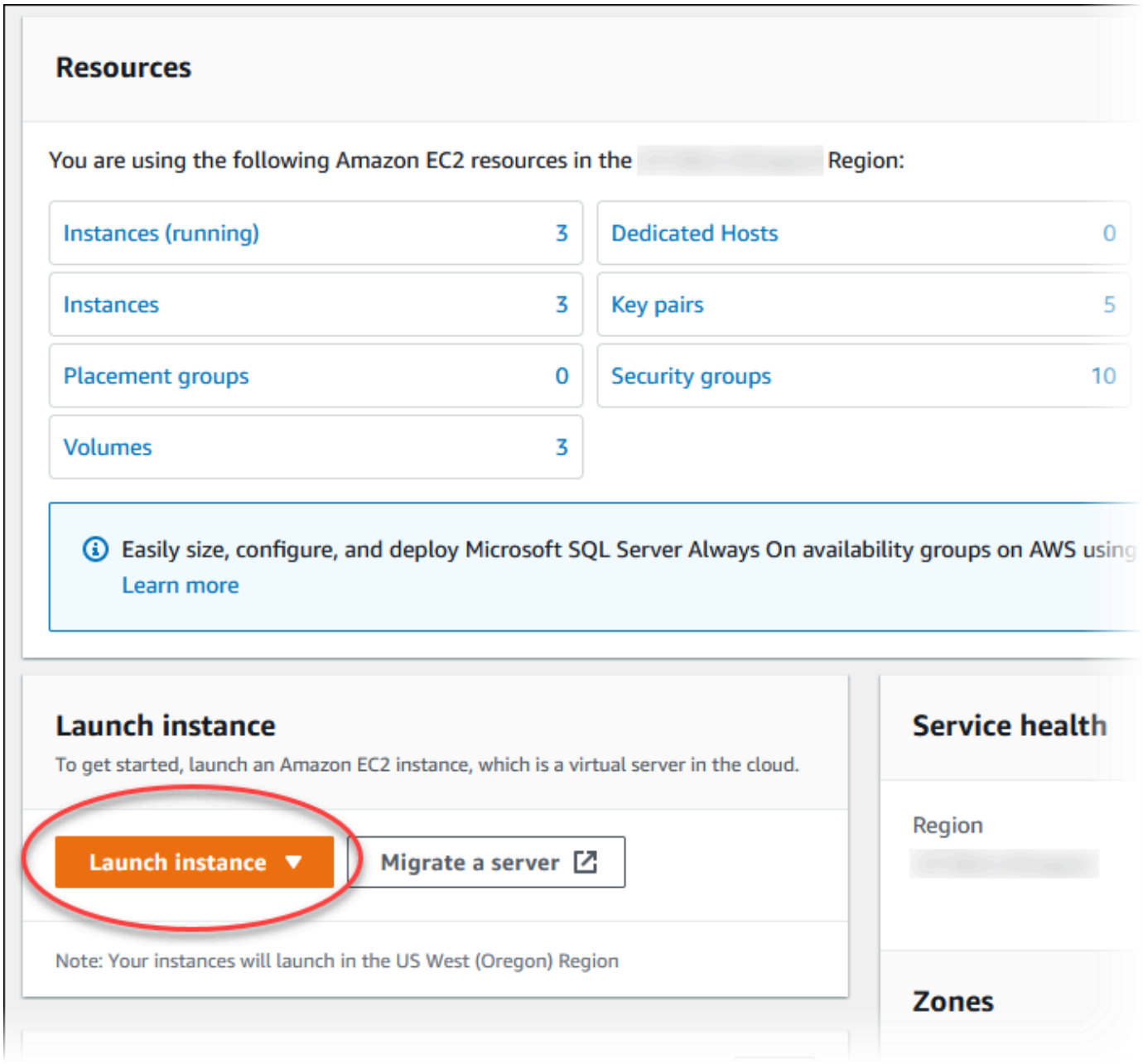
> **Note**
>
> 자습서를 완료하면 VPC의 각 가용 영역에 퍼블릭 서브넷과 프라이빗 서브넷이 있을 것입니다. 이 자습서에서는 AWS 계정에 대한 기본 VPC를 사용하고 EC2 인스턴스와 DB 인스턴스 간의 연결을 자동으로 설정합니다. 대신 이 시나리오에 맞게 새 VPC 구성하려면 자습서: DB 인스턴스에 사용할 Amazon VPC 생성(IPv4 전용)에서 작업을 완료하세요.

EC2 인스턴스를 시작하여 DB 인스턴스에 연결

VPC의 퍼블릭 서브넷에서 Amazon EC2 인스턴스를 생성합니다.

EC2 인스턴스 시작

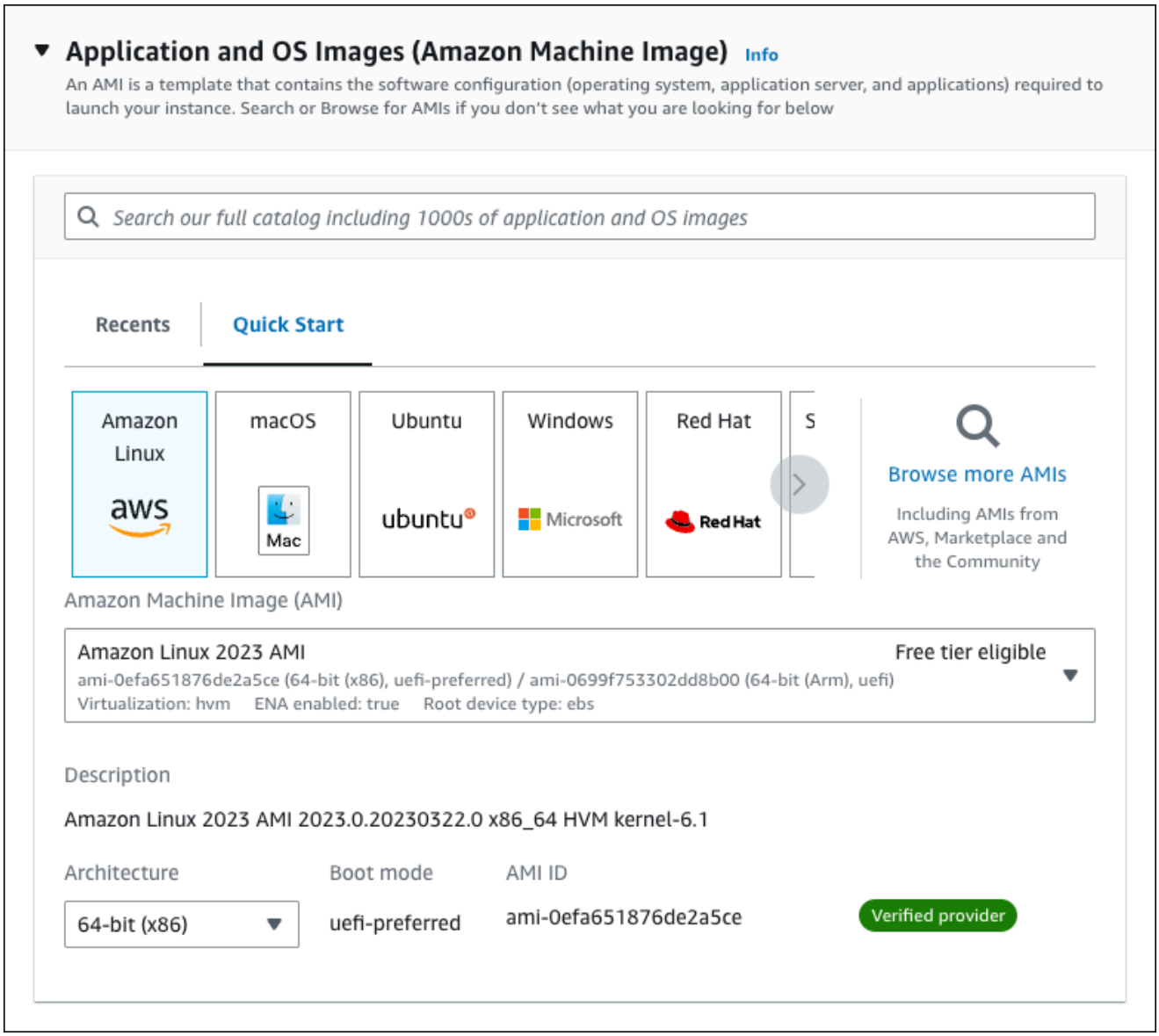
1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. AWS Management Console 콘솔의 오른쪽 상단에서 EC2 인스턴스를 생성하려는 AWS 리전을 선택합니다.
3. 다음과 같이 EC2 대시보드를 선택한 다음, 인스턴스 시작을 선택합니다.



4. 인스턴스 시작 페이지에서 다음 설정을 선택합니다.

a. Name and tags(이름 및 태그) 아래의 Name(이름)에 **`tutorial-ec2-instance-web-server`**을 입력하세요.

b. Application and OS Images (Amazon Machine Image)(애플리케이션 및 OS 이미지(Amazon Machine Image))에서 Amazon Linux를 선택한 다음 Amazon Linux 2023 AMI를 선택합니다. 다른 선택 항목에 대해서는 기본값을 그대로 유지합니다.



c. 인스턴스 유형에서 t2.micro를 선택합니다.

d. 키 페어(로그인)에서 기존 키 페어를 사용할 키 페어 이름을 선택합니다. Amazon EC2 인스턴스에 대한 새 키 페어를 생성하려면 새 키 페어 생성을 선택한 다음 키 페어 생성 창을 사용하여 생성합니다.

새 키 페어 생성에 대한 자세한 내용은 Amazon EC2 Linux 인스턴스용 사용 설명서의 키 페어 생성을 참조하세요.

e. 네트워크 설정에서 다음과 같이 이러한 값을 설정하고 다른 값은 기본값으로 유지합니다.

- SSH 트래픽 허용에서 EC2 인스턴스에 대한 SSH 연결 소스를 선택합니다.

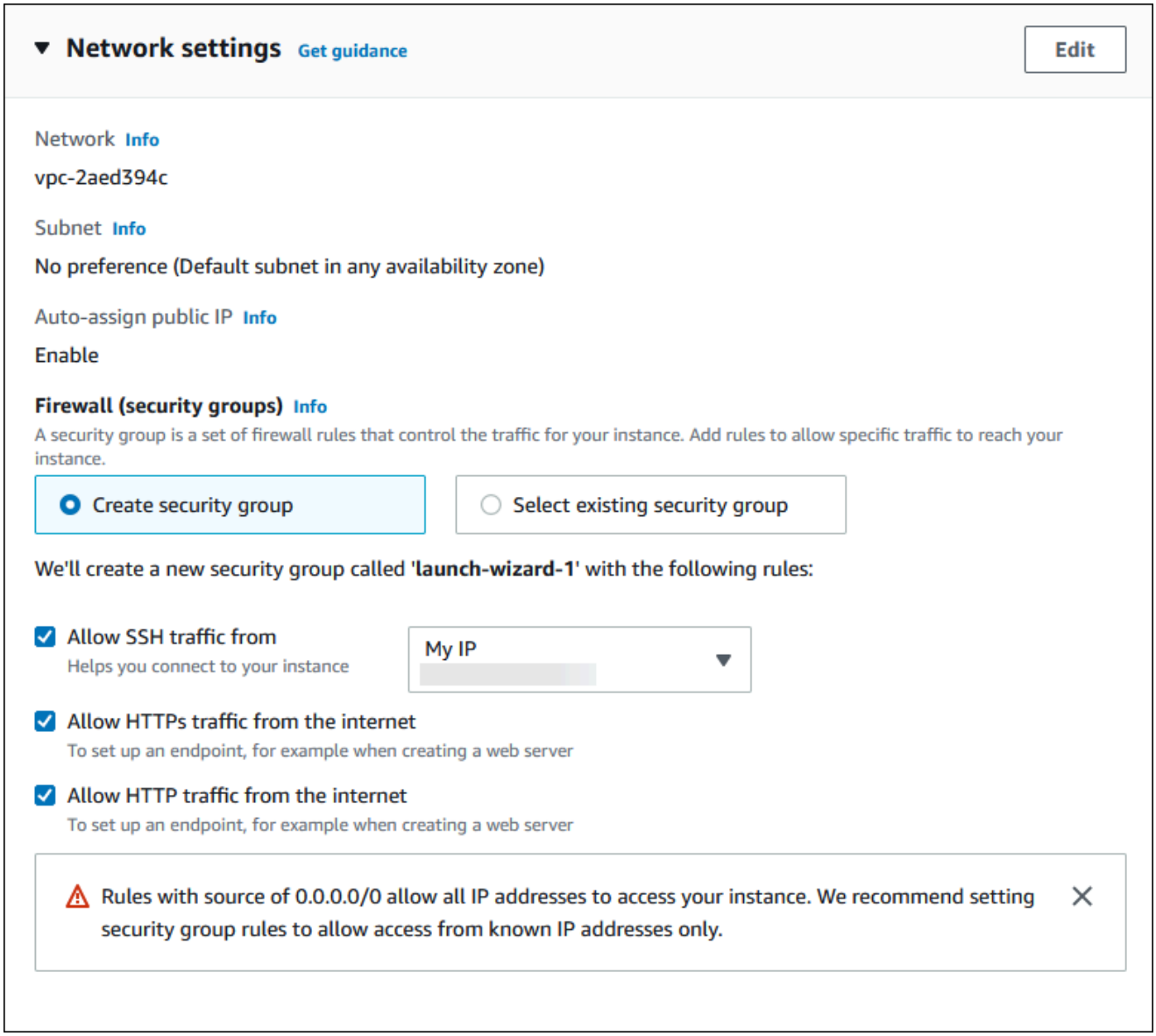
  표시된 IP 주소가 SSH 연결에 대해 올바른 경우 내 IP를 선택할 수 있습니다.

  그렇지 않으면 SSH(Secure Shell)를 사용하여 VPC의 EC2 인스턴스에 연결하는 데 사용할 IP 주소를 결정할 수 있습니다. 퍼블릭 IP 주소를 확인하려면 다른 브라우저 창 또는 탭에서 https://checkip.amazonaws.com의 서비스를 사용합니다. IP 주소의 예는 `203.0.113.25/32`입니다.

  대부분의 경우 고정 IP 주소가 없는 방화벽 뒤나 인터넷 서비스 제공업체(ISP)를 통해 연결하는 경우가 많습니다. 그렇다면 클라이언트 컴퓨터에서 사용하는 IP 주소 범위를 결정합니다.

  > **Warning**
  >
  > SSH 액세스에 `0.0.0.0/0`을 사용하는 경우 모든 IP 주소가 SSH를 사용하여 퍼블릭 인스턴스에 액세스할 수 있도록 활성화합니다. 이 방법은 테스트 환경에서 잠시 사용하는 것은 괜찮지만 프로덕션 환경에서는 안전하지 않습니다. 프로덕션에서는 SSH를 사용하여 인스턴스에 액세스할 수 있는 특정 IP 주소 또는 주소 범위만 인증합니다.

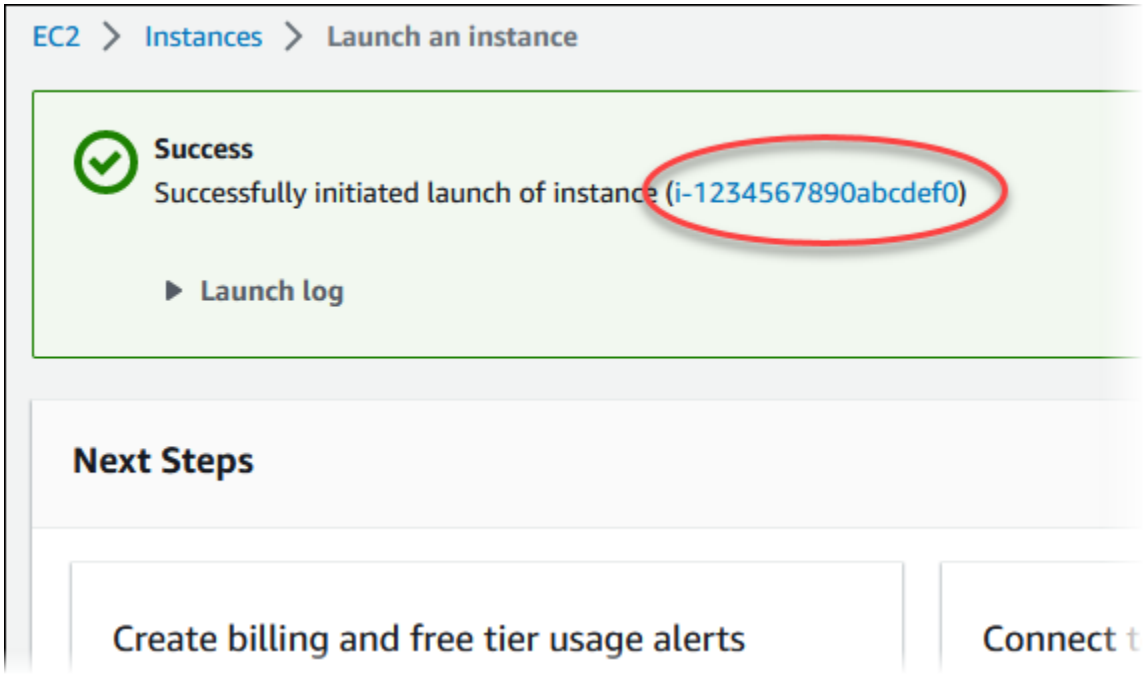
- 인터넷에서 오는 HTTPS 트래픽 허용을 켭니다.
- 인터넷에서 오는 HTTP 트래픽 허용을 켭니다.



f. 나머지 섹션에서 기본값은 그대로 둡니다.

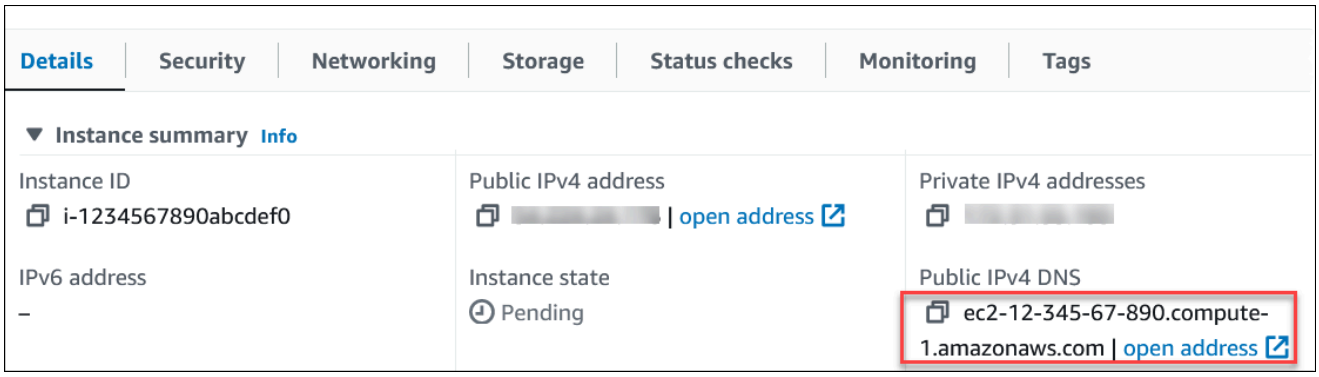
g. 요약 패널에서 인스턴스 구성 요약을 검토하고 준비가 되면 인스턴스 시작을 선택합니다.

5. 시작 상태 페이지에서, 새 EC2 인스턴스의 식별자(예: `i-1234567890abcdef0`)를 기록해 둡니다.

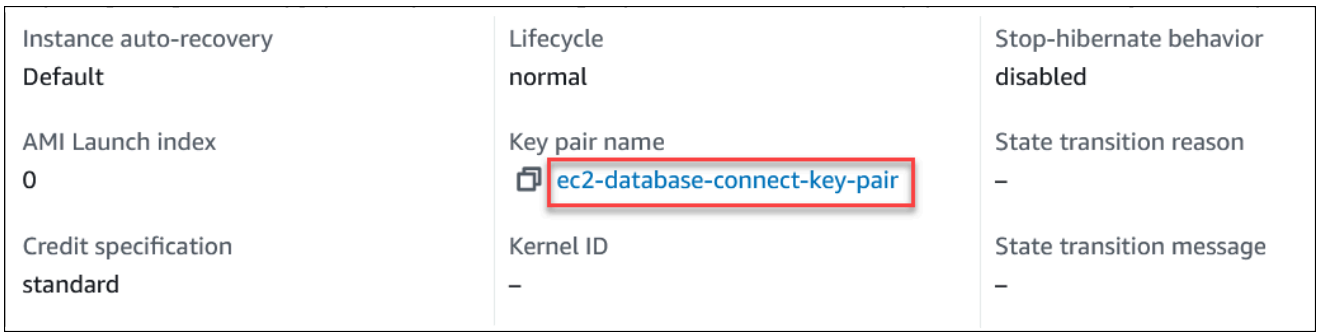


6. EC2 인스턴스 식별자를 선택하여 EC2 인스턴스 목록을 열고 EC2 인스턴스를 선택합니다.
7. 세부 정보 탭에서 SSH를 사용하여 연결할 때 필요한 다음 값을 기록해 둡니다.

   a. 인스턴스 요약에서 퍼블릭 IPv4 DNS의 값을 기록해 둡니다.



   b. 인스턴스 세부 정보에서 키 페어 이름의 값을 기록해 둡니다.



8. 인스턴스의 인스턴스 상태가 실행 중으로 읽힐 때까지 기다린 다음 계속합니다.

9. Amazon RDS DB 인스턴스 생성을 완료합니다.

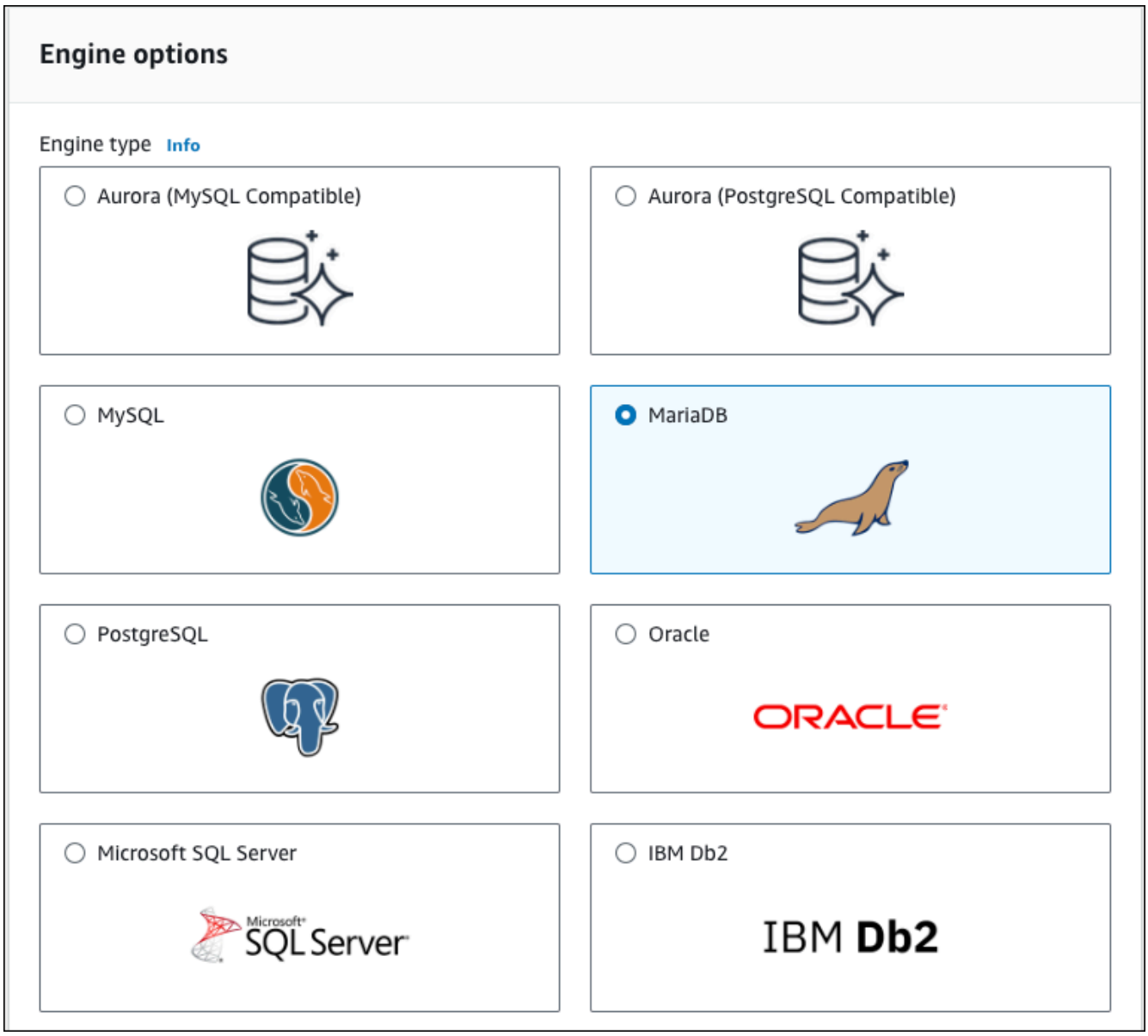
Amazon RDS DB 인스턴스 생성

웹 애플리케이션에서 사용되는 데이터를 유지 관리하는 RDS for MariaDB, RDS for MySQL 또는 RDS for PostgreSQL 인스턴스를 생성합니다.

RDS for MariaDB

MariaDB 인스턴스를 만들려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. AWS Management Console의 오른쪽 상단 모서리에서 AWS 리전을 확인합니다. EC2 인스턴스를 생성한 리전과 동일해야 합니다.
3. 탐색 창에서 데이터베이스를 선택합니다.
4. 데이터베이스 생성을 선택합니다.
5. 데이터베이스 생성 페이지에서 표준 생성을 선택합니다.
6. 엔진 옵션에서 MariaDB를 선택합니다.

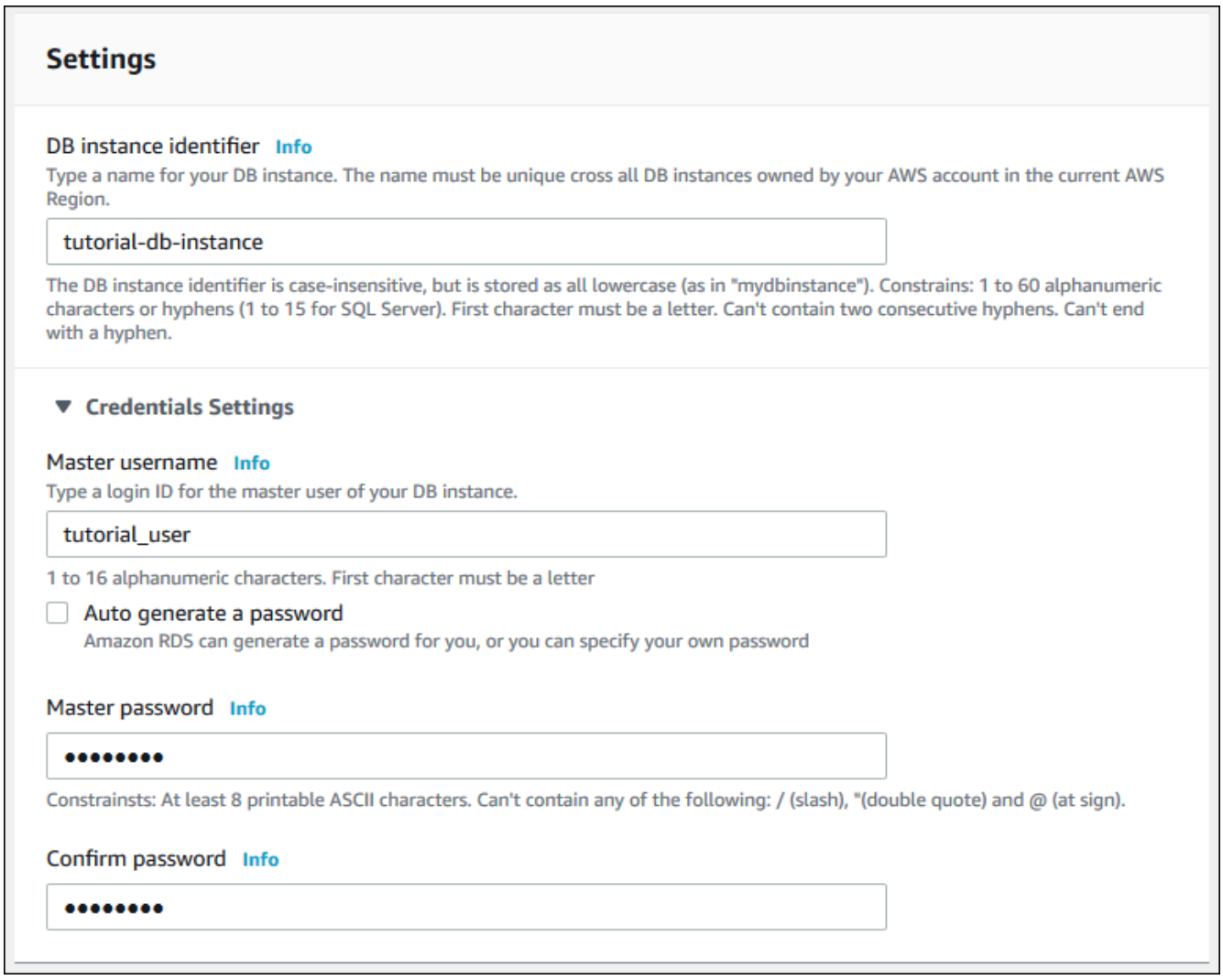


7. 템플릿에서 프리 티어 또는 샌드박스를 선택합니다. 프리 티어는 무료 요금제 계정에 표시됩니다. 샌드박스는 유료 요금제 계정에 표시됩니다.



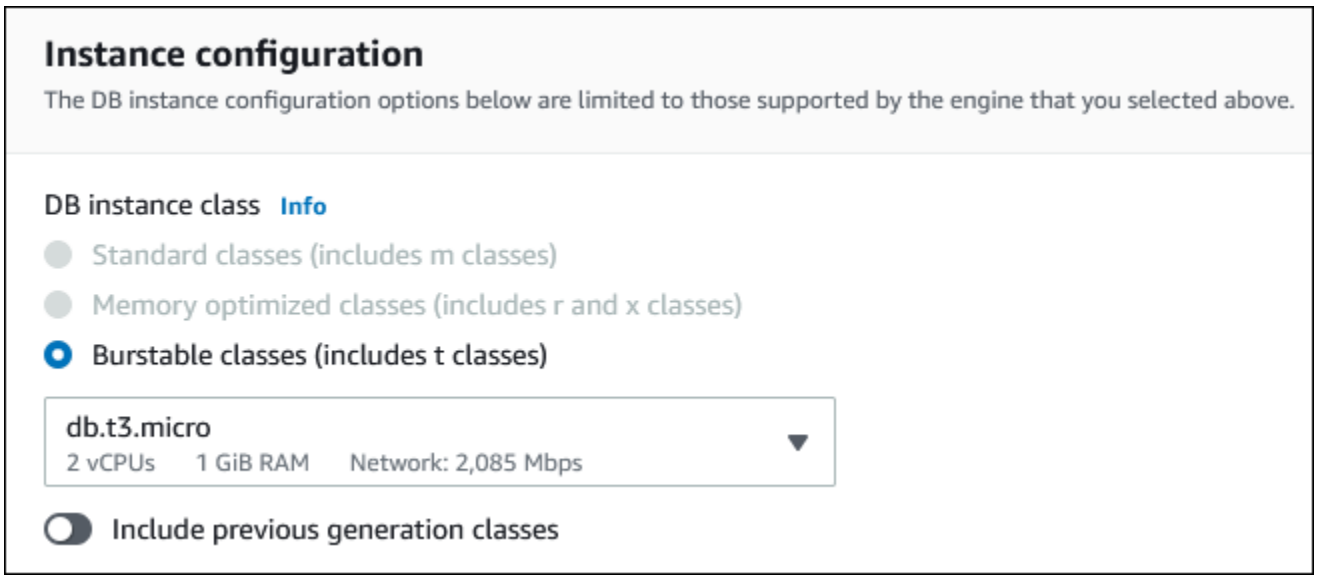
8. 가용성 및 내구성 섹션에서 기본값을 사용합니다.
9. 설정 섹션에서 이러한 값들을 설정합니다.

   - DB 인스턴스 식별자 – **`tutorial-db-instance`**를 입력합니다.
   - 마스터 사용자 이름 - **`tutorial_user`**를 입력합니다.
   - Auto generate a password(암호 자동 생성) - 옵션을 끈 상태로 둡니다.
   - 마스터 암호 – 암호를 입력합니다.
   - 암호 확인 – 암호를 다시 입력합니다.

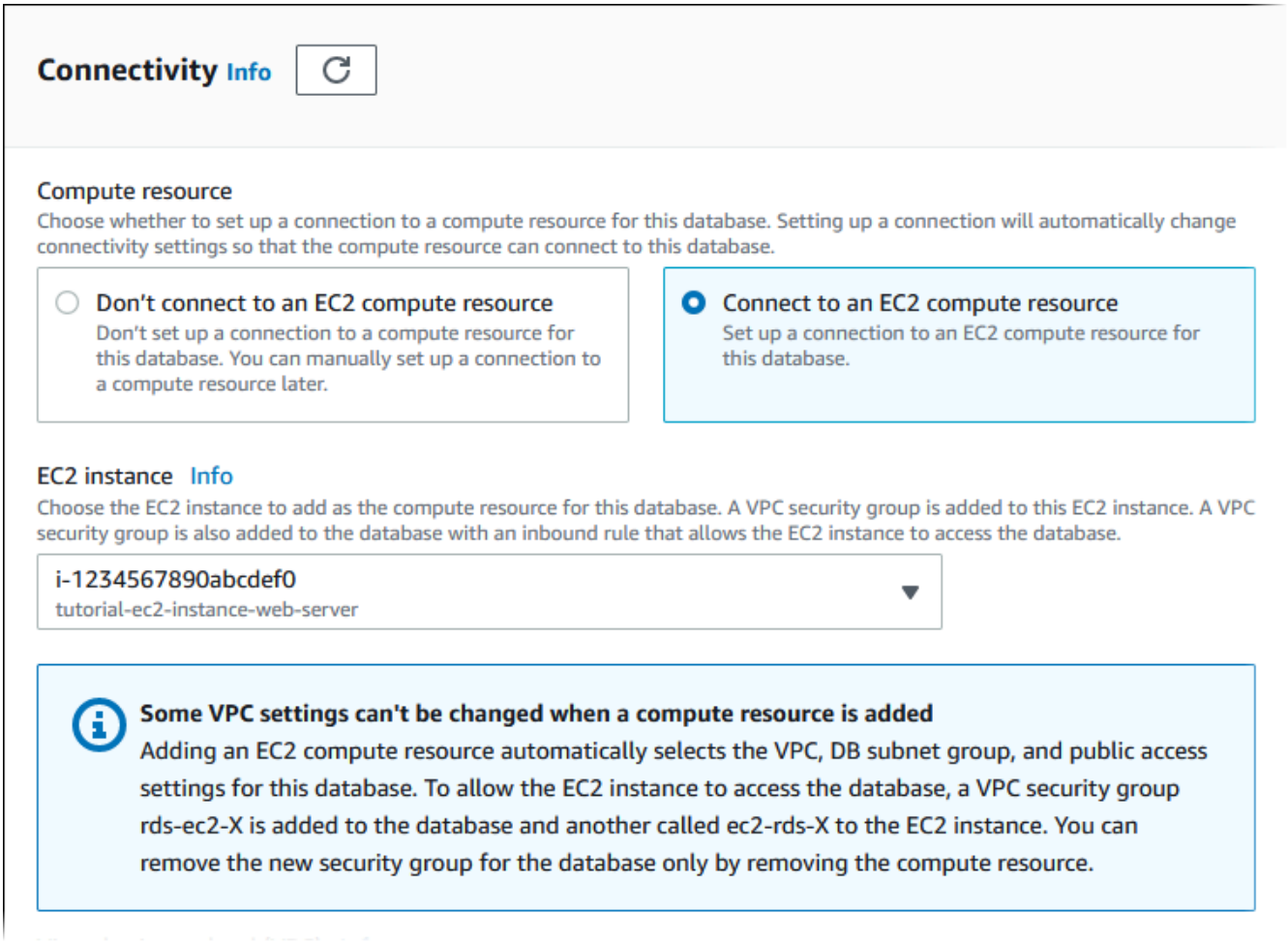


10. 인스턴스 구성] 섹션에서 다음 값을 설정합니다.

    - 버스트 가능 클래스(t 클래스 포함)
    - db.t3.micro

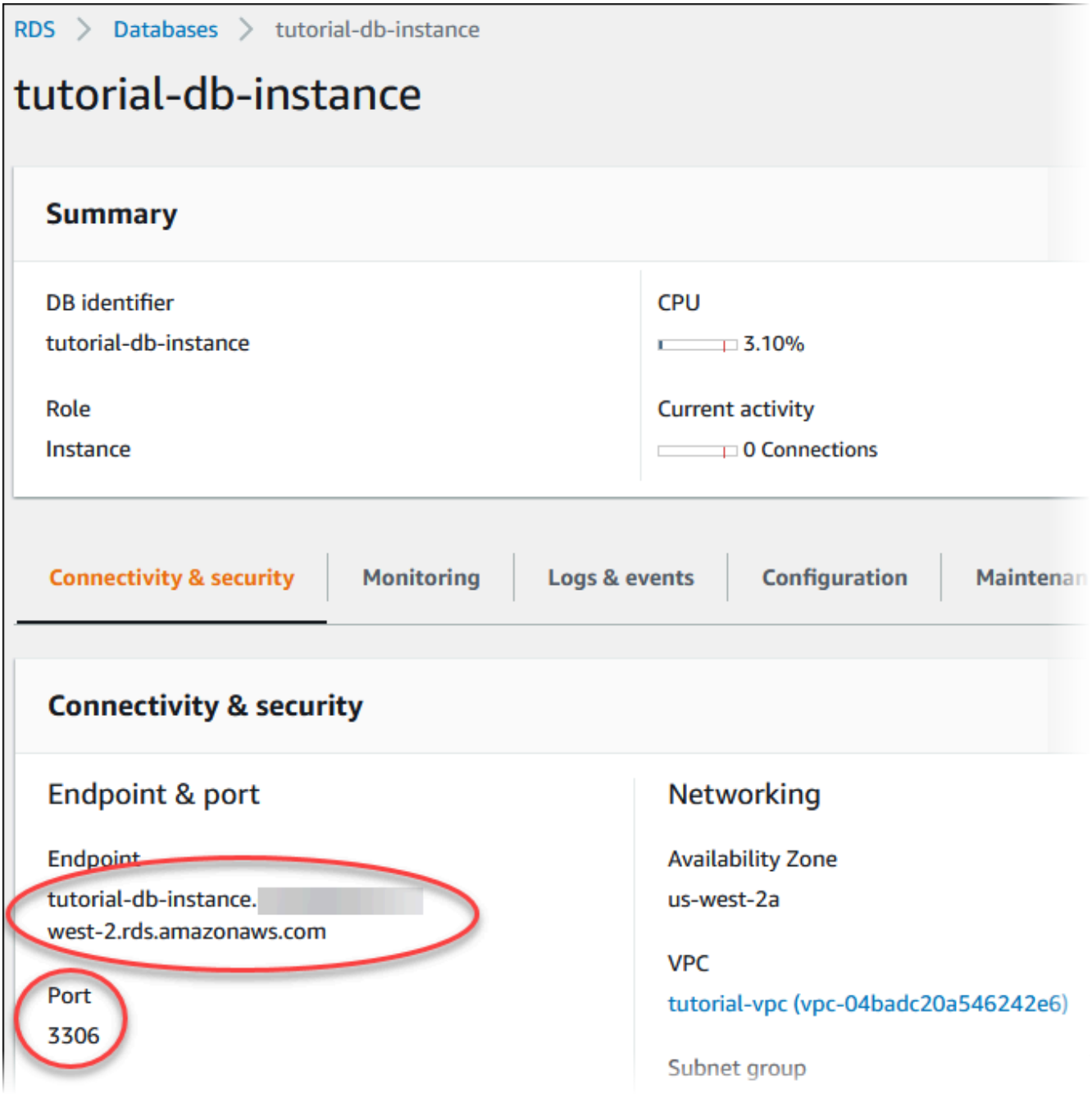


11. 스토리지 섹션에서 기본값으로 유지합니다.
12. 연결 섹션에서 다음 값을 설정하고 다른 값을 기본값으로 유지합니다.
    - 컴퓨팅 리소스에서 EC2 컴퓨팅 리소스에 연결을 선택합니다.
    - EC2 인스턴스의 경우 tutorial-ec2-instance-web-server와 같이 이전에 생성한 EC2 인스턴스를 선택합니다.



13. [데이터베이스 인증(Database authentication)] 섹션에서 [암호 인증(Password authentication)]이 선택되어 있는지 확인합니다.
14. 추가 구성 섹션을 열고 초기 데이터베이스 이름에 **`sample`**를 입력합니다. 다른 옵션은 기본 설정을 유지합니다.
15. MariaDB 인스턴스를 생성하려면 데이터베이스 생성을 선택합니다.

    새 DB 인스턴스가 데이터베이스목록에 생성 중의 상태로 나타납니다.
16. 새 DB 인스턴스의 상태가 사용 가능으로 나타날 때까지 기다립니다. 그런 다음, 세부 정보를 표시할 DB 인스턴스 이름을 선택합니다.
17. Connectivity & security(연결 및 보안) 섹션에서 DB 인스턴스의 엔드포인트 및 포트를 확인합니다.



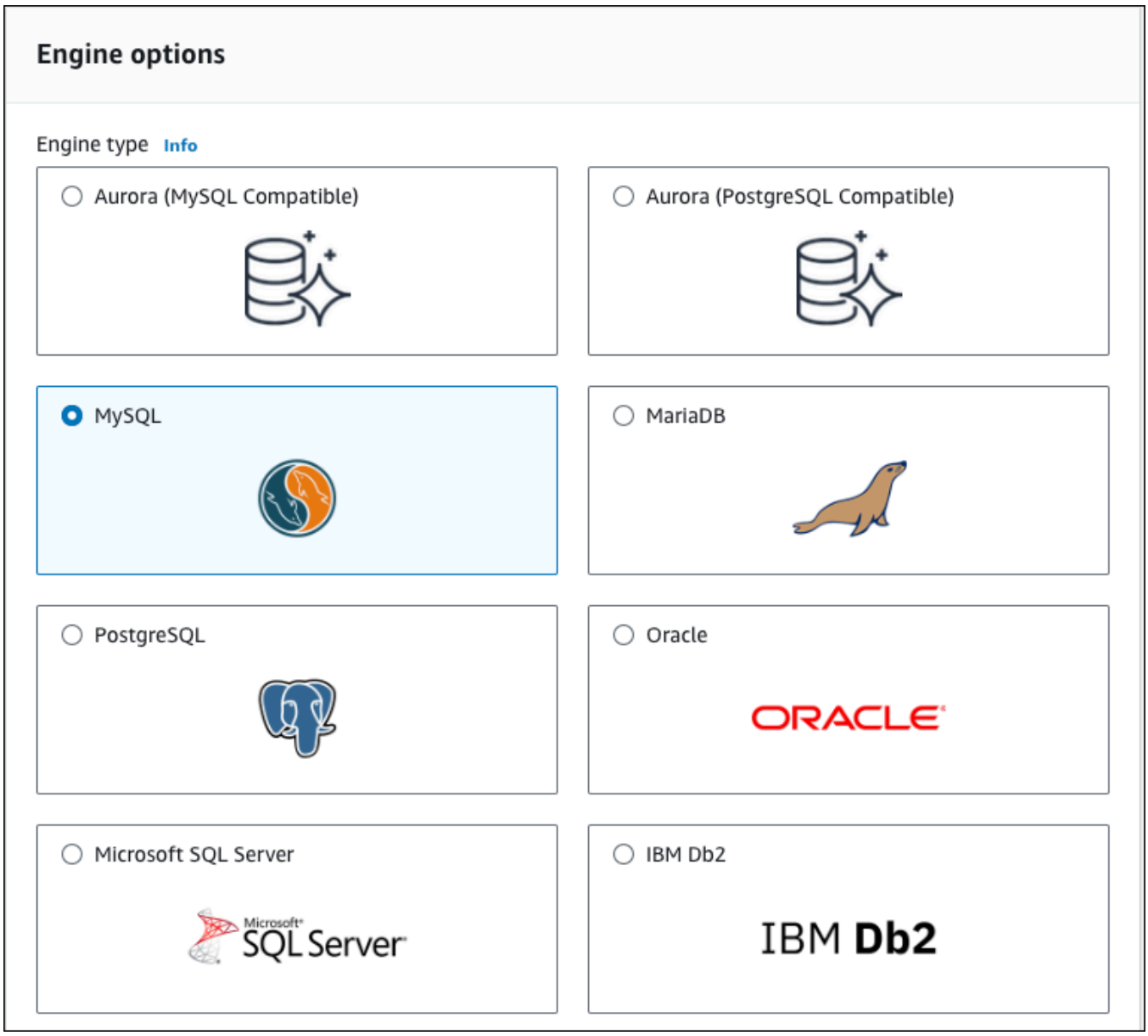
DB 인스턴스의 엔드포인트와 포트를 적어 둡니다. 이 정보를 사용하여 웹 서버를 RDS DB 인스턴스에 연결하게 됩니다.

18. EC2 인스턴스에 웹 서버 설치를 완료합니다.

RDS for MySQL

MySQL DB 인스턴스를 만들려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. AWS Management Console의 오른쪽 상단 모서리에서 AWS 리전을 확인합니다. EC2 인스턴스를 생성한 리전과 동일해야 합니다.
3. 탐색 창에서 데이터베이스를 선택합니다.
4. 데이터베이스 생성을 선택합니다.
5. 데이터베이스 생성 페이지에서 표준 생성을 선택합니다.
6. 엔진 옵션에서 MySQL을 선택합니다.

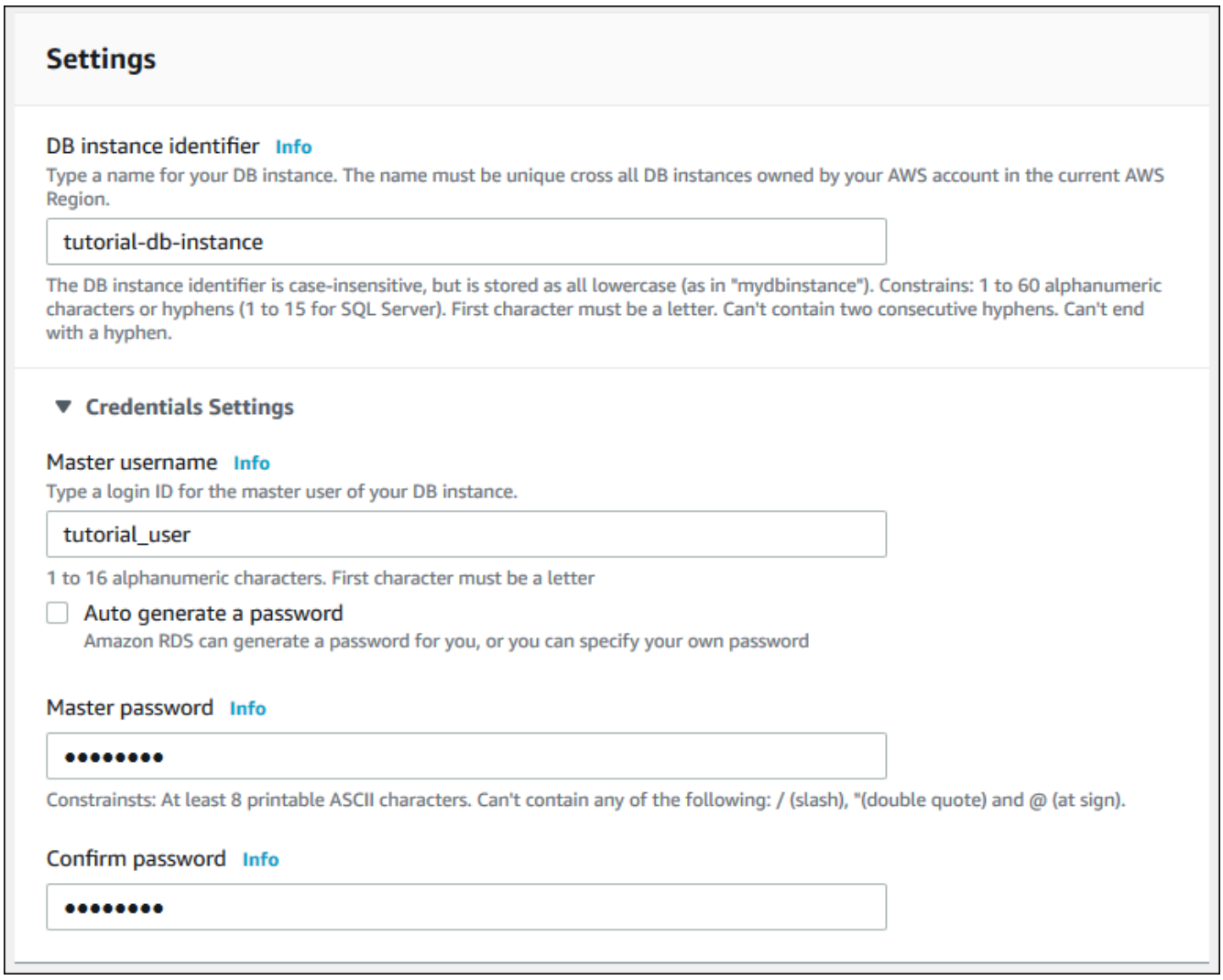


7. 템플릿에서 프리 티어 또는 샌드박스를 선택합니다. 프리 티어는 무료 요금제 계정에 표시됩니다. 샌드박스는 유료 요금제 계정에 표시됩니다.



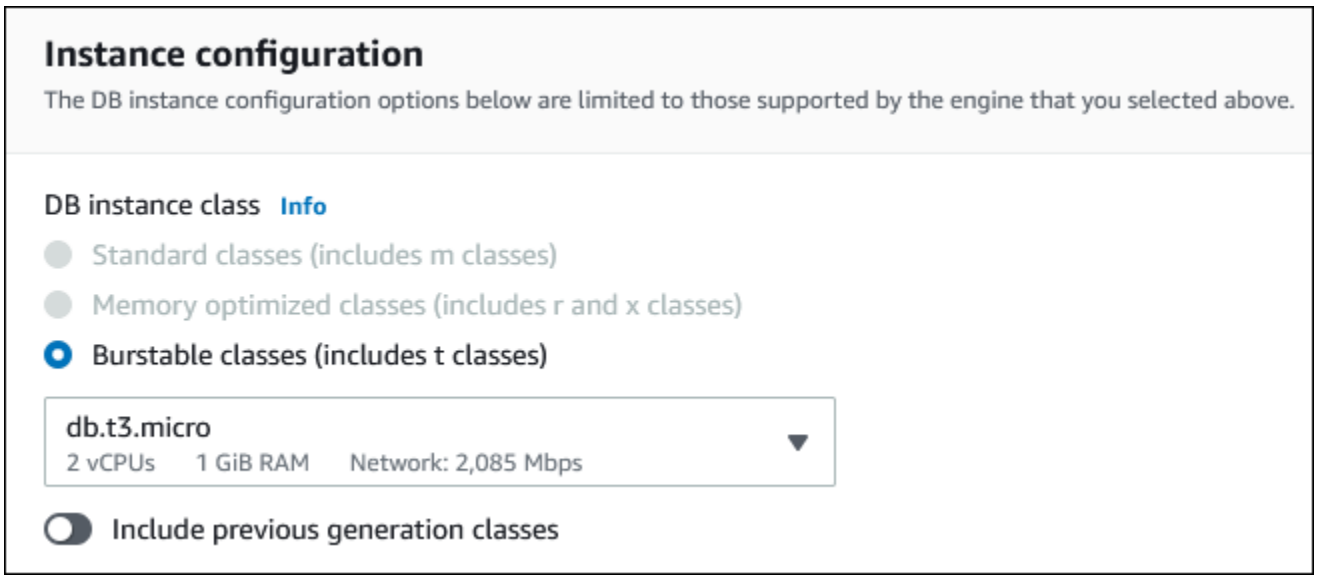
8. 가용성 및 내구성 섹션에서 기본값을 사용합니다.
9. 설정 섹션에서 이러한 값들을 설정합니다.

   - DB 인스턴스 식별자 – **`tutorial-db-instance`**를 입력합니다.
   - 마스터 사용자 이름 - **`tutorial_user`**를 입력합니다.
   - Auto generate a password(암호 자동 생성) - 옵션을 끈 상태로 둡니다.
   - 마스터 암호 – 암호를 입력합니다.
   - 암호 확인 – 암호를 다시 입력합니다.

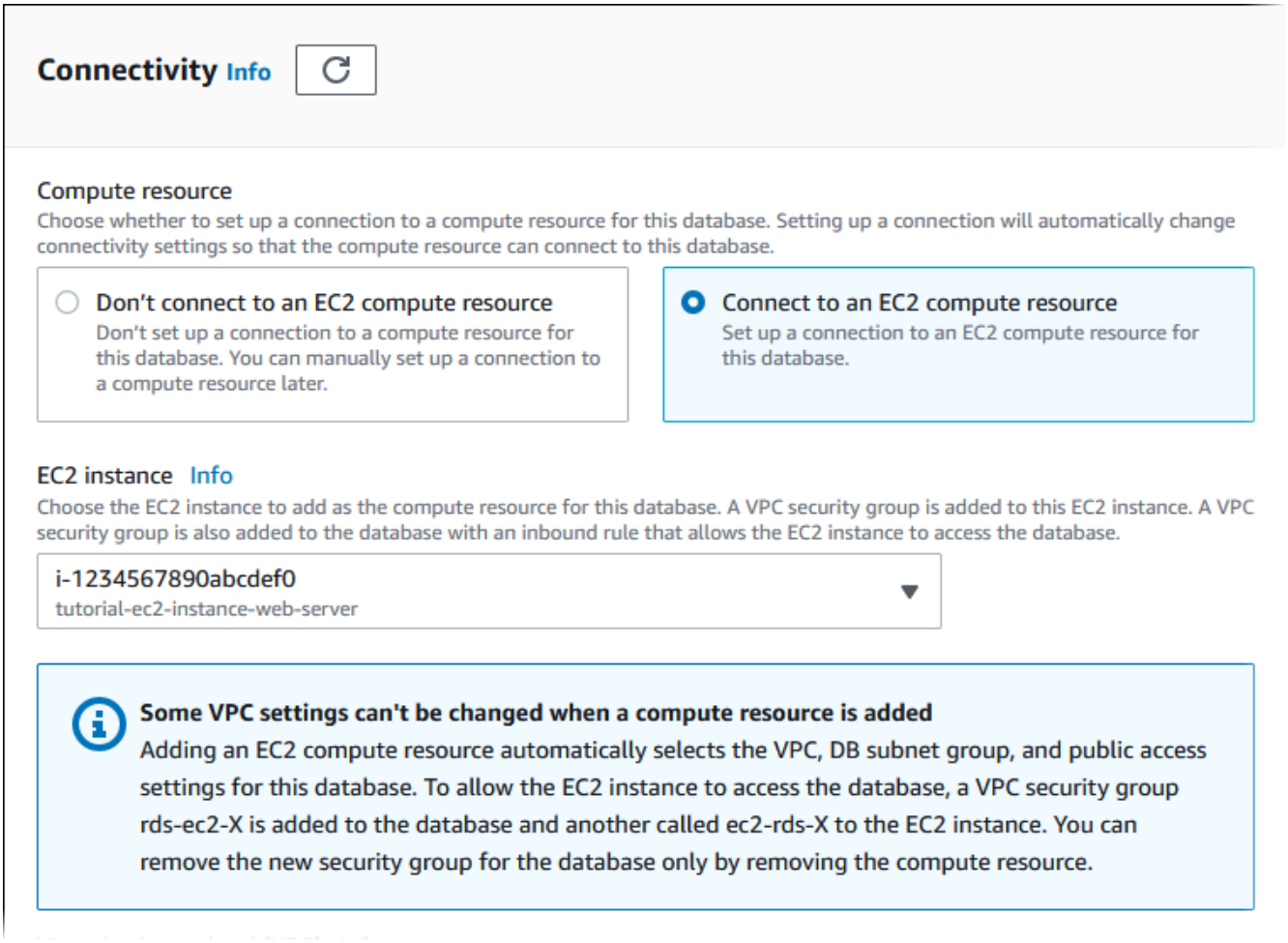


10. 인스턴스 구성] 섹션에서 다음 값을 설정합니다.

    - 버스트 가능 클래스(t 클래스 포함)
    - db.t3.micro

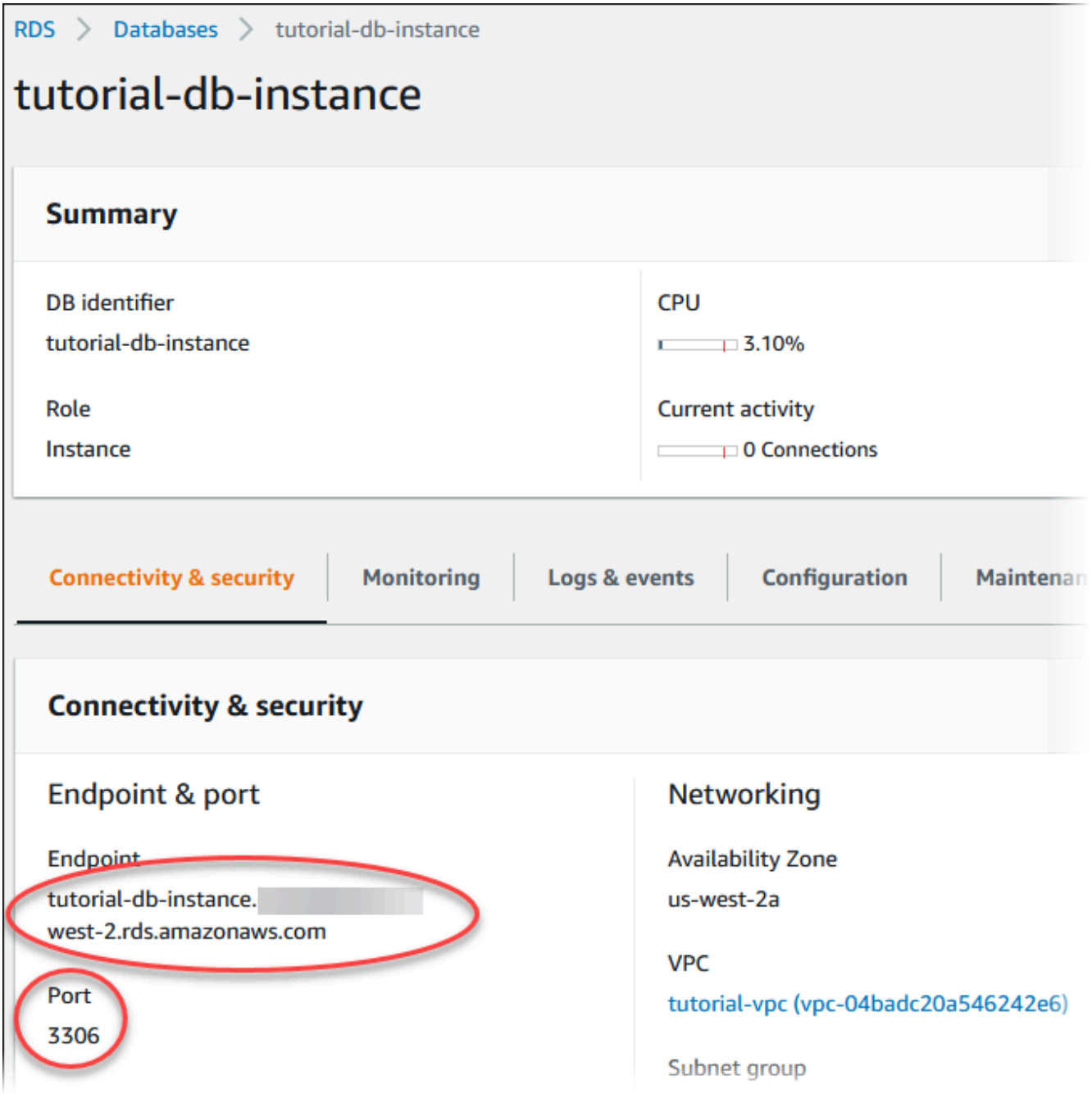


11. 스토리지 섹션에서 기본값으로 유지합니다.
12. 연결 섹션에서 다음 값을 설정하고 다른 값을 기본값으로 유지합니다.
    - 컴퓨팅 리소스에서 EC2 컴퓨팅 리소스에 연결을 선택합니다.
    - EC2 인스턴스의 경우 tutorial-ec2-instance-web-server와 같이 이전에 생성한 EC2 인스턴스를 선택합니다.



13. [데이터베이스 인증(Database authentication)] 섹션에서 [암호 인증(Password authentication)]이 선택되어 있는지 확인합니다.
14. 추가 구성 섹션을 열고 초기 데이터베이스 이름에 **`sample`**를 입력합니다. 다른 옵션은 기본 설정을 유지합니다.
15. MySQL DB 인스턴스를 생성하려면 [데이터베이스 생성(Create database)]을 선택합니다.

    새 DB 인스턴스가 데이터베이스목록에 생성 중의 상태로 나타납니다.
16. 새 DB 인스턴스의 상태가 사용 가능으로 나타날 때까지 기다립니다. 그런 다음, 세부 정보를 표시할 DB 인스턴스 이름을 선택합니다.
17. Connectivity & security(연결 및 보안) 섹션에서 DB 인스턴스의 엔드포인트 및 포트를 확인합니다.



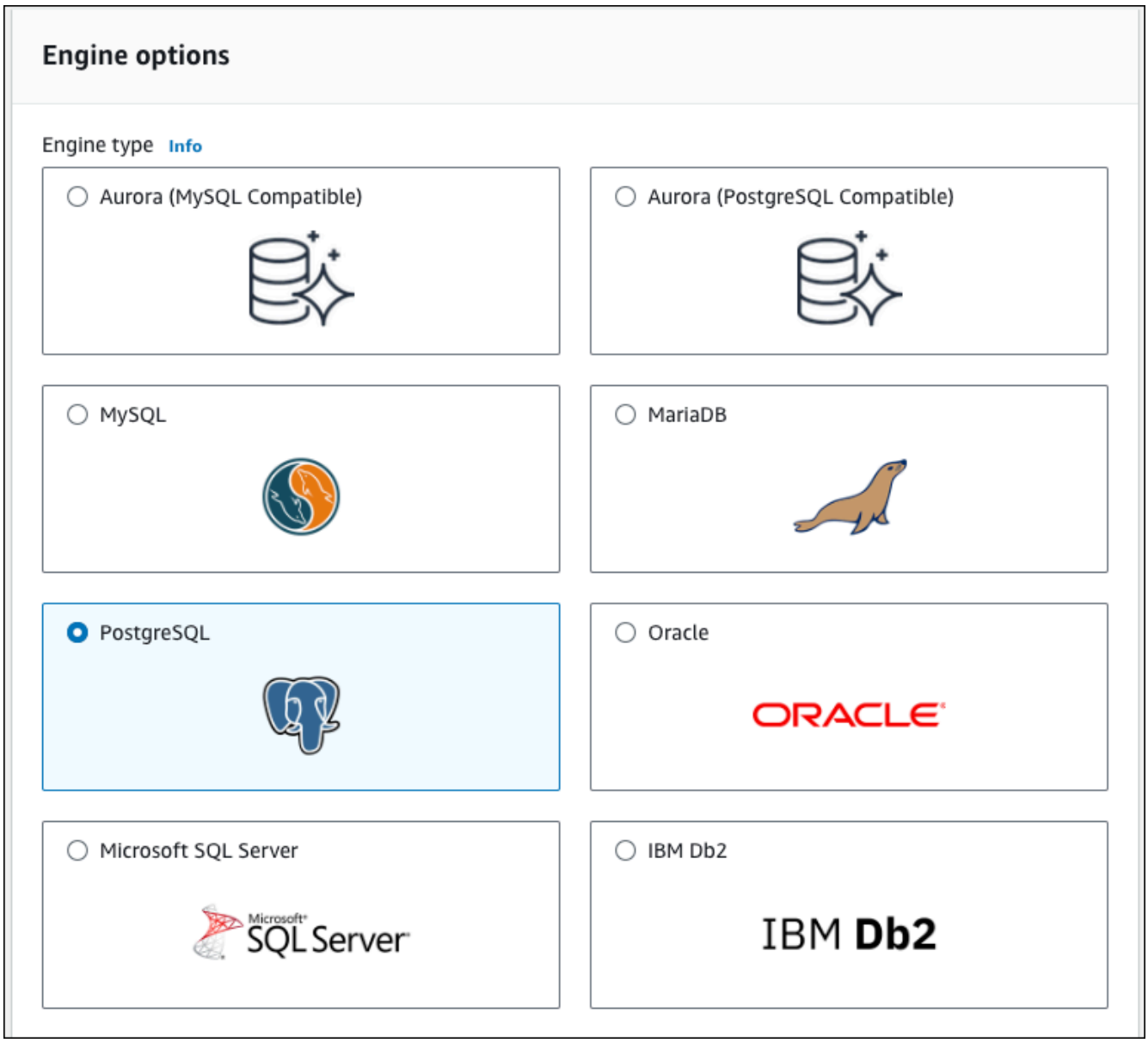
DB 인스턴스의 엔드포인트와 포트를 적어 둡니다. 이 정보를 사용하여 웹 서버를 RDS DB 인스턴스에 연결하게 됩니다.

18. EC2 인스턴스에 웹 서버 설치를 완료합니다.

RDS for PostgreSQL

PostgreSQL DB 인스턴스를 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. AWS Management Console의 오른쪽 상단 모서리에서 AWS 리전을 확인합니다. EC2 인스턴스를 생성한 리전과 동일해야 합니다.
3. 탐색 창에서 데이터베이스를 선택합니다.
4. 데이터베이스 생성을 선택합니다.
5. 데이터베이스 생성 페이지에서 표준 생성을 선택합니다.
6. 엔진 옵션에서 PostgreSQL을 선택합니다.

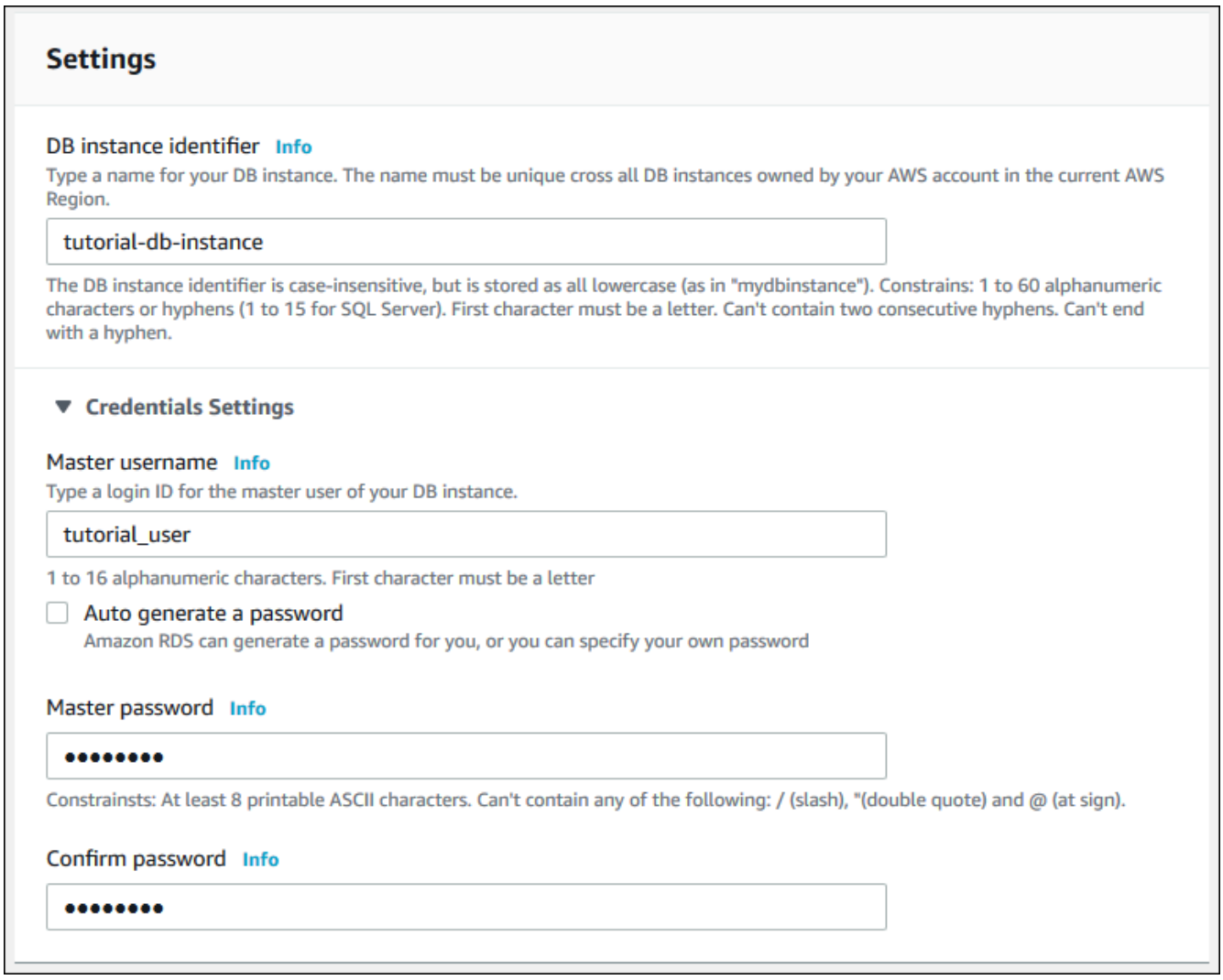


7. 템플릿에서 프리 티어 또는 샌드박스를 선택합니다. 프리 티어는 무료 요금제 계정에 표시됩니다. 샌드박스는 유료 요금제 계정에 표시됩니다.



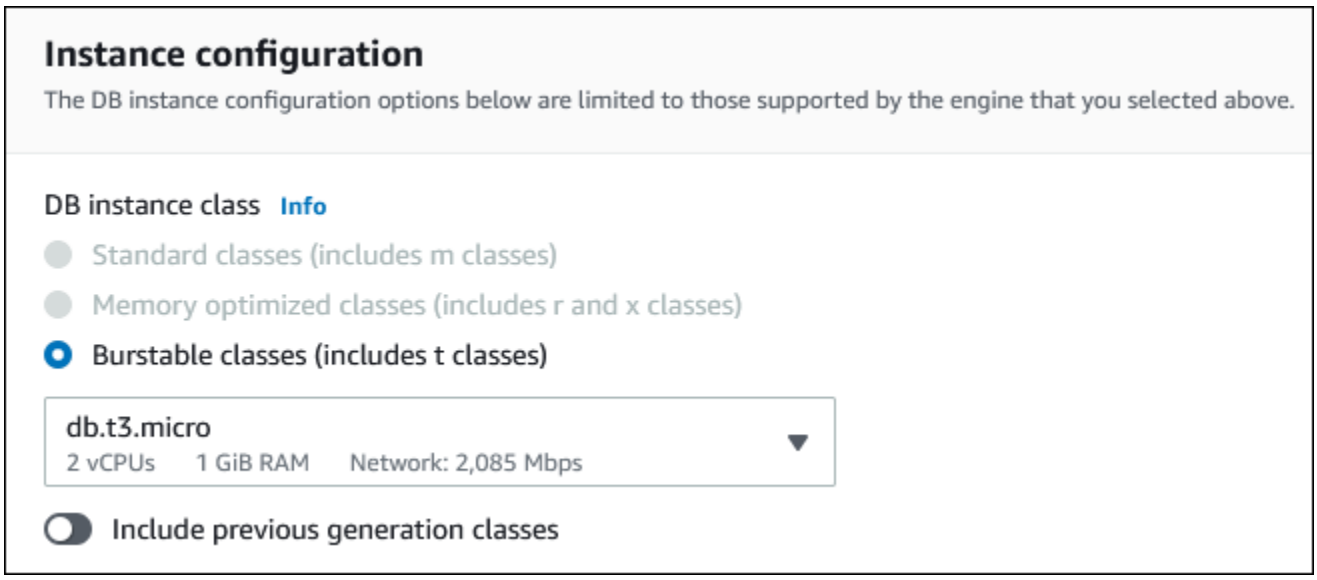
8. 가용성 및 내구성 섹션에서 기본값을 사용합니다.
9. 설정 섹션에서 이러한 값들을 설정합니다.

   - DB 인스턴스 식별자 – **`tutorial-db-instance`**를 입력합니다.
   - 마스터 사용자 이름 - **`tutorial_user`**를 입력합니다.
   - Auto generate a password(암호 자동 생성) - 옵션을 끈 상태로 둡니다.
   - 마스터 암호 – 암호를 입력합니다.
   - 암호 확인 – 암호를 다시 입력합니다.

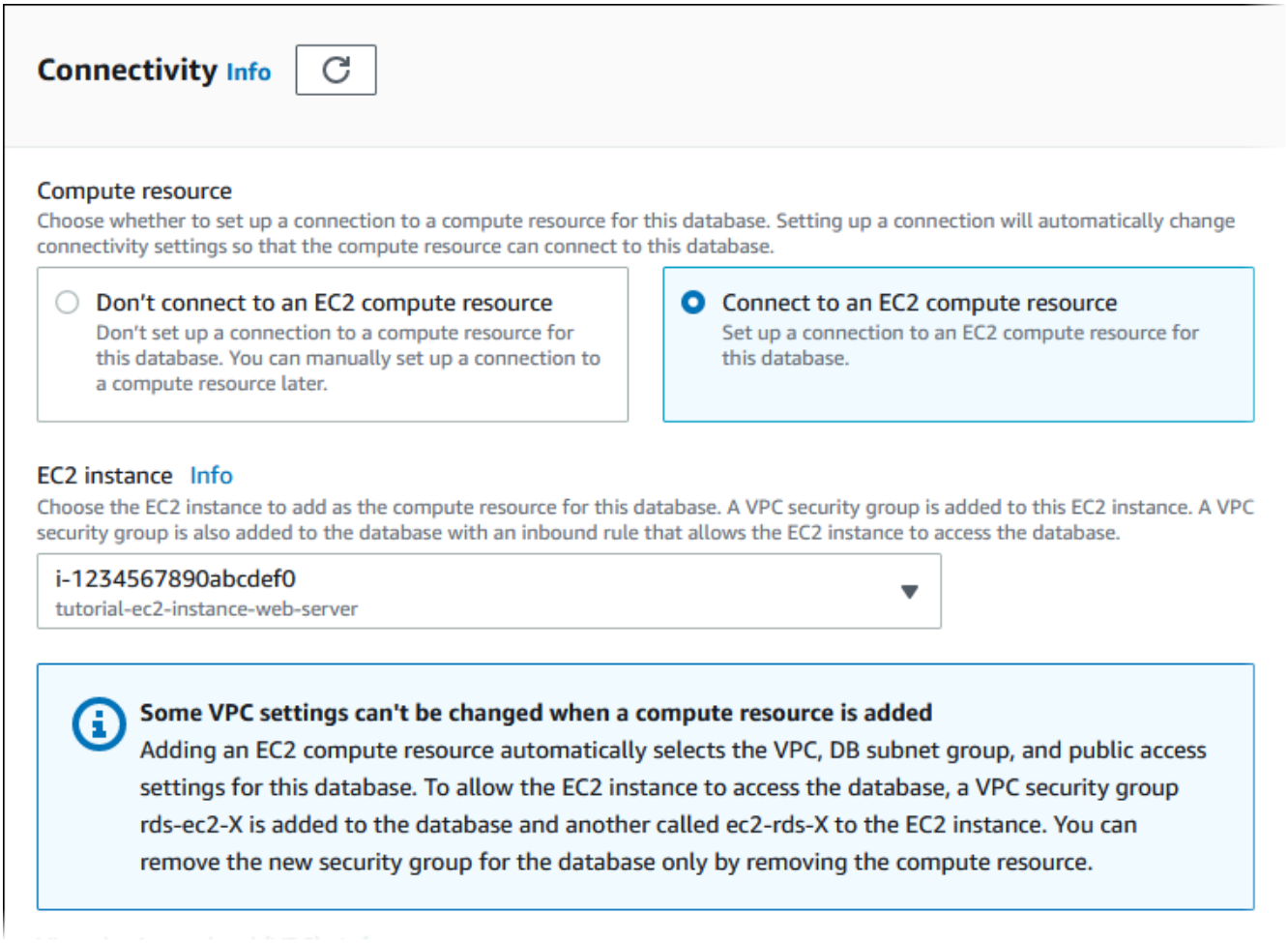


10. 인스턴스 구성] 섹션에서 다음 값을 설정합니다.

    - 버스트 가능 클래스(t 클래스 포함)
    - db.t3.micro

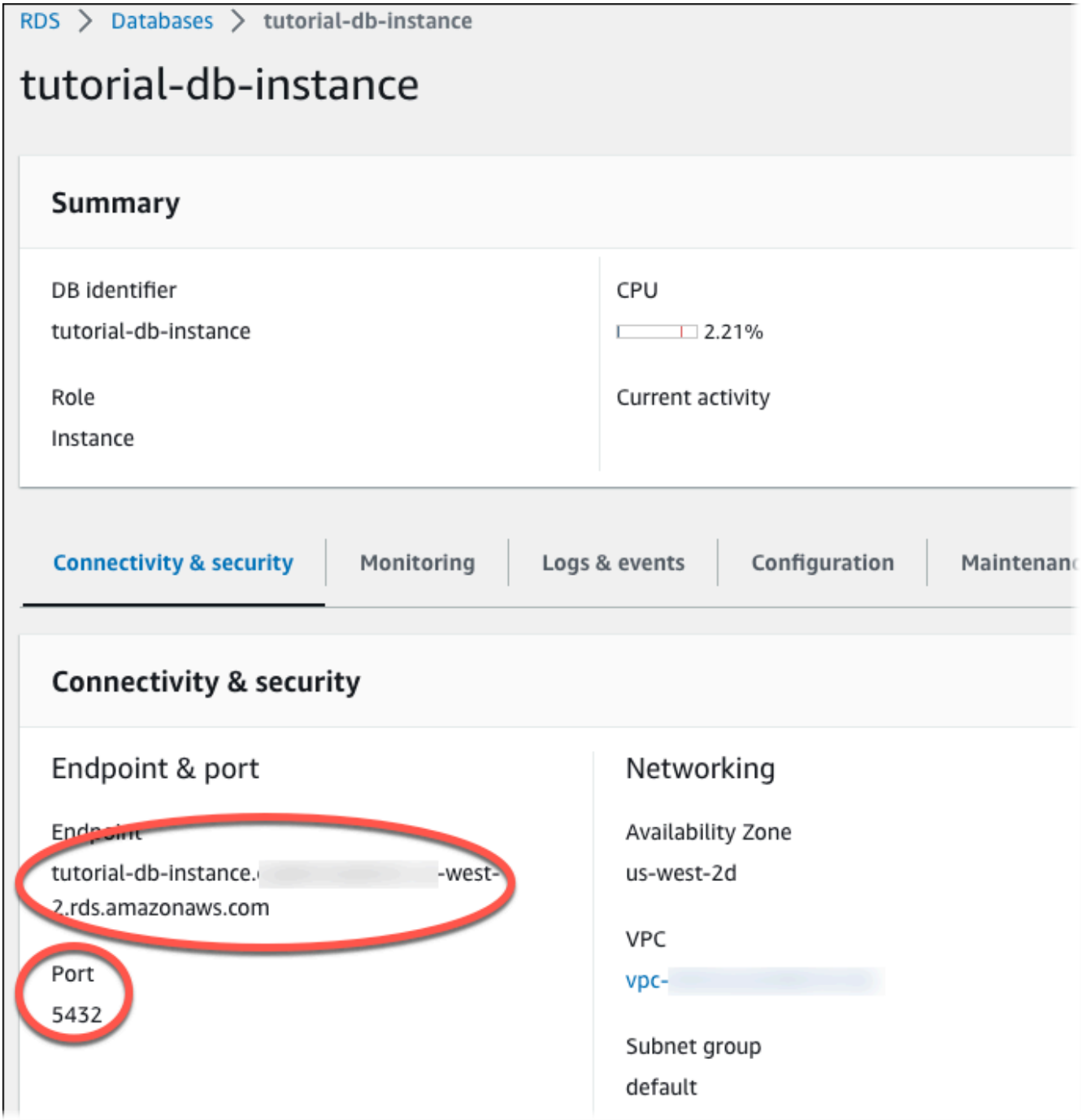


11. 스토리지 섹션에서 기본값으로 유지합니다.
12. 연결 섹션에서 다음 값을 설정하고 다른 값을 기본값으로 유지합니다.
    - 컴퓨팅 리소스에서 EC2 컴퓨팅 리소스에 연결을 선택합니다.
    - EC2 인스턴스의 경우 tutorial-ec2-instance-web-server와 같이 이전에 생성한 EC2 인스턴스를 선택합니다.



13. [데이터베이스 인증(Database authentication)] 섹션에서 [암호 인증(Password authentication)]이 선택되어 있는지 확인합니다.
14. 추가 구성 섹션을 열고 초기 데이터베이스 이름에 **`sample`**를 입력합니다. 다른 옵션은 기본 설정을 유지합니다.
15. PostgreSQL DB 인스턴스를 생성하려면 데이터베이스 생성을 선택합니다.

    새 DB 인스턴스가 데이터베이스목록에 생성 중의 상태로 나타납니다.
16. 새 DB 인스턴스의 상태가 사용 가능으로 나타날 때까지 기다립니다. 그런 다음, 세부 정보를 표시할 DB 인스턴스 이름을 선택합니다.
17. Connectivity & security(연결 및 보안) 섹션에서 DB 인스턴스의 엔드포인트 및 포트를 확인합니다.



DB 인스턴스의 엔드포인트와 포트를 적어 둡니다. 이 정보를 사용하여 웹 서버를 RDS DB 인스턴스에 연결하게 됩니다.

18. EC2 인스턴스에 웹 서버 설치를 완료합니다.

EC2 인스턴스에 웹 서버 설치

EC2 인스턴스를 시작하여 DB 인스턴스에 연결에서 생성한 EC2 인스턴스에 웹 서버를 설치합니다. 웹 서버는 Amazon RDS DB 인스턴스 생성에서 생성한 Amazon RDS DB 인스턴스에 연결됩니다.

PHP 및 MariaDB와 함께 Apache 웹 서버 설치

EC2 인스턴스에 연결하고 서버를 설치합니다.

EC2 인스턴스에 연결하고 PHP가 포함된 Apache 웹 서버를 설치하는 방법

1. Amazon EC2 사용 설명서에 있는 Linux 인스턴스에 연결의 단계를 따라 앞에서 만든 EC2 인스턴스에 연결합니다.

   SSH를 사용하여 EC2 인스턴스에 연결하는 것이 좋습니다. Windows, Linux 또는 Mac에 SSH 클라이언트 유틸리티가 설치된 경우 다음 명령 형식을 사용하여 인스턴스에 연결할 수 있습니다.

   ```
   ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
   ```

   예를 들어 `ec2-database-connect-key-pair.pem`이 Linux의 `/dir1`에 저장되어 있고, EC2 인스턴스의 퍼블릭 IPv4 DNS가 `ec2-12-345-678-90.compute-1.amazonaws.com`이라고 가정해 보겠습니다. SSH 명령은 다음과 같이 표시됩니다.

   ```
   ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
   user@ec2-12-345-678-90.compute-1.amazonaws.com
   ```

2. EC2 인스턴스에서 소프트웨어를 업데이트하여 최신 버그 수정 및 보안 업데이트를 받습니다. 이렇게 하려면 다음 명령을 사용하십시오.

   > **Note**
   >
   > `-y` 옵션을 사용하면 확인 여부를 묻지 않고 업데이트를 설치합니다. 설치 전에 업데이트 정보를 확인하려면 이 옵션을 생략합니다.

   ```
   sudo dnf update -y
   ```

3. 업데이트가 완료되면 다음 명령을 사용하여 Apache 웹 서버, PHP, MariaDB 또는 PostgreSQL 소프트웨어를 설치합니다. 이 명령은 여러 소프트웨어 패키지와 관련 종속 프로그램을 동시에 설치합니다.

   MariaDB & MySQL

   ```
   sudo dnf install -y httpd php php-mysqli mariadb105
   ```

   PostgreSQL

   ```
   sudo dnf install -y httpd php php-pgsql postgresql15
   ```

   오류가 발생하면 인스턴스가 Amazon Linux 2023 AMI로 실행되지 않은 것입니다. Amazon Linux 2 AMI를 사용하고 있는 것일 수 있습니다. 다음 명령을 사용하여 Amazon Linux 버전을 볼 수 있습니다.

   ```
   cat /etc/system-release
   ```

   자세한 내용은 인스턴스 소프트웨어 업데이트 단원을 참조하십시오.

4. 다음 명령을 사용하여 웹 서버를 시작합니다.

   ```
   sudo systemctl start httpd
   ```

   웹 서버가 제대로 설치되고 시작되었는지 테스트할 수 있습니다. 이렇게 하려면 웹 브라우저의 주소 표시줄에 EC2 인스턴스의 퍼블릭 Domain Name System(DNS) 이름을 입력합니다(예: `http://ec2-42-8-168-21.us-west-1.compute.amazonaws.com`). 웹 서버가 실행되고 있으면 Apache 테스트 페이지가 표시됩니다.

   Apache 테스트 페이지가 표시되지 않으면 자습서: DB 인스턴스에 사용할 Amazon VPC 생성(IPv4 전용)에서 생성한 VPC 보안 그룹에 대한 인바운드 규칙을 확인합니다. 인바운드 규칙에 웹 서버에 연결할 IP 주소에 대한 HTTP(포트 80) 액세스를 허용하는 규칙이 포함되어 있는지 확인합니다.

   > **Note**
   >
   > Apache 테스트 페이지는 문서의 루트 디렉터리 `/var/www/html`에 콘텐츠가 없는 경우에만 표시됩니다. 문서의 루트 디렉터리에 콘텐츠를 추가한 후에는 콘텐츠가 EC2 인스턴

스의 퍼블릭 DNS 주소에 나타납니다. 이 시점 이전에는 Apache 테스트 페이지에 나타납니다.

5. systemctl 명령을 사용하여 웹 서버가 시스템 부팅 때마다 시작되도록 구성합니다.

```
sudo systemctl enable httpd
```

ec2-user가 Apache 웹 서버의 기본 루트 디렉터리에 있는 파일을 관리할 수 있도록 하려면 /var/www 디렉터리의 소유권 및 권한을 변경합니다. 이 작업을 수행하는 방법에는 여러 가지가 있습니다. 본 자습서에서는 ec2-user를 apache 그룹에 추가하여 apache 그룹에 /var/www 디렉터리의 소유권을 부여하고 쓰기 권한을 할당합니다.

Apache 웹 서버에 대한 파일 권한 설정 방법

1. ec2-user 사용자를 apache 그룹에 추가합니다.

```
sudo usermod -a -G apache ec2-user
```

2. 권한을 새로 고치고 새 apache 그룹을 포함하려면 로그아웃합니다.

```
exit
```

3. 다시 로그인한 다음, apache 명령을 사용하여 groups 그룹이 있는지 확인합니다.

```
groups
```

출력 결과는 다음과 비슷합니다.

```
ec2-user adm wheel apache systemd-journal
```

4. /var/www 디렉터리 및 해당 콘텐츠의 그룹 소유권을 apache 그룹으로 변경합니다.

```
sudo chown -R ec2-user:apache /var/www
```

5. /var/www 및 그 하위 디렉터리의 디렉터리 권한을 변경해서 그룹 쓰기 권한을 추가하고 나중에 생성될 하위 디렉터리에서 그룹 ID를 설정합니다.

```
sudo chmod 2775 /var/www
```

```
find /var/www -type d -exec sudo chmod 2775 {} \;
```

6. /var/www 디렉터리 및 하위 디렉터리의 파일 권한을 계속 변경해서 그룹 쓰기 권한을 추가합니다.

```
find /var/www -type f -exec sudo chmod 0664 {} \;
```

이제 ec2-user(및 apache 그룹의 향후 멤버)는 Apache 문서 루트에서 파일을 추가, 삭제, 편집할 수 있습니다. 따라서 정적 웹 사이트 또는 PHP 애플리케이션과 같은 콘텐츠를 추가할 수 있습니다.

> **Note**
>
> HTTP 프로토콜을 실행하는 웹 서버는 송신하거나 수신하는 데이터에 대해 아무런 전송 보안 기능도 제공하지 않습니다. 웹 브라우저를 사용하여 HTTP 서버에 연결하면 네트워크 경로를 따라 어디서든 엿보려는 사람들이 많은 정보를 볼 수 있습니다. 이 정보에는 방문하는 URL, 수신하는 웹 페이지의 내용, HTML 양식의 내용(암호 포함)이 포함됩니다.
> 웹 서버를 안전하게 보호하기 위한 최선의 방법은 HTTPS(HTTP Secure) 지원 기능을 설치하는 것입니다. 이 프로토콜은 SSL/TLS 암호화로 데이터를 보호합니다. 자세한 내용은 Amazon EC2 사용 설명서에서 자습서: Amazon Linux AMI를 사용하여 SSL/TLS 구성을 참조하세요.

DB 인스턴스에 Apache 웹 서버 연결

이제 Amazon RDS DB 인스턴스에 연결되는 Apache 웹 서버에 콘텐츠를 추가합니다.

DB 인스턴스에 연결되는 Apache 웹 서버에 콘텐츠를 추가하는 방법

1. EC2 인스턴스에 계속 연결되어 있을 때 디렉터리를 /var/www로 변경하고 inc라는 새로운 하위 디렉터리를 생성합니다.

```
cd /var/www
mkdir inc
cd inc
```

2. inc라는 dbinfo.inc 디렉터리에서 새 파일을 생성한 다음 nano 또는 선택한 편집기를 호출하여 파일을 편집합니다.

```
>dbinfo.inc
nano dbinfo.inc
```

3. 다음 내용을 dbinfo.inc 파일에 추가합니다. 여기서 *db_instance_endpoint*는 DB 인스턴스에 대해 포트가 없는 DB 인스턴스 엔드포인트입니다.

> **Note**
>
> 웹 서버의 문서 루트에 속하지 않은 폴더에 사용자 이름과 암호 정보를 두는 것이 좋습니다. 이렇게 하면 보안 정보가 노출될 가능성이 줄어듭니다.
> 애플리케이션에서 적절한 암호로 master password를 변경해야 합니다.

```
<?php

define('DB_SERVER', 'db_instance_endpoint');
define('DB_USERNAME', 'tutorial_user');
define('DB_PASSWORD', 'master password');
define('DB_DATABASE', 'sample');
?>
```

4. dbinfo.inc 파일을 저장하고 닫습니다. nano를 사용하는 경우 Ctrl+S 및 Ctrl+X를 사용하여 파일을 저장하고 닫습니다.
5. 디렉터리를 /var/www/html로 변경합니다.

```
cd /var/www/html
```

6. html라는 SamplePage.php 디렉터리에서 새 파일을 생성한 다음 nano 또는 선택한 편집기를 호출하여 파일을 편집합니다.

```
>SamplePage.php
nano SamplePage.php
```

7. 다음 콘텐츠를 SamplePage.php 파일에 추가합니다.

   MariaDB & MySQL

```
<?php include "../inc/dbinfo.inc"; ?>
<html>
<body>
<h1>Sample page</h1>
<?php
```

```
  /* Connect to MySQL and select the database. */
  $connection = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD);

  if (mysqli_connect_errno()) echo "Failed to connect to MySQL: " .
 mysqli_connect_error();

  $database = mysqli_select_db($connection, DB_DATABASE);

  /* Ensure that the EMPLOYEES table exists. */
  VerifyEmployeesTable($connection, DB_DATABASE);

  /* If input fields are populated, add a row to the EMPLOYEES table. */
  $employee_name = htmlentities($_POST['NAME']);
  $employee_address = htmlentities($_POST['ADDRESS']);

  if (strlen($employee_name) || strlen($employee_address)) {
    AddEmployee($connection, $employee_name, $employee_address);
  }
?>

<!-- Input form -->
<form action="<?PHP echo $_SERVER['SCRIPT_NAME'] ?>" method="POST">
  <table border="0">
    <tr>
      <td>NAME</td>
      <td>ADDRESS</td>
    </tr>
    <tr>
      <td>
        <input type="text" name="NAME" maxlength="45" size="30" />
      </td>
      <td>
        <input type="text" name="ADDRESS" maxlength="90" size="60" />
      </td>
      <td>
        <input type="submit" value="Add Data" />
      </td>
    </tr>
  </table>
</form>

<!-- Display table data. -->
<table border="1" cellpadding="2" cellspacing="2">
  <tr>
```

```
    <td>ID</td>
    <td>NAME</td>
    <td>ADDRESS</td>
  </tr>

<?php

$result = mysqli_query($connection, "SELECT * FROM EMPLOYEES");

while($query_data = mysqli_fetch_row($result)) {
  echo "<tr>";
  echo "<td>",$query_data[0], "</td>",
       "<td>",$query_data[1], "</td>",
       "<td>",$query_data[2], "</td>";
  echo "</tr>";
}
?>

</table>

<!-- Clean up. -->
<?php

  mysqli_free_result($result);
  mysqli_close($connection);

?>

</body>
</html>


<?php

/* Add an employee to the table. */
function AddEmployee($connection, $name, $address) {
   $n = mysqli_real_escape_string($connection, $name);
   $a = mysqli_real_escape_string($connection, $address);

   $query = "INSERT INTO EMPLOYEES (NAME, ADDRESS) VALUES ('$n', '$a');";

   if(!mysqli_query($connection, $query)) echo("<p>Error adding employee data.</
p>");
}
```

```
/* Check whether the table exists and, if not, create it. */
function VerifyEmployeesTable($connection, $dbName) {
  if(!TableExists("EMPLOYEES", $connection, $dbName))
  {
     $query = "CREATE TABLE EMPLOYEES (
         ID int(11) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
         NAME VARCHAR(45),
         ADDRESS VARCHAR(90)
       )";

     if(!mysqli_query($connection, $query)) echo("<p>Error creating table.</
p>");
  }
}

/* Check for the existence of a table. */
function TableExists($tableName, $connection, $dbName) {
  $t = mysqli_real_escape_string($connection, $tableName);
  $d = mysqli_real_escape_string($connection, $dbName);

  $checktable = mysqli_query($connection,
      "SELECT TABLE_NAME FROM information_schema.TABLES WHERE TABLE_NAME = '$t'
 AND TABLE_SCHEMA = '$d'");

  if(mysqli_num_rows($checktable) > 0) return true;

  return false;
}
?>
```

PostgreSQL

```
<?php include "../inc/dbinfo.inc"; ?>

<html>
<body>
<h1>Sample page</h1>
<?php

/* Connect to PostgreSQL and select the database. */
```

```
$constring = "host=" . DB_SERVER . " dbname=" . DB_DATABASE . " user=" .
 DB_USERNAME . " password=" . DB_PASSWORD ;
$connection = pg_connect($constring);

if (!$connection){
 echo "Failed to connect to PostgreSQL";
 exit;
}

/* Ensure that the EMPLOYEES table exists. */
VerifyEmployeesTable($connection, DB_DATABASE);

/* If input fields are populated, add a row to the EMPLOYEES table. */
$employee_name = htmlentities($_POST['NAME']);
$employee_address = htmlentities($_POST['ADDRESS']);

if (strlen($employee_name) || strlen($employee_address)) {
  AddEmployee($connection, $employee_name, $employee_address);
}

?>

<!-- Input form -->
<form action="<?PHP echo $_SERVER['SCRIPT_NAME'] ?>" method="POST">
  <table border="0">
    <tr>
      <td>NAME</td>
      <td>ADDRESS</td>
    </tr>
    <tr>
      <td>
    <input type="text" name="NAME" maxlength="45" size="30" />
      </td>
      <td>
    <input type="text" name="ADDRESS" maxlength="90" size="60" />
      </td>
      <td>
    <input type="submit" value="Add Data" />
      </td>
    </tr>
  </table>
</form>
<!-- Display table data. -->
<table border="1" cellpadding="2" cellspacing="2">
```

```
  <tr>
    <td>ID</td>
    <td>NAME</td>
    <td>ADDRESS</td>
  </tr>

<?php

$result = pg_query($connection, "SELECT * FROM EMPLOYEES");

while($query_data = pg_fetch_row($result)) {
  echo "<tr>";
  echo "<td>",$query_data[0], "</td>",
       "<td>",$query_data[1], "</td>",
       "<td>",$query_data[2], "</td>";
  echo "</tr>";
}
?>
</table>

<!-- Clean up. -->
<?php

  pg_free_result($result);
  pg_close($connection);
?>
</body>
</html>


<?php

/* Add an employee to the table. */
function AddEmployee($connection, $name, $address) {
   $n = pg_escape_string($name);
   $a = pg_escape_string($address);
   echo "Forming Query";
   $query = "INSERT INTO EMPLOYEES (NAME, ADDRESS) VALUES ('$n', '$a');";

   if(!pg_query($connection, $query)) echo("<p>Error adding employee data.</
p>");
}

/* Check whether the table exists and, if not, create it. */
```

```
function VerifyEmployeesTable($connection, $dbName) {
  if(!TableExists("EMPLOYEES", $connection, $dbName))
  {
     $query = "CREATE TABLE EMPLOYEES (
         ID serial PRIMARY KEY,
         NAME VARCHAR(45),
         ADDRESS VARCHAR(90)
       )";

     if(!pg_query($connection, $query)) echo("<p>Error creating table.</p>");
  }
}
/* Check for the existence of a table. */
function TableExists($tableName, $connection, $dbName) {
  $t = strtolower(pg_escape_string($tableName)); //table name is case sensitive
  $d = pg_escape_string($dbName); //schema is 'public' instead of 'sample' db
 name so not using that

  $query = "SELECT TABLE_NAME FROM information_schema.TABLES WHERE TABLE_NAME =
 '$t';";
  $checktable = pg_query($connection, $query);

  if (pg_num_rows($checktable) >0) return true;
  return false;

}
?>
```

8. SamplePage.php 파일을 저장하고 닫습니다.
9. 웹 브라우저를 열고 http://*EC2 instance endpoint*/SamplePage.php(예: http://ec2-12-345-67-890.us-west-2.compute.amazonaws.com/SamplePage.php)를 검색하여 웹 서버에서 DB 인스턴스에 제대로 연결되는지 확인합니다.

SamplePage.php를 사용하여 DB 인스턴스에 데이터를 추가할 수 있습니다. 그러면 추가한 데이터가 페이지에 표시됩니다. 데이터가 테이블에 삽입되었는지 확인하려면 Amazon EC2 인스턴스에 MySQL을 설치합니다. 그런 다음 DB 인스턴스에 연결하여 테이블을 쿼리합니다.

MySQL 클라이언트를 설치하고 DB 인스턴스에 연결하는 방법에 대한 자세한 내용은 MySQL DB 인스턴스에 연결 섹션을 참조하세요.

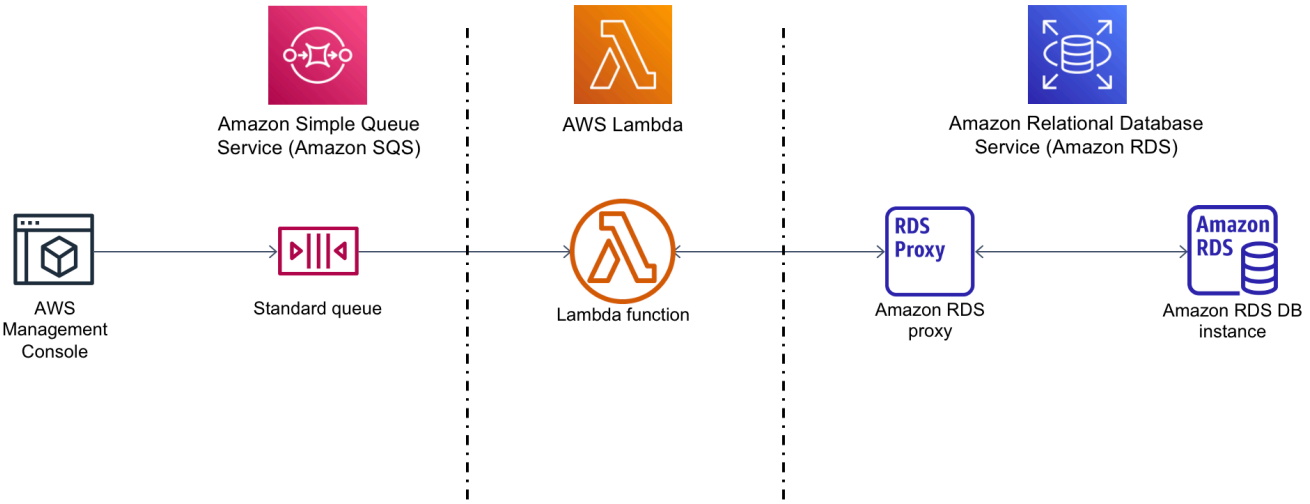
DB 인스턴스를 최대한 보호하려면 VPC 외부의 소스가 DB 인스턴스에 연결할 수 없는지 확인합니다.

웹 서버 및 데이터베이스 테스트를 완료한 후에는 DB 인스턴스와 Amazon EC2 인스턴스를 삭제해야 합니다.

- DB 인스턴스를 삭제하려면 DB 인스턴스 삭제의 지침을 따릅니다. 최종 스냅샷을 생성할 필요가 없습니다.
- Amazon EC2 인스턴스를 종료하려면 Amazon EC2 사용 설명서에서 인스턴스 종료 섹션의 지침을 따르세요.

자습서: Amazon RDS에 액세스하기 위해 Lambda 함수 사용

이 자습서에서는 Lambda 함수를 사용하여 RDS 프록시를 통해 Amazon Relational Database Service(Amazon RDS) 데이터베이스에 데이터를 기록합니다. Lambda 함수는 메시지가 추가될 때마다 Amazon Simple Queue Service(Amazon SQS) 대기열에서 레코드를 읽고 데이터베이스의 테이블에 새 항목을 기록합니다. 이 예제에서는 AWS Management Console을 사용하여 대기열에 메시지를 수동으로 추가합니다. 다음 다이어그램은 자습서를 완료하는 데 사용하는 AWS 리소스를 보여줍니다.



Amazon RDS를 사용하면 Microsoft SQL Server, MariaDB, MySQL, Oracle Database, PostgreSQL과 같은 일반적인 데이터베이스 제품을 사용하여 클라우드에서 관리형 관계형 데이터베이스를 실행할 수 있습니다. Lambda를 사용하여 데이터베이스에 액세스하면 새 고객이 웹 사이트에 등록하는 것과 같은 이벤트에 응답하여 데이터를 읽고 쓸 수 있습니다. 함수와 데이터베이스 인스턴스, 프록시는 수요가 많은 기간에 맞춰 자동으로 확장됩니다.

이 자습서를 완료하려면 다음 작업을 수행하세요.

1. RDS for MySQL 데이터베이스 인스턴스 및 프록시를 AWS 계정의 기본 VPC에 시작합니다.
2. 데이터베이스에 새 테이블을 생성하고 여기에 데이터를 기록하는 Lambda 함수를 생성 및 테스트합니다.
3. Amazon SQS 대기열을 생성하고 새 메시지가 추가될 때마다 Lambda 함수를 호출하도록 구성합니다.

4. AWS Management Console을 사용하여 대기열에 메시지를 추가하고 CloudWatch Logs를 사용하여 결과를 모니터링하면서 전체 설정을 테스트합니다.

이 단계를 완료하면 다음을 학습하게 됩니다.

- Amazon RDS를 사용하여 데이터베이스 인스턴스와 프록시를 생성하고 Lambda 함수를 프록시에 연결하는 방법
- Lambda를 사용하여 Amazon RDS 데이터베이스에서 생성 및 읽기 작업을 수행하는 방법
- Amazon SQS를 사용하여 Lambda 함수를 호출하는 방법

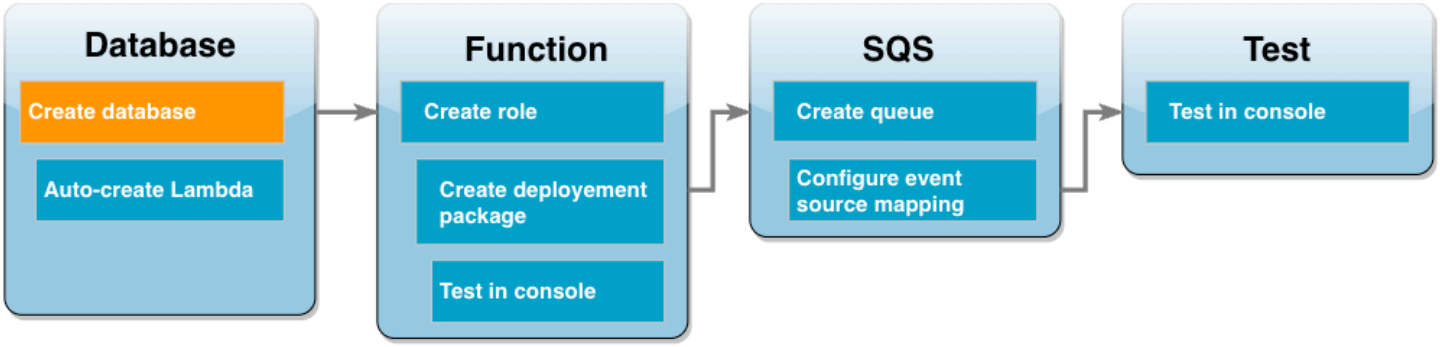
AWS Management Console 또는 AWS Command Line Interface(AWS CLI)를 사용하여 이 자습서를 완료할 수 있습니다.

사전 조건

시작하기 전에 다음 섹션에서 다음 단계를 완료하세요.

- AWS 계정에 등록
- 관리자 액세스 권한이 있는 사용자 생성

Amazon RDS DB 인스턴스 생성



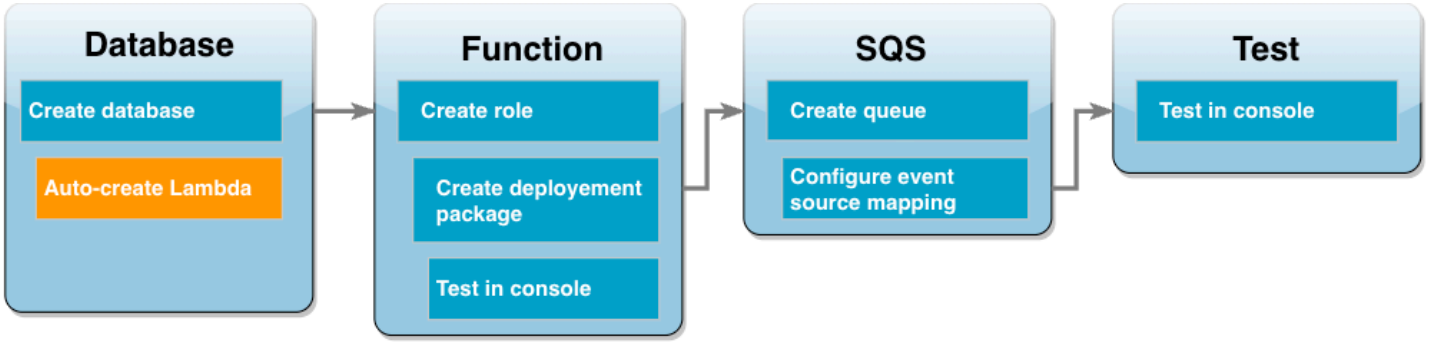
Amazon RDS DB 인스턴스는 AWS 클라우드에서 실행되는 격리된 데이터베이스 환경입니다. 인스턴스에 사용자가 만든 데이터베이스가 하나 이상 포함될 수 있습니다. 별도로 지정하지 않는 한 Amazon RDS는 AWS 계정에 포함된 기본 VPC에 새 데이터베이스 인스턴스를 생성합니다. Amazon VPC에 대한 자세한 내용은 Amazon Virtual Private Cloud 사용 설명서를 참조하세요.

이 자습서에서는 AWS 계정의 기본 VPC 새 인스턴스를 생성하고 해당 인스턴스에 이름이 `ExampleDB`인 데이터베이스를 생성합니다. AWS Management Console 또는 AWS CLI를 사용하여 DB 인스턴스 및 데이터베이스를 생성할 수 있습니다.

데이터베이스 인스턴스를 생성하는 방법

1. Amazon RDS 콘솔을 열고 데이터베이스 생성을 선택합니다.
2. 표준 생성 옵션을 선택한 상태로 두고 엔진 옵션에서 MySQL을 선택합니다.
3. 템플릿에서 프리 티어 또는 샌드박스를 선택합니다. 프리 티어는 프리 티어 계정에 표시됩니다. 샌드박스는 유료 요금제 계정에 표시됩니다.
4. Settings(설정)에서 DB instance identifier(DB 인스턴스 식별자)에 **`MySQLForLambda`**를 입력합니다.
5. 다음을 따라 사용자 이름과 암호를 설정합니다.
   a. 자격 증명 설정에서 마스터 사용자 이름을 `admin`으로 둡니다.
   b. 마스터 암호의 경우 데이터베이스에 액세스하기 위한 암호를 입력하고 확인합니다.
6. 다음을 수행하여 데이터베이스 이름을 지정합니다.
   - 나머지 기본 옵션을 모두 선택한 상태로 두고 아래로 스크롤하여 추가 구성 섹션으로 이동합니다.
   - 이 섹션을 확장하고 초기 데이터베이스 이름으로 **`ExampleDB`**를 입력합니다.
7. 나머지 기본 옵션을 모두 선택한 상태로 두고 데이터베이스 생성을 선택합니다.

Lambda 함수 및 프록시 생성



RDS 콘솔을 사용하여 데이터베이스와 동일한 VPC에 Lambda 함수와 프록시를 생성할 수 있습니다.

> **Note**
>
> 데이터베이스 생성이 완료되어 있으며 사용 가능 상태인 경우에만 이러한 관련 리소스를 생성할 수 있습니다.

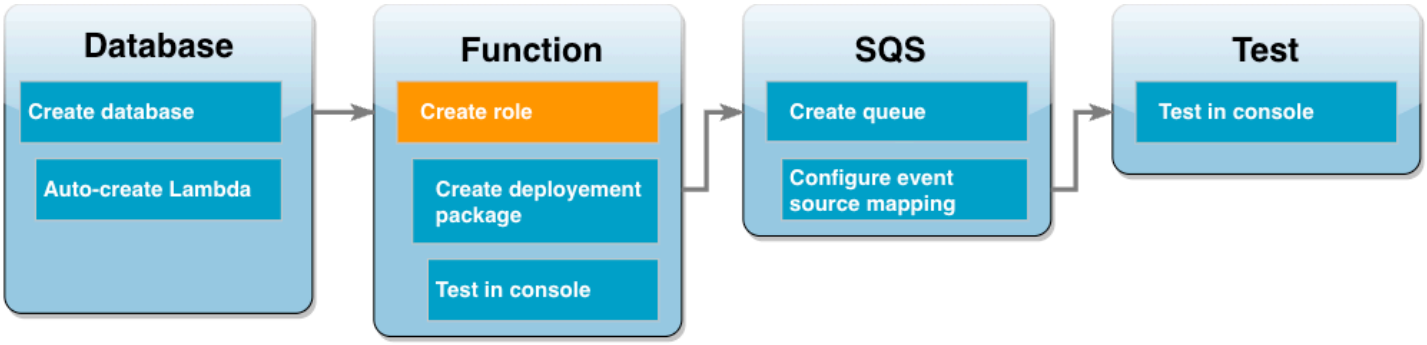
관련 함수 및 프록시를 생성하는 방법

1. 데이터베이스 페이지에서 데이터베이스가 사용 가능 상태인지 확인합니다. 사용 가능 상태이면 다음 단계를 진행합니다. 사용 가능 상태가 아니면 데이터베이스를 사용할 수 있을 때까지 기다립니다.
2. 데이터베이스를 선택하고 작업에서 Lambda 연결 설정을 선택합니다.
3. Lambda 연결 설정 페이지에서 새 함수 생성을 선택합니다.

   새 Lambda 함수 이름을 **`LambdaFunctionWithRDS`**로 설정합니다.
4. RDS 프록시 섹션에서 RDS 프록시를 사용하여 연결 옵션을 선택합니다. 그리고 새 프록시 생성을 선택합니다.
   - 데이터베이스 자격 증명에서 데이터베이스 사용자 이름 및 암호를 선택합니다.
   - 사용자 이름은 `admin`으로 지정합니다.
   - 암호에는 데이터베이스 인스턴스에 생성한 암호를 입력합니다.
5. 설정을 선택하여 프록시와 Lambda 함수 생성을 완료합니다.

마법사가 설정을 완료하고 새 함수를 검토할 수 있도록 Lambda 콘솔 링크를 제공합니다. Lambda 콘솔로 전환하기 전에 프록시 엔드포인트를 기록해 두세요.

함수 실행 역할 생성



Lambda 함수를 생성하기 전에 함수에 필요한 권한을 부여하는 실행 역할을 생성해야 합니다. 이 자습서에서는 데이터베이스 인스턴스가 포함된 VPC에 대한 네트워크 연결을 관리하고 Amazon SQS 대기열에서 메시지를 폴링할 수 있는 권한이 Lambda에 필요합니다.

Lambda 함수에 필요한 권한을 부여하기 위해 이 자습서에서는 IAM 관리형 정책을 사용합니다. 이러한 정책은 여러 가지 일반적인 사용 사례에서 권한을 부여하고 AWS 계정에서 제공됩니다. 관리형 정책에 대한 자세한 내용은 정책 모범 사례의 내용을 참조하세요.

Lambda 실행 역할 생성

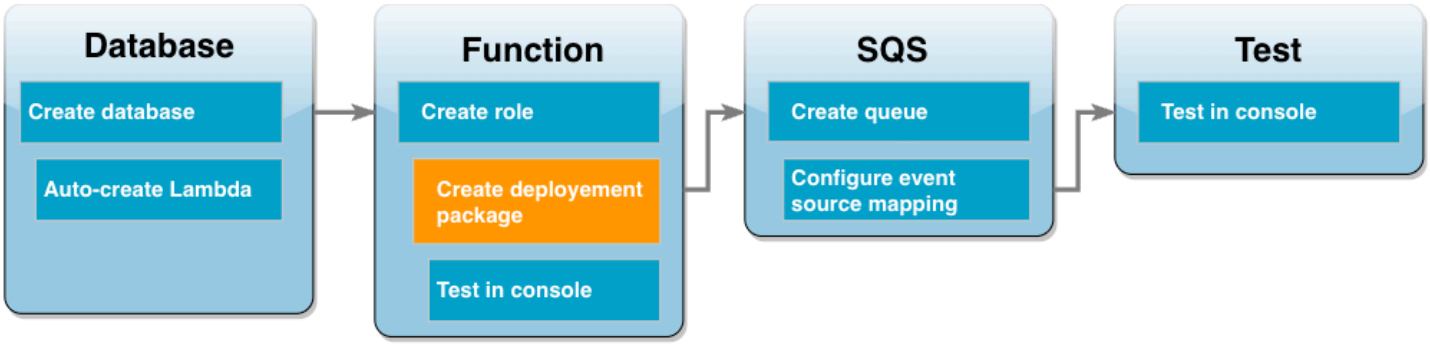
1. IAM 콘솔의 역할 페이지를 열고 Create role(역할 생성)을 선택합니다.
2. 신뢰할 수 있는 엔터티 유형으로 AWS 서비스를 선택한 다음 사용 사례로 Lambda를 선택합니다.
3. 다음을 선택합니다.
4. 다음을 수행하여 IAM 관리형 정책을 추가합니다.
   a. 정책 검색 상자를 사용하여 **AWSLambdaSQSQueueExecutionRole**을 검색합니다.
   b. 결과 목록에서 역할 옆의 확인란을 선택한 다음 필터 지우기를 선택합니다.
   c. 정책 검색 상자를 사용하여 **AWSLambdaVPCAccessExecutionRole**을 검색합니다.
   d. 결과 목록에서 역할 옆의 확인란을 선택한 다음 Next(다음)를 선택합니다.
5. Role name(역할 이름)에 **lambda-vpc-sqs-role**을 입력한 다음 Create role(역할 생성)을 선택합니다.

자습서 뒷부분에서는 방금 생성한 실행 역할의 Amazon 리소스 이름(ARN)이 필요합니다.

실행 역할 ARN 찾기

1. IAM 콘솔의 역할 페이지를 열고 역할(`lambda-vpc-sqs-role`)을 선택합니다.
2. 요약 섹션에 표시된 역할 ARN을 복사합니다.

Lambda 배포 패키지 생성



다음 예제 Python 코드는 PyMySQL 패키지를 사용하여 데이터베이스에 연결합니다. 함수를 처음 호출하면 `Customer`라는 새 테이블도 생성됩니다. 테이블은 다음 스키마를 사용하고, 여기서 `CustID`는 기본 키입니다.

```
Customer(CustID, Name)
```

또한 함수는 PyMySQL을 사용하여 이 테이블에 레코드를 추가합니다. 이 함수는 Amazon SQS 대기열에 추가할 메시지에 지정된 고객 ID 및 이름을 사용하여 레코드를 추가합니다.

이 코드는 핸들러 함수 외부에서 데이터베이스에 대한 연결을 생성합니다. 초기화 코드에서 연결을 생성하면 이후에 함수를 호출할 때 연결을 다시 사용할 수 있어 성능이 향상됩니다. 프로덕션 애플리케이션에서는 프로비저닝된 동시성을 사용하여 요청된 수의 데이터베이스 연결을 초기화할 수도 있습니다. 이러한 연결은 함수가 호출되는 즉시 사용할 수 있습니다.

```
import sys
import logging
import pymysql
import json
import os

# rds settings
user_name = os.environ['USER_NAME']
password = os.environ['PASSWORD']
rds_proxy_host = os.environ['RDS_PROXY_HOST']
db_name = os.environ['DB_NAME']

logger = logging.getLogger()
logger.setLevel(logging.INFO)

# create the database connection outside of the handler to allow connections to be
```

```
# re-used by subsequent function invocations.
try:
        conn = pymysql.connect(host=rds_proxy_host, user=user_name, passwd=password,
 db=db_name, connect_timeout=5)
except pymysql.MySQLError as e:
    logger.error("ERROR: Unexpected error: Could not connect to MySQL instance.")
    logger.error(e)
    sys.exit(1)

logger.info("SUCCESS: Connection to RDS for MySQL instance succeeded")

def lambda_handler(event, context):
    """
    This function creates a new RDS database table and writes records to it
    """
    message = event['Records'][0]['body']
    data = json.loads(message)
    CustID = data['CustID']
    Name = data['Name']

    item_count = 0
    sql_string = f"insert into Customer (CustID, Name) values(%s, %s)"

    with conn.cursor() as cur:
        cur.execute("create table if not exists Customer ( CustID  int NOT NULL, Name
 varchar(255) NOT NULL, PRIMARY KEY (CustID))")
        cur.execute(sql_string, (CustID, Name))
        conn.commit()
        cur.execute("select * from Customer")
        logger.info("The following items have been added to the database:")
        for row in cur:
            item_count += 1
            logger.info(row)
    conn.commit()

    return "Added %d items to RDS for MySQL table" %(item_count)
```

> **Note**
>
> 이 예제에서는 데이터베이스 액세스 자격 증명이 환경 변수로 저장됩니다. 프로덕션 애플리케이션에서는 보다 안전한 옵션으로 AWS Secrets Manager을 사용하는 것이 좋습니다. Lambda 함수가 VPC에 있는 경우 Secrets Manager에 연결하려면 VPC 엔드포인트를 생성해야 한다는

점을 참고하세요. 자세한 내용은 Virtual Private Cloud 내에서 Secrets Manager에 연결하는 방법을 참조하세요.

함수 코드에 PyMySQL 종속성을 포함하려면.zip 배포 패키지를 생성하세요. 다음 명령은 Linux, macOS 또는 Unix에서 작동합니다.

.zip 배포 패키지 생성

1. 예제 코드를 `lambda_function.py`라는 파일로 저장합니다.
2. `lambda_function.py` 파일을 생성한 디렉터리와 동일한 디렉터리에서 이름이 `package`인 새 디렉터리를 생성하고 PyMySQL 라이브러리를 설치합니다.

```
mkdir package
pip install --target package pymysql
```

3. 애플리케이션 코드와 PyMySQL 라이브러리가 포함된 zip 파일을 생성합니다. Linux 또는 MacOS에서는 다음 CLI 명령을 실행합니다. Windows에서는 선호하는 zip 도구를 사용하여 `lambda_function.zip` 파일을 생성합니다. `lambda_function.py` 원본 코드 파일과 종속 항목이 포함된 폴더는 .zip 파일의 루트에 설치해야 합니다.

```
cd package
zip -r ../lambda_function.zip .
cd ..
zip lambda_function.zip lambda_function.py
```

Python 가상 환경을 사용하여 배포 패키지를 생성할 수도 있습니다. .zip 파일 아카이브를 사용하여 Python Lambda 함수 배포를 참조하세요.

Lambda 함수 업데이트

방금 생성한 .zip 패키지를 사용하여 이제 Lambda 콘솔에서 Lambda 함수를 업데이트합니다. 함수가 데이터베이스에 액세스할 수 있도록 하려면 액세스 보안 인증으로 환경 변수도 구성해야 합니다.

Lambda 함수를 업데이트하는 방법

1. Lambda 콘솔의 함수 페이지를 열고 함수(`LambdaFunctionWithRDS`)를 선택합니다.
2. 런타임 설정 탭에서 편집을 선택하여 함수의 런타임을 Python 3.10으로 변경합니다.

3. 핸들러를 `lambda_function.lambda_handler`로 변경합니다.
4. 코드 탭에서 에서 업로드를 선택한 다음 .zip 파일을 선택합니다.
5. 이전 단계에서 생성한 `lambda_function.zip` 파일을 선택하고 저장을 선택합니다.

이제 앞서 생성한 실행 역할로 함수를 구성합니다. 이렇게 하면 함수에 데이터베이스 인스턴스에 액세스하고 Amazon SQS 대기열을 폴링하는 데 필요한 권한이 부여됩니다.

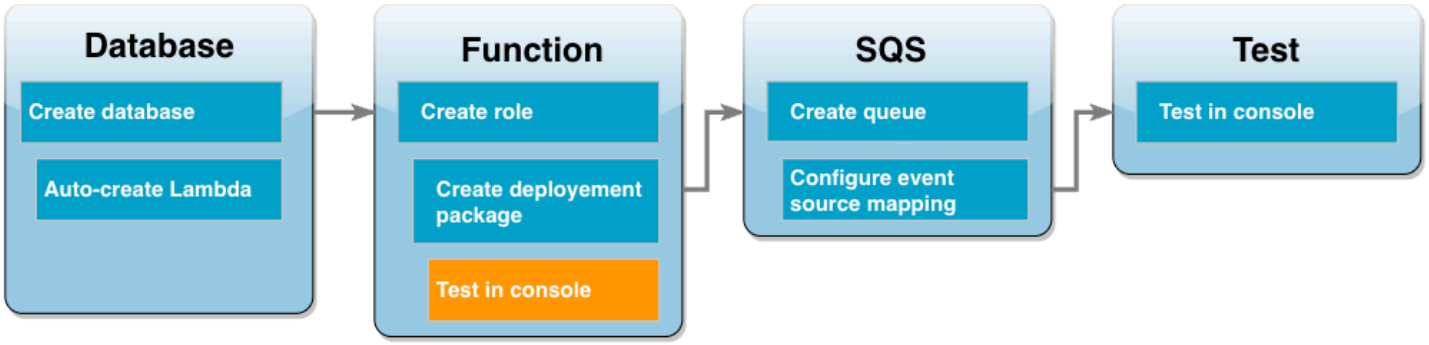
함수의 실행 역할을 구성하는 방법

1. Lambda 콘솔의 함수 페이지에서 구성 탭을 선택한 다음 권한을 선택합니다.
2. 실행 역할에서 편집을 선택합니다.
3. 기존 역할에서 실행 역할(`lambda-vpc-sqs-role`)을 선택합니다.
4. 저장을 선택합니다.

함수의 환경 변수를 구성하는 방법

1. Lambda 콘솔의 함수 페이지에서 구성 탭을 선택한 다음 환경 변수를 선택합니다.
2. 편집을 선택합니다.
3. 데이터베이스 액세스 자격 증명을 추가하려면 다음을 수행합니다.
    a. 환경 변수 추가를 선택한 다음 키에 **`USER_NAME`**을 입력하고 값에 **`admin`**을 선택합니다.
    b. 환경 변수 추가를 선택한 다음 키에 **`DB_NAME`**을 입력하고 값에 **`ExampleDB`**를 선택합니다.
    c. 환경 변수 추가를 선택한 다음 키에 **`PASSWORD`**를 입력하고 값에 데이터베이스를 생성할 때 선택한 암호를 입력합니다.
    d. 환경 변수 추가를 선택한 다음 키에 **`RDS_PROXY_HOST`**를 입력하고 값에 앞서 기록한 RDS 프록시 엔드포인트를 입력합니다.
    e. 저장을 선택합니다.

콘솔에서 Lambda 함수 테스트



이제 Lambda 콘솔을 사용하여 함수를 테스트할 수 있습니다. 자습서의 마지막 단계에서 Amazon SQS를 사용하여 함수를 호출할 때 함수가 수신할 데이터를 모방하는 테스트 이벤트를 생성합니다. 테스트 이벤트에는 함수가 생성하는 `Customer` 테이블에 추가할 고객 ID와 고객 이름을 지정하는 JSON 객체가 포함됩니다.

Lambda 함수를 테스트하려면

1. Lambda 콘솔의 함수 페이지를 열고 함수를 선택합니다.
2. 테스트 섹션을 선택합니다.
3. 새 이벤트 생성을 선택하고 이벤트 이름으로 **myTestEvent**를 입력합니다.
4. 다음 코드를 이벤트 JSON에 복사하고 저장을 선택합니다.

```
{
  "Records": [
    {
      "messageId": "059f36b4-87a3-44ab-83d2-661975830a7d",
      "receiptHandle": "AQEBwJnKyrHigUMZj6rYigCgxlaS3SLy0a...",
      "body": "{\n     \"CustID\": 1021,\n     \"Name\": \"Martha Rivera\"\n}",
      "attributes": {
        "ApproximateReceiveCount": "1",
        "SentTimestamp": "1545082649183",
        "SenderId": "AIDAIENQZJOLO23YVJ4VO",
        "ApproximateFirstReceiveTimestamp": "1545082649185"
      },
      "messageAttributes": {},
      "md5OfBody": "e4e68fb7bd0e697a0ae8f1bb342846b3",
      "eventSource": "aws:sqs",
      "eventSourceARN": "arn:aws:sqs:us-west-2:123456789012:my-queue",
      "awsRegion": "us-west-2"
    }
```

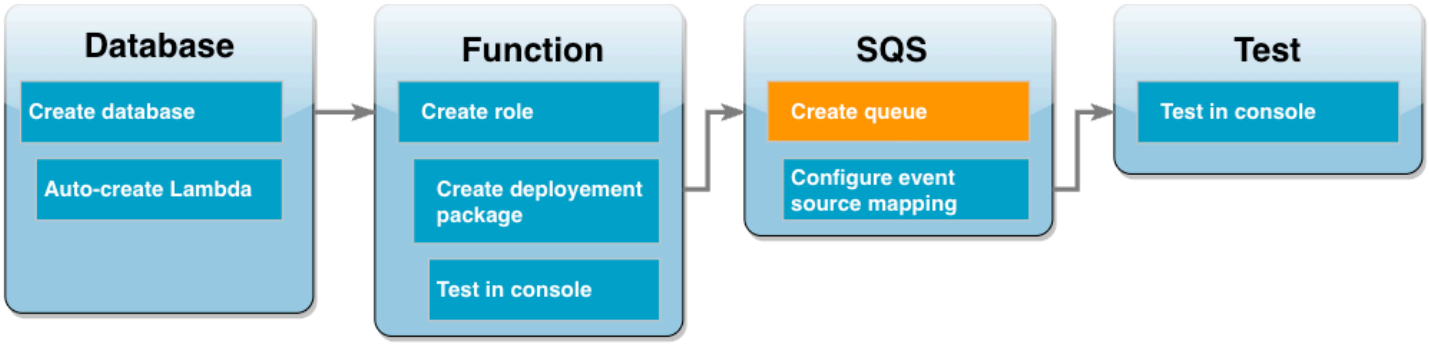
```
  ]
}
```

5. 테스트를 선택합니다.

실행 결과 탭의 함수 로그에 아래와 비슷한 결과가 표시될 것입니다.

```
[INFO] 2023-02-14T19:31:35.149Z bdd06682-00c7-4d6f-9abb-89f4bbb4a27f The following
 items have been added to the database:
[INFO] 2023-02-14T19:31:35.149Z bdd06682-00c7-4d6f-9abb-89f4bbb4a27f (1021, 'Martha
 Rivera')
```

Amazon SQS 대기열 생성

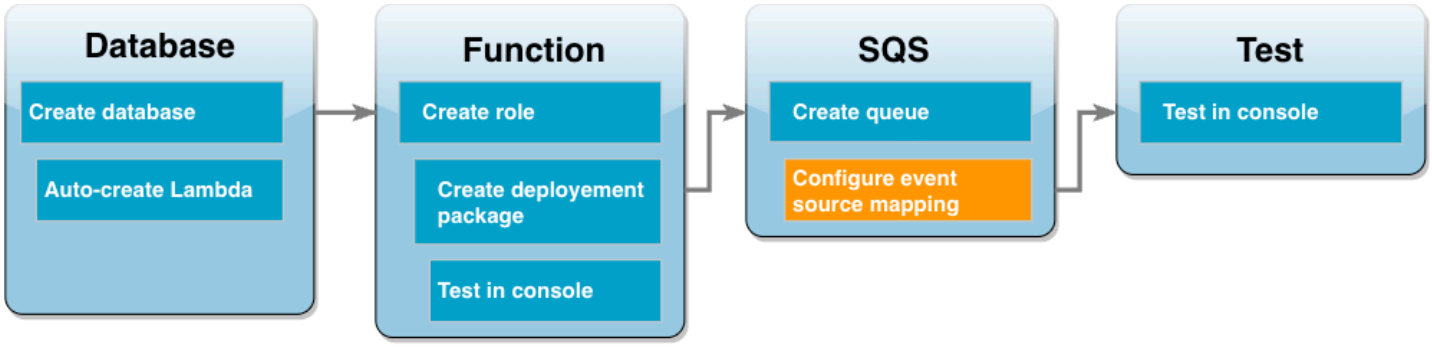


Lambda 함수와 Amazon RDS 데이터베이스 인스턴스의 통합을 성공적으로 테스트했습니다. 이제 자습서의 마지막 단계에서 Lambda 함수를 호출하는 데 사용할 Amazon SQS 대기열을 생성합니다.

Amazon SQS 대기열 생성(콘솔)

1. Amazon SQS 콘솔의 대기열 페이지를 열고 대기열 생성을 선택합니다.
2. 유형을 표준으로 두고 대기열 이름으로 **LambdaRDSQueue**를 입력합니다.
3. 기본 옵션을 모두 선택한 상태로 두고 Create queue(대기열 생성)를 선택합니다.

이벤트 소스 매핑을 생성하여 Lambda 함수 호출



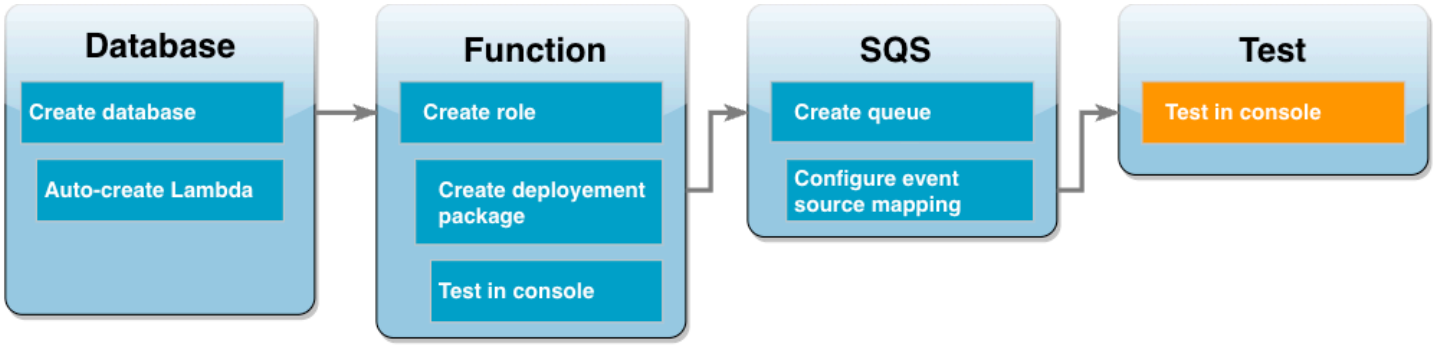
이벤트 소스 매핑은 스트림 또는 대기열에서 항목을 읽고 Lambda 함수를 호출하는 Lambda 리소스입니다. 이벤트 소스 매핑을 구성할 때 스트림 또는 대기열의 레코드가 단일 페이로드로 일괄 처리되도록 배치 크기를 지정할 수 있습니다. 이 예제에서는 대기열에 메시지를 보낼 때마다 Lambda 함수가 호출되도록 배치 크기를 1로 설정합니다. AWS CLI 또는 Lambda 콘솔을 사용하여 이벤트 소스 매핑을 구성할 수 있습니다.

이벤트 소스 매핑 생성(콘솔)

1. Lambda 콘솔의 함수 페이지를 열고 함수(`LambdaFunctionWithRDS`)를 선택합니다.
2. 함수 개요 섹션에서 트리거 추가를 선택합니다.
3. 소스로 Amazon SQS를 선택한 다음 대기열의 이름(`LambdaRDSQueue`)을 선택합니다.
4. 배치 크기에 **1**을 입력합니다.
5. 다른 모든 옵션은 기본값으로 두고 추가를 선택합니다.

이제 Amazon SQS 대기열에 메시지를 추가하여 전체 설정을 테스트할 준비가 되었습니다.

설정 테스트 및 모니터링



전체 설정을 테스트하려면 콘솔을 사용하여 Amazon SQS 대기열에 메시지를 추가합니다. 이후 CloudWatch Logs를 사용하여 Lambda 함수가 예상대로 데이터베이스에 레코드를 작성하고 있는지 확인합니다.

설정 테스트 및 모니터링

1. Amazon SQS 콘솔의 대기열 페이지를 열고 대기열(LambdaRDSQueue)을 선택합니다.
2. 메시지 전송 및 수신을 선택하고 다음 JSON을 메시지 전송 섹션의 메시지 본문에 붙여넣습니다.

```
{
    "CustID": 1054,
    "Name": "Richard Roe"
}
```

3. 메시지 전송을 선택합니다.

   메시지를 대기열로 보내면 Lambda가 이벤트 소스 매핑을 통해 함수를 호출합니다. Lambda에서 예상대로 함수를 호출했는지 확인하려면 CloudWatch Logs를 사용하여 함수가 고객 이름과 ID를 데이터베이스 테이블에 기록했는지 확인합니다.

4. CloudWatch 콘솔의 로그 그룹 페이지를 열고 함수(`/aws/lambda/LambdaFunctionWithRDS`)의 로그 그룹을 선택합니다.
5. 로그 스트림 섹션에서 가장 최근의 로그 스트림을 선택합니다.

   테이블에는 함수를 호출할 때마다 하나씩, 총 두 개의 고객 레코드가 포함되어야 합니다. 로그 스트림에 다음과 유사한 메시지가 표시됩니다.

```
[INFO] 2023-02-14T19:06:43.873Z 45368126-3eee-47f7-88ca-3086ae6d3a77 The following
 items have been added to the database:
[INFO] 2023-02-14T19:06:43.873Z 45368126-3eee-47f7-88ca-3086ae6d3a77 (1021, 'Martha
 Rivera')
[INFO] 2023-02-14T19:06:43.873Z 45368126-3eee-47f7-88ca-3086ae6d3a77 (1054,
 'Richard Roe')
```

리소스 정리

이 자습서 용도로 생성한 리소스를 보관하고 싶지 않다면 지금 삭제할 수 있습니다. 더 이상 사용하지 않는 AWS 리소스를 삭제하면 AWS 계정에 불필요한 요금이 발생하는 것을 방지할 수 있습니다.

Lambda 함수를 삭제하려면

1. Lambda 콘솔의 함수 페이지를 엽니다.
2. 생성한 함수를 선택합니다.
3. 작업, 삭제를 선택합니다.
4. 삭제를 선택합니다.

실행 역할을 삭제하려면

1. IAM 콘솔에서 역할 페이지를 엽니다.
2. 생성한 실행 역할을 선택합니다.
3. 역할 삭제(Delete role)를 선택합니다.
4. 예, 삭제(Yes, delete)를 선택합니다.

MySQL DB 인스턴스를 삭제하려면

1. Amazon RDS 콘솔의 데이터베이스 페이지를 엽니다.
2. 생성한 데이터베이스를 선택합니다.
3. 작업, 삭제를 선택합니다.
4. 최종 스냅샷 생성 확인란을 선택 해제합니다.
5. 텍스트 상자에 **`delete me`**를 입력합니다.
6. 삭제를 선택합니다.

Amazon SQS 대기열을 삭제하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/sqs/에서 Amazon SQS 콘솔을 엽니다.
2. 생성한 대기열을 선택합니다.
3. 삭제를 선택합니다.
4. 텍스트 상자에 **`delete`**를 입력합니다.
5. 삭제를 선택합니다.

Amazon RDS 자습서 및 샘플 코드

AWS 문서에는 일반적인 Amazon RDS 사용 사례를 안내하는 몇 가지 자습서가 포함되어 있습니다. 이들 자습서 중에는 Amazon RDS를 다른 AWS 서비스와 함께 사용하는 방법을 설명하는 자습서가 많습니다. 또한 GitHub에서 샘플 코드에 액세스할 수 있습니다.

> **Note**
>
> AWS 데이터베이스 블로그에서 더 많은 자습서를 찾을 수 있습니다. 교육에 대한 자세한 내용은 AWS Training and Certification을 참조하세요.

주제

- 이 안내서의 자습서
- 다른 AWS 안내서의 자습서
- Amazon RDS PostgreSQL에 대한 AWS 워크숍 및 랩 콘텐츠 포털
- Amazon RDS MySQL에 대한 AWS 워크숍 및 랩 콘텐츠 포털
- GitHub의 자습서 및 샘플 코드
- AWS SDK와 함께 이 서비스 사용

이 안내서의 자습서

이 안내서에 포함된 다음 자습서는 Amazon RDS의 일반적인 작업을 수행하는 방법을 설명합니다.

- 자습서: DB 인스턴스에 사용할 Amazon VPC 생성(IPv4 전용)

  Amazon VPC 서비스를 기반으로 하는 Virtual Private Cloud(VPC)에 DB 인스턴스를 포함하는 방법을 알아봅니다. 이 경우 VPC는 동일한 VPC의 Amazon EC2에서 실행 중인 웹 서버와 데이터를 공유합니다.
- 자습서: DB 인스턴스(듀얼 스택 모드)에 사용할 VPC 생성

  Amazon VPC 서비스를 기반으로 하는 Virtual Private Cloud(VPC)에 DB 인스턴스를 포함하는 방법을 알아봅니다. 이 경우 VPC는 동일한 VPC의 Amazon EC2와 데이터를 공유합니다. 이 자습서에서는 이중 스택 모드에서 실행되는 데이터베이스와 함께 작동하는 이 시나리오의 VPC를 생성합니다.

- 자습서: 웹 서버 및 Amazon RDS DB 인스턴스 생성

  PHP가 있는 Apache 웹 서버를 설치하고 MySQL 데이터베이스를 생성하는 방법을 알아봅니다. 이 웹 서버는 Amazon Linux를 사용하여 Amazon EC2 인스턴스에서 실행되며, MySQL 데이터베이스는 MySQL DB 인스턴스입니다. Amazon EC2 인스턴스 및 DB 인스턴스가 모두 Amazon VPC에서 실행됩니다.
- 자습서: DB 스냅샷에서 Amazon RDS DB 인스턴스 복원

  DB 스냅샷에서 DB 인스턴스를 복원하는 방법을 알아봅니다.
- 자습서: Amazon RDS에 액세스하기 위해 Lambda 함수 사용

  RDS 콘솔에서 프록시를 통해 데이터베이스에 액세스하기 위해 Lambda 함수를 생성하고, 테이블을 생성한 다음, 몇 가지 레코드를 추가하고, 테이블에서 레코드를 검색하는 방법을 알아봅니다. Lambda 함수를 호출하고 쿼리 결과를 확인하는 방법도 배웁니다.
- 자습서: 태그를 사용하여 중지할 DB 인스턴스를 지정합니다.

  태그를 사용하여 중지할 DB 인스턴스를 지정하는 방법을 알아봅니다.
- 자습서: Amazon EventBridge를 사용하여 DB 인스턴스의 상태 변경 로깅

  Amazon EventBridge 및 AWS Lambda를 사용하여 DB 인스턴스 상태 변경을 로깅하는 방법을 알아봅니다.
- 자습서: Amazon RDS에 대한 다중 AZ DB 클러스터 복제본 지연에 대한 Amazon CloudWatch 경보 생성

  다중 AZ DB 클러스터의 복제본 지연이 임계값을 초과하면 Amazon SNS 메시지를 보내는 CloudWatch 경보를 만드는 방법을 알아봅니다. 경보는 지정한 기간 동안 `ReplicaLag` 지표를 감시합니다. 이 작업은 Amazon SNS 주제나 Amazon EC2 Auto Scaling 정책으로 전송되는 알림입니다.

다른 AWS 안내서의 자습서

다른 AWS 안내서에 포함된 다음 자습서는 Amazon RDS의 태스크를 수행하는 방법을 설명합니다.

- AWS Secrets Manager 사용 설명서의 자습서: AWS 데이터베이스에 대한 암호 교체

  AWS 데이터베이스에 대한 보안 암호를 생성하여 일정에 따라 교체하도록 구성하는 방법을 알아봅니다. 교체를 수동으로 한 번 트리거한 후 보안 암호의 새 버전으로 계속해서 액세스할 수 있는지 확인합니다.
- AWS Elastic Beanstalk 개발자 안내서의 자습서 및 샘플

AWS Elastic Beanstalk와 함께 Amazon RDS 데이터베이스를 사용하는 애플리케이션을 배포하는 방법을 알아봅니다.

- Amazon Machine Learning Developer Guide의 Amazon RDS 데이터베이스의 데이터를 사용하여 Amazon ML 데이터 원본 생성

  MySQL DB 인스턴스에 저장된 데이터로 Amazon Machine Learning(Amazon ML) 데이터 원본 객체를 생성하는 방법을 알아봅니다.

- Amazon QuickSight 사용 설명서의 수동으로 VPC의 Amazon RDS 인스턴스에 대한 액세스 허용

  VPC의 Amazon RDS DB 인스턴스에 대한 QuickSight 액세스를 활성화하는 방법을 알아봅니다.

Amazon RDS PostgreSQL에 대한 AWS 워크숍 및 랩 콘텐츠 포털

다음의 워크숍 및 기타 실습 콘텐츠 모음은 Amazon RDS PostgreSQL의 특성과 기능을 이해하는 데 도움이 됩니다.

- DB 인스턴스 생성

  DB 인스턴스를 생성하는 방법을 알아봅니다.

- RDS 도구를 사용한 성능 모니터링

  AWS 및 SQL 도구(Cloudwatch, 고급 모니터링, 느린 쿼리 로그, 성능 개선 도우미, PostgreSQL 카탈로그 보기)를 사용하여 성능 문제를 이해하고, 데이터베이스의 성능을 개선할 수 있는 방법을 알아봅니다.

Amazon RDS MySQL에 대한 AWS 워크숍 및 랩 콘텐츠 포털

다음의 워크숍 및 기타 실습 콘텐츠 모음은 Amazon RDS MySQL의 특성과 기능을 이해하는 데 도움이 됩니다.

- DB 인스턴스 생성

  DB 인스턴스를 생성하는 방법을 알아봅니다.

- 성능 개선 도우미

  성능 개선 도우미를 사용하여 DB 인스턴스를 모니터링하고 튜닝하는 방법을 알아봅니다.

GitHub의 자습서 및 샘플 코드

GitHub의 다음 자습서와 샘플 코드는 Amazon RDS에서 일반적인 태스크를 수행하는 방법을 설명합니다.

- Amazon Relational Database Service 항목 추적기 생성

  작업 항목을 추적하고 보고하는 애플리케이션을 만드는 방법을 알아봅니다. 이 애플리케이션은 Amazon RDS, Amazon Simple Email Service, Elastic Beanstalk 및 SDK for Java 2.x를 사용합니다.

AWS SDK와 함께 이 서비스 사용

다양한 프로그래밍 언어에 대해 AWS 소프트웨어 개발 키트(SDK)을 사용할 수 있습니다. 각 SDK는 개발자가 선호하는 언어로 애플리케이션을 쉽게 구축할 수 있도록 하는 API, 코드 예시 및 설명서를 제공합니다.

SDK 설명서	코드 예시
AWS SDK for C++	AWS SDK for C++ 코드 예시
AWS CLI	AWS CLI 코드 예시
AWS SDK for Go	AWS SDK for Go 코드 예시
AWS SDK for Java	AWS SDK for Java 코드 예시
AWS SDK for JavaScript	AWS SDK for JavaScript 코드 예시
AWS SDK for Kotlin	AWS SDK for Kotlin 코드 예시
AWS SDK for .NET	AWS SDK for .NET 코드 예시
AWS SDK for PHP	AWS SDK for PHP 코드 예시
AWS Tools for PowerShell	AWS Tools for PowerShell 코드 예시
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 코드 예시
AWS SDK for Ruby	AWS SDK for Ruby 코드 예시

SDK 설명서	코드 예시
AWS SDK for Rust	AWS SDK for Rust 코드 예시
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP 코드 예시
AWS SDK for Swift	AWS SDK for Swift 코드 예시

이 서비스 관련 예시는 AWS SDK를 사용한 Amazon RDS용 코드 예제를 참조하세요.

> **예제 사용 가능 여부**
>
> 필요한 예제를 찾을 수 없습니까? 이 페이지 하단의 피드백 제공 링크를 사용하여 코드 예시를 요청하세요.

Amazon RDS의 모범 사례

Amazon RDS 작업 모범 사례에 대해서 알아봅니다. 새로운 모범 사례가 확인되는 대로 이 섹션을 업데이트할 예정입니다.

주제

- Amazon RDS 기본 운영 지침
- DB 인스턴스 RAM 권장 사항
- 데이터베이스 엔진 버전을 최신 상태로 유지
- AWS 데이터베이스 드라이버
- Enhanced Monitoring을 통한 운영 체제 문제 식별
- 지표를 통해 성능 문제 식별
- 쿼리 튜닝
- MySQL 작업 모범 사례
- MariaDB 작업 모범 사례
- Oracle 작업의 모범 사례
- PostgreSQL로 작업하기 위한 모범 사례
- SQL Server로 작업하기 위한 모범 사례
- DB 파라미터 그룹 작업
- DB 인스턴스 생성 자동화 모범 사례
- Amazon RDS의 새로운 비디오 특성

> **Note**
>
> Amazon RDS에 대한 일반적인 권장 사항은 Amazon RDS의 권장 사항 단원을 참조하십시오.

Amazon RDS 기본 운영 지침

다음은 Amazon RDS로 작업할 때 모든 사용자가 따라야 하는 기본 운영 지침입니다. Amazon RDS 서비스 수준 계약에 다음 지침을 따르도록 명시되어 있습니다.

- 지표를 사용하여 메모리, CPU, 복제본 지연 및 스토리지 사용량을 모니터링합니다. 사용 패턴이 변경되거나 배포 용량 한도에 거의 도달했을 때 알림을 받도록 Amazon CloudWatch를 설정하여 시스템 성능 및 가용성을 유지할 수 있습니다.
- 스토리지 용량 한도에 도달할 경우 DB 인스턴스를 확장합니다. 스토리지 및 메모리에 어느 정도 버퍼가 있어야만 애플리케이션에서 수요가 예기치 않게 늘어날 경우 이를 수용할 수 있습니다.
- 자동 백업을 활성화하고 일일 쓰기 IOPS가 낮은 동안 백업 창이 열리도록 설정합니다. 이때 백업이 데이터베이스 사용에 최소한의 영향을 미칩니다.
- 데이터베이스 작업량으로 인해 프로비저닝한 I/O보다 많이 필요할 경우 장애 조치 또는 데이터베이스 오류가 발생한 후에 복구 속도가 느려집니다. DB 인스턴스의 I/O 용량을 늘리려면 다음 중 일부 항목이나 모든 항목을 수행하십시오.
  - I/O 용량이 높은 다른 DB 인스턴스 클래스로 마이그레이션합니다.
  - 필요한 증분 양에 따라 마그네틱 스토리지를 범용 또는 프로비저닝된 IOPS 스토리지로 변환합니다. 사용 가능한 스토리지 유형에 대한 자세한 내용은 Amazon RDS 스토리지 유형 단원을 참조하십시오.

    프로비저닝된 IOPS 스토리지로 변환할 경우 프로비저닝된 IOPS에 최적화된 DB 인스턴스 클래스를 사용해야 합니다. 프로비저닝된 IOPS에 대한 자세한 내용은 프로비저닝된 IOPS SSD 스토리지 단원을 참조하십시오.
  - 이미 프로비저닝된 IOPS 스토리지를 사용하고 있는 경우 추가 처리 용량을 프로비저닝합니다.
- 클라이언트 애플리케이션이 DB 인스턴스의 DNS(Domain Name Service) 데이터를 캐시하는 경우 TTL(Time-to-Live) 값을 30초 미만으로 설정합니다. 장애 조치 후에 DB 인스턴스의 기본 IP 주소가 변경될 수 있습니다. DNS 데이터를 장기간 캐시하면 연결이 실패할 수 있습니다. 애플리케이션이 더 이상 사용되지 않는 IP 주소에 연결을 시도할 수 있습니다.
- DB 인스턴스의 장애 조치를 테스트하여 특정 사용 사례 절차에 소요되는 시간을 파악하세요. 또한 장애 조치를 테스트하여 DB 인스턴스에 액세스하는 애플리케이션이 장애 조치 후 자동으로 새 DB 인스턴스에 연결되는지 확인하세요.

DB 인스턴스 RAM 권장 사항

작업 집합이 거의 완전히 메모리에 상주하도록 RAM을 충분히 할당하는 것이 Amazon RDS 성능 모범 사례에 따르는 길입니다. 작업 집합은 인스턴스에서 자주 사용되는 데이터 및 인덱스입니다. DB 인스턴스를 많이 사용할수록 작업 집합이 커집니다.

작업 집합이 거의 전부 메모리에 있는지 확인하려면 (Amazon CloudWatch를 사용하여) DB 인스턴스에 작업 부하가 걸려 있는 동안 ReadIOPS 메트릭을 확인하십시오. ReadIOPS의 값은 작고 안정적이

어야 합니다. 경우에 따라 DB 인스턴스 클래스를 RAM이 더 많은 클래스로 스케일 업하면 ReadIOPS가 대폭 떨어질 수 있습니다. 이러한 경우 작업 세트가 메모리에 거의 완전히 저장되지 않았습니다. 규모 조정 작업 후 ReadIOPS가 더 이상 대폭 떨어지지 않을 때까지 계속 확장합니다. 그렇지 않으면 ReadIOPS가 매우 작은 양으로 감소됩니다. DB 인스턴스의 메트릭 모니터링에 대한 자세한 내용은 Amazon RDS 콘솔에서 지표 보기 단원을 참조하십시오.

데이터베이스 엔진 버전을 최신 상태로 유지

데이터베이스 엔진 버전을 정기적으로 업그레이드하여 보안, 성능 및 규정 준수를 유지합니다. Amazon RDS는 보안 패치, 성능 향상 및 새로운 기능을 포함하는 새로운 마이너 및 메이저 버전을 릴리스합니다. 오래된 데이터베이스 엔진을 실행하면 워크로드가 알려진 취약성, 호환성 문제에 노출되고 AWS 및 데이터베이스 공급업체의 지원이 감소할 수 있습니다.

중단을 최소화하려면 업그레이드를 계획할 때 다음 사항을 고려합니다.

- 스테이징 환경에서 테스트 - 프로덕션 데이터베이스를 업그레이드하기 전에 워크로드에 대해 새 버전을 검증합니다.
- Amazon RDS 관리형 업그레이드 사용 - 더 쉬운 패치 적용을 위해 마이너 버전 자동 업그레이드를 사용 설정합니다.
- 메이저 버전 업그레이드 예약 - 릴리스 정보를 검토하고, 애플리케이션 호환성을 테스트하고, 제어된 업그레이드 기간을 계획합니다.

정기적인 업그레이드를 통해 데이터베이스를 안전하게 유지하고 최적화하며 AWS 모범 사례에 맞게 조정할 수 있습니다.

AWS 데이터베이스 드라이버

애플리케이션 연결에 AWS 드라이버 제품군을 사용하는 것이 좋습니다. 더 빠른 전환 및 장애 조치 시간, AWS Secrets Manager, AWS Identity and Access Management(IAM) 및 페더레이션 ID를 사용한 인증을 지원하도록 설계된 드라이버입니다. AWS 드라이버는 DB 인스턴스 상태 모니터링과 인스턴스 토폴로지 파악을 통해 새 라이터를 결정합니다. 이 접근 방식은 전환 및 장애 조치 시간을 오픈 소스 드라이버의 경우 수십 초였던 것에 비해 10초 미만으로 단축합니다.

새로운 서비스 기능이 도입됨에 따라 AWS 드라이버 제품군의 목표는 이러한 서비스 기능에 대한 지원을 기본 제공하는 것입니다.

자세한 내용은 AWS 드라이버를 사용하여 DB 인스턴스에 연결 섹션을 참조하세요.

Enhanced Monitoring을 통한 운영 체제 문제 식별

확장 모니터링이 활성화되어 있으면 Amazon RDS는 DB 인스턴스가 실행되는 운영 체제(OS)에 대한 측정치를 실시간으로 제공합니다. 콘솔을 사용하여 DB 인스턴스에 대한 지표를 볼 수 있습니다. 또한 선택한 모니터링 시스템에서 Amazon CloudWatch Logs의 Enhanced Monitoring JSON 출력을 사용할 수 있습니다. Enhanced Monitoring에 대한 자세한 내용은 Enhanced Monitoring을 사용하여 OS 지표 모니터링 단원을 참조하세요.

지표를 통해 성능 문제 식별

리소스 부족이나 기타 일반적인 병목으로 인해 발생하는 성능 문제를 식별하기 위해 Amazon RDS DB 인스턴스에서 사용할 수 있는 지표를 모니터링할 수 있습니다.

성능 지표 보기

다양한 기간 동안의 평균, 최대, 최소 측정값을 보려면 성능 지표를 정기적으로 모니터링해야 합니다. 이렇게 하면 성능이 저하된 시점을 식별할 수 있습니다. 또한 특정 지표 임계값에 대해 Amazon CloudWatch 경보를 설정하여 해당 임계값에 이를 경우 알리도록 할 수 있습니다.

성능 문제를 해결하기 위해서는 시스템의 기준 성능을 파악해야 합니다. DB 인스턴스를 설정하고 일반적인 워크로드로 실행하는 경우 모든 성능 지표의 평균, 최댓값, 최솟값을 캡처합니다. 다양한 간격(예: 1시간, 24시간, 1주, 2주)으로 캡처하세요. 그러면 무엇이 정상인지를 알 수 있습니다. 이렇게 하면 작업의 최고 피크와 최저 피크 시간을 비교할 수 있습니다. 그런 다음 이 정보를 사용하여 성능이 표준 수준 이하로 떨어진 때를 식별할 수 있습니다.

다중 AZ DB 클러스터를 사용하는 경우 리더 DB 인스턴스에서 가장 최근에 적용된 트랜잭션과 라이터 DB 인스턴스에서 가장 최근 트랜잭션 사이의 시간 차이를 모니터링합니다. 이 차이를 복제본 지연이라고 부릅니다. 자세한 내용은 복제본 지연 시간 및 다중 AZ DB 클러스터 섹션을 참조하세요.

성능 개선 도우미 대시보드에서 결합된 성능 개선 도우미 및 CloudWatch 지표를 확인하고 DB 인스턴스를 모니터링할 수 있습니다. 이 모니터링 보기를 사용하려면 DB 인스턴스에서 성능 개선 도우미가 켜져 있어야 합니다. 이 모니터링에 대한 자세한 내용은 성능 개선 도우미 대시보드와 결합된 지표 보기 섹션을 참조하세요.

특정 기간에 대한 성과 분석 보고서를 생성하고 식별된 인사이트와 문제 해결을 위한 권장 사항을 볼 수 있습니다. 자세한 내용은 성능 개선 도우미에서 성과 분석 보고서 생성 섹션을 참조하세요.

성능 지표를 보려면

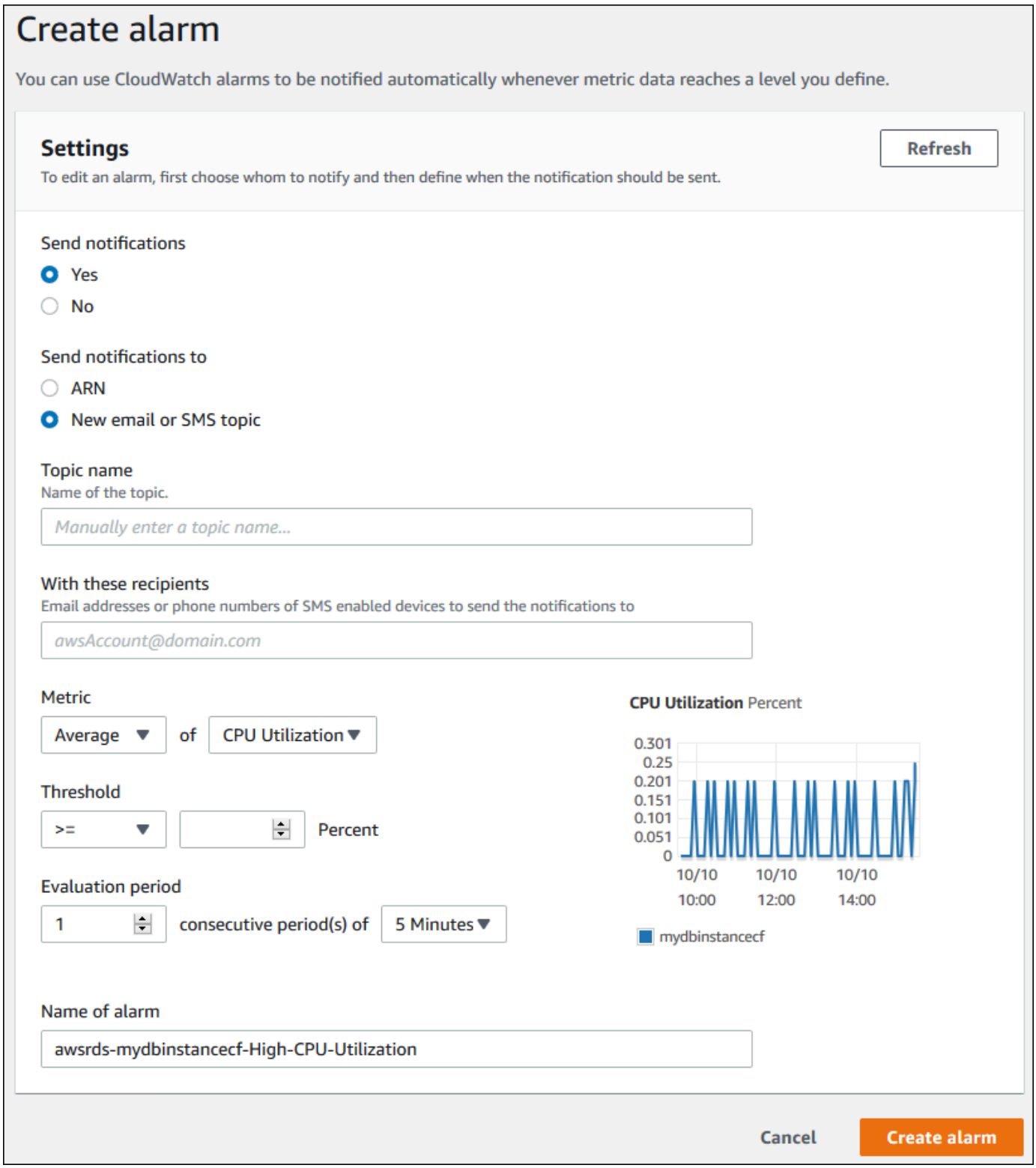
1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 후 DB 인스턴스를 선택합니다.
3. 모니터링을 선택합니다.

   대시보드는 성능 지표를 제공합니다. 지표는 기본적으로 지난 3시간 동안의 정보를 표시합니다.
4. 오른쪽 상단의 숫자 버튼을 사용하여 다른 페이지의 지표를 보거나, 설정을 조정하여 더 많은 지표를 표시합니다.
5. 다른 날짜의 데이터를 보려면 성능 지표의 시간 범위를 조정합니다. [Statistic], [Time Range] 및 [Period] 값을 변경하여 표시되는 정보를 조정할 수 있습니다. 예를 들어 지난 2주 동안 각 날짜의 지표에 대한 피크 값을 보고 싶을 수 있습니다. 그렇다면 통계를 최대로, 시간 범위를 지난 2주로, 기간을 일로 설정합니다.

CLI 또는 API를 사용하여 성능 지표를 볼 수도 있습니다. 자세한 내용은 Amazon RDS 콘솔에서 지표 보기 섹션을 참조하세요.

CloudWatch 경보를 설정하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 후 DB 인스턴스를 선택합니다.
3. Logs & events(로그 및 이벤트)를 선택합니다.
4. CloudWatch 경보 섹션에서 경보 생성을 선택합니다.



5. Send notifications(알림 보내기)에서 예를 선택하고 알림 받을 대상에서 New email or SMS topic(새로운 이메일 또는 SMS 주제)을 선택합니다.

6. 주제 이름에 알림의 이름을 입력하고, 수신자에 이메일 주소 혹은 전화번호 목록을 쉼표로 구분하여 입력합니다.
7. 지표에서 설정할 경보 통계 및 지표를 선택합니다.
8. 임계값에서 지표가 임계값보다 크거나 작아야 하는지 아니면 임계값과 같아야 하는지 지정하고 임계값을 지정합니다.
9. 평가 기간은 경보에 대한 평가 기간을 선택합니다. consecutive period(s) of(연속 기간) 상자에서 임계값이 얼마나 유지되어야 경보를 발생할지 기간을 선택합니다.
10. 경보 이름에 경보 이름을 입력합니다.
11. [Create Alarm]을 선택합니다.

CloudWatch 경보 섹션에 경보가 나타납니다.

성능 지표 평가

한 개의 DB 인스턴스에는 서로 다른 많은 카테고리의 지표가 있으며, 허용되는 값을 결정하는 방법은 지표에 따라 다릅니다.

CPU

- CPU 사용률 – 사용된 컴퓨터 처리 용량의 백분율입니다.

메모리

- 여유 메모리 – DB 인스턴스에서 사용 가능한 RAM을 바이트 단위로 나타냅니다. [Monitoring] 탭 지표의 붉은색 선이 CPU, 메모리 및 스토리지 지표의 75%에 표시됩니다. 인스턴스 메모리 소비가 주기적으로 이 선을 넘는 경우 이는 워크로드를 확인하거나 인스턴스를 업그레이드해야 함을 나타냅니다.
- 스왑 사용량 – DB 인스턴스에서 사용하는 스왑 공간을 바이트 단위로 나타냅니다.

디스크 공간

- 여유 스토리지 공간 – DB 인스턴스에서 현재 사용하지 않고 있는 디스크 공간을 메가바이트 단위로 나타냅니다.

입력/출력 작업

- IOPS 읽기, IOPS 쓰기 – 초당 디스크 읽기 또는 쓰기 작업의 평균 횟수입니다.
- 읽기 지연 시간, 쓰기 지연 시간 – 읽기 또는 쓰기 작업의 평균 시간(밀리초)입니다.
- 읽기 처리량, 쓰기 처리량 – 초당 디스크에서 읽거나 디스크에 쓴 평균 크기(메가바이트)입니다.
- 대기열 길이 – 디스크에 쓰기 위해 또는 디스크에서 읽기 위해 대기 중인 I/O 작업의 수입니다.

네트워크 트래픽

- 네트워크 수신 처리량, 네트워크 전송 처리량 – DB 인스턴스에 대한 수신 또는 전송 네트워크 트래픽 속도(초당 바이트)입니다.

데이터베이스 연결

- DB 연결 – DB 인스턴스에 연결된 클라이언트 세션의 수입니다.

사용 가능한 각 성능 지표에 대한 자세한 내용은 Amazon CloudWatch로 Amazon RDS 지표 모니터링 단원을 참조하십시오.

일반적으로 성능 지표에 허용되는 값은 기준이 무엇인지 그리고 애플리케이션 무엇을 수행하는지에 따라 다릅니다. 기준과의 일관된 차이 또는 추세를 조사하십시오. 특정 지표 유형에 대한 참고 정보는 다음과 같습니다.

- CPU 또는 RAM 사용량이 많음 - CPU 또는 RAM 사용량 값을 높게 설정하는 것이 적합할 수 있습니다. 예를 들어 해당 애플리케이션의 목표(처리량 또는 동시성)와 일치하고 예상되는 결과라면 그럴 수 있습니다.
- 디스크 공간 사용량 – 총 디스크 용량의 85퍼센트 이상이 계속 사용될 경우 디스크 공간 사용량을 검사합니다. 인스턴스에서 데이터를 삭제할 수 있는지 또는 다른 시스템에 데이터를 아카이브하여 공간을 확보할 수 있는지 확인합니다.
- 네트워크 트래픽 – 네트워크 트래픽의 경우 시스템 관리자에게 문의하여 해당 도메인 네트워크 및 인터넷 연결의 기대 처리량을 확인합니다. 처리량이 기대값보다 항상 낮으면 네트워크 트래픽을 검사합니다.
- 데이터베이스 연결 – 인스턴스 성능 저하 및 응답 시간 지연과 함께 사용자 연결 수가 많을 경우 데이터베이스 연결 제한을 고려해 봅니다. DB 인스턴스에 대한 최적의 사용자 연결 수는 해당 인스턴스 클래스와, 수행하는 작업의 복잡성에 따라 다릅니다. 데이터베이스 연결 수를 지정하려면 DB 인스턴스를 파라미터 그룹과 연결합니다. 이 그룹에서 사용자 연결 파라미터를 0(무제한) 이외의 값으로

설정합니다. 기존 파라미터 그룹을 사용하거나 새로 하나 만들 수 있습니다. 자세한 내용은 Amazon RDS의 파라미터 그룹 섹션을 참조하세요.

- IOPS 지표 – IOPS 지표의 기대값은 디스크 사양 및 서버 구성에 따라 다르므로 해당 기준에 일반적인 값을 파악합니다. 값이 기준과 계속 차이가 나는지 검사합니다. 최적의 IOPS 성능을 위해, 일반적인 작업 세트가 메모리에 적합하고 읽기 및 쓰기 작업을 최소화하는지 확인합니다.

성능 지표와 관련된 문제의 경우 성능을 개선하기 위한 첫 번째 단계는 가장 많이 사용되고 비용이 가장 많이 드는 쿼리를 튜닝하는 것입니다. 그러한 쿼리를 튜닝하여 시스템 리소스에 대한 부담이 낮아지는지 확인하세요. 자세한 내용은 쿼리 튜닝 섹션을 참조하세요.

쿼리가 조정되었는데도 문제가 지속되면 Amazon RDS DB 인스턴스 클래스를 업그레이드하는 것을 고려해 보세요. 문제와 관련된 리소스(CPU, RAM, 디스크 공간, 네트워크 대역폭, I/O 용량)를 추가하여 업그레이드할 수 있습니다.

쿼리 튜닝

DB 인스턴스 성능을 향상하는 가장 좋은 방법 중 하나는 일반적으로 가장 많이 사용하는 쿼리와 리소스를 가장 많이 사용하는 쿼리를 튜닝하는 것입니다. 이때 실행 비용이 절감되도록 튜닝합니다. 쿼리 개선에 대한 자세한 내용은 다음 리소스를 사용하세요.

- MySQL – MySQL 설명서에서 SELECT 문 최적화를 참조하세요. 추가 쿼리 튜닝 리소스에 대한 내용은 MySQL 성능 튜닝 및 최적화 리소스를 참조하세요.
- Oracle – Oracle Database 설명서의 데이터베이스 SQL 튜닝 안내서를 참조하세요.
- SQL Server – Microsoft 설명서의 쿼리 분석을 참조하세요. 또한 Microsoft 설명서의 시스템 동적 관리 보기에서 설명한 실행 관련, 인덱스 관련 및 I/O 관련 데이터 관리 보기(DMV)를 사용하여 SQL Server 쿼리 문제를 해결할 수도 있습니다.

  쿼리 튜닝의 공통적인 부분은 효율적인 인덱스를 만드는 것입니다. DB 인스턴스의 인덱스 개선에 대한 자세한 내용은 Microsoft 설명서의 데이터베이스 엔진 튜닝 관리자를 참조하세요. RDS for SQL Server에서 튜닝 관리자 사용에 대한 자세한 내용은 데이터베이스 엔진 튜닝 관리자를 사용하여 Amazon RDS for SQL Server DB 인스턴스의 데이터베이스 워크로드 분석 섹션을 참조하세요.
- PostgreSQL – 쿼리 계획을 분석하는 방법은 PostgreSQL 설명서에서 EXPLAIN 사용을 참조하세요. 이 정보를 참조하여 쿼리 성능을 높이기 위해 쿼리나 기본 테이블을 수정할 수 있습니다.

  최상의 성능을 위해 쿼리에 조인을 지정하는 방법에 대한 자세한 내용은 명시적 JOIN 절을 사용하여 플래너 제어를 참조하세요.

- MariaDB – MariaDB 설명서에서 쿼리 최적화를 참조하세요.

MySQL 작업 모범 사례

MySQL 데이터베이스의 테이블 크기와 테이블 개수는 모두 성능에 영향을 미칠 수 있습니다.

테이블 크기

일반적으로 파일 크기에 대한 운영 체제 제약 조건에 따라 MySQL 데이터베이스의 유효 최대 테이블 크기가 결정됩니다. 즉, 이 한도는 일반적으로 내부 MySQL 제약 조건에 의해 결정되지 않습니다.

MySQL DB 인스턴스에서 데이터베이스의 테이블이 너무 크게 늘어나지 않도록 합니다. 스토리지는 일반적으로 64TiB로 제한되기는 하지만 프로비저닝 스토리지 제한에 따라 MySQL 테이블 파일의 최대 크기를 16TB로 제한합니다. 파일 크기가 16TB 제한을 넘지 않도록 라지 테이블을 분할합니다. 이 접근 방식을 수행하면 성능 및 복구 시간도 향상할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 파일 크기 제한 섹션을 참조하세요.

크기가 100GB 이상인 매우 큰 테이블은 읽기 및 쓰기(DML 문 포함, 특히 DDL 문 포함)의 성능에 부정적인 영향을 미칠 수 있습니다. larges 테이블의 인덱스는 선택 성능을 크게 높일 수 있지만 DML 문의 성능을 저하시킬 수도 있습니다. ALTER TABLE과 같은 DDL 문은 큰 테이블에 대해 상당히 느려질 수 있습니다. 해당 작업은 경우에 따라 테이블을 완전히 다시 작성할 수 있기 때문입니다. 이러한 DDL 문은 작업을 실행하는 동안 테이블을 잠글 수 있습니다.

MySQL의 읽기 및 쓰기에 필요한 메모리 용량은 작업에 관련된 테이블에 따라 달라집니다. 적어도 많이 사용되는 테이블의 인덱스를 유지하기에는 충분한 RAM을 확보하는 것이 좋습니다. 데이터베이스에서 가장 큰 10개의 테이블과 인덱스를 찾으려면 다음 쿼리를 사용합니다.

```
select table_schema, TABLE_NAME, dat, idx from
(SELECT table_schema, TABLE_NAME,
       ( data_length ) / 1024 / 1024 as dat,
       ( index_length ) / 1024 / 1024 as idx
FROM information_schema.TABLES
order by 3 desc ) a
order by 3 desc
limit 10;
```

테이블 수

기본 파일 시스템에서는 테이블을 나타내는 파일 수가 제한될 수 있습니다. 그러나 MySQL에는 테이블 수에 제한이 없습니다. 그럼에도 불구하고 MySQL InnoDB 스토리지 엔진의 총 테이블 수는 해당 테이

블의 크기에 관계없이 성능 저하의 원인이 될 수 있습니다. 운영 체제에 미치는 영향을 최소화하기 위해 동일한 MySQL DB 인스턴스의 여러 데이터베이스에 걸쳐 테이블을 분할할 수 있습니다. 이렇게 하면 디렉터리의 파일 수가 제한되지만 전반적인 문제는 해결되지 않습니다.

많은 수의 테이블(10,000개 이상)로 인해 성능 저하가 발생하는 경우, 이는 MySQL이 스토리지 파일을 열고 닫는 것을 비롯하여 스토리지 파일을 작업에 사용하기 때문에 발생합니다. 이 문제를 해결하기 위해 `table_open_cache` 및 `table_definition_cache` 파라미터의 크기를 늘릴 수 있습니다. 하지만 이들 파라미터의 값을 늘리면 MySQL이 사용하는 메모리 용량이 크게 늘어날 수 있으며 가용 메모리를 전부 사용하게 될 수도 있습니다. 자세한 내용은 MySQL 설명서의 MySQL에서 테이블을 열고 닫는 방법 섹션을 참조하세요.

또한 테이블이 너무 많으면 MySQL 시작 시간에 크게 영향을 미칠 수 있습니다. 정상 종료 및 재시작과 충돌 복구가 모두 영향을 받을 수 있습니다. 특히 MySQL 8.0 이전 버전에서는 더욱 그렇습니다.

DB 인스턴스의 모든 데이터베이스에 걸쳐 총 테이블 수를 1만 개 미만으로 유지하는 것이 좋습니다. MySQL 데이터베이스에 많은 수의 테이블이 있는 사용 사례는 MySQL 8.0의 테이블 백만 개 섹션을 참조하세요.

스토리지 엔진

Amazon RDS for MySQL의 특정 시점으로 복구 및 스냅샷 복원 기능을 사용하려면 충돌 복구 가능 스토리지 엔진이 필요합니다. 이러한 기능은 InnoDB 스토리지 엔진에만 지원됩니다. MySQL은 다양한 기능을 가진 여러 스토리지 엔진을 지원하지만, 모든 엔진이 충돌 복구와 데이터 내구성에 최적화되어 있지는 않습니다. 예를 들어 MyISAM 스토리지 엔진은 안정적인 충돌 복구를 지원하지 않으며, 이로 인해 특정 시점으로 복구 또는 스냅샷 복원이 의도한 대로 작동하지 못할 수 있습니다. 그 결과 충돌 후 MySQL을 다시 시작하면 데이터가 손실되거나 손상될 수 있습니다.

InnoDB는 Amazon RDS에서 MySQL DB 인스턴스용으로 권장되고 지원되는 스토리지 엔진입니다. InnoDB 인스턴스는 Aurora로 마이그레이션할 수 있지만, MyISAM 인스턴스는 마이그레이션할 수 없습니다. 하지만 MyISAM은 강력한 전체 텍스트 검색 기능이 필요한 경우 InnoDB보다 나은 성능을 발휘합니다. 그래도 Amazon RDS와 함께 MyISAM을 사용하도록 선택할 경우 지원되지 않는 MySQL 스토리지 엔진에 대한 자동 백업에 개략적으로 설명되어 있는 단계를 따르면 특정한 시나리오에서 스냅샷 복원 기능에 유용할 수 있습니다.

기존 MyISAM 테이블을 InnoDB 테이블로 변환하려면 MySQL 설명서의 MyISAM에서 InnoDB로 테이블 변환에 요약된 프로세스를 사용하면 됩니다. MyISAM과 InnoDB는 각기 다른 장점과 단점을 갖고 있으므로 전환하기 전에 이 전환이 애플리케이션에 미치는 영향을 충분히 평가해야 합니다.

또한 연동 스토리지 엔진은 현재 Amazon RDS for MySQL에서 지원되지 않습니다.

MariaDB 작업 모범 사례

MariaDB 데이터베이스의 테이블 크기와 테이블 개수는 모두 성능에 영향을 미칠 수 있습니다.

테이블 크기

일반적으로 파일 크기에 대한 운영 체제 제약 조건에 따라 MariaDB 데이터베이스의 유효 최대 테이블 크기가 결정됩니다. 즉, 이 한도는 일반적으로 내부 MariaDB 제약 조건에 의해 결정되지 않습니다.

MariaDB DB 인스턴스에서 데이터베이스의 테이블이 너무 크게 늘어나지 않도록 합니다. 스토리지는 일반적으로 64TiB로 제한되기는 하지만 프로비저닝 스토리지 제한에 따라 MariaDB 테이블 파일의 최대 크기를 16TB로 제한합니다. 파일 크기가 16TB 제한을 넘지 않도록 라지 테이블을 분할합니다. 이 접근 방식을 수행하면 성능 및 복구 시간도 향상할 수 있습니다.

크기가 100GB 이상인 매우 큰 테이블은 읽기 및 쓰기(DML 문 포함, 특히 DDL 문 포함)의 성능에 부정적인 영향을 미칠 수 있습니다. larges 테이블의 인덱스는 선택 성능을 크게 높일 수 있지만 DML 문의 성능을 저하시킬 수도 있습니다. ALTER TABLE과 같은 DDL 문은 큰 테이블에 대해 상당히 느려질 수 있습니다. 해당 작업은 경우에 따라 테이블을 완전히 다시 작성할 수 있기 때문입니다. 이러한 DDL 문은 작업을 실행하는 동안 테이블을 잠글 수 있습니다.

MariaDB의 읽기 및 쓰기에 필요한 메모리 용량은 작업에 관련된 테이블에 따라 달라집니다. 적어도 많이 사용되는 테이블의 인덱스를 유지하기에는 충분한 RAM을 확보하는 것이 좋습니다. 데이터베이스에서 가장 큰 10개의 테이블과 인덱스를 찾으려면 다음 쿼리를 사용합니다.

```
select table_schema, TABLE_NAME, dat, idx from
(SELECT table_schema, TABLE_NAME,
       ( data_length ) / 1024 / 1024 as dat,
       ( index_length ) / 1024 / 1024 as idx
FROM information_schema.TABLES
order by 3 desc ) a
order by 3 desc
limit 10;
```

테이블 수

기본 파일 시스템에서는 테이블을 나타내는 파일 수가 제한될 수 있습니다. 그러나 MariaDB에는 테이블 수에 제한이 없습니다. 그럼에도 불구하고 MariaDB InnoDB 스토리지 엔진의 총 테이블 수는 해당 테이블의 크기에 관계없이 성능 저하의 원인이 될 수 있습니다. 운영 체제에 미치는 영향을 최소화하기 위해 동일한 MariaDB DB 인스턴스의 여러 데이터베이스에 걸쳐 테이블을 분할할 수 있습니다. 이렇게 하면 디렉터리의 파일 수가 제한되지만 전반적인 문제는 해결되지 않습니다.

많은 수의 테이블(1만 개 이상)로 인해 성능 저하가 발생하는 경우, 이는 MariaDB가 스토리지 파일을 작업에 사용하기 때문에 발생합니다. 이 작업에는 MariaDB 스토리지가 파일을 열고 닫는 작업이 포함됩니다. 이 문제를 해결하기 위해 `table_open_cache` 및 `table_definition_cache` 파라미터의 크기를 늘릴 수 있습니다. 하지만 이들 파라미터의 값을 늘리면 MariaDB가 사용하는 메모리 용량이 크게 늘어날 수 있으며 사용 가능한 메모리를 모두 사용할 수도 있습니다. 자세한 내용은 MariaDB 설명서의 table_open_cache 최적화 섹션을 참조하세요.

또한 테이블이 너무 많으면 MariaDB 시작 시간에 크게 영향을 미칠 수 있습니다. 정상 종료 및 재시작과 충돌 복구가 모두 영향을 받을 수 있습니다. DB 인스턴스의 모든 데이터베이스에 걸쳐 총 테이블 수를 1만 개 미만으로 유지하는 것이 좋습니다.

스토리지 엔진

Amazon RDS for MariaDB의 특정 시점으로 복구 및 스냅샷 복원 기능을 사용하려면 충돌 복구 가능 스토리지 엔진이 필요합니다. MariaDB는 다양한 기능을 가진 여러 스토리지 엔진을 지원하지만, 모든 엔진이 충돌 복구와 데이터 내구성에 최적화되어 있지는 않습니다. 예를 들어, Aria가 충돌 안정성을 개선한 MyISAM 대체 스토리지 엔진이지만, 여전히 특정 시점으로 복구 또는 스냅샷 복원이 의도한 대로 작동하지 못할 수 있습니다. 그 결과 충돌 후 MariaDB를 다시 시작하면 데이터가 손실되거나 손상될 수 있습니다. InnoDB는 Amazon RDS에서 MariaDB DB 인스턴스용으로 권장되고 지원되는 스토리지 엔진입니다. 그래도 Amazon RDS와 함께 Aria를 사용하도록 선택할 경우 지원되지 않는 MariaDB 스토리지 엔진에 대한 자동 백업에 개략적으로 설명되어 있는 단계를 따르면 특정한 시나리오에서 스냅샷 복원 기능에 유용할 수 있습니다.

기존 MyISAM 테이블을 InnoDB 테이블로 변환하려면 MariaDB 설명서의 MyISAM에서 InnoDB로 테이블 변환에 요약된 프로세스를 사용하면 됩니다. MyISAM과 InnoDB는 각기 다른 장점과 단점을 갖고 있으므로 전환하기 전에 이 전환이 애플리케이션에 미치는 영향을 충분히 평가해야 합니다.

Oracle 작업의 모범 사례

Amazon RDS for Oracle 작업의 모범 사례에 대한 자세한 내용은 Amazon Web Services에서 Oracle Database 실행의 모범 사례를 참조하세요.

2020 AWS 가상 워크숍에는 Amazon RDS에서 프로덕션 Oracle 데이터베이스를 실행하는 방법에 대한 프레젠테이션이 포함되어 있습니다. 프레젠테이션 비디오는 여기에서 볼 수 있습니다.

PostgreSQL로 작업하기 위한 모범 사례

RDS for PostgreSQL로 성능을 향상할 수 있는 두 가지 중요한 영역 중 하나는 DB 인스턴스에 데이터를 로드할 때입니다. 또 다른 하나는 PostgreSQL Autovacuum 기능을 사용할 때입니다. 다음 섹션에서는 이런 영역에 대해 권장하는 모범 사례를 다룹니다.

Amazon RDS에서 다른 일반적인 PostgreSQL DBA 태스크를 구현하는 방법에 대한 자세한 내용은 Amazon RDS for PostgreSQL의 일반적인 DBA 태스크 섹션을 참조하세요.

PostgreSQL DB 인스턴스에 데이터 로드

Amazon RDS PostgreSQL DB 인스턴스에 데이터를 로드할 때 DB 인스턴스 설정과 DB 파라미터 그룹 값을 수정합니다. 가장 효율적으로 DB 인스턴스에 데이터를 가져오도록 설정합니다.

DB 인스턴스 설정을 다음과 같이 수정합니다.

- DB 인스턴스 백업 비활성화(backup_retention을 0으로 설정)
- Multi-AZ 비활성화

다음 설정을 포함하도록 DB 파라미터 그룹을 수정합니다. 또한 파라미터 설정을 테스트하여 DB 인스턴스에 가장 효율적인 설정을 찾습니다.

- `maintenance_work_mem` 파라미터의 값을 늘립니다. PostgreSQL 리소스 사용 파라미터에 대한 자세한 내용은 PostgreSQL 설명서를 참조하세요.
- 미리 쓰기(WAL) 로그에 대한 쓰기 수를 줄이도록 `max_wal_size` 및 `checkpoint_timeout` 파라미터의 값을 늘립니다.
- `synchronous_commit` 파라미터 비활성화
- PostgreSQL autovacuum 파라미터를 비활성화합니다.
- 가져오려는 테이블 모두가 로깅되는지 확인합니다. 로깅되지 않는 테이블에 저장된 데이터는 장애 조치 중에 손실될 수 있습니다. 자세한 내용은 로그되지 않은 테이블 생성을 참조하세요.

이런 설정과 함께 `pg_dump -Fc`(압축) 또는 `pg_restore -j`(병렬) 명령을 사용합니다.

로드 작업이 완료되면 DB 인스턴스와 DB 파라미터를 일반 설정으로 되돌립니다.

PostgreSQL Autovacuum 기능 사용

PostgreSQL 데이터베이스용 autovacuum 기능은 PostgreSQL DB 인스턴스의 상태를 유지 관리하는 데 사용할 것을 강력히 권장하는 중요한 기능입니다. Autovacuum은 VACUUM 및 ANALYZE 명령의 실행을 자동화합니다. PostgreSQL에서는 autovacuum을 반드시 사용해야 하며 Amazon RDS에서는 필수 사항은 아니지만 훌륭한 성능을 달성하는 데 매우 중요한 기능입니다. 이 기능은 모든 새로운 Amazon RDS for PostgreSQL DB 인스턴스에 대해 기본적으로 활성화되며, 관련 구성 파라미터는 적절히 기본적으로 설정됩니다.

데이터베이스 관리자는 이 유지 관리 작업을 파악하고 이해할 필요가 있습니다. Autovacuum에 대한 PostgreSQL 설명서는 Autovacuum 데몬을 참조하세요.

Autovacuum은 "리소스"를 일정 부분 사용하는 작업하지만, 백그라운드에서 작동하며 사용자 작업에 최대한 많은 리소스가 할당되도록 합니다. Autovacuum이 활성화되어 있을 때는 업데이트되거나 삭제된 튜플 수가 많은 테이블이 있는지 확인합니다. 또한 이 기능은 트랜잭션 ID 랩어라운드로 인해 매우 오래된 데이터가 손실되지 않도록 보호합니다. 자세한 내용은 트랜잭션 ID 랩어라운드 실패 방지를 참조하십시오.

Autovacuum을 더 나은 성능을 위해 줄일 수 있는 높은 오버헤드 작업으로 생각하면 안 됩니다. 오히려, autovacuum을 실행하지 않으면 업데이트 및 삭제 속도가 빠른 테이블이 시간이 흐르면서 빠르게 성능이 저하됩니다.

> **Important**
>
> Autovacuum을 실행하지 않으면 훨씬 더 많은 주입식 vacuum 작업을 수행하기 위해 결국은 필연적으로 중단될 수 있습니다. 경우에 따라 autovacuum을 지나치게 보수적으로 사용하여 RDS for PostgreSQL DB 인스턴스를 사용할 수 없게 될 수 있습니다. 이러한 경우 PostgreSQL 데이터베이스는 자체 보호를 위해 종료됩니다. 그 시점에서 Amazon RDS는 DB 인스턴스에서 직접 단일 사용자 모드 전체 vacuum을 수행해야 합니다. 이렇게 전체 vacuum을 수행하면 몇 시간 동안 정전이 발생할 수 있습니다. 따라서 기본적으로 활성화되는 autovaccum은 아예 해제하지 않는 것이 좋습니다.

Autovacuum 파라미터에 따라 autovacuum의 작동 시점과 정도가 결정됩니다. `autovacuum_vacuum_threshold` 및 `autovacuum_vacuum_scale_factor` 파라미터에 따라 autovacuum 실행 시점이 결정됩니다. `autovacuum_max_workers`, `autovacuum_nap_time`, `autovacuum_cost_limit` 및 `autovacuum_cost_delay` 파라미터에 따라 autovacuum의 작동 정도가 결정됩니다. Autovacuum에 대한 자세한 정보, Autovacuum의 실행 조건, 필요한 파라미터에 관해서는 PostgreSQL 문서의 일상적인 Vacuuming을 참조하세요.

다음 쿼리를 통해 table1로 명명된 테이블에 있는 '사용되지 않는' 튜플의 수를 알 수 있습니다.

```
SELECT relname, n_dead_tup, last_vacuum, last_autovacuum FROM
pg_catalog.pg_stat_all_tables
WHERE n_dead_tup > 0 and relname = 'table1';
```

쿼리의 결과는 다음과 같은 내용일 것입니다.

```
relname | n_dead_tup | last_vacuum | last_autovacuum
---------+------------+-------------+-----------------
 tasks   |   81430522 |             |
(1 row)
```

Amazon RDS for PostgreSQL 모범 사례 동영상

2020 AWS re:Invent 컨퍼런스에는 Amazon RDS에서 PostgreSQL 작업의 새로운 기능과 모범 사례에 대한 프레젠테이션이 포함되었습니다. 프레젠테이션 비디오는 여기에서 볼 수 있습니다.

SQL Server로 작업하기 위한 모범 사례

SQL Server DB 인스턴스를 포함한 Multi-AZ 배포를 위한 모범 사례에는 다음이 포함됩니다.

- Amazon RDS RDS DB 이벤트를 사용하여 장애 조치를 모니터링합니다. 예를 들어 DB 인스턴스가 장애 조치할 때 문자 메시지 또는 이메일로 알림 서비스를 받을 수 있습니다. Amazon RDS 이벤트에 대한 자세한 내용은 Amazon RDS 이벤트 알림 작업 단원을 참조하십시오.
- 애플리케이션이 DNS 값을 캐시하는 경우 TTL(time to live)을 30초 미만으로 설정합니다. TTL을 이렇게 설정하면 장애 조치가 발생할 경우 도움이 됩니다. 장애 조치 시 IP 주소가 바뀔 수 있으며 캐시된 값은 더 이상 사용하지 못할 수 있습니다.
- 다음 모드에서는 Multi-AZ에 필수적인 트랜잭션 로깅이 해제되므로, 이들 모드를 활성화하지 않는 것이 좋습니다.
  - 단순 복구 모드
  - 오프라인 모드
  - 읽기 전용 모드
- 테스트를 통해 DB 인스턴스를 장애 조치하는 데 얼마나 오래 걸리는지 확인합니다. 장애 조치 시간은 데이터베이스의 유형, 인스턴스 클래스, 사용하는 스토리지 유형에 따라 변할 수 있습니다. 장애 조치가 발생할 경우 작업을 계속 수행할 수 있는 애플리케이션의 능력도 테스트해야 합니다.

- 장애 조치 시간을 단축하려면 다음을 수행합니다.
  - 워크로드에 대해 충분한 프로비저닝된 IOPS가 할당되어 있는지 확인하십시오. I/O가 부적합하면 장애 조치 시간이 길어질 수 있습니다. 데이터베이스 복구에 I/O가 필요합니다.
  - 더 작은 트랜잭션을 사용하십시오. 데이터베이스 복구는 트랜잭션에 의존하므로, 큰 트랜잭션을 여러 개의 작은 트랜잭션으로 나눌 수 있다면 장애 조치 시간이 단축될 것입니다.
- 장애 조치 중에 지연 시간이 증가할 것이라는 점을 고려하십시오. 장애 조치 프로세스의 일부로서, Amazon RDS는 데이터를 새 대기 인스턴스로 자동으로 복제합니다. 이 복제는 새 데이터가 서로 다른 두 DB 인스턴스로 커밋되고 있음을 의미합니다. 따라서 대기 DB 인스턴스가 새 기본 DB 인스턴스를 따라잡을 때까지 약간의 지연 시간이 발생할 수 있습니다.
- 모든 가용 영역에 애플리케이션을 배포합니다. 한 가용 영역이 다운되더라도 다른 가용 영역에 있는 애플리케이션을 계속 사용할 수 있습니다.

SQL Server의 다중 AZ 배포를 사용하여 작업할 때, Amazon RDS가 인스턴스에 있는 모든 SQL Server 데이터베이스의 복제본을 생성합니다. 특정 데이터베이스의 보조 복제본이 생기지 않도록 하려면 해당 데이터베이스에 대해 다중 AZ를 사용하지 않는 별개의 DB 인스턴스를 설정합니다.

Amazon RDS for SQL Server 모범 사례 동영상

2019년 AWS re:Invent 컨퍼런스에는 Amazon RDS에서 SQL Server 작업의 새로운 기능과 모범 사례에 대한 프레젠테이션이 포함되었습니다. 프레젠테이션 비디오는 여기에서 볼 수 있습니다.

DB 파라미터 그룹 작업

DB 파라미터 그룹 변경 내용을 프로덕션 DB 인스턴스에 적용하기 전에 테스트 DB 인스턴스에 적용해 보는 것이 좋습니다. DB 파라미터 그룹에 DB 엔진 파라미터를 잘못 설정하면 성능 저하나 시스템 불안정 등의 의도하지 않은 부작용이 있을 수 있습니다. DB 엔진 파라미터를 수정할 때 항상 주의를 기울이고 DB 파라미터 그룹을 수정하기 전에 DB 인스턴스를 백업하십시오.

DB 인스턴스 백업에 대한 자세한 내용은 데이터 백업, 복원 및 내보내기 단원을 참조하십시오.

DB 인스턴스 생성 자동화 모범 사례

기본 설정된 마이너 버전의 데이터베이스 엔진을 사용하여 DB 인스턴스를 생성하는 것이 Amazon RDS 모범 사례입니다. AWS CLI, Amazon RDS API 또는 AWS CloudFormation을 사용하여 DB 인스턴스 생성을 자동화할 수 있습니다. 이러한 방법을 사용하는 경우 메이저 버전만 지정하면 Amazon RDS에서 기본 설정된 마이너 버전으로 인스턴스가 자동으로 생성됩니다. 예를 들어 PostgreSQL 12.5

가 기본 설정된 마이너 버전인 경우 `create-db-instance`(으)로 버전 12를 지정하면 DB 인스턴스는 버전 12.5가 됩니다.

기본 설정된 마이너 버전을 확인하려면 다음 예에서와 같이 `describe-db-engine-versions` 옵션을 사용하여 `--default-only` 명령을 실행하면 됩니다.

```
aws rds describe-db-engine-versions --default-only --engine postgres

{
    "DBEngineVersions": [
        {
            "Engine": "postgres",
            "EngineVersion": "12.5",
            "DBParameterGroupFamily": "postgres12",
            "DBEngineDescription": "PostgreSQL",
            "DBEngineVersionDescription": "PostgreSQL 12.5-R1",
            ...some output truncated...
        }
    ]
}
```

프로그래밍 방식으로 DB 인스턴스를 생성하는 방법에 대한 자세한 내용은 다음 리소스를 참조하세요.

- AWS CLI 사용 - create-db-instance
- Amazon RDS API – CreateDBInstance 사용
- AWS CloudFormation 사용 - AWS::RDS::DBInstance

Amazon RDS의 새로운 비디오 특성

2023 AWS re:Invent 컨퍼런스에는 새 Amazon RDS 특성에 관한 프레젠테이션이 포함되어 있습니다. 프레젠테이션 비디오는 여기에서 볼 수 있습니다.

Amazon RDS에 대한 프로그래밍 방식 액세스

Amazon RDS는 프로그래밍 방식으로 Amazon RDS 리소스를 관리할 수 있는 다음 도구를 제공합니다.

AWS Command Line Interface (AWS CLI)

명령줄 쉘에서 AWS CLI를 사용하여 RDS 리소스를 생성하고 관리할 수 있습니다. AWS CLI는 Amazon RDS와 같은 AWS 서비스의 API에 대한 직접 액세스를 제공합니다. Amazon RDS에 대한 명령의 구문 및 예시는 AWS CLI 명령 참조에서 rds 섹션을 참조하세요.

AWS CloudFormation

이 AWS 코드형 인프라(IaC) 도구를 사용하면 원하는 모든 Amazon RDS 리소스를 설명하는 템플릿을 생성하고 AWS CloudFormation이 해당 리소스를 프로비저닝하고 구성해 줍니다. 자세한 내용은 the section called “을 사용하여 리소스 생성AWS CloudFormation” 단원을 참조하십시오.

AWS 소프트웨어 개발 키트(SDK)

AWS는 널리 사용되는 많은 기술 및 프로그래밍 언어를 위한 SDK를 제공합니다. 이를 통해 애플리케이션 내에서 해당 언어 또는 기술로 AWS 서비스를 더 쉽게 직접적으로 호출할 수 있습니다. 이러한 SDK에 대한 자세한 내용은 AWS에서 애플리케이션 개발 및 관리를 위한 도구를 참조하세요.

Amazon RDS API

이 API는 Amazon RDS의 프로토콜 수준 인터페이스입니다. 이 API를 사용할 때는 모든 HTTPS 요청을 올바르게 포맷하고 모든 요청에 유효한 디지털 서명을 추가해야 합니다. 자세한 내용은 Amazon RDS API 참조 단원을 참조하십시오.

콘솔 투 코드

이 도구를 사용하면 Amazon RDS 콘솔에서 수행하는 작업에 대한 코드를 생성하고 AWS CloudFormation과 같은 다른 도구에서 해당 코드를 사용할 수 있습니다. 자세한 내용은 the section called “콘솔 투 코드” 단원을 참조하십시오.

콘솔 투 코드를 사용하여 Amazon RDS 콘솔 작업을 위한 코드를 생성합니다.

콘솔은 리소스를 생성하고 프로토타입을 테스트하기 위한 안내 경로를 제공합니다. 대규모로 동일한 리소스를 생성하려면 자동화 코드가 필요합니다. 콘솔 투 코드는 자동화 코드를 시작하는 데 도움이 되

는 Amazon Q Developer의 기능입니다. 콘솔 투 코드는 기본값과 사용자가 제공한 파라미터 값을 포함하여 콘솔 작업을 기록합니다. 그런 다음 생성형 AI를 사용하여 선택한 작업에 대해 원하는 언어와 형식으로 코드를 제안합니다. 콘솔 워크플로는 사용자가 지정한 파라미터 값이 모두 유효한지 확인하기 때문에 콘솔 투 코드를 사용하여 생성하는 코드는 호환되는 파라미터 값을 가집니다. 코드를 시작점으로 사용하여 특정 사용 사례에서 프로덕션에 바로 사용할 수 있도록 사용자 지정할 수 있습니다.

예를 들어 콘솔 투 코드를 사용하면 RDS DB 인스턴스 생성을 기록하고 AWS CloudFormation JSON 형식으로 코드를 생성하도록 선택할 수 있습니다. 그런 다음 해당 코드를 복사하고 AWS CloudFormation 템플릿에서 사용할 수 있도록 사용자 지정할 수 있습니다.

콘솔 투 코드는 현재 다음과 같은 언어 및 형식으로 코드형 인프라(IAC)를 생성할 수 있습니다.

- CDK Java
- CDK Python
- CDK TypeScript
- CloudFormation JSON
- CloudFormation YAML

콘솔 투 코드를 사용하는 방법에 대한 자세한 내용 및 지침은 Amazon Q Developer 사용 설명서의 Amazon Q Developer 콘솔 투 코드를 사용하여 AWS 서비스 자동화를 참조하세요.

Amazon RDS DB 인스턴스 구성

이 단원에서는 Amazon RDS DB 인스턴스를 설정하는 방법을 보여줍니다. DB 인스턴스를 생성하기 전에 DB 인스턴스를 실행할 DB 인스턴스 클래스를 결정합니다. 또한 AWS 리전을 선택하여 DB 인스턴스를 실행할 위치를 결정합니다. 그런 다음 DB 인스턴스를 생성합니다.

옵션 그룹과 DB 파라미터 그룹을 사용하여 DB 인스턴스를 구성할 수 있습니다.

- 옵션 그룹은 Amazon RDS DB 인스턴스에서 사용 가능한 옵션이라는 기능을 지정합니다.
- DB 파라미터 그룹은 하나 이상의 DB 인스턴스에 적용되는 엔진 구성 값의 컨테이너 역할을 합니다.

사용 가능한 옵션과 파라미터는 DB 엔진 및 DB 엔진 버전에 따라 다릅니다. DB 인스턴스를 생성할 때 옵션 그룹과 DB 파라미터 그룹을 지정할 수 있습니다. DB 인스턴스를 수정하여 지정할 수도 있습니다.

주제

- Amazon RDS DB 인스턴스 생성
- AWS CloudFormation을 사용하여 Amazon RDS 리소스 생성
- Amazon RDS DB 인스턴스에 연결
- 옵션 그룹 작업
- Amazon RDS의 파라미터 그룹
- Amazon RDS DB 인스턴스 설정을 사용하여 Amazon ElastiCache 캐시 생성
- AWS Database Migration Service를 사용하여 EC2 데이터베이스를 Amazon RDS로 자동 마이그레이션
- 자습서: 사용자 지정 파라미터 및 새로운 옵션 그룹을 사용하여 MySQL DB 인스턴스 만들기

Amazon RDS DB 인스턴스 생성

Amazon RDS의 기본 빌딩 블록은 데이터베이스를 생성하는 DB 인스턴스입니다. DB 인스턴스를 생성할 때는 엔진 고유의 특성을 선택합니다. 또한 데이터베이스 서버가 실행되는 AWS 인스턴스의 스토리지 용량, CPU, 메모리 등도 선택합니다.

주제

- DB 인스턴스 사전 조건
- DB 인스턴스 생성
- DB 인스턴스에 대한 설정

DB 인스턴스 사전 조건

> **Important**
>
> Amazon RDS DB 인스턴스를 생성하기 전에 Amazon RDS 환경 설정에서 작업을 완료합니다.

다음은 DB 인스턴스를 생성하기 위한 전제 조건입니다.

주제

- DB 인스턴스의 네트워크 구성
- 추가 사전 조건

DB 인스턴스의 네트워크 구성

Amazon VPC 서비스 기반의 Virtual Private Cloud(VPC)에서만 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 또한 가용 영역이 2개 이상 있는 AWS 리전에 있어야 합니다. DB 인스턴스에 대해 선택한 DB 서브넷 그룹은 2개 이상의 가용 영역을 포함해야 합니다. 이 구성을 사용하면 DB 인스턴스를 생성할 때 다중 AZ 배포를 구성하거나 향후 쉽게 DB 인스턴스로 이동할 수 있습니다.

새 DB 인스턴스와 동일한 VPC의 Amazon EC2 인스턴스 간에 연결을 설정하려는 경우 DB 인스턴스 생성 시 설정합니다. 동일한 VPC의 EC2 인스턴스 이외의 리소스에서 DB 인스턴스에 연결하려는 경우 네트워크 연결을 수동으로 구성합니다.

주제

- EC2 인스턴스와의 자동 네트워크 연결 구성
- 네트워크 수동 구성

EC2 인스턴스와의 자동 네트워크 연결 구성

RDS DB 인스턴스를 생성할 때 AWS Management Console을 사용하여 EC2 인스턴스와 새 DB 인스턴스 간의 연결을 설정할 수 있습니다. 이렇게 하면 RDS가 VPC 및 네트워크 설정을 자동으로 구성합니다. DB 인스턴스는 EC2 인스턴스가 DB 인스턴스에 액세스할 수 있도록 EC2 인스턴스와 동일한 VPC에 생성됩니다.

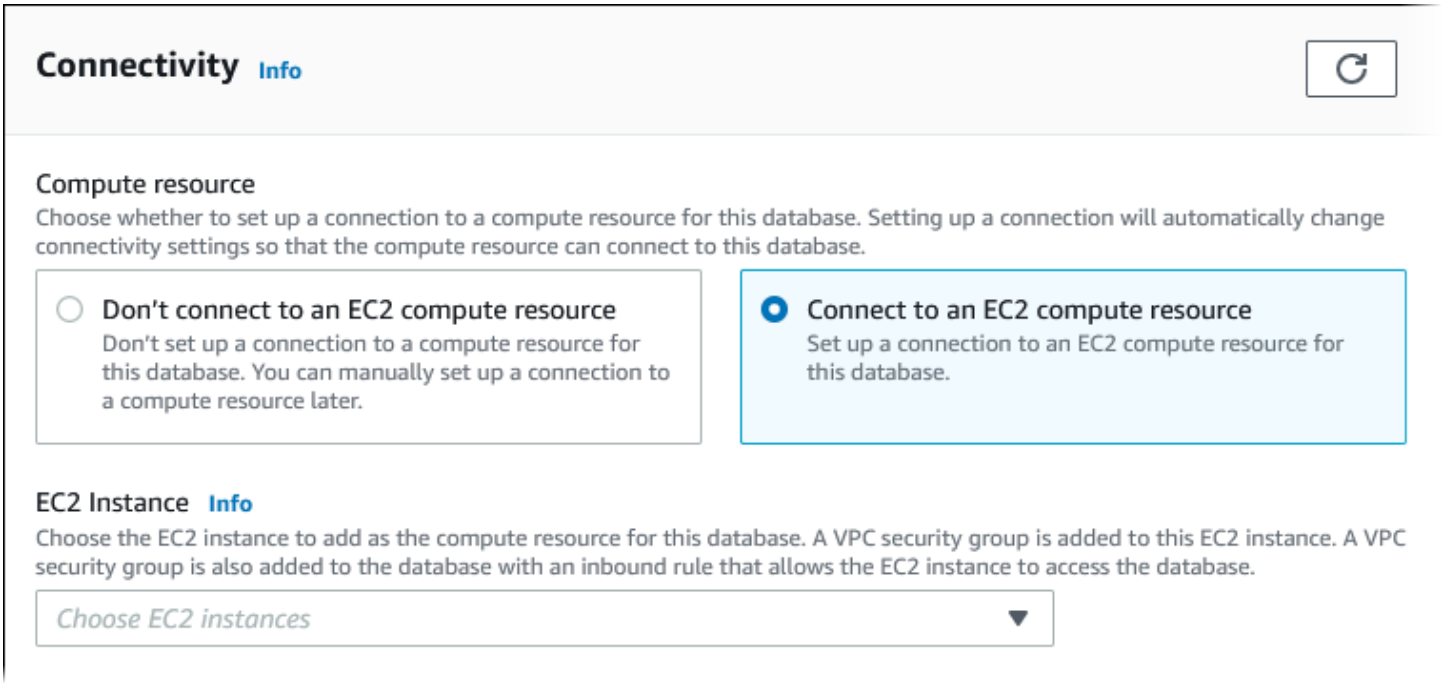
다음은 EC2 인스턴스를 DB 인스턴스와 연결하기 위한 요구 사항입니다.

- DB 인스턴스를 생성하려면 먼저 EC2 인스턴스가 AWS 리전에 있어야 합니다.

  AWS 리전에 EC2 인스턴스가 없는 경우 콘솔은 인스턴스를 생성할 수 있는 링크를 제공합니다.
- DB 인스턴스를 만드는 사용자는 다음 작업을 수행할 수 있는 권한이 있어야 합니다.
  - `ec2:AssociateRouteTable`
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateRouteTable`
  - `ec2:CreateSubnet`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeRouteTables`
  - `ec2:DescribeSecurityGroups`
  - `ec2:DescribeSubnets`
  - `ec2:ModifyNetworkInterfaceAttribute`
  - `ec2:RevokeSecurityGroupEgress`

이 옵션을 사용하면 프라이빗 DB 인스턴스가 생성됩니다. DB 인스턴스는 프라이빗 서브넷만 있는 DB 서브넷 그룹을 사용하여 VPC 내 리소스에 대한 액세스를 제한합니다.

EC2 인스턴스를 DB 인스턴스에 연결하려면 데이터베이스 생성 페이지의 연결 섹션에서 EC2 컴퓨팅 리소스에 연결을 선택합니다.



Connect to an EC2 compute resource(EC2 컴퓨팅 리소스에 연결)를 선택하면 RDS가 다음 옵션을 자동으로 설정합니다. Don't connect to an EC2 compute resource(EC2 컴퓨팅 리소스에 연결하지 않음)를 선택하여 EC2 인스턴스와의 연결을 설정하지 않도록 선택하지 않는 한 이러한 설정을 변경할 수 없습니다.

콘솔 옵션	자동 설정
네트워크 유형	RDS는 네트워크 유형을 IPv4로 설정합니다. 현재 듀얼 스택 모드는 EC2 인스턴스와 DB 인스턴스 간의 연결을 설정할 때 지원되지 않습니다.
Virtual Private Cloud(VPC)	RDS는 VPC를 EC2 인스턴스와 연결된 VPC로 설정합니다.
DB 서브넷 그룹	RDS에는 EC2 인스턴스와 동일한 가용 영역에 프라이빗 서브넷이 있는 DB 서브넷 그룹이 필요합니다. 이 요구 사항을 충족하는 DB 서브넷 그룹이 있다면 RDS는 기존 DB 서브넷 그룹을 사용합니다. 기본적으로 이 옵션은 Automatic setup(자동 설정)으로 설정되어 있습니다. Automatic setup(자동 설정)을 선택하고 이 요구 사항을 충족하는 DB 서브넷 그룹이 없으면 다음 작업이 수행됩니다. RDS는 가용

콘솔 옵션	자동 설정
	영역 중 하나가 EC2 인스턴스와 동일한 3개의 가용 영역에서 3개의 사용 가능한 프라이빗 서브넷을 사용합니다. 가용 영역에서 프라이빗 서브넷을 사용할 수 없는 경우 RDS는 가용 영역에 프라이빗 서브넷을 생성합니다. 그런 다음 RDS는 DB 서브넷 그룹을 생성합니다. 프라이빗 서브넷을 사용할 수 있는 경우 RDS는 서브넷과 연결된 라우팅 테이블을 사용하고 생성된 모든 서브넷을 이 라우팅 테이블에 추가합니다. 프라이빗 서브넷을 사용할 수 없는 경우 RDS는 인터넷 게이트웨이 액세스가 없는 라우팅 테이블을 생성하고 생성한 서브넷을 라우팅 테이블에 추가합니다. RDS를 사용하면 기존 DB 서브넷 그룹도 사용할 수 있습니다. 선택한 기존 DB 서브넷 그룹을 사용하려면 Choose existing(기존 항목 선택)을 선택합니다.
공개 액세스(Public access)	RDS는 DB 인스턴스에 공개적으로 액세스할 수 없도록 아니요를 선택합니다. 보안을 위해 데이터베이스를 비공개로 유지하고 인터넷에서 액세스할 수 없도록 하는 것이 가장 좋습니다.

콘솔 옵션	자동 설정
VPC 보안 그룹(방화벽)	RDS는 DB 인스턴스와 연결된 새 보안 그룹을 생성합니다. 보안 그룹의 이름은 `rds-ec2-`*n*이며, 여기서 *n*은 숫자입니다. 이 보안 그룹에는 EC2 VPC 보안 그룹(방화벽)이 소스인 인바운드 규칙이 포함됩니다. DB 인스턴스와 연결된 이 보안 그룹을 통해 EC2 인스턴스가 DB 인스턴스에 액세스할 수 있습니다. RDS는 EC2 인스턴스와 연결된 새 보안 그룹도 생성합니다. 보안 그룹의 이름은 `ec2-rds-`*n*이며, 여기서 *n*은 숫자입니다. 이 보안 그룹에는 DB 인스턴스의 VPC 보안 그룹을 소스로 하는 아웃바운드 규칙이 포함됩니다. 이 보안 그룹을 사용하면 EC2 인스턴스가 DB 인스턴스로 트래픽을 보낼 수 있습니다. Create new(새로 생성)를 선택하고 새 보안 그룹의 이름을 입력하여 다른 새 보안 그룹을 추가할 수 있습니다. Choose existing(기존 항목 선택)을 선택하고 추가할 보안 그룹을 선택하여 기존 보안 그룹을 추가할 수 있습니다.
가용 영역	가용성 및 내구성(단일 AZ 배포)에서 단일 DB 인스턴스를 선택한 경우 RDS는 EC2 인스턴스의 가용 영역을 선택합니다. 가용성 및 내구성에서 다중 AZ DB 인스턴스를 선택한 경우(다중 AZ DB 인스턴스 배포), RDS는 배포에서 단일 DB 인스턴스에 대해 EC2 인스턴스의 가용 영역을 선택합니다. RDS는 다른 DB 인스턴스에 대해 서로 다른 가용 영역을 임의로 선택합니다. 기본 DB 인스턴스 또는 대기 복제본은 EC2 인스턴스와 동일한 가용 영역에서 생성됩니다. 다중 AZ DB 인스턴스를 선택할 때 DB 인스턴스와 EC2 인스턴스가 서로 다른 가용 영역에 있는 경우 교차 가용 영역 비용이 발생할 가능성이 있습니다.

이러한 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 섹션을 참조하세요.

DB 인스턴스를 생성한 후에 이러한 설정을 변경할 경우 변경 사항이 EC2 인스턴스와 DB 인스턴스 간의 연결에 영향을 미칠 수 있습니다.

네트워크 수동 구성

동일한 VPC의 EC2 인스턴스 이외의 리소스에서 DB 인스턴스에 연결하려는 경우 네트워크 연결을 수동으로 구성합니다. AWS Management Console을 사용하여 DB 인스턴스를 생성하는 경우에는 Amazon RDS에서 VPC를 자동으로 생성할 수 있습니다. 또는 DB 인스턴스에서 기존 VPC를 사용하거나 새 VPC를 생성할 수 있습니다. 어떤 접근 방식이든 RDS DB 인스턴스와 함께 VPC를 사용하려면 2개 이상의 가용 영역마다 서브넷이 1개 이상 있어야 합니다.

기본적으로 Amazon RDS는 DB 인스턴스를 자동으로 가용 영역에 생성합니다. 특정 가용 영역을 선택하려면 가용성 및 내구성 설정을 단일 DB 인스턴스로 변경해야 합니다. 이렇게 하면 VPC의 가용 영역 중에서 선택할 수 있는 가용 영역 설정이 노출됩니다. 하지만 다중 AZ 배포를 선택하는 경우 RDS는 기본 또는 라이터 DB 인스턴스의 가용 영역을 자동으로 선택하고 가용 영역 설정이 표시되지 않습니다.

경우에 따라 기본 VPC가 없거나 VPC를 생성하지 않았을 수 있습니다. 이러한 경우에는 콘솔을 사용해 DB 인스턴스를 생성할 때 Amazon RDS에서 자동으로 VPC를 생성하도록 할 수 있습니다. 그렇지 않으면 다음을 수행하세요.

- DB 인스턴스를 배포하려는 AWS 리전에서 두 개 이상의 가용 영역에 각각 한 개 이상의 서브넷을 갖는 VPC를 생성합니다. 자세한 내용은 VPC에서 DB 인스턴스를 사용한 작업 및 자습서: DB 인스턴스에 사용할 Amazon VPC 생성(IPv4 전용) 섹션을 참조하세요.
- DB 인스턴스에 대한 연결 권한을 부여할 수 있도록 VPC 보안 그룹을 지정합니다. 자세한 내용은 보안 그룹을 생성하여 VPC 내부의 DB 인스턴스에 대한 액세스를 제공 및 보안 그룹을 통한 액세스 제어 섹션을 참조하세요.
- DB 인스턴스에서 VPC의 서브넷 2개 이상을 사용할 수 있도록 RDS DB 서브넷 그룹을 지정합니다. 자세한 내용은 DB 서브넷 그룹을 사용한 작업 섹션을 참조하세요.

DB 인스턴스와 동일한 VPC에 있지 않은 리소스에 연결하려는 경우 VPC에서 DB 인스턴스에 액세스하는 시나리오에서 적절한 시나리오를 참조하십시오.

추가 사전 조건

DB 인스턴스를 만들려면 먼저 다음과 같은 추가 사전 조건을 고려하십시오.

- AWS Identity and Access Management(IAM) 보안 인증 정보를 사용하여 AWS에 연결하는 경우 AWS 계정에 특정 IAM 정책이 있어야 합니다. 이 정책은 Amazon RDS 작업을 수행하는 데 필요한 권한을 부여합니다. 자세한 내용은 Amazon RDS의 자격 증명 및 액세스 관리 섹션을 참조하세요.

IAM을 사용하여 RDS 콘솔에 액세스하려면 IAM 사용자 자격 증명으로 AWS Management Console에 로그인합니다. 그런 다음 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔로 이동합니다.

- DB 인스턴스에 대한 구성 파라미터를 사용자 지정하려면 필요한 파라미터 설정으로 DB 파라미터 그룹을 지정합니다. DB 파라미터 그룹의 생성 및 수정에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.

> **Important**
>
> Amazon RDS for Db2용 BYOL 모델을 사용하는 경우, DB 인스턴스를 만들기 전에 먼저 IBM Site ID 및 IBM Customer ID를 포함하는 사용자 지정 파라미터 그룹을 만들어야 합니다. 자세한 내용은 Db2에 기존 보유 라이선스 사용 섹션을 참조하세요.

- DB 인스턴스에 지정할 TCP/IP 포트 번호를 정합니다. 일부 기업에서는 방화벽이 RDS DB 인스턴스에 대한 기본 포트 연결을 차단하는 경우도 있습니다. 이처럼 기업 방화벽이 기본 포트를 차단할 경우 DB 인스턴스에 다른 포트를 선택해야 합니다. Amazon RDS DB 엔진의 기본 포트는 다음과 같습니다.

RDS for Db2	RDS for MariaDB	RDS for MySQL	RDS for Oracle	RDS for PostgreSQL	RDS for SQL Server
50000	3306	3306	1521	5432	1433

RDS for SQL Server의 경우 다음 포트는 예약되어 있고 DB 인스턴스 `1234`, `1434`, `3260`, `3343`, `3389`, `47001`, 및 `49152-49156`을 생성할 때 사용할 수 없습니다.

DB 인스턴스 생성

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 Amazon RDS DB 인스턴스를 생성할 수 있습니다.

> **Note**
>
> RDS for Db2의 경우 RDS for Db2 DB 인스턴스를 생성하기 전에 라이선스 모델에 필요한 항목을 설정하는 것이 좋습니다. 자세한 내용은 Amazon RDS for Db2 라이선스 옵션 섹션을 참조하세요.

콘솔

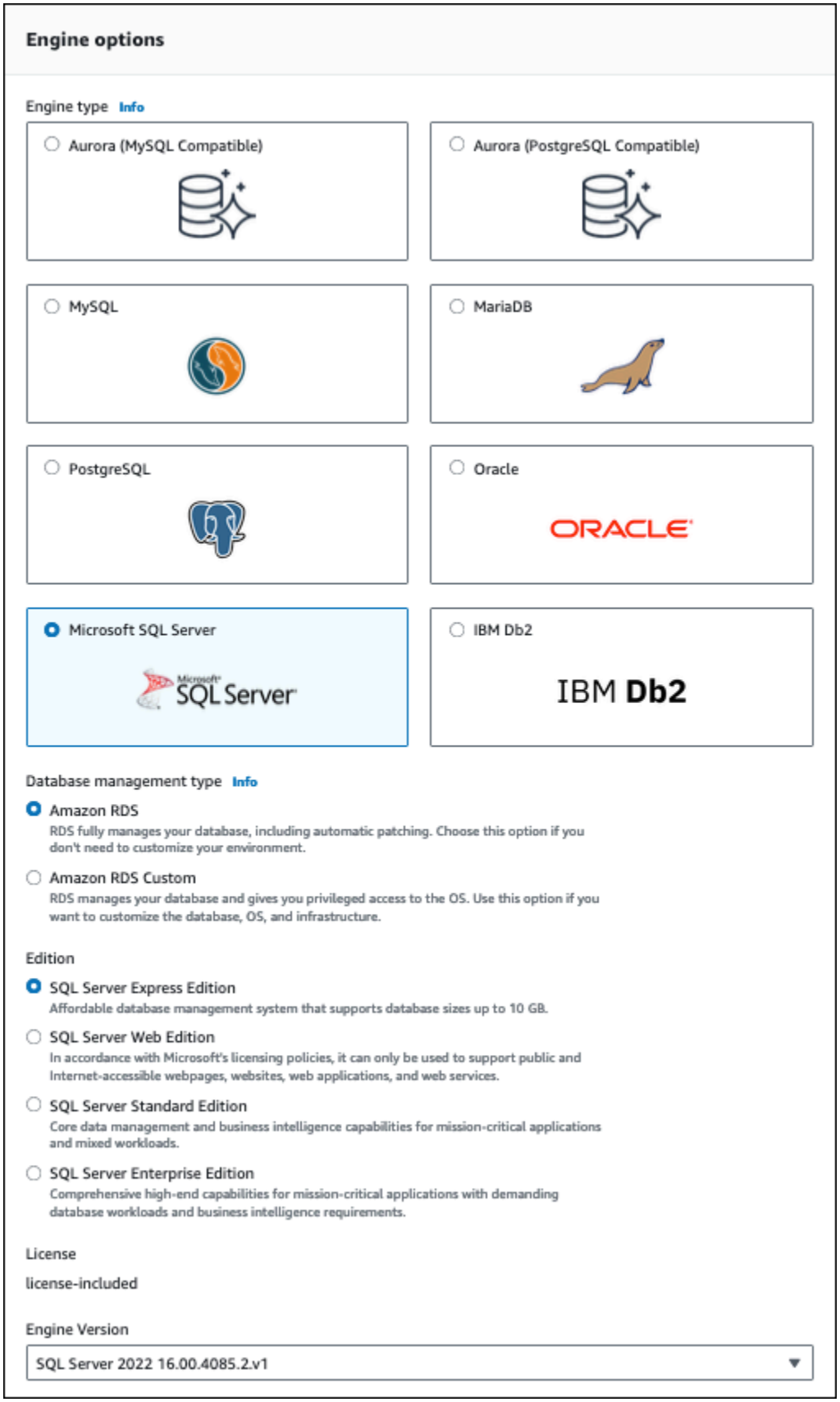
간편 생성(Easy Create)을 활성화하거나 활성화하지 않고 AWS Management Console을 사용하여 DB 인스턴스를 생성할 수 있습니다. Easy create(간편 생성)를 활성화한 경우에는 DB 엔진 유형, DB 인스턴스 크기 및 DB 인스턴스 식별자만 지정합니다. Easy create(간편 생성)는 다른 구성 옵션에서도 기본 설정을 사용합니다. Easy create(간편 생성)가 활성화되지 않은 경우에는 데이터베이스를 생성할 때 가용성, 보안, 백업 및 유지 관리에 대한 옵션을 포함하여 더 많은 구성 옵션을 지정합니다.

> **Note**
>
> 다음 절차에서는 표준 생성(Standard Create)이 활성화되고 간편 생성(Easy Create)이 활성화되지 않습니다. 이 절차에서는 Microsoft SQL Server를 예로 사용합니다.
> 간편 생성(Easy Create)을 사용하여 각 엔진의 샘플 DB 인스턴스를 생성하고 연결하는 방법을 안내하는 예제는 Amazon RDS 시작하기 섹션을 참조하세요.

DB 인스턴스를 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 콘솔의 오른쪽 상단에서 DB 인스턴스를 생성하려는 AWS 리전을 선택합니다.
3. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
4. 데이터베이스 생성에서 표준 생성을 선택합니다.
5. 엔진 유형에서 IBM Db2, MariaDB, Microsoft SQL Server, MySQL, Oracle 또는 PostgreSQL을 선택합니다.

   여기서는 Microsoft SQL Server가 표시됩니다.


Engine options
Engine type Info
Aurora (MySQL Compatible)
Aurora (PostgreSQL Compatible)
MySQL
MariaDB
PostgreSQL
Oracle
ORACLE
Microsoft SQL Server
SQL Server
IBM Db2
IBM Db2
Database management type Info
Amazon RDS
RDS fully manages your database, including automatic patching. Choose this option if you don't need to customize your environment.
Amazon RDS Custom
RDS manages your database and gives you privileged access to the OS. Use this option if you want to customize the database, OS, and infrastructure.
Edition
SQL Server Express Edition
Affordable database management system that supports database sizes up to 10 GB.
SQL Server Web Edition
In accordance with Microsoft's licensing policies, it can only be used to support public and Internet-accessible webpages, websites, web applications, and web services.
SQL Server Standard Edition
Core data management and business intelligence capabilities for mission-critical applications and mixed workloads.
SQL Server Enterprise Edition
Comprehensive high-end capabilities for mission-critical applications with demanding database workloads and business intelligence requirements.
License
license-included
Engine Version
SQL Server 2022 16.00.4085.2.v1

6. Oracle 또는 SQL Server를 사용하는 경우 데이터베이스 관리 유형에서 Amazon RDS 또는 Amazon RDS Custom을 선택합니다.

   Amazon RDS가 여기에 표시됩니다. RDS Custom에 대한 자세한 내용은 Amazon RDS Custom 섹션을 참조하세요.

7. Oracle 또는 SQL Server를 사용하는 경우 에디션에서 사용할 DB 엔진 에디션을 선택합니다.

   MySQL에는 에디션에 대한 옵션이 하나 뿐이며, MariaDB와 PostgreSQL에는 없습니다.

8. 버전에서 엔진 버전을 선택합니다.

9. 템플릿에서 사용 사례에 맞는 템플릿을 선택합니다. 프로덕션을 선택하면 이후 단계에서 다음 설정이 미리 선택됩니다.

   - 다중 AZ 장애 조치 옵션
   - 프로비저닝된 IOPS(io1) 스토리지 옵션
   - 삭제 방지 활성화 옵션

   어떤 프로덕션 환경이든 이 기능을 권장합니다.

   > **Note**
   >
   > 템플릿 옵션은 에디션별로 다릅니다.

10. Settings(설정) 섹션에서 Credential Settings(보안 인증 정보 설정)를 엽니다. 뒤이어 다음과 같이 하세요.

    a. (선택 사항) 마스터 사용자 이름(Master username) 값을 변경합니다.

    b. 다음 자격 증명 관리 옵션 중 하나를 선택합니다.

       - AWS Secrets Manager에서 관리됨

         암호화 키 선택에서 Secrets Manager가 만드는 KMS 키 또는 사용자가 만든 키를 선택합니다.

> **Note**
>
> 자격 증명을 관리하기 위한 가장 안전한 기법으로 AWS Secrets Manager를 사용하는 것이 좋습니다. 추가 요금이 발생합니다. 자세한 내용은 Amazon RDS 및 AWS Secrets Manager를 통한 암호 관리 섹션을 참조하세요.

- 자체 관리형

  암호를 지정하려면 암호 자동 생성 확인란을 선택 취소합니다(선택되어 있는 경우). 마스터 암호 및 마스터 암호 확인에 동일한 암호를 입력합니다.

11. (선택 사항) 이 DB 인스턴스의 컴퓨팅 리소스에 대한 연결을 설정합니다.

    DB 클러스터를 생성하는 동안 Amazon EC2 인스턴스와 새 DB 인스턴스 간의 연결을 구성할 수 있습니다. 자세한 내용은 EC2 인스턴스와의 자동 네트워크 연결 구성 섹션을 참조하세요.

12. VPC 보안 그룹(방화벽) 아래의 연결 섹션에서 새로 만들기를 선택하면 로컬 컴퓨터의 IP 주소가 데이터베이스에 액세스할 수 있도록 허용하는 인바운드 규칙과 함께 VPC 보안 그룹이 생성됩니다.

13. 나머지 섹션에서 DB 인스턴스 설정을 지정합니다. 각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하십시오.

14. 데이터베이스 생성을 선택합니다.

    자동 생성된 암호를 사용하기로 한 경우에는 데이터베이스 페이지에 View credential details(자격 증명 세부 정보 보기) 버튼이 나타납니다.

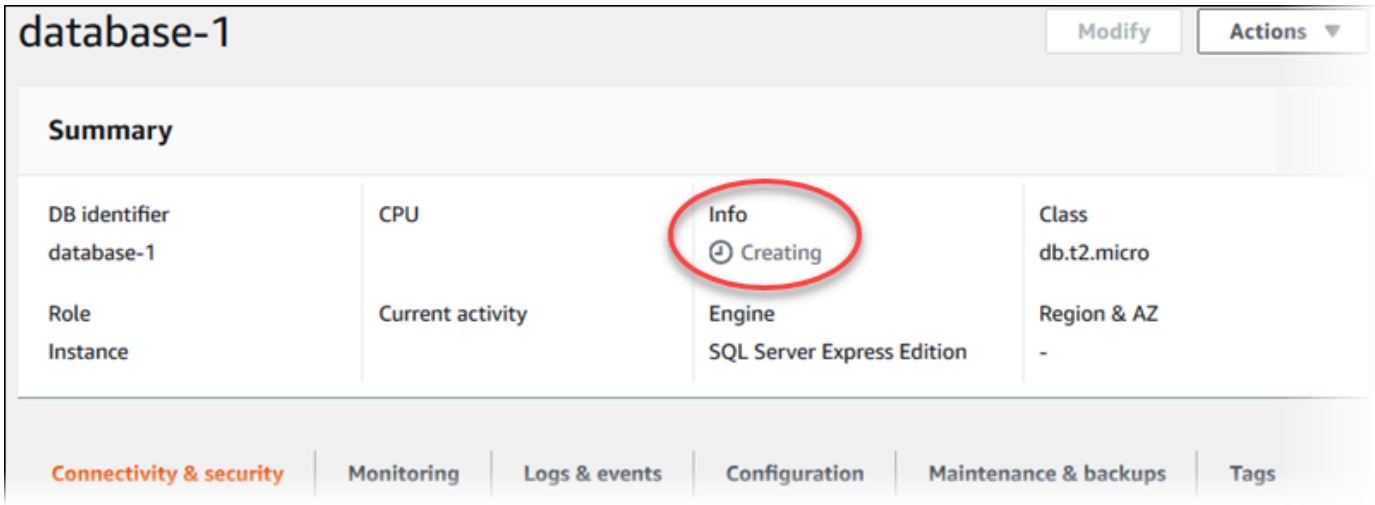
    DB 인스턴스의 마스터 사용자 이름 및 암호를 보려면 View credential details(자격 증명 세부 정보 보기)를 선택합니다.

    DB 인스턴스를 마스터 사용자로 연결하려면 화면에 나타난 사용자 이름과 암호를 사용합니다.

    > **Important**
    >
    > 마스터 사용자 암호를 다시 볼 수는 없습니다. 따라서 기록을 해두지 않으면 이를 변경해야 합니다. DB 인스턴스가 사용 가능한 상태가 되고 난 후에 마스터 사용자 암호를 변경해야 하는 경우에는 DB 인스턴스를 수정하여 암호를 변경합니다. DB 인스턴스 변경에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

15. 데이터베이스에서 새 DB 인스턴스의 이름을 선택합니다.

RDS 콘솔에 새 DB 인스턴스의 세부 정보가 표시됩니다. DB 인스턴스를 만들고 사용할 준비가 될 때까지 DB 인스턴스의 상태는 Creating입니다. 상태가 사용 가능으로 변경되면 DB 인스턴스에 연결할 수 있습니다. 할당된 DB 인스턴스 클래스와 스토리지에 따라 새 인스턴스를 사용할 수 있을 때까지 몇 분 정도 걸릴 수 있습니다.



AWS CLI

> **Note**
>
> AWS Marketplace를 통한 Db2 라이선스를 사용하려면 먼저 AWS Marketplace를 구독하고 AWS Management Console를 사용하여 IBM에 등록해야 합니다. 자세한 내용은 Db2 Marketplace 리스팅 구독 및 IBM으로 등록 섹션을 참조하세요.

AWS CLI를 사용하여 DB 인스턴스를 생성하려면 다음 파라미터와 함께 create-db-instance 명령을 호출합니다.

- `--db-instance-identifier`
- `--db-instance-class`
- `--vpc-security-group-ids`
- `--db-subnet-group`
- `--engine`
- `--master-username`
- `--master-user-password` 또는 `--manage-master-user-password`

- `--allocated-storage`
- `--backup-retention-period`

각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하세요.

이 예에서는 Microsoft SQL Server를 사용합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance \
    --engine sqlserver-se \
    --db-instance-identifier mymsftsqlserver \
    --allocated-storage 250 \
    --db-instance-class db.t3.large \
    --vpc-security-group-ids mysecuritygroup \
    --db-subnet-group mydbsubnetgroup \
    --master-username masterawsuser \
    --manage-master-user-password \
    --backup-retention-period 3
```

Windows의 경우:

```
aws rds create-db-instance ^
    --engine sqlserver-se ^
    --db-instance-identifier mydbinstance ^
    --allocated-storage 250 ^
    --db-instance-class db.t3.large ^
    --vpc-security-group-ids mysecuritygroup ^
    --db-subnet-group mydbsubnetgroup ^
    --master-username masterawsuser ^
    --manage-master-user-password ^
    --backup-retention-period 3
```

다음과 비슷한 출력이 생성됩니다.

```
DBINSTANCE  mydbinstance  db.t3.large  sqlserver-se  250  sa  creating  3  ****  n
 10.50.2789
SECGROUP  default  active
PARAMGRP  default.sqlserver-se-14  in-sync
```

RDS API

> **Note**
>
> AWS Marketplace를 통한 Db2 라이선스를 사용하려면 먼저 AWS Marketplace를 구독하고 AWS Management Console를 사용하여 IBM에 등록해야 합니다. 자세한 내용은 Db2 Marketplace 리스팅 구독 및 IBM으로 등록 섹션을 참조하세요.

Amazon RDS API를 사용하여 DB 인스턴스를 생성하려면 CreateDBInstance 작업을 호출합니다.

각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하십시오.

DB 인스턴스에 대한 설정

다음 표에는 DB 인스턴스를 생성할 때 선택하는 설정에 대한 세부 정보가 나와 있습니다. 이 표에는 각 설정이 지원되는 DB 엔진도 나와 있습니다.

콘솔, create-db-instance CLI 명령 또는 CreateDBInstance RDS API 작업을 사용하여 DB 인스턴스를 생성할 수 있습니다.

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

할당된 스토리지	DB 인스턴스에 할당할 스토리지 양(기가바이트 단위)입니다. 경우에 따라 DB 인스턴스에 대해 데이터베이스의 크기보다 많은 양의 스토리지를 할당하면 I/O 성능을 개선할 수 있습니다.
	
	자세한 내용은 Amazon RDS DB 인스턴스 스토리지 섹션을 참조하세요.
		CLI 옵션:
		
		`--allocated-storage`
		
		API 파라미터:
		
		`AllocatedStorage`
			모두

아키텍처 설정	Oracle 멀티테넌트 아키텍처를 선택하면 RDS for Oracle에서 컨테이너 데이터베이스(CDB)가 생성됩니다. 이 옵션을 선택하지 않으면 RDS for Oracle에서 비 CDB가 생성됩니다. 비CDB에는 기존의 Oracle 데이터베이스 아키텍처가 사용됩니다.	CLI 옵션: `--engine oracle-ee-cdb` (Oracle 멀티테넌트)	Oracle

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진
	CDB에는 플러그형 데이터베이스(PDB)가 포함될 수 있지만 비CDB에는 포함될 수 없습니다. Oracle Database 21c는 CDB 아키텍처만 사용합니다. Oracle Database 19c는 CDB 또는 비CDB 아키텍처를 사용할 수 있습니다. Oracle Database 19c 이전의 릴리스는 비CDB 아키텍처만 사용합니다. 자세한 내용은 RDS for Oracle CDB 개요 섹션을 참조하세요.	`--engine oracle-se2-cdb` (Oracle 멀티테넌트) `--engine oracle-ee` (기존) `--engine oracle-se2` (기존) API 파라미터: `Engine`	

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진
아키텍처 구성	이러한 설정은 아키텍처 설정으로 Oracle 멀티테넌트 아키텍처를 선택한 경우에만 유효합니다. 다음 추가 설정 중 하나를 선택합니다. • 다중 테넌트 구성 에서 RDS for Oracle CDB 인스턴스에는 데이터베이스 에디션 및 필요한 옵션 라이선스에 따라 1~30개의 테넌트 데이터베이스가 포함될 수 있습니다. Oracle 데이터베이스의 경우 테넌트 데이터베이스는 PDB입니다. 애플리케이션 PDB 및 프록시 PDB는 지원되지 않습니다. DB 인스턴스는 1개의 초기 테넌트 데이터베이스와 함께 생성됩니다. 테넌트 데이터베이스 이름, 테넌트 데이터베이스 마스터 사용자 이름, 테넌트 데이터베이스 마스터 암호 및 테넌트 데이터베이스 문자 집합의 값을 선택합니다. 다중 테넌트 구성은 영구적입니다. 따라서 다중 테넌트 구성을 단일 테넌트 구성으로 다시 변환할 수 없습니다. 다중 테넌트 구성에 지원되는 최소 릴리스 업데이트(RU)는 19.0.0.0.ru-2022-01.rur-2022.r1입니다. **Note** Amazon RDS 구성은 Oracle DB 엔진뿐만 아니라 RDS 플랫폼의 기능이기 때문에 '멀티테넌트'가 아닌 '다중 테넌트'라	CLI 옵션: `--multi-tenant` (다중 테넌트 구성) `--no-multi-tenant`(단일 테넌트 구성) API 파라미터: `MultiTenant`	Oracle

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진
	고 합니다. 마찬가지로 RDS 용어 '테넌트'는 Oracle PDB뿐만 아니라 RDS 구성의 모든 테넌트를 나타냅니다. RDS 설명서에서 하이픈이 포함되지 않은 용어 'Oracle 멀티테넌트'는 온프레미스 및 RDS 배포 모두와 호환되는 Oracle 데이터베이스 CDB 아키텍처만을 의미합니다. • 단일 테넌트 구성의 경우 RDS for Oracle CDB에는 PDB 1개가 포함됩니다. 이는 CDB 생성 시 기본 구성입니다. 초기 PDB를 삭제하거나 PDB를 더 추가할 수 없습니다. 나중에 CDB의 단일 테넌트 구성을 다중 테넌트 구성으로 변환할 수 있지만 다시 단일 테넌트 구성으로 변환할 수는 없습니다. 어떤 구성을 선택하든 CDB에는 처음의 PDB 한 개가 포함됩니다. 다중 테넌트 구성에서는 나중에 RDS API를 사용하여 더 많은 PDB를 만들 수 있습니다. 자세한 내용은 RDS for Oracle CDB 개요 섹션을 참조하세요.		

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

자동 마이너 버전 업그레이드	DB 엔진의 기본 마이너 버전 업그레이드가 제공되면 DB 인스턴스가 자동으로 이를 받을 수 있게 하려면 마이너 버전 자동 업그레이드 사용을 선택합니다. 이는 기본 설정 동작입니다. Amazon RDS는 유지 관리 기간에 자동 마이너 버전 업그레이드를 수행합니다. 마이너 버전 자동 업그레이드 사용을 선택하지 않으면 새 마이너 버전이 출시될 때 DB 인스턴스가 자동으로 업그레이드되지 않습니다.
	
	자세한 내용은 마이너 엔진 버전 자동 업그레이드 섹션을 참조하세요.
		CLI 옵션:
		
		`--auto-minor-version-upgrade`
		
		`--no-auto-minor-version-upgrade`
		
		API 파라미터:
		
		`AutoMinorVersionUpgrade`
			모두

가용 영역	DB 인스턴스의 가용 영역입니다. 가용 영역을 지정하지 않으려면 기본값으로 No Preference를 사용합니다.
	
	자세한 내용은 리전, 가용 영역 및 로컬 영역 섹션을 참조하세요.
		CLI 옵션:
		
		`--availability-zone`
		
		API 파라미터:
		
		`AvailabilityZone`
			모두

AWS KMS key	암호화를 암호화 활성화로 설정한 경우에만 사용할 수 있습니다. 이 DB 인스턴스를 암호화하는 데 사용할 AWS KMS key을(를) 선택합니다. 자세한 내용은 Amazon RDS 리소스 암호화 섹션을 참조하세요.	CLI 옵션:
		
		`--kms-key-id`
		
		API 파라미터:
		
		`KmsKeyId`
			모두

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진
AWS License Manager 구성	AWS License Manager 라이선스 구성의 이름을 입력합니다. 이름은 100자 이하여야 하며 a~z, A~Z, 0~9만 포함해야 합니다. 자세한 내용은 the section called “AWS License Manager과 통합” 섹션을 참조하세요.	CLI 옵션: 자세한 내용은 AWS License Manager CLI 섹션을 참조하세요. API 파라미터: 자세한 내용은 AWS License Manager API 섹션을 참조하세요.	Db2
백업 복제	재해 복구를 위해 추가 리전에 백업을 생성하려면 다른 AWS 리전에서 복제 활성화(Enable replication in another AWS Region)를 선택합니다. 그런 다음 추가 백업을 위한 대상 리전(Destination Region)을 선택합니다.	DB 인스턴스를 생성할 때는 사용할 수 없습니다. AWS CLI 또는 RDS API를 사용하여 교차 리전 백업을 활성화하는 방법에 대한 자세한 내용은 교차 리전 자동 백업 활성화 섹션을 참조하세요.	Oracle PostgreSQL SQL Server

백업 보관 기간	DB 인스턴스의 자동 백업을 보존할 기간(단위: 일)입니다. 중요한 DB 인스턴스라면 이 값을 **1** 이상으로 설정합니다.
	
	자세한 내용은 백업 소개 섹션을 참조하세요.
		CLI 옵션:
		
		`--backup-retention-period`
		
		API 파라미터:
		
		`BackupRetentionPeriod`
			모두

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

백업 대상	AWS 클라우드를 선택하여 자동 백업 및 수동 스냅샷을 상위 AWS 리전에 저장합니다. Outposts(온프레미스)(Outposts (on-premises))를 선택하여 Outpost에 로컬로 저장합니다.
	
	이 옵션 설정은 Outposts 기반 RDS에만 적용됩니다. 자세한 내용은 Amazon RDS on AWS Outposts DB 인스턴스 생성 섹션을 참조하세요.
		CLI 옵션:
		
		`--backup-target`
		
		API 파라미터:
		
		`BackupTarget`
			MySQL, PostgreSQL, SQL Server

백업 기간	Amazon RDS가 자동으로 DB 인스턴스를 백업하는 기간입니다. 데이터베이스를 백업할 특정 기간을 지정하지 않으려면 기본값으로 기본 설정 없음을 사용합니다.
	
	자세한 내용은 백업 소개 섹션을 참조하세요.
		CLI 옵션:
		
		`--preferred-backup-window`
		
		API 파라미터:
		
		`PreferredBackupWindow`
			모두

인증 기관	DB 인스턴스에서 사용하는 서버 인증서의 CA(인증 기관)입니다.
	
	자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 섹션을 참조하세요.
		CLI 옵션:
		
		`--ca-certificate-identifier`
		
		RDS API 파라미터:
		
		`CACertificateIdentifier`
			모두

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

문자 집합	DB 인스턴스에 대한 문자 세트입니다. DB 문자 집합에 대한 기본값 AL32UTF8은 Unicode 5.0 UTF-8 Universal 문자 집합을 나타냅니다. DB 인스턴스를 생성한 후에는 DB 문자 집합을 변경할 수 없습니다.
	
	단일 테넌트 구성에서 기본이 아닌 DB 문자 집합은 CDB가 아니라 PDB에만 영향을 줍니다. 자세한 내용은 CDB 아키텍처의 단일 테넌트 구성 섹션을 참조하세요.
	
	DB 문자 집합은 NCHAR 문자 집합이라고 하는 국가별 문자 집합과 다릅니다. DB 문자 집합과 달리 NCHAR 문자 집합은 데이터베이스 메타데이터에 영향을 주지 않고 NCHAR 데이터 형식(NCHAR, NVARCHAR2 및 NCLOB) 열에 대한 인코딩을 지정합니다.
	
	자세한 내용은 RDS for Oracle 문자 집합 섹션을 참조하세요.
		CLI 옵션:
		
		`--character-set-name`
		
		API 파라미터:
		
		`CharacterSetName`
			Oracle

콜레이션	DB 인스턴스에 대한 서버 수준 콜레이션입니다.
	
	자세한 내용은 Microsoft SQL Server의 서버 수준 콜레이션 섹션을 참조하세요.
		CLI 옵션:
		
		`--character-set-name`
		
		API 파라미터:
		
		`CharacterSetName`
			SQL 서버

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

스냅샷으로 태그 복사	이 옵션은 스냅샷을 생성할 때 DB 인스턴스 태그를 DB 스냅샷에 복사합니다.
	
	자세한 내용은 Amazon RDS 리소스에 태그 지정 섹션을 참조하세요.
		CLI 옵션:
		
		`--copy-tags-to-snapshot`
		
		`--no-copy-tags-to-snapshot`
		
		RDS API 파라미터:
		
		`CopyTagsToSnapshot`
			모두

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

데이터베이스 인증	사용하려는 데이터베이스 인증 옵션입니다.
	
	데이터베이스 암호로만 데이터베이스 사용자를 인증할 수 있도록 Password authentication(암호 인증)을 선택합니다.
	
	사용자 및 역할을 통해 데이터베이스 암호 및 사용자 자격 증명으로 데이터베이스 사용자를 인증할 수 있도록 Password and IAM DB authentication(암호 및 IAM DB 인증)을 선택합니다. 자세한 내용은 MariaDB, MySQL 및 PostgreSQL IAM 데이터베이스 인증 섹션을 참조하세요. 이 옵션은 MySQL 및 PostgreSQL에 대해서만 지원됩니다.
	
	AWS Managed Microsoft AD로 생성한 AWS Directory Service를 통해 데이터베이스 암호 및 Kerberos 인증으로 데이터베이스 사용자를 인증할 수 있도록 [암호 및 Kerberos 인증(Password and Kerberos authentication)]을 선택합니다. 그다음에는 디렉터리를 선택하거나 새 디렉터리 생성을 선택합니다.
	
	자세한 내용은 다음 중 하나를 참조하십시오.
	
	• Amazon RDS for Db2에 대한 Kerberos 인증 사용
	• Amazon RDS for MySQL에 Kerberos 인증 사용
		IAM:
		
		CLI 옵션:
		
		`--enable-iam-database-authentication`
		
		`--no-enable-iam-database-authentication`
		
		RDS API 파라미터:
		
		`EnableIAMDatabaseAuthentication`
		
		Kerberos:
		
		CLI 옵션:
		
		`--domain`
		
		`--domain-iam-role-name`
		
		RDS API 파라미터:
		
		`Domain`
		
		`DomainIAMRoleName`
			인증 유형에 따라 다름

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진
	• Amazon RDS for Oracle에 대한 Kerberos 인증 구성 • Amazon RDS for PostgreSQL과 함께 Kerberos 인증 사용		
데이터베이스 관리 유형	환경을 사용자 지정할 필요가 없는 경우 Amazon RDS를 선택합니다. 데이터베이스, OS 및 인프라를 사용자 지정하려는 경우 Amazon RDS Custom을 선택합니다. 자세한 내용은 Amazon RDS Custom 섹션을 참조하세요.	CLI 및 API의 경우 데이터베이스 엔진 유형을 지정합니다.	Oracle SQL Server

데이터베이스 포트	DB 인스턴스에 액세스하는 데 사용할 포트입니다. 기본 포트가 표시됩니다.
	Note
	일부 기업에서는 방화벽이 기본 MariaDB, MySQL 및 PostgreSQL 포트에 대한 연결을 차단합니다. 이처럼 기업 방화벽이 기본 포트를 차단할 경우 DB 인스턴스에 다른 포트를 입력해야 합니다.
		CLI 옵션:
		`--port`
		RDS API 파라미터:
		`Port`
			모두

DB 엔진 버전	사용할 데이터베이스 엔진의 버전입니다.	CLI 옵션:
		`--engine-version`
		RDS API 파라미터:
		`EngineVersion`
			모두

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

DB 인스턴스 클래스	DB 인스턴스에 대한 구성입니다. 예를 들어, db.t3.small DB 인스턴스 클래스에는 2GiB 메모리, vCPU 2개, 가상 코어 1개, 가변 ECU 1개, 중간 정도의 I/O 용량이 있습니다.
	
	가능하면 일반 쿼리 작업 세트가 메모리에 상주할 수 있을 정도로 큰 DB 인스턴스 클래스를 선택합니다. 작업 세트가 메모리에 상주할 경우 시스템의 디스크 쓰기가 불필요하여 성능이 향상됩니다. 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요.
	
	RDS for Oracle에서 [추가 메모리 구성 포함(Include additional memory configurations)]을 선택할 수 있습니다. 이러한 구성은 vCPU에 대한 메모리 비율이 높도록 최적화되었습니다. 예를 들어, db.r5.6xlarge.tpc2.mem4x는 코어당 2개의 스레드(tpc2)가 있고 표준 db.r5.6xlarge DB 인스턴스의 메모리가 4배인 db.r5.8x DB 인스턴스입니다. 자세한 내용은 RDS for Oracle DB 인스턴스 클래스 섹션을 참조하세요.
		CLI 옵션:
		
		`--db-instance-class`
		
		RDS API 파라미터:
		
		`DBInstanceClass`
			모두

DB 인스턴스 식별자	DB 인스턴스의 이름입니다. 온프레미스 서버와 동일한 방식으로 DB 인스턴스의 이름을 지정합니다. DB 인스턴스 식별자는 최대 63자의 영숫자 문자를 포함할 수 있으며 선택한 AWS 리전의 계정에 대해 고유해야 합니다.	CLI 옵션:
		
		`--db-instance-identifier`
		
		RDS API 파라미터:
		
		`DBInstanceIdentifier`
			모두

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

DB 파라미터 그룹	DB 인스턴스의 파라미터 그룹입니다. 기본 파라미터 그룹을 사용하거나 사용자 지정 파라미터 그룹을 생성할 수 있습니다.
	
	RDS for Db2용 BYOL 모델을 사용하는 경우, DB 인스턴스를 만들기 전에 먼저 IBM Site ID 및 IBM Customer ID를 포함하는 사용자 지정 파라미터 그룹을 만들어야 합니다. 자세한 내용은 Db2에 기존 보유 라이선스 사용 섹션을 참조하세요.
	
	자세한 내용은 Amazon RDS의 파라미터 그룹 섹션을 참조하세요.
		CLI 옵션:
		
		`--db-parameter-group-name`
		
		RDS API 파라미터:
		
		`DBParameterGroupName`
			모두

DB 서브넷 그룹	DB 클러스터에 사용할 DB 서브넷 그룹입니다.
	기존 DB 서브넷 그룹을 사용하려면 Choose existing(기존 항목 선택)을 선택합니다. 그런 다음 Existing DB subnet groups(기존 DB 서브넷 그룹) 드롭다운 목록에서 필요한 서브넷 그룹을 선택합니다.
	
	RDS가 호환되는 DB 서브넷 그룹을 선택하도록 하려면 Automatic setup(자동 설정)을 선택합니다. 해당 그룹이 없으면 RDS는 클러스터에 대한 새 서브넷 그룹을 생성합니다.
	
	자세한 내용은 DB 서브넷 그룹을 사용한 작업 섹션을 참조하세요.
		CLI 옵션:
		
		`--db-subnet-group-name`
		
		RDS API 파라미터:
		
		`DBSubnetGroupName`
			모두

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

전용 로그 볼륨	전용 로그 볼륨(DLV)을 사용하여 데이터베이스 테이블이 들어 있는 볼륨과 분리된 스토리지 볼륨에 데이터베이스 트랜잭션 로그를 저장합니다.
	
	자세한 내용은 전용 로그 볼륨(DLV) 사용 섹션을 참조하세요.
		CLI 옵션:
		
		`--dedicated-log-volume`
		
		RDS API 파라미터:
		
		`DedicatedLogVolume`
			모두

삭제 방지	DB 인스턴스가 삭제되지 않도록 방지하려면, 삭제 방지를 활성화합니다. AWS Management Console을 사용하여 프로덕션 DB 인스턴스를 생성할 경우 기본적으로 삭제 방지가 활성화됩니다.
	
	자세한 내용은 DB 인스턴스 삭제 섹션을 참조하세요.
		CLI 옵션:
		
		`--deletion-protection`
		
		`--no-deletion-protection`
		
		RDS API 파라미터:
		
		`DeletionProtection`
			모두

암호화	이 DB 인스턴스에 대해 유휴 암호화를 활성화하기 위한 [Enable Encryption].
	
	자세한 내용은 Amazon RDS 리소스 암호화 섹션을 참조하세요.
		CLI 옵션:
		
		`--storage-encrypted`
		
		`--no-storage-encrypted`
		
		RDS API 파라미터:
		
		`StorageEncrypted`
			모두

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

확장 모니터링	DB 인스턴스가 실행되는 운영 체제에 대한 실시간 측정치 수집을 활성화하려면 [Enable enhanced monitoring]을 선택합니다.
	
	자세한 내용은 Enhanced Monitoring을 사용하여 OS 지표 모니터링 섹션을 참조하세요.
		CLI 옵션:
		
		`--monitoring-interval`
		
		`--monitoring-role-arn`
		
		RDS API 파라미터:
		
		`MonitoringInterval`
		
		`MonitoringRoleArn`
			모두

엔진 유형	이 DB 인스턴스에 사용할 데이터베이스 엔진을 선택합니다.	CLI 옵션:
		
		`--engine`
		
		RDS API 파라미터:
		
		`Engine`
			모두

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

초기 데이터베이스 이름	DB 인스턴스에 있는 데이터베이스의 이름입니다. 이름을 입력하지 않으면 Amazon RDS가 DB 인스턴스(Oracle 및 PostgreSQL 제외)에서 데이터베이스를 생성하지 않습니다. 이름은 데이터베이스 엔진에서 예약한 단어일 수 없으며, DB 엔진에 따라 다른 제약 조건이 있습니다.
	
	Db2:
	
	• 1–8자의 영숫자로 구성되어야 합니다.
	• a~z, A~Z, @, $ 또는 #로 시작하고 그 뒤에 a~z, A~Z, 0~9, _, @, # 또는 $가 와야 합니다.
	• 공백은 포함할 수 없습니다.
	• 자세한 내용은 추가 고려 사항 섹션을 참조하세요.
	
	MariaDB 및 MySQL:
	
	• 1–64자의 영숫자로 구성되어야 합니다.
	
	Oracle:
	
	• 1–8자의 영숫자로 구성되어야 합니다.
	• NULL일 수는 없습니다. 기본 값은 ORCL입니다.
	• 문자로 시작해야 합니다.
		CLI 옵션:
		
		`--db-name`
		
		RDS API 파라미터:
		
		`DBName`
			SQL Server를 제외한 모든 항목

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진
	PostgreSQL: • 1–63자의 영숫자로 구성되어야 합니다. • 글자나 밑줄로 시작해야 합니다. 이후 문자는 글자, 밑줄 또는 숫자(0~9)가 될 수 있습니다. • 초기 데이터베이스 이름은 `postgres`입니다.		

라이선스	라이선스 모델의 유효 값:
	• Db2용 기존 보유 라이선스 사용(BYOL) 또는 마켓플레이스 라이선스
	• MariaDB의 경우 general-public-license.
	• Microsoft SQL Server의 경우 license-included.
	• MySQL의 경우 general-public-license.
	• Oracle의 경우 license-included 또는 bring-your-own-license.
	• PostgreSQL의 경우 postgreql-license.
		CLI 옵션:
		`--license-model`
		RDS API 파라미터:
		`LicenseModel`
			모두

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

로그 내보내기	Amazon CloudWatch Logs에 게시할 데이터베이스 로그 파일의 유형입니다.
	
	자세한 내용은 Amazon CloudWatch Logs에 데이터베이스 로그 게시 섹션을 참조하세요.
		CLI 옵션:
		
		`--enable-cloudwatch-logs-exports`
		
		RDS API 파라미터:
		
		`EnableCloudwatchLogsExports`
			모두

유지보수 윈도우	대기 중인 DB 인스턴스 설정 변경이 적용되기 위해 경과해야 하는 기간(30분)입니다. 이 시간이 중요하지 않은 경우 [No Preference]를 선택합니다.
	
	자세한 내용은 Amazon RDS 유지 관리 기간 섹션을 참조하세요.
		CLI 옵션:
		
		`--preferred-maintenance-window`
		
		RDS API 파라미터:
		
		`PreferredMaintenanceWindow`
			모두

AWS Secrets Manager에서 마스터 보안 인증 정보 관리	Secrets Manager에서 보안 암호의 마스터 사용자 암호를 관리하려면 AWS Secrets Manager에서 마스터 자격 증명 관리를 선택합니다.
	
	원하는 경우 보안 암호를 보호하는 데 사용할 KMS 키를 선택합니다. 자신의 계정에서 KMS 키를 선택하거나 다른 계정의 키를 입력할 수 있습니다.
	
	자세한 내용은 Amazon RDS 및 AWS Secrets Manager를 통한 암호 관리 섹션을 참조하세요.
		CLI 옵션:
		
		`--manage-master-user-password | --no-manage-master-user-password`
		
		`--master-user-secret-kms-key-id`
		
		RDS API 파라미터:
		
		`ManageMasterUserPassword`
		
		`MasterUserSecretKmsKeyId`
			모두

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

마스터 암호	마스터 사용자 계정의 암호입니다. 암호에 있는 인쇄 가능한 ASCII 문자(/, ", 공백 및 @ 제외)의 수는 다음과 같습니다. 이는 DB 엔진에 따라 달라집니다.
	• Db2: 8~255
	• Oracle: 8–30
	• MariaDB 및 MySQL: 8–41
	• SQL Server 및 PostgreSQL: 8–128
		CLI 옵션:
		`--master-user-password`
		RDS API 파라미터:
		`MasterUserPassword`
			모두

마스터 사용자 이름	모든 데이터베이스 권한을 사용해 DB 인스턴스에 로그인하기 위해 마스터 사용자 이름으로 사용할 이름입니다. 이름 지정 시 다음과 같은 제한 사항이 있습니다.
	• 이름에 1~16자의 영숫자와 밑줄을 포함할 수 있습니다.
	• 첫 번째 자리는 문자여야 합니다.
	• 이름은 데이터베이스 엔진에서 예약한 단어는 사용할 수 없습니다.
	DB 인스턴스가 생성된 후에는 마스터 사용자 이름을 변경할 수 없습니다.
	Db2의 경우 자체 관리형 Db2 인스턴스 이름과 동일한 마스터 사용자 이름을 사용하는 것이 좋습니다.
	마스터 사용자에게 부여된 권한에 대한 자세한 내용은 마스터 사용자 계정 권한 섹션을 참조하세요.
		CLI 옵션:
		`--master-username`
		RDS API 파라미터:
		`MasterUsername`
			모두

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

Microsoft SQL Server Windows 인증	Microsoft SQL Server Windows 인증을 활성화한 다음 권한이 있는 도메인 사용자가 Windows 인증을 사용하여 이 SQL Server 인스턴스로 권한을 부여할 수 있도록 허용할 디렉터리를 찾습니다.	CLI 옵션:
		`--domain`
		`--domain-iam-role-name`
		RDS API 파라미터:
		`Domain`
		`DomainIAMRoleName`
			SQL 서버

다중 AZ 배포	장애 조치를 위해 다른 가용 영역에 DB 인스턴스의 수동 보조 복제본을 생성하려면 Create a standby instance(대기 인스턴스를 생성)를 선택합니다. 이때 고가용성을 유지하려면 프로덕션 워크로드를 위한 다중 AZ를 권장합니다.
	개발 및 테스트의 경우 Do not create a standby instance(대기 인스턴스를 생성하지 않음)를 선택할 수 있습니다.
	자세한 내용은 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 섹션을 참조하세요.
		CLI 옵션:
		`--multi-az`
		`--no-multi-az`
		RDS API 파라미터:
		`MultiAZ`
			모두

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

국가별 문자 집합 (NCHAR)	DB 인스턴스에 국가별 문자 집합으로, 일반적으로 NCHAR 문자 집합이라고 합니다. 국가별 문자 집합을 AL16UTF16(기본값) 또는 UTF-8로 설정할 수 있습니다. DB 인스턴스를 생성한 후에는 국가별 문자 집합을 변경할 수 없습니다.
	
	국가별 문자 집합은 DB 문자 집합과 다릅니다. DB 문자 집합과 달리 국가별 문자 집합은 데이터베이스 메타데이터에 영향을 주지 않고 NCHAR 데이터 형식(NCHAR, NVARCHAR2 및 NCLOB) 열에 대한 인코딩을 지정합니다.
	
	자세한 내용은 RDS for Oracle 문자 집합 섹션을 참조하세요.
		CLI 옵션:
		
		`--nchar-character-set-name`
		
		API 파라미터:
		
		`NcharCharacterSetName`
			Oracle

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

네트워크 유형	DB 인스턴스에서 지원하는 IP 주소 지정 프로토콜입니다.
	
	IPv4(기본값) 리소스가 인터넷 프로토콜 버전 4 (IPv4) 주소 지정 프로토콜을 통해서만 DB 인스턴스와 통신할 수 있도록 지정합니다.
	
	듀얼 스택 모드리소스가 IPv4, 인터넷 프로토콜 버전 6 (IPv6) 또는 둘 다를 통해 DB 인스턴스와 통신할 수 있도록 지정합니다. IPv6 주소 지정 프로토콜을 통해 DB 인스턴스와 통신해야 하는 리소스가 있는 경우 이중 스택 모드를 사용합니다. 또한 IPv6 CIDR 블록을 지정한 DB 서브넷 그룹의 모든 서브넷과 연결해야 합니다.
	
	자세한 내용은 Amazon RDS IP 주소 지정 단원을 참조하십시오.
		CLI 옵션:
		
		`--network-type`
		
		RDS API 파라미터:
		
		`NetworkType`
			모두

옵션 그룹	DB 인스턴스의 옵션 그룹입니다. 기본 옵션 그룹을 사용하거나 사용자 지정 옵션 그룹을 생성할 수 있습니다.
	
	자세한 내용은 옵션 그룹 작업 섹션을 참조하세요.
		CLI 옵션:
		
		`--option-group-name`
		
		RDS API 파라미터:
		
		`OptionGroupName`
			모두

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

성능 개선 도우미	데이터베이스 성능을 분석하고 문제를 해결할 수 있도록 DB 인스턴스 로드를 모니터링하려면 Performance Insights 활성화를 선택합니다.
	
	보존 기간을 선택하여 성능 개선 도우미 데이터 기록을 얼마나 유지할지 결정합니다. 프리 티어의 보존 설정은 기본값(7일)입니다. 성능 데이터를 더 오래 보존하려면 1~24개월을 지정하십시오. 보존 기간에 대한 자세한 내용은 성능 개선 도우미의 요금 및 데이터 보존 단원을 참조하십시오.
	
	이 데이터베이스 볼륨을 암호화하는 데 사용되는 키를 보호하는 데 사용할 KMS 키를 선택합니다. 자신의 계정에서 KMS 키를 선택하거나 다른 계정의 키를 입력할 수 있습니다.
	
	자세한 내용은 성능 개선 도우미를 통한 Amazon RDS 모니터링 섹션을 참조하세요.
		CLI 옵션:
		
		`--enable-performance-insights`
		
		`--no-enable-performance-insights`
		
		`--performance-insights-retention-period`
		
		`--performance-insights-kms-key-id`
		
		RDS API 파라미터:
		
		`EnablePerformanceInsights`
		
		`PerformanceInsightsRetentionPeriod`
		
		`PerformanceInsightsKMSKeyId`
			Db2를 제외한 모든 항목

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

프로비저닝된 IOPS	DB 인스턴스의 프로비저닝된 IOPS(초당 I/O 작업 수) 값입니다. 이 설정은 스토리지 유형에서 다음 중 하나를 선택한 경우에만 사용할 수 있습니다.
	• 범용 SSD(gp3)
	• 프로비저닝된 IOPS SSD(io1)
	• 프로비저닝된 IOPS SSD(io2)
	자세한 내용은 Amazon RDS DB 인스턴스 스토리지 섹션을 참조하세요.
		CLI 옵션:
		`--iops`
		RDS API 파라미터:
		`Iops`
			모두

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

공개 액세스(Public access)	DB 인스턴스에 퍼블릭 IP 주소를 부여하려면(즉 VPC 외부에서 액세스할 수 있음) 예를 선택합니다. 공개적으로 액세스가 가능하려면 DB 인스턴스도 VPC의 퍼블릭 서브넷에 있어야 합니다.
	
	VPC 내부에서만 DB 인스턴스에 액세스할 수 있게 하려면 [No].
	
	자세한 내용은 VPC에 있는 DB 인스턴스를 인터넷에서 숨기기 섹션을 참조하세요.
	
	VPC 외부에서 DB 인스턴스에 연결하려면 DB 인스턴스에 공개적으로 액세스할 수 있어야 합니다. 또한 DB 인스턴스 보안 그룹의 인바운드 규칙을 사용하여 액세스 권한을 부여해야 하며, 다른 요구 사항도 충족해야 합니다. 자세한 내용은 Amazon RDS DB 인스턴스에 연결할 수 없음 섹션을 참조하세요.
	
	DB 인스턴스에 공개적으로 액세스할 수 없는 경우 AWS Site-to-Site VPN 연결 또는 AWS Direct Connect 연결을 사용하여 프라이빗 네트워크에서 액세스합니다. 자세한 내용은 인터네트워크 트래픽 개인 정보 섹션을 참조하세요.
		CLI 옵션:
		
		`--publicly-accessible`
		
		`--no-publicly-accessible`
		
		RDS API 파라미터:
		
		`PubliclyAccessible`
			모두

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

RDS 추가 지원	지원되는 메이저 엔진 버전이 RDS 표준 지원 종료 날짜를 지나서도 계속 실행되도록 하려면 RDS 추가 지원 활성화를 선택합니다.
	
	DB 클러스터를 생성할 때 Amazon RDS는 기본적으로 RDS 확장 지원을 사용합니다. RDS 표준 지원 종료일 후에 새 DB 인스턴스가 생성되지 않도록 하고 RDS 확장 지원에 대한 요금이 부과되지 않도록 하려면 이 설정을 비활성화하세요. 기존 DB 인스턴스에는 RDS 확장 지원 요금 시작일이 되기 전까지는 요금이 부과되지 않습니다.
	
	자세한 내용은 Amazon RDS를 사용한 Amazon RDS 추가 지원 섹션을 참조하세요.
		CLI 옵션:
		
		`--engine-lifecycle-support`
		
		RDS API 파라미터:
		
		`EngineLifecycleSupport`
			MySQL
			
			PostgreSQL

RDS 프록시	Create an RDS Proxy(RDS 프록시 생성)를 선택하여 DB 인스턴스의 프록시를 생성합니다. Amazon RDS는 프록시에 대한 IAM 역할과 Secrets Manager 암호를 자동으로 생성합니다. 자세한 내용은 Amazon RDS Proxy 섹션을 참조하세요.	DB 인스턴스를 생성할 때는 사용할 수 없습니다.	MariaDB MySQL PostgreSQL

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

Storage autoscaling(스토리지 Autoscaling)	Enable storage autoscaling(스토리지 Autoscaling 활성화)를 사용하면 필요할 경우 Amazon RDS가 스토리지를 자동으로 확장하여 DB 인스턴스의 스토리지 공간이 부족해지는 문제를 방지할 수 있습니다.
	
	Maximum storage threshold(최대 스토리지 임계값)를 사용하여 Amazon RDS가 DB 인스턴스의 스토리지를 자동으로 확장할 수 있는 상한선을 설정합니다. 기본값은 1,000GiB입니다.
	
	자세한 내용은 Amazon RDS 스토리지 Autoscaling을 사용한 용량 자동 관리 섹션을 참조하세요.
		CLI 옵션:
		
		`--max-allocated-storage`
		
		RDS API 파라미터:
		
		`MaxAllocatedStorage`
			모두

스토리지 처리량(throughput)	DB 인스턴스의 스토리지 처리량(throughput) 값입니다. 이 설정은 스토리지 유형으로 범용 SSD(gp3)를 선택한 경우에만 사용할 수 있습니다.
	
	자세한 내용은 gp3 스토리지(권장) 섹션을 참조하세요.
		CLI 옵션:
		
		`--storage-throughput`
		
		RDS API 파라미터:
		
		`StorageThroughput`
			모두

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

스토리지 유형	DB 인스턴스의 스토리지 유형입니다.
	
	범용 SSD(gp3)를 선택하면 고급 설정에서 프로비저닝된 IOPS 및 스토리지 처리량(throughput)을 추가로 프로비저닝할 수 있습니다.
	
	프로비저닝된 IOPS SSD(io1) 또는 프로비저닝된 IOPS SSD(io2)를 선택하는 경우 프로비저닝된 IOPS 값을 입력합니다.
	
	자세한 내용은 Amazon RDS 스토리지 유형 섹션을 참조하세요.
		CLI 옵션:
		
		`--storage-type`
		
		RDS API 파라미터:
		
		`StorageType`
			모두

서브넷 그룹	이 DB 인스턴스에 연결할 DB 서브넷 그룹입니다.
	
	자세한 내용은 DB 서브넷 그룹을 사용한 작업 섹션을 참조하세요.
		CLI 옵션:
		
		`--db-subnet-group-name`
		
		RDS API 파라미터:
		
		`DBSubnetGroupName`
			모두

테넌트 데이터베이스 이름	Oracle 아키텍처의 다중 테넌트 구성에 있는 초기 PDB의 이름입니다. 이 설정은 아키텍처 구성으로 다중 테넌트 구성을 선택한 경우에만 사용할 수 있습니다.
	
	테넌트 데이터베이스 이름은 이름이 `RDSCDB`인 CDB의 이름과 달라야 합니다. CDB 이름은 변경할 수 없습니다.
		CLI 옵션:
		
		`--db-name`
		
		RDS API 파라미터:
		
		`DBName`
			Oracle

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

테넌트 데이터베이스 마스터 사용자 이름	모든 데이터베이스 권한을 사용해 테넌트 데이터베이스(PDB)에 로그인하기 위해 마스터 사용자 이름으로 사용할 이름입니다. 이 설정은 아키텍처 구성으로 다중 테넌트 구성을 선택한 경우에만 사용할 수 있습니다.
	
	이름 지정 시 다음과 같은 제한 사항이 있습니다.
	
	• 이름에 1~16자의 영숫자와 밑줄을 포함할 수 있습니다.
	• 첫 번째 자리는 문자여야 합니다.
	• 이름은 데이터베이스 엔진에서 예약한 단어는 사용할 수 없습니다.
	
	다음을 수행할 수 없습니다.
	
	• 테넌트 데이터베이스를 만든 후 테넌트 마스터 사용자 이름을 변경합니다.
	• 테넌트 마스터 사용자 이름으로 CDB에 로그인합니다.
		CLI 옵션:
		
		`--master-username`
		
		RDS API 파라미터:
		
		`MasterUsername`
			Oracle

테넌트 데이터베이스 마스터 암호	테넌트 데이터베이스(PDB)의 마스터 사용자 계정 암호입니다. 이 설정은 아키텍처 구성으로 다중 테넌트 구성을 선택한 경우에만 사용할 수 있습니다.
	
	암호는 8~30자의 인쇄 가능한 ASCII 문자로 구성되어야 합니다(/, ", 공백 및 @ 제외).
		CLI 옵션:
		
		`--master-password`
		
		RDS API 파라미터:
		
		`MasterPassword`
			Oracle

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

테넌트 데이터베이스 문자 집합	초기 테넌트 데이터베이스의 문자 집합입니다. 이 설정은 아키텍처 구성으로 다중 테넌트 구성을 선택한 경우에만 사용할 수 있습니다. RDS for Oracle CDB 인스턴스만 지원됩니다.
	
	테넌트 데이터베이스 문자 집합에 대한 기본값 AL32UTF8은 Unicode 5.0 UTF-8 Universal 문자 집합을 나타냅니다. CDB 문자 집합과 다른 테넌트 데이터베이스 문자 집합을 선택할 수 있습니다.
	
	자세한 내용은 RDS for Oracle 문자 집합 섹션을 참조하세요.
		CLI 옵션:
		
		`--character-set-name`
		
		RDS API 파라미터:
		
		`CharacterSetName`
			Oracle

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진

테넌트 데이터베이스 내셔널 문자 집합	테넌트 데이터베이스의 내셔널 문자 집합으로, 일반적으로 NCHAR 문자 집합이라고 합니다. 이 설정은 아키텍처 구성으로 다중 테넌트 구성을 선택한 경우에만 사용할 수 있습니다. RDS for Oracle CDB 인스턴스만 지원됩니다.
	
	국가별 문자 집합을 AL16UTF16(기본값) 또는 UTF-8로 설정할 수 있습니다. 테넌트 데이터베이스를 생성한 후에는 국가별 문자 집합을 변경할 수 없습니다.
	
	테넌트 데이터베이스 내셔널 문자 집합은 테넌트 데이터베이스 문자 집합과 다릅니다. 내셔널 문자 집합은 NCHAR 데이터 유형(NCHAR, NVARCHAR2 및NCLOB)을 사용하는 열의 인코딩만 지정하고 데이터베이스 메타데이터에는 영향을 주지 않습니다.
	
	자세한 내용은 RDS for Oracle 문자 집합 섹션을 참조하세요.
		CLI 옵션:
		
		`--nchar-character-set-name`
		
		API 파라미터:
		
		`NcharCharacterSetName`
			Oracle

시간대	DB 인스턴스의 시간대입니다. 표준 시간대를 선택하지 않으면 DB 인스턴스는 기본 표준 시간대를 사용합니다. DB 인스턴스가 생성된 후에는 시간대를 변경할 수 없습니다.
	
	자세한 내용은 Amazon RDS for Db2 DB 인스턴스의 현지 시간대 및 Microsoft SQL Server DB 인스턴스의 현지 시간대 단원을 참조하십시오.
		CLI 옵션:
		
		`--timezone`
		
		RDS API 파라미터:
		
		`Timezone`
			Db2
			
			SQL Server
			
			RDS Custom for SQL Server

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	지원되는 DB 엔진
Virtual Private Cloud(VPC)	이 DB 인스턴스와 연결할 Amazon VPC 서비스 기반 VPC입니다. 자세한 내용은 Amazon VPC 및 Amazon RDS 섹션을 참조하세요.	CLI 및 API의 경우 VPC 보안 그룹 ID를 지정합니다.	모두

VPC 보안 그룹(방화벽)	DB 인스턴스에 연결할 보안 그룹입니다.
	
	자세한 내용은 VPC 보안 그룹 개요 섹션을 참조하세요.
		CLI 옵션:
		
		`--vpc-security-group-ids`
		
		RDS API 파라미터:
		
		`VpcSecurityGroupIds`
			모두

AWS CloudFormation을 사용하여 Amazon RDS 리소스 생성

Amazon RDS는 리소스 및 인프라를 생성하고 관리하는 데 소요되는 시간을 줄일 수 있도록 AWS 리소스를 모델링하고 설정하는 데 도움이 되는 서비스인 AWS CloudFormation과 통합됩니다. DB 인스턴스 및 DB 파라미터 그룹과 같이 원하는 모든 AWS 리소스(DB 클러스터 및 DB 클러스터 파라미터 그룹를 설명하는 템플릿을 생성하고 AWS CloudFormation은 해당 리소스를 자동으로 프로비저닝하고 구성합니다.

AWS CloudFormation을 사용할 때 템플릿을 재사용하여 RDS 리소스를 일관되고 반복적으로 설정할 수 있습니다. 리소스를 한 번 설명한 후 여러 AWS 계정 및 리전에서 동일한 리소스를 반복적으로 프로비저닝할 수 있습니다.

RDS과AWS CloudFormation템플릿

AWS CloudFormation 템플릿은 JSON 또는 YAML로 서식 지정된 텍스트 파일입니다. 이 템플릿은 AWS CloudFormation 스택에서 프로비저닝할 리소스에 대해 설명합니다. JSON 또는 YAML에 익숙하지 않은 경우 AWS CloudFormation Designer를 사용하면 AWS CloudFormation 템플릿을 시작하는 데 도움이 됩니다. 자세한 내용은 AWS CloudFormation 사용 설명서에서 AWS CloudFormation Designer이란 무엇입니까?를 참조하세요.

RDS는 AWS CloudFormation에서 리소스 생성을 지원합니다. 이러한 리소스에 대한 JSON 및 YAML 템플릿의 예를 비롯한 자세한 내용은 AWS CloudFormation 사용 설명서에서 RDS 리소스 유형 참조를 참조하세요.

AWS CloudFormation에 대해 자세히 알아보기

AWS CloudFormation에 대한 자세한 내용은 다음 리소스를 참조하세요.

- AWS CloudFormation
- AWS CloudFormation 사용 설명서
- AWS CloudFormation API 참조
- AWS CloudFormation 명령줄 인터페이스 사용 설명서

Amazon RDS DB 인스턴스에 연결

DB 인스턴스에 연결하려면 먼저 DB 인스턴스를 생성해야 합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성을 참조하세요. Amazon RDS가 DB 인스턴스를 프로비저닝한 후에는 해당 DB 엔진용 표준 클라이언트 애플리케이션 또는 유틸리티를 사용하여 DB 인스턴스에 연결합니다. 연결 문자열에는 호스트 파라미터로 DB 인스턴스 엔드포인트의 DNS 주소를 지정합니다. 또한 DB 인스턴스 엔드포인트의 포트 번호를 포트 파라미터로 지정합니다.

Amazon RDS DB 인스턴스의 연결 정보 또는 VPC의 DB 인스턴스에 액세스하기 위한 시나리오를 찾는 방법에 대한 자세한 내용은 다음 주제를 참조하세요.

- Amazon RDS DB 인스턴스에 대한 연결 정보 찾기
- VPC에서 DB 인스턴스에 액세스하는 시나리오

Amazon RDS DB 인스턴스에 대한 연결 정보 찾기

DB 인스턴스의 연결 정보에는 엔드포인트, 포트 및 유효한 데이터베이스 사용자(예: 마스터 사용자)가 포함됩니다. 예를 들어 MySQL DB 인스턴스의 엔드포인트 값이 `mydb.123456789012.us-east-1.rds.amazonaws.com`이라고 가정합니다. 이 경우 포트 값은 `3306`이고 데이터베이스 사용자는 `admin`입니다. 이 정보를 바탕으로 연결 문자열에 다음 값을 지정합니다.

- 호스트 또는 호스트 이름 또는 DNS 이름에 `mydb.123456789012.us-east-1.rds.amazonaws.com`을 지정합니다.
- 포트에 대해 `3306`을 지정합니다.
- 사용자에 `admin`을 지정합니다.

엔드포인트는 DB 인스턴스마다 고유하며 포트 및 사용자 값이 다를 수 있습니다. 다음 목록은 각 DB 엔진의 가장 일반적인 포트를 보여 줍니다.

- Db2 - 50,000
- MariaDB – 3306
- Microsoft SQL Server – 1433
- MySQL – 3306
- Oracle – 1521
- PostgreSQL – 5432

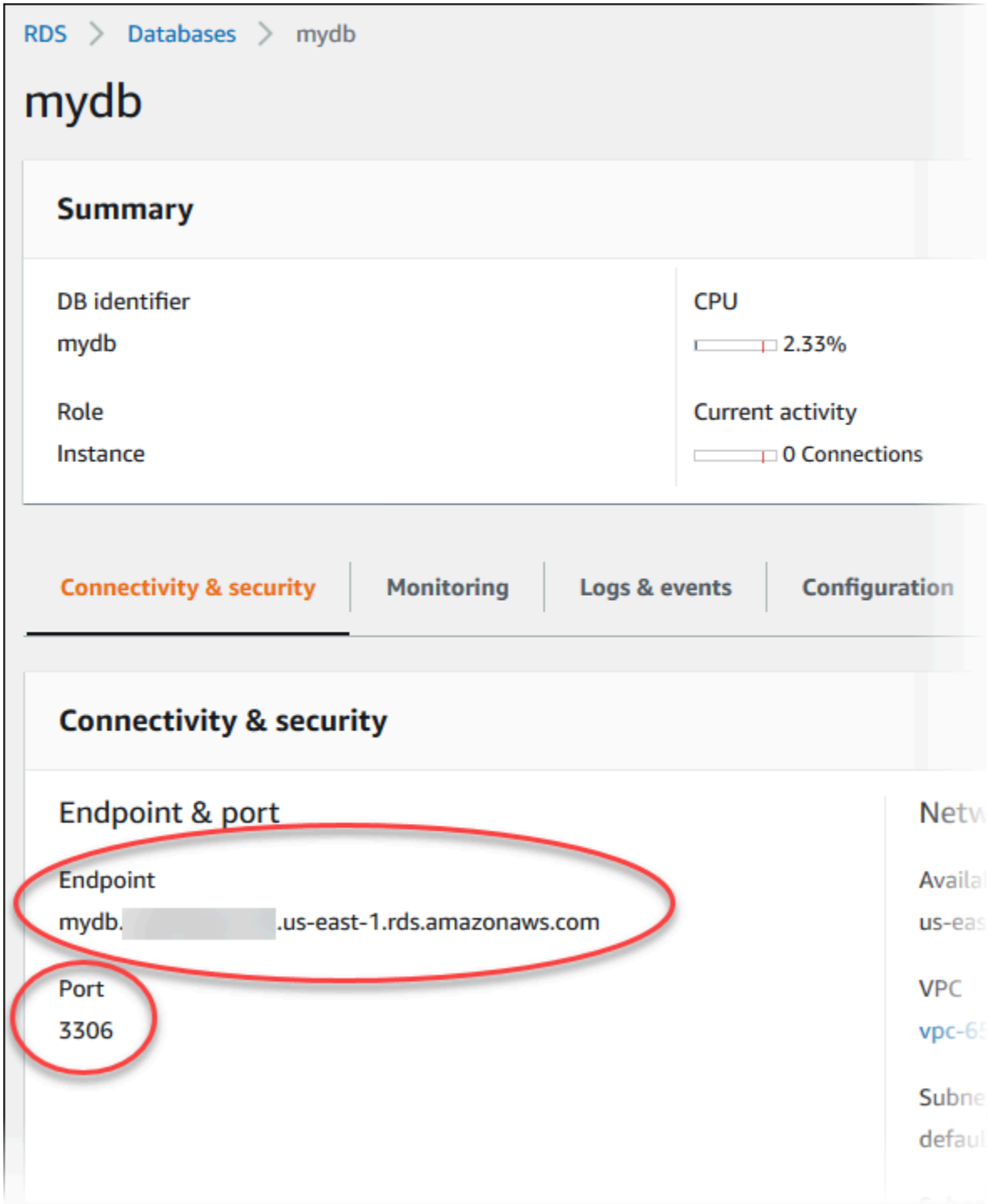
DB 인스턴스에 연결하려면 DB 엔진에 대해 임의의 클라이언트를 사용합니다. 예를 들어 mysql 유틸리티를 사용하여 MariaDB 또는 MySQL DB 인스턴스에 연결할 수 있습니다. Microsoft SQL Server Management Studio를 사용하여 SQL Server DB 인스턴스에 연결할 수 있습니다. Oracle SQL Developer를 사용하여 Oracle DB 인스턴스에 연결할 수 있습니다. 마찬가지로, psql 명령줄 유틸리티를 사용하여 PostgreSQL DB 인스턴스에 연결할 수 있습니다.

DB 인스턴스에 대한 연결 정보를 찾으려면 AWS Management Console을 사용합니다. 또한 AWS Command Line Interface(AWS CLI) describe-db-instances 명령 또는 RDS API DescribeDBInstances 작업을 사용할 수도 있습니다.

콘솔

AWS Management Console에서 DB 인스턴스에 대한 연결 정보를 찾으려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [데이터베이스(Database)] 를 선택하여 DB 인스턴스 목록을 표시합니다.
3. DB 인스턴스의 이름을 선택하여 세부 정보를 표시합니다.
4. Connectivity & security(연결 및 보안) 탭에서 엔드포인트를 복사합니다. 또한 포트 번호를 적어 둡니다. DB 인스턴스에 연결하려면 엔드포인트와 포트 번호가 모두 필요합니다.



5. 마스터 사용자 이름을 찾아야 하는 경우 [구성(Configuration)] 탭을 선택하고 [마스터 사용자 이름(Master username)] 값을 확인합니다.

AWS CLI

AWS CLI를 사용하여 DB 인스턴스의 연결 정보를 찾으려면 describe-db-instances 명령을 호출합니다. 이 호출에서 DB 인스턴스 ID, 엔드포인트, 포트 및 마스터 사용자 이름을 쿼리합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-instances \
  --query "*[].[DBInstanceIdentifier,Endpoint.Address,Endpoint.Port,MasterUsername]"
```

Windows의 경우:

```
aws rds describe-db-instances ^
  --query "*[].[DBInstanceIdentifier,Endpoint.Address,Endpoint.Port,MasterUsername]"
```

다음과 유사하게 출력되어야 합니다.

```
[
    [
        "mydb",
        "mydb.123456789012.us-east-1.rds.amazonaws.com",
        3306,
        "admin"
    ],
    [
        "myoracledb",
        "myoracledb.123456789012.us-east-1.rds.amazonaws.com",
        1521,
        "dbadmin"
    ],
    [
        "mypostgresqldb",
        "mypostgresqldb.123456789012.us-east-1.rds.amazonaws.com",
        5432,
        "postgresadmin"
    ]
]
```

RDS API

Amazon RDS API를 사용하여 DB 인스턴스의 연결 정보를 찾으려면 DescribeDBInstances 작업을 호출합니다. 출력에서 엔드포인트 주소, 엔드포인트 포트 및 마스터 사용자 이름의 값을 찾습니다.

VPC에서 DB 인스턴스에 액세스하는 시나리오

Amazon Virtual Private Cloud(Amazon VPC)를 사용하여 Amazon RDS DB 인스턴스와 같은 AWS 리소스를 Virtual Private Cloud(VPC)에서 시작할 수 있습니다. Amazon VPC를 사용하면 가상 네트워킹 환경을 완벽하게 제어할 수 있습니다. 자기만의 IP 주소 범위를 선택하고, 서브넷을 생성하고, 라우팅 및 액세스 제어 목록을 구성할 수 있습니다.

VPC 보안 그룹은 VPC 내부의 DB 인스턴스에 대한 액세스를 제어합니다. 각 VPC 보안 그룹 규칙을 설정하면 특정 소스가 해당 VPC 보안 그룹과 연결되어 있는 VPC의 DB 인스턴스에 액세스할 수 있습니다. 소스는 주소 범위(예: 203.0.113.0/24) 또는 다른 VPC 보안 그룹일 수 있습니다. VPC 보안 그룹을 소스로 지정하면 소스 VPC 보안 그룹을 사용하는 모든 인스턴스(일반적으로 애플리케이션 서버)에서 수신 트래픽이 허용됩니다.

DB 인스턴스에 연결하기 전에 사용 사례에 맞게 VPC를 구성합니다. 다음은 VPC에서 DB 인스턴스에 액세스하는 일반적인 시나리오입니다.

- 동일한 VPC의 Amazon EC2 인스턴스가 VPC의 – DB 인스턴스에 액세스 VPC에서 DB 인스턴스의 일반적인 사용 사례는 동일한 VPC의 EC2 인스턴스에서 실행 중인 애플리케이션 서버와 데이터를 공유하는 것입니다. EC2 인스턴스는 DB 인스턴스와 상호 작용하는 애플리케이션을 통해 웹 서버를 실행할 수 있습니다.
- 다른 VPC의 EC2 인스턴스가 VPC의 DB 인스턴스에 액세스 – 경우에 따라 액세스하는 데 사용 중인 EC2 인스턴스와 다른 VPC에 DB 인스턴스가 있을 수 있습니다. 그렇다면 VPC 피어링을 사용하여 DB 인스턴스에 액세스할 수 있습니다.
- 클라이언트 애플리케이션이 인터넷을 통해 VPC의 DB 인스턴스에 액세스 – 클라이언트 애플리케이션에서 인터넷을 통해 VPC의 DB 인스턴스에 액세스하려면, 단일 퍼블릭 서브넷으로 VPC를 구성합니다. 인터넷을 통한 통신을 활성화하도록 인터넷 게이트웨이도 구성합니다.

  VPC 외부에서 DB 인스턴스에 연결하려면 DB 인스턴스에 공개적으로 액세스할 수 있어야 합니다. 또한 DB 인스턴스 보안 그룹의 인바운드 규칙을 사용하여 액세스 권한을 부여해야 하며, 기타 요구 사항을 충족해야 합니다. 자세한 내용은 Amazon RDS DB 인스턴스에 연결할 수 없음 단원을 참조하십시오.
- 프라이빗 네트워크로 액세스하는 VPC의 DB 인스턴스 - DB 인스턴스에 공개적으로 액세스할 수 없는 경우 다음 옵션 중 하나를 사용하여 프라이빗 네트워크에서 액세스할 수 있습니다.
  - AWS Site-to-Site VPN 연결
  - AWS Direct Connect 연결
  - AWS Client VPN 연결

자세한 내용은 VPC에서 DB 인스턴스에 액세스하는 시나리오 단원을 참조하십시오.

AWS 드라이버를 사용하여 DB 인스턴스에 연결

더 빠른 전환 및 장애 조치 시간, AWS Secrets Manager, AWS Identity and Access Management(IAM) 및 페더레이션 ID를 사용한 인증을 지원하도록 설계된 AWS 드라이버 제품군입니다. AWS 드라이버는 DB 인스턴스 상태 모니터링과 인스턴스 토폴로지 파악을 통해 새 기본 인스턴스를 결정합니다. 이 접근 방식은 전환 및 장애 조치 시간을 오픈 소스 드라이버의 경우 수십 초였던 것에 비해 10초 미만으로 단축합니다.

다음 표에는 각 드라이버에 지원되는 기능이 나와 있습니다. 새로운 서비스 기능이 도입됨에 따라 AWS 드라이버 제품군의 목표는 이러한 서비스 기능에 대한 지원을 기본 제공하는 것입니다.

기능	AWS JDBC 드라이버	AWS Python 드라이버	MySQL용 AWS ODBC 드라이버
장애 조치 지원	예	예	예
향상된 장애 조치 모니터링	예	예	예
읽기/쓰기 분할	예	예	아니요
드라이버 메타데이터 연결	예	해당 사항 없음	N/A
원격 측정	예	예	아니요
Secrets Manager	예	예	예
IAM 인증.	예	예	예
페더레이션 ID(AD FS)	예	예	아니요
페더레이션 ID(Okta)	예	아니요	아니요
다중 AZ DB 클러스터	예	예	아니요

AWS 드라이버에 대한 자세한 내용은 RDS for MariaDB, RDS for MySQL 또는 RDS for PostgreSQL DB 인스턴스에 대한 해당 언어 드라이버를 참조하세요.

> **Note**
>
> RDS for MariaDB에서 지원되는 유일한 기능은 AWS Secrets Manager, AWS Identity and Access Management(IAM) 및 페더레이션 ID를 사용한 인증입니다.

특정 DB 엔진을 실행하는 DB 인스턴스에 연결

특정 DB 엔진을 실행 중인 DB 인스턴스에 연결하는 방법을 알아보려면 DB 엔진의 지침을 따르세요.

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

RDS Proxy와의 연결 관리

또한 Amazon RDS 프록시를 사용하여 RDS for MariaDB, RDS for Microsoft SQL Server, RDS for MySQL, RDS for PostgreSQL DB 인스턴스에 대한 연결을 관리할 수 있습니다. RDS Proxy를 사용하면 애플리케이션이 데이터베이스 연결을 풀링하고 공유하여 확장성을 향상할 수 있습니다. 자세한 내용은 Amazon RDS Proxy 단원을 참조하십시오.

데이터베이스 인증 옵션

Amazon RDS는 데이터베이스 사용자를 인증하는 다음과 같은 방법을 지원합니다.

- 암호 인증 – DB 인스턴스가 모든 사용자 계정 관리 작업을 수행합니다. 여러분은 사용자를 생성하고 SQL 문을 사용하여 암호를 지정합니다. 사용할 수 있는 SQL 문은 DB 엔진에 따라 다릅니다.
- AWS Identity and Access Management(IAM)데이터베이스 인증 - DB 인스턴스에 연결할 때 암호를 사용할 필요가 없습니다. 대신에 인증 토큰을 사용합니다.
- Kerberos 인증 – Kerberos 및 Microsoft Active Directory를 통해 데이터베이스 사용자의 외부 인증을 사용합니다. Kerberos는 티켓과 대칭 키 암호화를 사용하여 네트워크를 통해 암호를 전송할 필요가 없는 네트워크 인증 프로토콜입니다. Kerberos는 Active Directory에 내장되어 있으며 데이터베이스와 같은 네트워크 리소스에 대해 사용자를 인증하도록 설계되었습니다.

IAM 데이터베이스 인증 및 Kerberos 인증은 특정 DB 엔진 및 버전에만 사용할 수 있습니다.

자세한 내용은 Amazon RDS을 사용한 데이터베이스 인증 섹션을 참조하세요.

암호화된 연결

애플리케이션에서 SSL(Secure Socket Layer) 또는 TLS(전송 계층 보안)를 사용하여 DB 인스턴스에 대한 연결을 암호화할 수 있습니다. 각 DB 엔진에는 SSL/TLS를 구현하기 위한 고유한 프로세스가 있습니다. 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 단원을 참조하십시오.

옵션 그룹 작업

DB 엔진 중에는 데이터와 데이터베이스를 더욱 쉽게 관리할 뿐만 아니라 데이터베이스 보안을 더욱 강화한 추가 기능을 가진 엔진도 있습니다. Amazon RDS는 옵션 그룹을 사용하여 이러한 기능을 활성화하고 구성합니다. 옵션 그룹은 Amazon RDS DB 인스턴스에서 사용 가능한 옵션이라는 기능을 지정할 수 있습니다. 옵션은 여러 설정을 통해 옵션의 활용 방식을 지정합니다. 이후 DB 인스턴스와 옵션 그룹을 연동시키면 지정한 옵션과 옵션 설정이 연동된 DB 인스턴스에서 활성화됩니다.

Amazon RDS는 다음 데이터베이스 엔진 옵션을 지원합니다.

데이터베이스 엔진	관련 설명서
`Db2`	RDS for Db2 DB 인스턴스 옵션
`MariaDB`	MariaDB 데이터베이스 엔진을 위한 옵션
`Microsoft SQL Server`	Microsoft SQL Server 데이터베이스 엔진의 옵션
`MySQL`	MySQL DB 인스턴스 옵션
`Oracle`	Oracle DB 인스턴스에 옵션 추가
`PostgreSQL`	PostgreSQL은 옵션 및 옵션 그룹을 사용하지 않습니다. PostgreSQL은 확장 및 모듈을 사용하여 추가 기능을 제공합니다. 자세한 내용은 지원되는 PostgreSQL 확장 버전 섹션을 참조하세요.

옵션 그룹 개요

Amazon RDS는 새로운 DB 인스턴스마다 비어있는 기본 옵션 그룹을 제공합니다. 이 기본 옵션 그룹은 수정하거나 삭제할 수 없지만, 생성되는 새로운 옵션 그룹은 기본 옵션 그룹에서 해당 설정을 가져옵니다. 옵션을 DB 인스턴스에 적용하려면 다음을 수행해야 합니다.

1. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 복사 또는 수정합니다.
2. 하나 이상의 옵션을 옵션 그룹에 추가합니다.
3. 옵션 그룹을 DB 인스턴스에 연동시킵니다.

옵션 그룹을 DB 인스턴스와 연결하려면 DB 인스턴스를 수정합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

DB 인스턴스와 DB 스냅샷은 모두 옵션 그룹과 연동시킬 수 있습니다. 어떤 경우에는 DB 스냅샷에서 복원하거나 DB 인스턴스에 대해 특정 시점으로의 복원을 수행할 수 있습니다. 이 경우 DB 스냅샷 또는 DB 인스턴스에 연결된 옵션 그룹은 기본적으로 복원된 DB 인스턴스와 연결됩니다. 다른 옵션 그룹을 복구된 DB 인스턴스와 연동시킬 수 있습니다. 다만 새로운 옵션 그룹에는 원래 옵션 그룹에 포함되었던 지속적 또는 영구적 옵션이 있어야 합니다. 지속적이거나 영구적인 옵션은 다음과 같이 설명됩니다.

DB 인스턴스에서 옵션을 실행하려면 추가 메모리가 필요합니다. 따라서 DB 인스턴스의 현재 용도에 따라 옵션 사용을 위해 대용량 인스턴스를 시작해야 하는 경우가 있을 수 있습니다. 예를 들어 Oracle Enterprise Manager Database Control은 약 300MB의 RAM을 사용합니다. 스몰 DB 인스턴스에 대해 이 옵션을 활성화할 경우 성능 문제 또는 메모리 부족 오류가 발생할 수 있습니다.

지속적이거나 영구적인 옵션

두 가지 유형인 지속적이거나 영구적인 옵션은 옵션 그룹에 추가할 때 세심한 주의가 필요합니다.

DB 인스턴스가 이 옵션 그룹과 연결되어 있는 동안에는 지속적인 옵션을 해당 옵션 그룹에서 제거할 수 없습니다. 지속적인 옵션의 한 가지 예는 Microsoft SQL Server Transparent Data Encryption(TDE)을 위한 TDE 옵션입니다. 옵션 그룹에서 지속적인 옵션을 제거하려면 먼저 모든 DB 인스턴스를 옵션 그룹에서 연동 해제해야 합니다. 어떤 경우에는 DB 스냅샷에서 복원 또는 특정 시점으로의 복원을 수행할 수 있습니다. 이러한 경우 해당 DB 스냅샷과 연결되어 있는 옵션 그룹에 지속적인 옵션이 포함된 경우에는 복원된 DB 인스턴스만 이 옵션 그룹과 연결할 수 있습니다.

반면 Oracle Advanced Security TDE를 위한 TDE 옵션처럼 영구적인 옵션은 옵션 그룹에서 절대로 제거할 수 없습니다. 영구적인 옵션을 사용 중인 DB 인스턴스의 옵션 그룹은 변경할 수 있습니다. 그러나 DB 인스턴스에 연결된 옵션 그룹에는 동일한 영구 옵션이 포함되어야 합니다. 어떤 경우에는 DB 스냅샷에서 복원 또는 특정 시점으로의 복원을 수행할 수 있습니다. 이러한 경우 해당 DB 스냅샷과 연결되어 있는 옵션 그룹에 영구적인 옵션이 포함된 경우에는 옵션 그룹이 있는 복원된 DB 인스턴스만 이 영구 옵션과 연결할 수 있습니다.

Oracle DB 인스턴스의 경우 `Timezone` 또는 `OLS` 옵션(또는 둘 다)이 있는 공유 DB 스냅샷을 복사할 수 있습니다. 이렇게 하려면 DB 스냅샷을 복사할 때 이 옵션이 포함된 대상 옵션 그룹을 지정하십시오. OLS 옵션은 Oracle 버전 12.2 이상을 실행하는 Oracle DB 인스턴스에 대해서만 영구적이고 지속적입니다. 이러한 옵션에 대한 자세한 내용은 Oracle 시간대 및 Oracle 레이블 보안 단원을 참조하세요.

VPC 고려 사항

DB 인스턴스와 연결된 옵션 그룹은 DB 인스탄스의 VPC에도 연결됩니다. 즉, 인스턴스를 다른 VPC로 복원하려고 하면 DB 인스턴스에 할당된 옵션 그룹을 사용할 수 없습니다. DB 인스턴스를 다른 VPC로 복원하는 경우 다음 중 하나를 수행할 수 있습니다.

- 기본 옵션 그룹을 해당 DB 인스턴스에 배정.
- 해당 VPC에 연결된 옵션 그룹을 할당합니다.
- 새 옵션 그룹을 생성하여 해당 DB 인스턴스에 배정.

Oracle TDE와 같은 지속적 또는 영구적 옵션의 경우 새 옵션 그룹을 생성해야 합니다. 이 옵션 그룹에는 DB 인스턴스를 다른 VPC로 복원할 때 지속적 또는 영구적 옵션을 포함해야 합니다.

옵션 특성은 옵션 설정에 따라 달라집니다. 예를 들어 Oracle 고급 보안 옵션인 `NATIVE_NETWORK_ENCRYPTION`은 DB 인스턴스로 전송 및 수신되는 네트워크 트래픽에 암호화 알고리즘을 지정할 수 있는 설정이 있습니다. 옵션 설정 중에는 Amazon RDS 사용에 최적화하여 변경할 수 없는 설정도 있습니다.

함께 사용할 수 없는 옵션

일부 옵션은 함께 사용할 수 없습니다. 한 옵션 또는 다른 옵션을 사용할 수 있지만 두 옵션을 동시에 사용할 수 없습니다. 함께 사용할 수 없는 옵션은 다음과 같습니다.

- Oracle Enterprise Manager Database Express, , 및 Oracle Management Agent for Enterprise Manager Cloud Control
- Oracle 기본 네트워크 암호화, , 및 Oracle 보안 소켓 Layer

옵션 그룹 생성

기본 옵션 그룹에서 설정을 가져오는 새 옵션 그룹을 생성할 수 있습니다. 그런 다음, 하나 이상의 옵션을 새 옵션 그룹에 추가합니다. 아니면 이미 기존 옵션 그룹이 있는 경우에는 기존 옵션 그룹과 모든 옵션을 새 옵션 그룹에 복사할 수 있습니다. 자세한 내용은 옵션 그룹 생성 단원을 참조하십시오.

생성된 새 옵션 그룹에는 옵션이 없습니다. 옵션을 옵션 그룹에 추가하는 방법을 배우려면 옵션 그룹에 옵션 추가 단원을 참조하십시오. 원하는 옵션을 추가했다면 이제 옵션 그룹을 DB 인스턴스에 연동시킬 수 있습니다. 이렇게 하면 DB 인스턴스에서 옵션을 사용할 수 있습니다. 옵션 그룹과 DB 인스턴스의 연동에 대한 자세한 내용은 옵션 그룹 작업에서 해당 엔진의 설명서를 참조하세요.

콘솔

옵션 그룹을 생성하는 한 가지 방법은 AWS Management Console을 사용하는 것입니다.

콘솔을 이용하여 새 옵션 그룹을 생성하려면,

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 그룹 생성을 선택합니다.
4. 보안 그룹 생성 창에서 다음과 같이 합니다.
   a. [이름(Name)]에 AWS 계정 내에서 쉽게 식별할 수 있는 옵션 그룹 이름을 입력합니다. 이름은 글자, 숫자 및 하이픈만 사용 가능합니다.
   b. Description에 옵션 그룹에 대한 간략한 설명을 입력합니다. 이 설명은 표시 용도로만 사용됩니다.
   c. Engine에서 원하는 DB 엔진을 선택합니다.
   d. 메이저 엔진 버전에서 DB 엔진의 원하는 메이저 버전을 선택합니다.
5. 계속하려면 생성을 선택합니다. 작업을 취소하려면 Cancel을 선택합니다.

AWS CLI

옵션 그룹을 생성하려면 AWS CLI `create-option-group` 명령을 다음 필수 파라미터와 함께 사용합니다.

- `--option-group-name`
- `--engine-name`
- `--major-engine-version`
- `--option-group-description`

Example

다음은 `testoptiongroup`이라는 이름의 옵션 그룹을 생성하는 예제입니다. 이 옵션 그룹은 Oracle Enterprise Edition DB 엔진과 연동됩니다. 설명은 인용 부호로 묶여 있습니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds create-option-group \
    --option-group-name testoptiongroup \
    --engine-name oracle-ee \
    --major-engine-version 19 \
    --option-group-description "Test option group for Oracle Database 19c EE"
```

Windows의 경우:

```
aws rds create-option-group ^
    --option-group-name testoptiongroup ^
    --engine-name oracle-ee ^-
    --major-engine-version 19 ^
    --option-group-description "Test option group for Oracle Database 19c EE"
```

RDS API

옵션 그룹을 생성하려면 Amazon RDS API CreateOptionGroup 작업을 호출합니다. 다음 파라미터를 포함합니다.

- OptionGroupName
- EngineName
- MajorEngineVersion
- OptionGroupDescription

옵션 그룹 생성

AWS CLI 또는 Amazon RDS API를 사용하여 옵션 그룹을 복사할 수 있습니다. 옵션 그룹을 복사하면 편리할 수 있습니다. 기존 옵션 그룹이 있으며 이 그룹의 사용자 지정 파라미터 및 값의 대부분을 새 옵션 그룹에 포함하려는 경우를 예로 들 수 있습니다. 프로덕션 환경에서 사용하는 옵션 그룹 사본을 만든 다음 사본 설정을 변경하여 다른 옵션 설정을 테스트할 수도 있습니다.

> **Note**
>
> 현재는 옵션 그룹을 다른 AWS 리전에 복사할 수 없습니다.

AWS CLI

옵션 그룹을 복사하려면 AWS CLI copy-option-group 명령을 사용합니다. 다음 필수 옵션을 포함합니다.

- `--source-option-group-identifier`
- `--target-option-group-identifier`
- `--target-option-group-description`

Example

다음은 `new-option-group`이라는 이름의 옵션 그룹을 생성하는 예제입니다. 이 옵션 그룹은 옵션 그룹 `my-option-group`의 로컬 사본입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds copy-option-group \
    --source-option-group-identifier my-option-group \
    --target-option-group-identifier new-option-group \
    --target-option-group-description "My new option group"
```

Windows의 경우:

```
aws rds copy-option-group ^
    --source-option-group-identifier my-option-group ^
    --target-option-group-identifier new-option-group ^
    --target-option-group-description "My new option group"
```

RDS API

옵션 그룹을 복사하려면 Amazon RDS API CopyOptionGroup 작업을 호출합니다. 다음 필수 파라미터를 포함합니다.

- `SourceOptionGroupIdentifier`

- `TargetOptionGroupIdentifier`
- `TargetOptionGroupDescription`

옵션 그룹에 옵션 추가

옵션을 기존 옵션 그룹에 추가할 수 있습니다. 원하는 옵션을 추가했다면 이제 옵션 그룹을 DB 인스턴스에 연동시켜 옵션을 DB 인스턴스에서 사용할 수 있습니다. 옵션 그룹과 DB 인스턴스의 연동에 대한 자세한 정보는 옵션 그룹 작업에 나열된 구체적인 해당 DB 엔진 설명서를 참조하십시오.

다음 두 가지 경우에 옵션 그룹 변경 사항이 즉시 적용되어야 합니다.

- `OEM` 옵션처럼 포트 값을 추가하거나 업데이트하는 옵션을 추가하는 경우.
- 포트 값이 포함된 옵션이 있는 옵션 그룹을 추가하거나 삭제하는 경우.

이러한 경우 콘솔에서 즉시 적용 옵션을 선택합니다. 또는 AWS CLI를 사용할 때 `--apply-immediately` 옵션을 포함하거나 Amazon RDS API를 사용할 때 `ApplyImmediately` 파라미터를 `true`로 설정할 수 있습니다. 포트 값이 포함되지 않은 옵션도 즉시 적용하거나, 혹은 DB 인스턴스의 다음 유지 관리 기간에 적용할 수 있습니다.

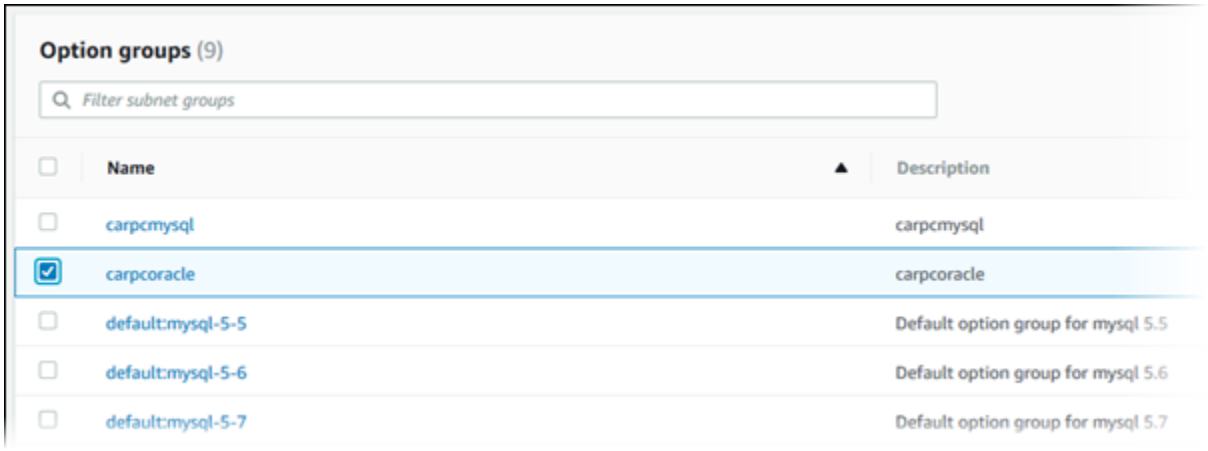
> **Note**
>
> 옵션 그룹의 옵션 값으로 보안 그룹을 지정하는 경우 옵션 그룹을 수정하여 보안 그룹을 관리합니다. DB 인스턴스를 수정하여 이 보안 그룹을 변경하거나 제거할 수 없습니다. 또한 보안 그룹은 AWS Management Console이나 AWS CLI 명령 `describe-db-instances`의 출력에 있는 DB 인스턴스 세부 정보에 나타나지 않습니다.

콘솔

AWS Management Console을 사용하여 옵션을 옵션 그룹에 추가할 수 있습니다.

콘솔을 사용하여 옵션을 옵션 그룹에 추가하려면,

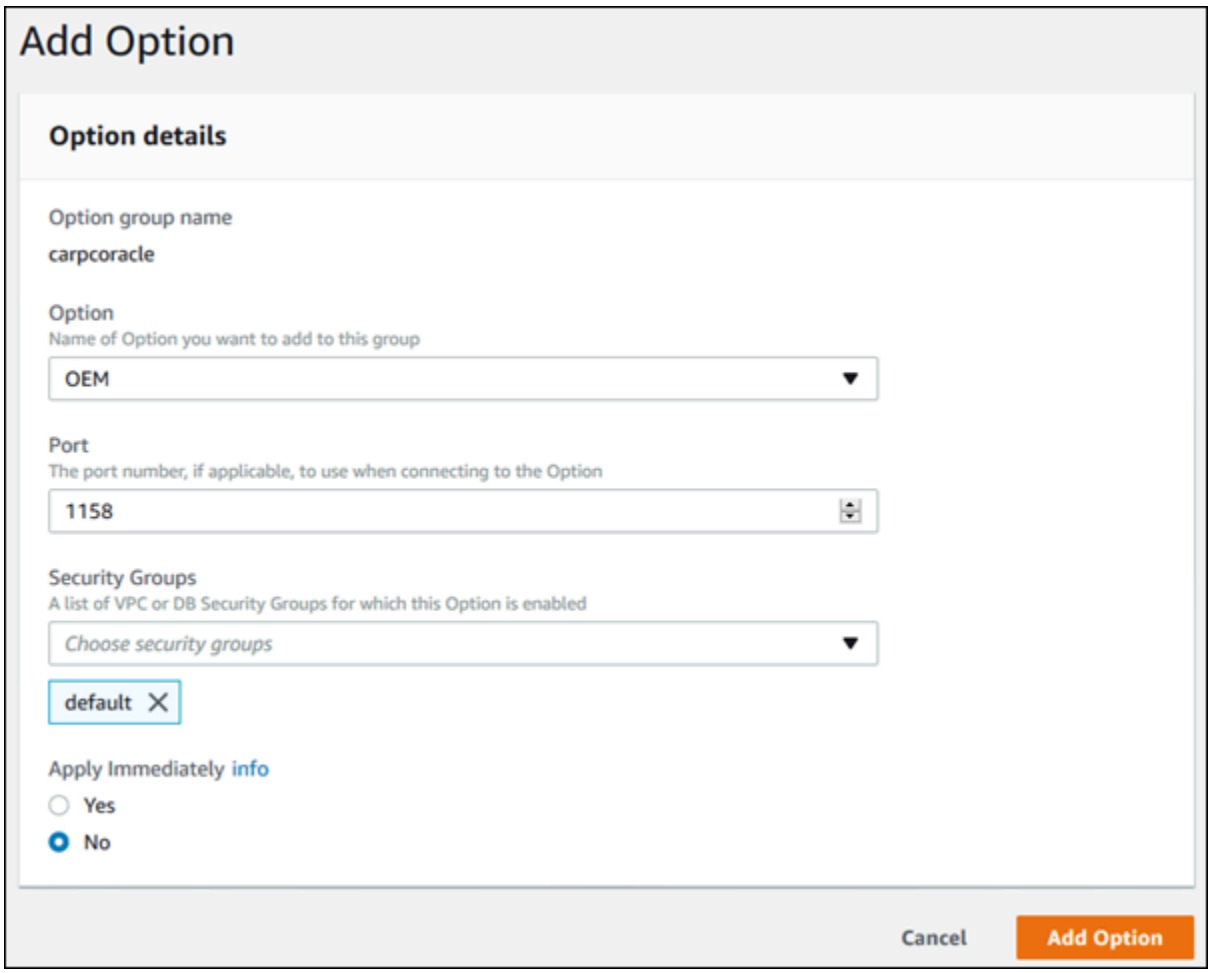
1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 수정하려는 옵션 그룹을 선택한 다음 옵션 추가를 선택합니다.



4. 옵션 추가 창에서 다음과 같이 합니다.

   a. 추가할 옵션을 선택합니다. 선택하는 옵션에 따라 값을 추가로 입력해야 하는 경우도 있습니다. 예를 들어, OEM 옵션을 선택하면 포트 값을 입력하고 보안 그룹도 지정해야 합니다.

   b. 옵션을 추가하는 즉시 연동된 모든 DB 인스턴스에서 옵션을 활성화하려면 Apply Immediately에서 Yes를 선택합니다. No(기본 설정)를 선택하면 다음 유지 관리 기간에 연동된 모든 DB 인스턴스에서 옵션이 활성화됩니다.



5. 원하는 대로 설정이 되었으면 옵션 추가를 선택합니다.

AWS CLI

옵션을 옵션 그룹에 추가하려면 추가할 옵션과 함께 AWS CLI add-option-to-option-group 명령을 실행합니다. 새로운 옵션을 연동된 모든 DB 인스턴스에서 즉시 활성화하려면 `--apply-immediately` 파라미터를 추가합니다. 기본적으로 이 옵션은 다음 유지 관리 기간에 연동된 모든 DB 인스턴스에서 활성화됩니다. 다음 필수 파라미터를 포함합니다.

- `--option-group-name`

Example

다음은 `testoptiongroup`이라는 이름의 옵션 그룹에 `America/Los_Angeles` 설정으로 `Timezone` 옵션을 추가한 후 즉시 활성화하는 예제입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds add-option-to-option-group \
    --option-group-name testoptiongroup \
    --options "OptionName=Timezone,OptionSettings=[{Name=TIME_ZONE,Value=America/
Los_Angeles}]" \
    --apply-immediately
```

Windows의 경우:

```
aws rds add-option-to-option-group ^
    --option-group-name testoptiongroup ^
    --options "OptionName=Timezone,OptionSettings=[{Name=TIME_ZONE,Value=America/
Los_Angeles}]" ^
    --apply-immediately
```

명령은 다음과 유사하게 출력됩니다.

```
...{
       "OptionName": "Timezone",
       "OptionDescription": "Change time zone",
       "Persistent": true,
       "Permanent": false,
       "OptionSettings": [
           {
               "Name": "TIME_ZONE",
               "Value": "America/Los_Angeles",
               "DefaultValue": "UTC",
               "Description": "Specifies the timezone the user wants to change the
 system time to",
               "ApplyType": "DYNAMIC",
               "DataType": "STRING",
               "AllowedValues": "Africa/Cairo,...",
               "IsModifiable": true,
               "IsCollection": false
           }
       ],
       "DBSecurityGroupMemberships": [],
       "VpcSecurityGroupMemberships": []
   }...
```

Example

다음 예에서는 Oracle OEM 옵션을 옵션 그룹에 추가합니다. 또한 사용자 지정 포트를 지정하고 이 포트에 사용할 한 쌍의 Amazon EC2 VPC 보안 그룹을 지정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds add-option-to-option-group \
    --option-group-name testoptiongroup \
    --options OptionName=OEM,Port=5500,VpcSecurityGroupMemberships="sg-test1,sg-test2"
 \
    --apply-immediately
```

Windows의 경우:

```
aws rds add-option-to-option-group ^
    --option-group-name testoptiongroup ^
    --options OptionName=OEM,Port=5500,VpcSecurityGroupMemberships="sg-test1,sg-test2"
 ^
    --apply-immediately
```

명령은 다음과 유사하게 출력됩니다.

```
OPTIONGROUP  False  oracle-ee  19 arn:aws:rds:us-east-1:1234567890:og:testoptiongroup
 Test Option Group  testoptiongroup vpc-test
OPTIONS Oracle 12c EM Express   OEM     False   False   5500
VPCSECURITYGROUPMEMBERSHIPS     active  sg-test1
VPCSECURITYGROUPMEMBERSHIPS     active  sg-test2
```

Example

다음은 Oracle 옵션 NATIVE_NETWORK_ENCRYPTION을 옵션 그룹에 추가한 후 옵션 설정을 지정하는 예제입니다. 지정할 옵션 설정이 없는 경우에는 기본 값을 사용합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds add-option-to-option-group \
    --option-group-name testoptiongroup \
    --options '[{"OptionSettings":
[{"Name":"SQLNET.ENCRYPTION_SERVER","Value":"REQUIRED"},
{"Name":"SQLNET.ENCRYPTION_TYPES_SERVER","Value":"AES256,AES192,DES"}],"OptionName":"NATIVE_NET
 \
```

```
    --apply-immediately
```

Windows의 경우:

```
aws rds add-option-to-option-group ^
    --option-group-name testoptiongroup ^
    --options "OptionSettings"=[{"Name"="SQLNET.ENCRYPTION_SERVER","Value"="REQUIRED"},
{"Name"="SQLNET.ENCRYPTION_TYPES_SERVER","Value"="AES256\,AES192\,DES"}],"OptionName"="NATIVE_N
 ^
    --apply-immediately
```

명령은 다음과 유사하게 출력됩니다.

```
...{
    "OptionName": "NATIVE_NETWORK_ENCRYPTION",
    "OptionDescription": "Native Network Encryption",
    "Persistent": false,
    "Permanent": false,
    "OptionSettings": [
        {
            "Name": "SQLNET.ENCRYPTION_TYPES_SERVER",
             "Value": "AES256,AES192,DES",
            "DefaultValue":
 "RC4_256,AES256,AES192,3DES168,RC4_128,AES128,3DES112,RC4_56,DES,RC4_40,DES40",
            "Description": "Specifies list of encryption algorithms in order of
 intended use",
            "ApplyType": "STATIC",
            "DataType": "STRING",
            "AllowedValues":
 "RC4_256,AES256,AES192,3DES168,RC4_128,AES128,3DES112,RC4_56,DES,RC4_40,DES40",
            "IsModifiable": true,
            "IsCollection": true
        },
        {
            "Name": "SQLNET.ENCRYPTION_SERVER",
            "Value": "REQUIRED",
            "DefaultValue": "REQUESTED",
            "Description": "Specifies the desired encryption behavior",
            "ApplyType": "STATIC",
            "DataType": "STRING",
            "AllowedValues": "ACCEPTED,REJECTED,REQUESTED,REQUIRED",
            "IsModifiable": true,
            "IsCollection": false
```

```
        },...
```

RDS API

Amazon RDS API를 사용하여 옵션을 옵션 그룹에 추가하려면 추가할 옵션과 함께 ModifyOptionGroup 작업을 호출합니다. 새로운 옵션을 연동된 모든 DB 인스턴스에서 즉시 활성화하려면 `ApplyImmediately` 파라미터를 추가하고 `true`로 설정합니다. 기본적으로 이 옵션은 다음 유지 관리 기간에 연동된 모든 DB 인스턴스에서 활성화됩니다. 다음 필수 파라미터를 포함합니다.

- `OptionGroupName`

옵션 그룹의 옵션 및 옵션 설정 표시하기

옵션 그룹의 옵션과 옵션 설정을 모두 표시할 수 있습니다.

콘솔

AWS Management Console을 사용하여 옵션 그룹의 옵션과 옵션 설정을 모두 표시할 수 있습니다.

옵션 그룹의 옵션과 옵션 설정을 표시하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 세부 정보를 표시할 옵션 그룹의 이름을 선택합니다. 옵션 그룹의 옵션과 옵션 설정이 나열됩니다.

AWS CLI

옵션 그룹의 옵션과 옵션 설정을 표시하려면 AWS CLI `describe-option-groups` 명령을 사용합니다. 표시할 옵션과 설정이 포함된 옵션 그룹 이름을 지정합니다. 옵션 그룹 이름을 지정하지 않으면 모든 옵션 그룹이 표시됩니다.

Example

다음은 모든 옵션 그룹에 포함된 옵션과 옵션 설정을 표시하는 예제입니다.

```
aws rds describe-option-groups
```

Example

다음은 testoptiongroup이라는 이름의 옵션 그룹에 포함된 옵션과 옵션 설정을 표시하는 예제입니다.

```
aws rds describe-option-groups --option-group-name testoptiongroup
```

RDS API

옵션 그룹의 옵션과 옵션 설정을 표시하려면 Amazon RDS API DescribeOptionGroups 작업을 사용합니다. 표시할 옵션과 설정이 포함된 옵션 그룹 이름을 지정합니다. 옵션 그룹 이름을 지정하지 않으면 모든 옵션 그룹이 표시됩니다.

옵션 설정 수정

옵션 설정을 변경할 수 있는 옵션은 추가한 이후에도 언제든지 설정을 변경할 수 있습니다. 옵션 그룹의 옵션 또는 옵션 설정을 변경할 경우에는 해당 옵션 그룹과 연동되어 있는 모든 DB 인스턴스에 변경 사항이 적용됩니다. 다양한 옵션에서 이용할 수 있는 설정에 대한 자세한 정보는 옵션 그룹 작업에서 해당 엔진의 설명서를 참조하세요.

다음 두 가지 경우에 옵션 그룹 변경 사항이 즉시 적용되어야 합니다.

- OEM 옵션처럼 포트 값을 추가하거나 업데이트하는 옵션을 추가하는 경우.
- 포트 값이 포함된 옵션이 있는 옵션 그룹을 추가하거나 삭제하는 경우.

이러한 경우 콘솔에서 즉시 적용 옵션을 선택합니다. 또는 --apply-immediately를 사용할 때 AWS CLI 옵션을 포함하거나 RDS API를 사용할 때 ApplyImmediately 파라미터를 true로 설정할 수 있습니다. 포트 값이 포함되지 않은 옵션도 즉시 적용하거나, 혹은 DB 인스턴스의 다음 유지 관리 기간에 적용할 수 있습니다.

> **Note**
>
> 옵션 그룹의 옵션 값으로 보안 그룹을 지정하는 경우 옵션 그룹을 수정하여 보안 그룹을 관리합니다. DB 인스턴스를 수정하여 이 보안 그룹을 변경하거나 제거할 수 없습니다. 또한 보안 그룹은 AWS Management Console이나 AWS CLI 명령 describe-db-instances의 출력에 있는 DB 인스턴스 세부 정보에 나타나지 않습니다.

콘솔

AWS Management Console을 사용하여 옵션 설정을 변경할 수 있습니다.

콘솔을 사용하여 옵션 설정을 변경하려면,

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 옵션을 수정하려는 옵션 그룹을 선택한 후 옵션 수정을 선택합니다.
4. 옵션 수정 창의 설치된 옵션에서 설정을 수정하려는 옵션을 선택합니다. 원하는 설정을 변경합니다.
5. 옵션을 추가하는 즉시 활성화하려면 Apply Immediately에서 Yes를 선택합니다. No(기본 설정)를 선택하면 다음 유지 관리 기간에 연동된 모든 DB 인스턴스에서 옵션이 활성화됩니다.
6. 원하는 대로 설정이 되었으면 Modify Option을 선택합니다.

AWS CLI

옵션 설정을 수정하려면 변경할 옵션 그룹 및 옵션과 함께 AWS CLI `add-option-to-option-group` 명령을 사용합니다. 기본적으로 이 옵션은 다음 유지 관리 기간에 연동된 모든 DB 인스턴스에서 활성화됩니다. 변경 사항을 연동된 모든 DB 인스턴스에 즉시 적용하려면 `--apply-immediately` 파라미터를 포함합니다. 옵션 설정을 변경하려면 `--settings` 인수를 사용합니다.

Example

다음은 `testoptiongroup`이라는 이름의 옵션 그룹에서 Oracle Enterprise Manager Database Control(OEM)이 사용하는 포트 설정을 변경한 다음 이 변경 사항을 즉시 적용하는 예제입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds add-option-to-option-group \
    --option-group-name testoptiongroup \
    --options OptionName=OEM,Port=5432,DBSecurityGroupMemberships=default \
    --apply-immediately
```

Windows의 경우:

```
aws rds add-option-to-option-group ^
    --option-group-name testoptiongroup ^
```

```
    --options OptionName=OEM,Port=5432,DBSecurityGroupMemberships=default ^
    --apply-immediately
```

명령은 다음과 유사하게 출력됩니다.

```
OPTIONGROUP   False  oracle-ee  19  arn:aws:rds:us-east-1:1234567890:og:testoptiongroup
   Test Option Group    testoptiongroup
OPTIONS Oracle 12c EM Express   OEM     False   False   5432
DBSECURITYGROUPMEMBERSHIPS   default  authorized
```

Example

다음은 Oracle 옵션 NATIVE_NETWORK_ENCRYPTION의 설정을 변경하는 예제입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds add-option-to-option-group \
    --option-group-name testoptiongroup \
    --options '[{"OptionSettings":
[{"Name":"SQLNET.ENCRYPTION_SERVER","Value":"REQUIRED"},
{"Name":"SQLNET.ENCRYPTION_TYPES_SERVER","Value":"AES256,AES192,DES,RC4_256"}],"OptionName":"NA
 \
    --apply-immediately
```

Windows의 경우:

```
aws rds add-option-to-option-group ^
    --option-group-name testoptiongroup ^
    --options "OptionSettings"=[{"Name"="SQLNET.ENCRYPTION_SERVER","Value"="REQUIRED"},
{"Name"="SQLNET.ENCRYPTION_TYPES_SERVER","Value"="AES256\,AES192\,DES
\,RC4_256"}],"OptionName"="NATIVE_NETWORK_ENCRYPTION" ^
    --apply-immediately
```

명령은 다음과 유사하게 출력됩니다.

```
OPTIONGROUP   False  oracle-ee  19  arn:aws:rds:us-east-1:1234567890:og:testoptiongroup
   Test Option Group    testoptiongroup
OPTIONS Oracle Advanced Security - Native Network Encryption
 NATIVE_NETWORK_ENCRYPTION       False   False
OPTIONSETTINGS
 RC4_256,AES256,AES192,3DES168,RC4_128,AES128,3DES112,RC4_56,DES,RC4_40,DES40 STATIC
 STRING
```

```
    RC4_256,AES256,AES192,3DES168,RC4_128,AES128,3DES112,RC4_56,DES,RC4_40,DES40
  Specifies list of encryption algorithms in order of intended use
    True     True    SQLNET.ENCRYPTION_TYPES_SERVER  AES256,AES192,DES,RC4_256
OPTIONSETTINGS  ACCEPTED,REJECTED,REQUESTED,REQUIRED    STATIC  STRING  REQUESTED
  Specifies the desired encryption behavior  False   True  SQLNET.ENCRYPTION_SERVER
  REQUIRED
OPTIONSETTINGS  SHA1,MD5   STATIC  STRING  SHA1,MD5    Specifies list of
  checksumming algorithms in order of intended use      True    True
  SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER  SHA1,MD5
OPTIONSETTINGS  ACCEPTED,REJECTED,REQUESTED,REQUIRED  STATIC  STRING
  REQUESTED     Specifies the desired data integrity behavior   False   True
  SQLNET.CRYPTO_CHECKSUM_SERVER  REQUESTED
```

RDS API

옵션 설정을 수정하려면 변경할 옵션 그룹 및 옵션과 함께 Amazon RDS API `ModifyOptionGroup` 명령을 사용합니다. 기본적으로 이 옵션은 다음 유지 관리 기간에 연동된 모든 DB 인스턴스에서 활성화됩니다. 변경 사항을 연동된 모든 DB 인스턴스에 즉시 적용하려면 `ApplyImmediately` 파라미터를 포함시키고 `true`로 설정합니다.

옵션 그룹에서 옵션 제거

일부 옵션은 옵션 그룹에서 제거할 수 있지만 그렇지 않은 옵션도 있습니다. 지속적인 옵션은 DB 인스턴스와 해당 옵션 그룹의 연동을 해제해야만 옵션 그룹에서 제거가 가능합니다. 영구적인 옵션은 옵션 그룹에서 절대로 제거할 수 없습니다. 제거할 수 있는 옵션에 대한 자세한 정보는 옵션 그룹 작업에 나열된 구체적인 해당 엔진 설명서를 참조하십시오.

옵션 그룹에서 모든 옵션을 제거하더라도 Amazon RDS가 옵션 그룹을 삭제하지는 않습니다. 비어 있는 옵션 그룹과 연동되어 있는 DB 인스턴스도 활성화된 옵션이 없을 뿐 연동 상태는 유지합니다. 또는 DB 인스턴스를 기본(비어있는) 옵션 그룹과 연동시키면 DB 인스턴스에서 모든 옵션을 제거할 수 있습니다.

콘솔

AWS Management Console을 사용하여 옵션 그룹에서 옵션을 제거할 수 있습니다.

console을 사용하여 옵션 그룹에서 옵션을 제거하려면,

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.

3. 옵션을 제거하려는 옵션 그룹을 선택한 후 옵션 삭제를 선택합니다.
4. 옵션 삭제 창에서 다음과 같이 합니다.
   - 삭제하려는 옵션 확인란을 선택합니다.
   - 삭제 후 바로 적용하려면 즉시 적용에서 예를 선택합니다. No(기본 설정)를 선택하면 다음 유지 관리 기간에 연동된 모든 DB 인스턴스에서 옵션이 삭제됩니다.



5. 원하는 대로 설정이 되었으면 Yes, Delete를 선택합니다.

AWS CLI

옵션 그룹에서 옵션을 제거하려면 삭제할 옵션과 함께 AWS CLI `remove-option-from-option-group` 명령을 사용합니다. 기본적으로 이 옵션은 다음 유지 관리 기간에 연동된 모든 DB 인스턴스에서 제거됩니다. 변경 사항을 즉시 적용하려면 `--apply-immediately` 파라미터를 추가합니다.

Example

다음은 `testoptiongroup`이라는 이름의 옵션 그룹에서 Oracle Enterprise Manager Database Control(OEM) 옵션을 제거한 후 변경 사항을 즉시 적용하는 예제입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds remove-option-from-option-group \
    --option-group-name testoptiongroup \
    --options OEM \
    --apply-immediately
```

Windows의 경우:

```
aws rds remove-option-from-option-group ^
    --option-group-name testoptiongroup ^
    --options OEM ^
    --apply-immediately
```

명령은 다음과 유사하게 출력됩니다.

```
OPTIONGROUP    testoptiongroup oracle-ee   19    Test option group
```

RDS API

옵션 그룹에서 옵션을 제거하려면 Amazon RDS API ModifyOptionGroup 작업을 사용합니다. 기본적으로 이 옵션은 다음 유지 관리 기간에 연동된 모든 DB 인스턴스에서 제거됩니다. 변경 사항을 즉시 적용하려면 ApplyImmediately 파라미터를 추가하고 true로 설정합니다.

다음 파라미터를 포함합니다.

- OptionGroupName
- OptionsToRemove.OptionName

옵션 그룹 삭제

다음 기준을 충족하는 경우에만 옵션 그룹을 삭제할 수 있습니다.

- Amazon RDS 리소스와 연결된 상태가 아니어야 합니다. 옵션 그룹은 DB 인스턴스, 수동 DB 스냅샷 또는 자동화된 DB 스냅샷에 연결할 수 있습니다.
- 기본 옵션 그룹이 아니어야 합니다.

DB 인스턴스와 DB 스냅샷에서 사용하는 옵션 그룹을 식별하려면 다음 CLI 명령을 사용하면 됩니다.

```
aws rds describe-db-instances \
    --query 'DBInstances[*].
[DBInstanceIdentifier,OptionGroupMemberships[].OptionGroupName]'
```

```
aws rds describe-db-snapshots | jq -r '.DBSnapshots[] | "\(.DBInstanceIdentifier),
\(.OptionGroupName)"' | sort | uniq
```

RDS 리소스와 연결된 옵션 그룹을 삭제하려고 하면 다음과 같은 오류가 반환됩니다.

```
An error occurred (InvalidOptionGroupStateFault) when calling the DeleteOptionGroup
 operation: The option group 'optionGroupName' cannot be deleted because it is in use.
```

옵션 그룹과 연결된 Amazon RDS 리소스를 찾으려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 세부 정보를 표시할 옵션 그룹의 이름을 선택합니다.
4. 연결된 Amazon RDS 리소스의 연결된 인스턴스 및 스냅샷 섹션을 확인하십시오.

DB 인스턴스가 옵션 그룹과 연결되어 있으면 DB 인스턴스가 다른 옵션 그룹을 사용하도록 수정하십시오. 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

수동 DB 스냅샷이 옵션 그룹과 연결되어 있으면 DB 스냅샷이 다른 옵션 그룹을 사용하도록 수정합니다. AWS CLI `modify-db-snapshot` 명령을 사용하여 수정할 수 있습니다.

> **Note**
>
> 자동화된 DB 스냅샷의 옵션 그룹은 수정할 수 없습니다.

콘솔

옵션 그룹을 삭제하는 한 가지 방법은 AWS Management Console을 사용하는 것입니다.

콘솔을 사용해 옵션 그룹을 삭제하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.

2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 옵션 그룹을 선택합니다.
4. 그룹 삭제를 선택합니다.
5. 확인 페이지에서 삭제를 선택하여 옵션 그룹 삭제를 완료하거나 취소를 선택하여 삭제를 취소합니다.

AWS CLI

옵션 그룹을 삭제하려면 AWS CLI `delete-option-group` 명령을 다음 필수 파라미터와 함께 사용합니다.

- `--option-group-name`

Example

다음 예제에서는 이름이 `testoptiongroup`인 옵션 그룹을 삭제합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds delete-option-group \
    --option-group-name testoptiongroup
```

Windows의 경우:

```
aws rds delete-option-group ^
    --option-group-name testoptiongroup
```

RDS API

옵션 그룹을 삭제하려면 Amazon RDS API `DeleteOptionGroup` 작업을 호출합니다. 다음 파라미터를 포함합니다.

- `OptionGroupName`

Amazon RDS의 파라미터 그룹

[데이터베이스 파라미터(Database parameters)]에서 데이터베이스 구성 방법을 지정합니다. 예를 들어 데이터베이스 파라미터는 메모리를 비롯하여 데이터베이스에 할당할 리소스의 양을 지정할 수 있습니다.

DB 인스턴스 및 Multi-AZ DB 클러스터를 파라미터 그룹과 연결하여 데이터베이스 구성을 관리합니다. Amazon RDS는 기본 설정으로 파라미터 그룹을 정의합니다. 맞춤형 설정으로 자신만의 파라미터 그룹을 정의할 수 있습니다.

> **Note**
>
> 일부 DB 엔진은 데이터베이스의 옵션 그룹에 옵션으로 추가할 수 있는 추가 기능을 제공합니다. 옵션 그룹에 대한 자세한 내용은 옵션 그룹 작업 섹션을 참조하세요.

주제

- 파라미터 그룹 개요
- Amazon RDS DB 인스턴스용 DB 파라미터 그룹
- 다중 AZ DB 클러스터용 DB 클러스터 파라미터 그룹 작업
- DB 파라미터 그룹 비교
- DB 파라미터 지정

파라미터 그룹 개요

DB 파라미터 그룹은 하나 이상의 DB 인스턴스에 적용되는 엔진 구성 값의 컨테이너 역할을 합니다.

DB 클러스터 파라미터 그룹은 다중 AZ DB 클러스터에만 적용됩니다. 다중 AZ DB 클러스터에서 DB 클러스터 파라미터 그룹의 설정은 클러스터의 모든 DB 인스턴스에 적용됩니다. DB 엔진 및 DB 엔진 버전에 대한 기본 DB 파라미터 그룹은 DB 클러스터의 각 DB 인스턴스에 사용됩니다.

주제

- 기본 및 사용자 지정 파라미터 그룹
- 정적 및 동적 DB 인스턴스 파라미터
- 정적 및 동적 DB 클러스터 파라미터
- 문자 집합 파라미터

- 지원되는 파라미터 및 파라미터 값

기본 및 사용자 지정 파라미터 그룹

DB 파라미터 그룹을 지정하지 않고 DB 인스턴스를 만드는 경우 DB 인스턴스에서는 기본 DB 파라미터 그룹을 사용합니다. 이와 마찬가지로 DB 클러스터 파라미터 그룹을 지정하지 않고 다중 AZ 클러스터를 생성할 경우 이 DB 클러스터에서는 기본 DB 클러스터 파라미터 그룹을 사용합니다. 각 기본 파라미터 그룹에는 인스턴스의 엔진, 컴퓨팅 클래스 및 할당된 스토리지에 따른 데이터베이스 엔진 기본값과 Amazon RDS 시스템 기본값이 들어 있습니다.

기본 DB 파라미터 그룹의 파라미터 설정은 수정할 수 없습니다. 대신에 다음 작업을 할 수 있습니다.

1. 새 파라미터 그룹을 생성해야 합니다.
2. 원하는 파라미터의 설정을 변경합니다. 파라미터 그룹에서 모든 DB 엔진 파라미터를 수정할 수 있는 것은 아닙니다.
3. DB 인스턴스 또는 DB 클러스터를 수정하여 새로운 파라미터 그룹을 연결하세요.

   새 DB 파라미터 그룹을 DB 인스턴스와 연결하면 연결은 즉시 이루어집니다. DB 인스턴스 수정에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요. 다중 AZ DB 클러스터 수정에 대한 자세한 내용은 Amazon RDS용 다중 AZ DB 클러스터 수정 섹션을 참조하세요.

> **Note**
>
> 사용자 지정 파라미터 그룹을 사용하도록 DB 인스턴스를 수정하고 DB 인스턴스를 시작하면 RDS는 시작 프로세스의 일부로 DB 인스턴스를 자동으로 재부팅합니다. AlwaysOn 또는 미러링 옵션이 활성화된 RDS for SQL Server 다중 AZ 인스턴스의 경우 시작 프로세스 후 인스턴스를 재부팅하면 장애 조치가 수행될 수 있습니다.

RDS는 DB 인스턴스가 재부팅된 후에만 수정된 정적 및 동적 파라미터를 새로 연결된 파라미터 그룹에 적용합니다. 그러나 DB 파라미터 그룹을 DB 인스턴스에 연결한 후 DB 파라미터 그룹에서 동적 파라미터를 수정하면 이러한 변경 사항이 재부팅 없이 즉시 적용됩니다. DB 파라미터 그룹 변경에 대한 자세한 내용은 [g26]Amazon RDS DB 인스턴스 수정[/g26][g25]???[/g25] 단원을 참조하세요.

DB 파라미터 그룹 내의 파라미터를 업데이트하는 경우 변경 사항은 이 파라미터 그룹과 연결된 모든 DB 인스턴스에 적용됩니다. 이와 마찬가지로 다중 AZ DB 클러스터 파라미터 그룹 내의 파라미터를 업데이트할 경우, 변경 사항은 이 DB 클러스터 파라미터 그룹과 연결된 모든 Aurora DB 클러스터에 적용됩니다.

파라미터 그룹을 처음부터 생성하지 않으려면 AWS CLI copy-db-parameter-group 명령 또는 copy-db-cluster-parameter-group 명령을 사용하여 기존 파라미터 그룹을 복사할 수 있습니다. 경우에 따라 파라미터 그룹을 복사하는 것이 유용할 수 있습니다. 예를 들어, 기존 DB 파라미터 그룹의 사용자 지정 파라미터 및 값의 대부분을 새 DB 파라미터 그룹에 포함하고자 할 수 있습니다.

정적 및 동적 DB 인스턴스 파라미터

DB 인스턴스 파라미터는 정적이거나 동적입니다. 이들은 다음과 같은 차이가 있습니다.

- 정적 파라미터를 변경하고 DB 파라미터 그룹을 저장한 후 연결된 DB 인스턴스를 수동으로 재부팅하면 파라미터 변경 내용이 적용됩니다. 정적 파라미터의 경우 콘솔은 `ApplyMethod`로 항상 `pending-reboot`를 사용합니다.
- 동적 파라미터를 변경하면 기본적으로 파라미터 변경이 재부팅 없이 즉시 적용됩니다. AWS Management Console을 사용하여 DB 인스턴스 파라미터 값을 변경하는 경우에는 동적 파라미터의 `ApplyMethod`로 항상 `immediate`를 사용합니다. 연결된 DB 인스턴스가 재부팅된 후로 파라미터 변경을 연기하려면 AWS CLI 또는 RDS API를 사용합니다. 파라미터 변경을 위해 `ApplyMethod`를 `pending-reboot`로 설정합니다.

> **Note**
>
> AWS CLI 또는 RDS API에서 RDS for SQL Server DB 인스턴스에 동적 파라미터와 함께 `pending-reboot`를 사용하면 오류가 생성됩니다. RDS for SQL Server에 `apply-immediately`를 사용합니다.

파라미터 값 변경을 위한 AWS CLI 사용과 관련한 자세한 내용은 modify-db-parameter-group 섹션을 참조하세요. 파라미터 값 변경을 위한 RDS API 사용과 관련한 자세한 내용은 ModifyDBParameterGroup 섹션을 참조하세요.

DB 인스턴스에서 연결된 DB 파라미터 그룹에 대한 최신 변경 내용을 사용하고 있지 않은 경우 콘솔에 DB 파라미터 그룹이 재시작-보류 중 상태로 표시됩니다. 이 상태로 인해 다음번 유지 관리 기간 중에 자동 재부팅이 되지는 않습니다. 최신 파라미터 변경 내용을 이 DB 인스턴스에 적용하려면 해당 DB 인스턴스를 수동으로 재부팅해야 합니다.

정적 및 동적 DB 클러스터 파라미터

DB 클러스터 파라미터는 정적이거나 동적입니다. 이들은 다음과 같은 차이가 있습니다.

- 정적 파라미터를 변경하고 DB 클러스터 파라미터 그룹을 저장하면 연결된 DB 클러스터를 수동으로 재부팅한 후에 파라미터 변경 내용이 적용됩니다. 정적 파라미터의 경우 콘솔은 `ApplyMethod`로 항상 `pending-reboot`를 사용합니다.
- 동적 파라미터를 변경하면 기본적으로 파라미터 변경이 재부팅 없이 즉시 적용됩니다. AWS Management Console을 사용하여 DB 클러스터 파라미터 값을 변경하는 경우에는 동적 파라미터의 `ApplyMethod`로 항상 `immediate`를 사용합니다. 연결된 DB 클러스터가 재부팅된 후로 파라미터 변경을 연기하려면 AWS CLI 또는 RDS API를 사용합니다. 파라미터 변경을 위해 `ApplyMethod`를 `pending-reboot`로 설정합니다.

파라미터 값 변경을 위한 AWS CLI 사용과 관련한 자세한 내용은 modify-db-cluster-parameter-group 섹션을 참조하세요. 파라미터 값 변경을 위한 RDS API 사용과 관련한 자세한 내용은 ModifyDBClusterParameterGroup 섹션을 참조하세요.

문자 집합 파라미터

DB 인스턴스 또는 다중 AZ DB 클러스터를 만들기 전에 파라미터 그룹에 있는 데이터베이스의 문자 세트 또는 데이터 정렬과 관련된 파라미터를 모두 설정합니다. 인스턴스 또는 클러스터 안에 데이터베이스를 생성하기 전에도 이 설정을 수행해야 합니다. 이렇게 하면 기본 데이터베이스와 새 데이터베이스가 지정한 문자 세트 및 데이터 정렬 값을 사용하게 됩니다. 문자 세트 또는 데이터 정렬 파라미터를 변경해도 기존 데이터베이스에는 변경된 파라미터가 적용되지 않습니다.

일부 DB 엔진의 경우 다음과 같이 `ALTER DATABASE` 명령을 사용하여 기존 데이터베이스의 문자 세트 또는 데이터 정렬 값을 변경할 수 있습니다.

```
ALTER DATABASE database_name CHARACTER SET character_set_name COLLATE collation;
```

데이터베이스의 문자 집합 또는 데이터 정렬 값 변경에 대한 자세한 내용은 DB 엔진 설명서를 참조하세요.

지원되는 파라미터 및 파라미터 값

DB 엔진에서 지원되는 파라미터를 확인하려면 DB 인스턴스 또는 DB 클러스터에서 사용되는 DB 파라미터 그룹 및 DB 클러스터 파라미터 그룹에서 파라미터를 확인합니다. 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 값 보기 및 DB 클러스터 파라미터 그룹의 파라미터 값 보기(을)를 참조하세요.

대부분의 경우 표현식, 수식 및 함수를 사용하여 정수 및 부울 파라미터 값을 지정할 수 있습니다. 함수에 수학 로그식을 넣을 수 있습니다. 그러나 일부 파라미터는 파라미터 값에 대한 표현식, 수식 및 함수를 지원하지 않습니다. 자세한 내용은 DB 파라미터 지정 섹션을 참조하세요.

파라미터 그룹에 파라미터를 잘못 설정하면 성능 저하나 시스템 불안정 등의 의도하지 않은 부작용이 있을 수 있습니다. 데이터베이스 파라미터를 수정할 때 항상 주의하고 DB 파라미터 그룹을 수정하기 전에 데이터를 백업하세요. 파라미터 그룹 변경 내용을 프로덕션 DB 인스턴스나 DB 클러스터에 적용하기 전에 테스트 DB 인스턴스나 DB 클러스터에 적용해 봐야 합니다.

Amazon RDS DB 인스턴스용 DB 파라미터 그룹

DB 인스턴스는 DB 파라미터 그룹을 사용합니다. 다음 섹션에서는 DB 인스턴스 파라미터 그룹 구성 및 관리에 대해 설명합니다.

주제

- Amazon RDS에서 DB 파라미터 그룹 생성
- Amazon RDS의 DB 인스턴스에 DB 파라미터 그룹 연결
- Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정
- Amazon RDS에서 DB 파라미터 그룹의 파라미터를 기본값으로 재설정
- Amazon RDS에서 DB 파라미터 그룹 복사
- Amazon RDS에 DB 파라미터 그룹 나열
- Amazon RDS에서 DB 파라미터 그룹의 파라미터 값 보기
- Amazon RDS에서 DB 파라미터 그룹 삭제

Amazon RDS에서 DB 파라미터 그룹 생성

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 새 DB 파라미터 그룹을 생성할 수 있습니다.

DB 파라미터 그룹 이름에는 다음과 같은 제한이 적용됩니다.

- 이름은 1~255자의 문자, 숫자 또는 하이픈으로 구성되어야 합니다.

  기본 파라미터 그룹 이름에는 마침표(예: `default.mysql8.0`)가 포함될 수 있습니다. 하지만 사용자 지정 파라미터 그룹 이름에는 마침표를 포함할 수 없습니다.
- 첫 번째 자리는 문자여야 합니다.
- 이름은 하이픈으로 끝나거나 2개 연속 하이픈을 포함할 수 없습니다.

콘솔

DB 파라미터 그룹을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.
3. [Create parameter group]을 선택합니다.
4. 파라미터 그룹 이름에 새 DB 파라미터 그룹의 이름을 입력합니다.
5. 설명에 새 DB 클러스터 파라미터 그룹에 대한 설명을 입력합니다.
6. 엔진 유형에서 DB 엔진을 선택합니다.
7. 파라미터 그룹 패밀리에서 DB 파라미터 그룹 패밀리를 선택합니다.
8. 유형에서 DB 파라미터 그룹을 선택합니다.
9. 생성(Create)을 선택합니다.

AWS CLI

DB 파라미터 그룹을 생성하려면 AWS CLI`create-db-parameter-group` 명령을 사용합니다. 다음 예에서는 'My new parameter group'이라는 설명과 함께 mydbparametergroup이라는 MySQL 버전 8.0용 DB 파라미터 그룹을 생성합니다.

다음 필수 파라미터를 포함합니다.

- `--db-parameter-group-name`
- `--db-parameter-group-family`
- `--description`

사용 가능한 모든 파라미터 그룹 패밀리를 나열하려면 다음 명령을 사용합니다.

```
aws rds describe-db-engine-versions --query "DBEngineVersions[].DBParameterGroupFamily"
```

> **Note**
>
> 출력에 중복이 있습니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --db-parameter-group-family MySQL8.0 \
    --description "My new parameter group"
```

Windows의 경우:

```
aws rds create-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --db-parameter-group-family MySQL8.0 ^
    --description "My new parameter group"
```

다음과 비슷한 출력이 생성됩니다.

```
DBPARAMETERGROUP  mydbparametergroup  mysql8.0  My new parameter group
```

RDS API

DB 파라미터 그룹을 생성하려면 RDS API `CreateDBParameterGroup` 작업을 사용합니다.

다음 필수 파라미터를 포함합니다.

- `DBParameterGroupName`
- `DBParameterGroupFamily`
- `Description`

Amazon RDS의 DB 인스턴스에 DB 파라미터 그룹 연결

사용자 지정 설정을 사용하여 사용자의 DB 파라미터 그룹을 생성할 수 있습니다. AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB 파라미터 그룹을 DB 인스턴스와 연결할 수 있습니다. DB 인스턴스를 생성하거나 수정할 때 이 작업을 수행할 수 있습니다.

DB 파라미터 그룹 생성에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹 생성 단원을 참조하세요. DB 인스턴스 생성에 대한 자세한 내용은 Amazon RDS DB 인스턴스 생성을(를) 참조하세요. DB 인스턴스 수정에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하세요.

> **Note**
>
> 새 DB 파라미터 그룹을 DB 인스턴스와 연결하면 수정된 정적 파라미터 및 동적 파라미터는 DB 인스턴스가 재부팅된 후에만 적용됩니다. 그러나 DB 파라미터 그룹을 DB 인스턴스에 연결한 후 DB 파라미터 그룹에서 동적 파라미터를 수정하면 이러한 변경 사항이 재부팅 없이 즉시 적용됩니다.

콘솔

DB 파라미터 그룹을 DB 인스턴스와 연결하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 다음 변경하려는 DB 인스턴스를 선택합니다.
3. 수정을 선택합니다. Modify DB instance(DB 인스턴스 수정) 페이지가 나타납니다.
4. DB 파라미터 그룹 설정을 변경합니다.
5. [Continue]를 수정 사항을 요약한 내용을 확인합니다.
6. (선택 사항) 즉시 적용을 선택하여 변경 내용을 즉시 적용합니다. 일부의 경우 이 옵션을 선택하면 중단이 발생할 수 있습니다. 자세한 내용은 수정 예약 설정 사용 섹션을 참조하세요.
7. 확인 페이지에서 변경 내용을 검토합니다. 변경 내용이 정확할 경우 DB 인스턴스 수정을 선택하여 변경 내용을 저장합니다.

   또는 뒤로를 선택하여 변경 내용을 편집하거나 취소를 선택하여 변경 내용을 취소합니다.

AWS CLI

DB 파라미터 그룹을 DB 인스턴스와 연결하려면 다음 옵션과 함께 AWS CLI `modify-db-instance` 명령을 사용합니다.

- `--db-instance-identifier`
- `--db-parameter-group-name`

다음 예제에서는 mydbpg DB 파라미터 그룹을 database-1 DB 인스턴스와 연결합니다. `--apply-immediately`를 사용하면 변경 내용이 즉시 적용됩니다. `--no-apply-immediately`를 사용하여

다음 유지 관리 기간 동안 변경 사항을 적용합니다. 자세한 내용은 수정 예약 설정 사용 섹션을 참조하세요.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier database-1 \
    --db-parameter-group-name mydbpg \
    --apply-immediately
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier database-1 ^
    --db-parameter-group-name mydbpg ^
    --apply-immediately
```

RDS API

DB 파라미터 그룹을 DB 인스턴스와 연결하려면 RDS API `ModifyDBInstance` 작업을 다음 파라미터와 함께 사용합니다.

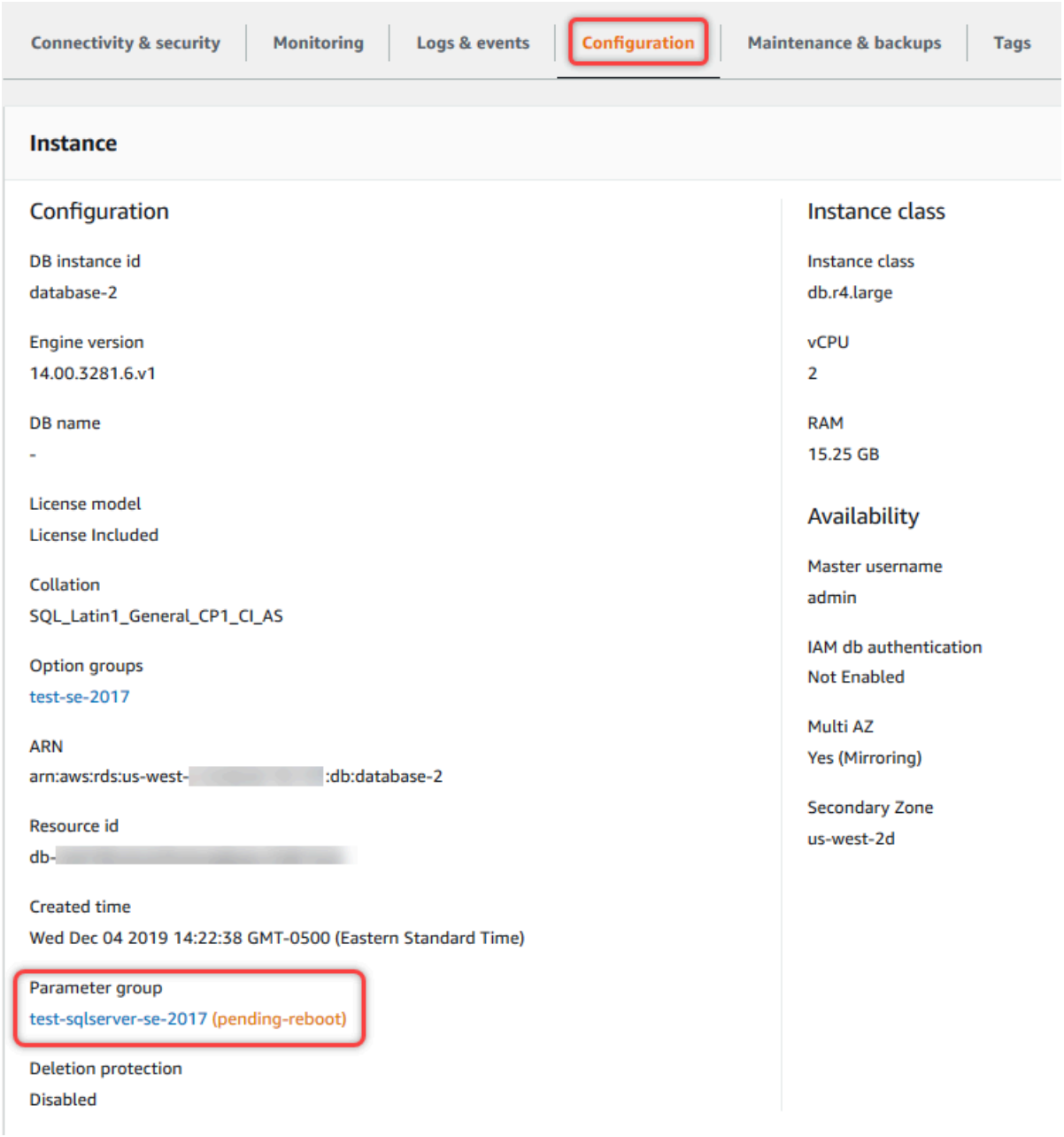
- `DBInstanceName`
- `DBParameterGroupName`

Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정

고객이 생성한 DB 파라미터 그룹의 파라미터 값은 수정할 수 있지만, 기본 DB 파라미터 그룹의 파라미터 값은 변경할 수 없습니다. 고객이 생성한 DB 파라미터 그룹의 파라미터를 변경하면 DB 파라미터 그룹과 연결된 모든 DB 인스턴스에 해당 변경 내용이 적용됩니다.

일부 파라미터에 대한 변경 사항은 재부팅 없이 DB 인스턴스에 즉시 적용됩니다. 다른 파라미터에 대한 변경 내용은 DB 인스턴스를 재부팅한 후에만 적용됩니다. RDS 콘솔에는 구성 탭에서 DB 인스턴스와 연결된 DB 파라미터 그룹의 상태가 표시됩니다. 예를 들어 DB 인스턴스에서 연결된 DB 파라미터 그룹에 대한 최신 변경 내용을 사용하고 있지 않다고 가정해 봅시다. 이 경우 RDS 콘솔에 DB 파라미터 그룹이 pending-reboot(재시작 보류 중) 상태로 표시됩니다. 최신 파라미터 변경 내용을 이 DB 인스턴스에 적용하려면 해당 DB 인스턴스를 수동으로 재부팅해야 합니다. 파라미터를 수정한 후에는 항상

DB 인스턴스를 재부팅하는 것이 좋습니다. DB 인스턴스를 재부팅하지 않으면 장애 조치 작업이 평소보다 오래 걸릴 수 있습니다.



콘솔

DB 파라미터 그룹의 파라미터를 수정하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.

3. 목록에서 수정할 파라미터 그룹의 이름을 선택합니다.
4. 파라미터 그룹 작업에서 편집을 선택합니다.
5. 수정할 파라미터의 값을 변경합니다. 대화 상자 오른쪽 위의 화살표 키를 사용하여 파라미터를 스크롤할 수 있습니다.

   기본 파라미터 그룹의 값은 변경할 수 없습니다.
6. 변경 사항 저장을 선택합니다.

AWS CLI

DB 파라미터 그룹을 수정하려면 AWS CLI`modify-db-parameter-group` 명령을 다음 필수 옵션과 함께 사용합니다.

- `--db-parameter-group-name`
- `--parameters`

다음 예에서는 mydbparametergroup이라는 DB 파라미터 그룹에서 `max_connections` 및 `max_allowed_packet` 값을 수정합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --parameters
 "ParameterName=max_connections,ParameterValue=250,ApplyMethod=immediate" \

 "ParameterName=max_allowed_packet,ParameterValue=1024,ApplyMethod=immediate"
```

Windows의 경우:

```
aws rds modify-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --parameters
 "ParameterName=max_connections,ParameterValue=250,ApplyMethod=immediate" ^

 "ParameterName=max_allowed_packet,ParameterValue=1024,ApplyMethod=immediate"
```

다음과 같은 출력이 생성됩니다.

```
DBPARAMETERGROUP  mydbparametergroup
```

RDS API

DB 파라미터 그룹을 수정하려면 RDS API `ModifyDBParameterGroup` 작업을 다음 필수 파라미터와 함께 사용합니다.

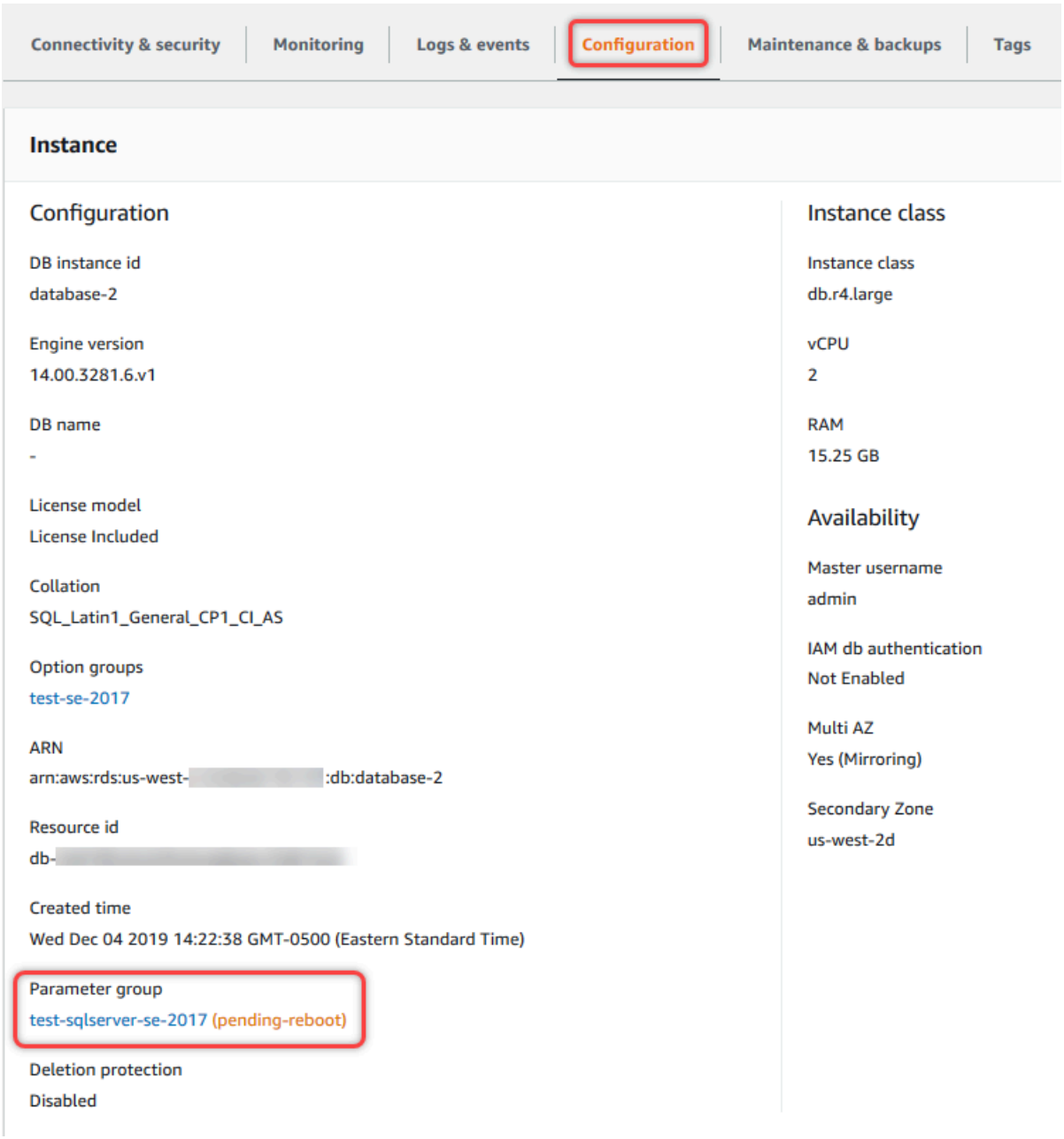
- `DBParameterGroupName`
- `Parameters`

Amazon RDS에서 DB 파라미터 그룹의 파라미터를 기본값으로 재설정

고객이 생성한 DB 파라미터 그룹의 파라미터 값을 기본값으로 재설정할 수 있습니다. 고객이 생성한 DB 파라미터 그룹의 파라미터를 변경하면 DB 파라미터 그룹과 연결된 모든 DB 인스턴스에 해당 변경 내용이 적용됩니다.

콘솔을 사용하는 경우 특정 파라미터를 기본값으로 재설정할 수 있습니다. 하지만 DB 파라미터 그룹의 모든 파라미터를 한꺼번에 손쉽게 재설정할 수는 없습니다. AWS CLI 또는 RDS API를 사용하는 경우 특정 파라미터를 기본값으로 재설정할 수 있습니다. DB 파라미터 그룹의 모든 파라미터를 한꺼번에 재설정할 수도 있습니다.

일부 파라미터에 대한 변경 사항은 재부팅 없이 DB 인스턴스에 즉시 적용됩니다. 다른 파라미터에 대한 변경 내용은 DB 인스턴스를 재부팅한 후에만 적용됩니다. RDS 콘솔에는 구성 탭에서 DB 인스턴스와 연결된 DB 파라미터 그룹의 상태가 표시됩니다. 예를 들어 DB 인스턴스에서 연결된 DB 파라미터 그룹에 대한 최신 변경 내용을 사용하고 있지 않다고 가정해 봅시다. 이 경우 RDS 콘솔에 DB 파라미터 그룹이 pending-reboot(재시작 보류 중) 상태로 표시됩니다. 최신 파라미터 변경 내용을 이 DB 인스턴스에 적용하려면 해당 DB 인스턴스를 수동으로 재부팅해야 합니다.



> **Note**
>
> 기본 DB 파라미터 그룹에서 파라미터는 항상 기본값으로 설정됩니다.

콘솔

DB 파라미터 그룹의 파라미터를 기본값으로 재설정하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.
3. 목록에서 파라미터 그룹을 선택합니다.
4. 파라미터 그룹 작업에서 편집을 선택합니다.
5. 기본값으로 재설정할 파라미터를 선택합니다. 대화 상자 오른쪽 위의 화살표 키를 사용하여 파라미터를 스크롤할 수 있습니다.

   기본 파라미터 그룹의 값은 변경할 수 없습니다.
6. Reset을 선택한 다음 Reset parameters를 선택하여 확인합니다.

AWS CLI

DB 파라미터 그룹의 파라미터를 일부 또는 모두 재설정하려면 필수 옵션 AWS CLI과 함께 `reset-db-parameter-group`--db-parameter-group-name 명령을 사용합니다.

DB 파라미터 그룹의 모든 파라미터를 수정하려면 `--reset-all-parameters` 옵션을 지정합니다. 특정 파라미터를 수정하려면 `--parameters` 옵션을 지정합니다.

다음 예제에서는 mydbparametergroup이라는 DB 파라미터 그룹의 모든 파라미터를 기본값으로 수정합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds reset-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --reset-all-parameters
```

Windows의 경우:

```
aws rds reset-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --reset-all-parameters
```

다음 예제에서는 mydbparametergroup이라는 DB 파라미터 그룹에서 `max_connections` 및 `max_allowed_packet` 옵션을 재설정합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds reset-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --parameters "ParameterName=max_connections,ApplyMethod=immediate" \
                 "ParameterName=max_allowed_packet,ApplyMethod=immediate"
```

Windows의 경우:

```
aws rds reset-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --parameters "ParameterName=max_connections,ApplyMethod=immediate" ^
                 "ParameterName=max_allowed_packet,ApplyMethod=immediate"
```

다음과 같은 출력이 생성됩니다.

```
DBParameterGroupName  mydbparametergroup
```

RDS API

DB 파라미터 그룹의 파라미터를 기본값으로 재설정하려면 필수 파라미터 `ResetDBParameterGroup`과 함께 RDS API `DBParameterGroupName` 명령을 사용합니다.

DB 파라미터 그룹의 모든 파라미터를 재설정하려면 `ResetAllParameters` 파라미터를 `true`로 설정합니다. 특정 파라미터를 재설정하려면 `Parameters` 파라미터를 지정합니다.

Amazon RDS에서 DB 파라미터 그룹 복사

생성하는 사용자 지정 DB 파라미터 그룹을 복사할 수 있습니다 파라미터 그룹을 복사하는 것이 편리한 솔루션이 될 수 있습니다. DB 파라미터 그룹을 만들고 이 그룹의 사용자 지정 파라미터 및 값의 대부분을 새 DB 파라미터 그룹에 포함하려는 경우를 예로 들 수 있습니다. AWS Management Console을 사용하여 DB 파라미터 그룹을 복사할 수 있습니다. AWS CLI copy-db-parameter-group 명령 또는 RDS API CopyDBParameterGroup 작업을 사용할 수도 있습니다.

DB 파라미터 그룹을 복사한 후 5분 이상 기다렸다가 해당 DB 파라미터 그룹을 기본 파라미터 그룹으로 사용하는 첫 번째 DB 인스턴스를 생성하십시오. 이렇게 하면 파라미터 그룹이 사용되기 전에

Amazon RDS에서 복사 작업을 완전히 마칠 수 있습니다. 이는 DB 인스턴스의 기본 데이터베이스를 생성할 때 필수적인 파라미터에 특히 중요합니다. 한 가지 예는 `character_set_database` 파라미터로 정의되는 기본 데이터베이스에 대한 문자 집합입니다. Amazon RDS 콘솔의 파라미터 그룹 옵션이나 describe-db-parameters 명령을 사용하여 DB 파라미터 그룹이 생성되었는지 확인하십시오.

> **Note**
>
> 기본 파라미터 그룹은 복사할 수 없습니다. 하지만 기본 파라미터 그룹을 바탕으로 하는 새로운 파라미터 그룹을 만들 수 있습니다.
> DB 파라미터 그룹을 다른 AWS 계정 또는 AWS 리전에 복사할 수 없습니다.

콘솔

DB 파라미터 그룹을 복사하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.
3. 목록에서 복사할 사용자 지정 파라미터 그룹을 선택합니다.
4. 파라미터 그룹 작업에서 복사를 선택합니다.
5. 새로운 DB 파라미터 그룹 식별자에 새로운 파라미터 그룹의 이름을 입력합니다.
6. 설명에 새로운 파라미터 그룹에 대한 설명을 입력합니다.
7. [Copy]를 선택합니다.

AWS CLI

DB 파라미터 그룹을 복사하려면 AWS CLI `copy-db-parameter-group` 명령을 다음 필수 옵션과 함께 사용합니다.

- `--source-db-parameter-group-identifier`
- `--target-db-parameter-group-identifier`
- `--target-db-parameter-group-description`

다음 예에서는 DB 파라미터 그룹 `mygroup2`을 복사하여 `mygroup1`라는 새 DB 파라미터 그룹을 생성합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds copy-db-parameter-group \
    --source-db-parameter-group-identifier mygroup1 \
    --target-db-parameter-group-identifier mygroup2 \
    --target-db-parameter-group-description "DB parameter group 2"
```

Windows의 경우:

```
aws rds copy-db-parameter-group ^
    --source-db-parameter-group-identifier mygroup1 ^
    --target-db-parameter-group-identifier mygroup2 ^
    --target-db-parameter-group-description "DB parameter group 2"
```

RDS API

DB 파라미터 그룹을 복사하려면 RDS API CopyDBParameterGroup 작업을 다음 필수 파라미터와 함께 사용합니다.

- SourceDBParameterGroupIdentifier
- TargetDBParameterGroupIdentifier
- TargetDBParameterGroupDescription

Amazon RDS에 DB 파라미터 그룹 나열

AWS 계정에 대해 생성한 DB 파라미터 그룹을 나열할 수 있습니다.

> **Note**
>
> 특정 DB 엔진과 버전에 대한 DB 인스턴스를 생성할 때 기존 파라미터 템플릿에서 기본 파라미터 그룹이 자동으로 생성됩니다. 이 기본 파라미터 그룹은 기본 파라미터 설정을 포함하며 수정할 수 없습니다. 사용자 지정 파라미터 그룹을 생성할 때 파라미터 설정을 수정할 수 있습니다.

콘솔

AWS 계정에 대한 모든 DB 파라미터 그룹을 나열하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.

   DB 파라미터 그룹이 목록에 나타납니다.

AWS CLI

AWS 계정에 사용할 수 있는 모든 DB 파라미터 그룹을 나열하려면 AWS CLI`describe-db-parameter-groups` 명령을 사용합니다.

Example

다음 예에서는 AWS 계정에 사용할 수 있는 모든 DB 파라미터 그룹을 나열합니다.

```
aws rds describe-db-parameter-groups
```

다음과 같은 응답이 반환됩니다.

```
DBPARAMETERGROUP  default.mysql8.0     mysql8.0  Default parameter group for MySQL8.0
DBPARAMETERGROUP  mydbparametergroup   mysql8.0  My new parameter group
```

다음은 mydbparamgroup1 파라미터 그룹을 설명하는 예제입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-parameter-groups \
    --db-parameter-group-name mydbparamgroup1
```

Windows의 경우:

```
aws rds describe-db-parameter-groups ^
    --db-parameter-group-name mydbparamgroup1
```

다음과 같은 응답이 반환됩니다.

```
DBPARAMETERGROUP  mydbparametergroup1  mysql8.0  My new parameter group
```

RDS API

AWS 계정에 사용할 수 있는 모든 DB 파라미터 그룹을 나열하려면 RDS API `DescribeDBParameterGroups` 작업을 사용합니다.

Amazon RDS에서 DB 파라미터 그룹의 파라미터 값 보기

DB 파라미터 그룹의 모든 파라미터와 해당 값 목록을 가져올 수 있습니다.

콘솔

DB 파라미터 그룹의 파라미터 값을 보려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.

   DB 파라미터 그룹이 목록에 나타납니다.
3. 파라미터 그룹의 이름을 선택하여 파라미터 목록을 봅니다.

AWS CLI

DB 파라미터 그룹의 파라미터 값을 보려면 AWS CLI`describe-db-parameters` 명령을 다음 필수 파라미터와 함께 사용합니다.

- `--db-parameter-group-name`

Example

다음 예에서는 mydbparametergroup이라는 DB 파라미터 그룹에 대한 파라미터와 파라미터 값을 나열합니다.

```
aws rds describe-db-parameters --db-parameter-group-name mydbparametergroup
```

다음과 같은 응답이 반환됩니다.

```
DBPARAMETER  Parameter Name           Parameter Value  Source           Data Type
 Apply Type  Is Modifiable
```

```
DBPARAMETER  allow-suspicious-udfs                    engine-default   boolean
 static      false
DBPARAMETER  auto_increment_increment                 engine-default   integer
 dynamic     true
DBPARAMETER  auto_increment_offset                    engine-default   integer
 dynamic     true
DBPARAMETER  binlog_cache_size         32768          system           integer
 dynamic     true
DBPARAMETER  socket                    /tmp/mysql.sock  system         string
 static      false
```

RDS API

DB 파라미터 그룹의 파라미터 값을 보려면 RDS API `DescribeDBParameters` 명령을 다음 필수 파라미터와 함께 사용하세요.

- `DBParameterGroupName`

Amazon RDS에서 DB 파라미터 그룹 삭제

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB 파라미터 그룹을 삭제할 수 있습니다. 파라미터 그룹은 DB 인스턴스와 연결되지 않은 경우에만 삭제할 수 있습니다.

콘솔

DB 파라미터 그룹을 삭제하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.

   DB 파라미터 그룹이 목록에 나타납니다.
3. 삭제할 파라미터 그룹의 이름을 선택합니다.
4. 작업을 선택한 후 삭제를 선택합니다.
5. 파라미터 그룹 이름을 검토한 다음 삭제를 선택합니다.

AWS CLI

DB 파라미터 그룹을 삭제하려면 AWS CLI `delete-db-parameter-group` 명령을 다음 필수 파라미터와 함께 사용합니다.

- `--db-parameter-group-name`

Example

다음 예제에서는 mydbparametergroup이라는 DB 파라미터 그룹을 삭제합니다.

```
aws rds delete-db-parameter-group --db-parameter-group-name mydbparametergroup
```

RDS API

DB 파라미터 그룹을 삭제하려면 RDS API `DeleteDBParameterGroup` 명령을 다음 필수 파라미터와 함께 사용합니다.

- `DBParameterGroupName`

다중 AZ DB 클러스터용 DB 클러스터 파라미터 그룹 작업

다중 AZ DB 클러스터는 DB 클러스터 파라미터 그룹을 사용합니다. 다음 섹션에서는 DB 클러스터 파라미터 그룹 구성 및 관리에 대해 설명합니다.

주제

- DB 클러스터 파라미터 그룹 생성
- DB 클러스터 파라미터 그룹의 파라미터 수정
- DB 클러스터 파라미터 그룹의 파라미터 재설정
- DB 클러스터 파라미터 그룹 복사
- DB 클러스터 파라미터 그룹 나열
- DB 클러스터 파라미터 그룹의 파라미터 값 보기
- DB 클러스터 파라미터 그룹 삭제

DB 클러스터 파라미터 그룹 생성

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 새 DB 클러스터 파라미터 그룹을 생성할 수 있습니다.

DB 클러스터 파라미터 그룹을 생성한 후 해당 DB 클러스터 파라미터 그룹을 사용하는 DB 클러스터를 생성하려면 5분 이상 기다려야 합니다. 이렇게 하면 새 DB 클러스터에서 사용하기 전에 Amazon RDS

에서 파라미터 그룹을 완전히 생성할 수 있습니다. Amazon RDS 콘솔의 파라미터 그룹 페이지 또는 describe-db-cluster-parameters 명령을 사용하여 DB 클러스터 파라미터 그룹이 생성되었는지 확인할 수 있습니다.

DB 클러스터 파라미터 그룹 이름에는 다음과 같은 제한이 적용됩니다.

- 이름은 1~255자의 문자, 숫자 또는 하이픈으로 구성되어야 합니다.

  기본 파라미터 그룹 이름에는 마침표(예: `default.mysql5.7`)가 포함될 수 있습니다. 하지만 사용자 지정 파라미터 그룹 이름에는 마침표를 포함할 수 없습니다.
- 첫 번째 자리는 문자여야 합니다.
- 이름은 하이픈으로 끝나거나 2개 연속 하이픈을 포함할 수 없습니다.

콘솔

DB 클러스터 파라미터 그룹을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.
3. [Create parameter group]을 선택합니다.
4. 파라미터 그룹 이름에 새 DB 클러스터 파라미터 그룹의 이름을 입력합니다.
5. 설명에 새 DB 클러스터 파라미터 그룹에 대한 설명을 입력합니다.
6. 엔진 유형에서 데이터베이스 엔진을 선택합니다.
7. 파라미터 그룹 패밀리에서 DB 파라미터 그룹 패밀리를 선택합니다.
8. Create를 선택합니다.

AWS CLI

DB 클러스터 파라미터 그룹을 생성하려면 AWS CLI `create-db-cluster-parameter-group` 명령을 사용합니다.

다음 예에서는 "My new cluster parameter group(내 새로운 클러스터 파라미터 그룹)"이라는 설명과 함께 mydbclusterparametergroup이라는 RDS for MySQL 버전 8.0용 DB 클러스터 파라미터 그룹을 생성합니다.

다음 필수 파라미터를 포함합니다.

- `--db-cluster-parameter-group-name`
- `--db-parameter-group-family`
- `--description`

사용 가능한 모든 파라미터 그룹 패밀리를 나열하려면 다음 명령을 사용합니다.

```
aws rds describe-db-engine-versions --query "DBEngineVersions[].DBParameterGroupFamily"
```

> **Note**
>
> 출력에 중복이 있습니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbclusterparametergroup \
    --db-parameter-group-family mysql8.0 \
    --description "My new cluster parameter group"
```

Windows의 경우:

```
aws rds create-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbclusterparametergroup ^
    --db-parameter-group-family mysql8.0 ^
    --description "My new cluster parameter group"
```

다음과 비슷한 출력이 생성됩니다.

```
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "mydbclusterparametergroup",
        "DBParameterGroupFamily": "mysql8.0",
        "Description": "My new cluster parameter group",
        "DBClusterParameterGroupArn": "arn:aws:rds:us-east-1:123456789012:cluster-
pg:mydbclusterparametergroup2"
    }
```

```
}
```

RDS API

DB 클러스터 파라미터 그룹을 생성하려면 RDS API `CreateDBClusterParameterGroup` 작업을 사용합니다.

다음 필수 파라미터를 포함합니다.

- `DBClusterParameterGroupName`
- `DBParameterGroupFamily`
- `Description`

DB 클러스터 파라미터 그룹의 파라미터 수정

고객이 생성한 DB 클러스터 파라미터 그룹에서 파라미터 값을 수정할 수 있습니다. 기본 DB 클러스터 파라미터 그룹에서는 파라미터 값을 변경할 수 없습니다. 고객이 생성한 DB 클러스터 파라미터 그룹의 파라미터를 변경하면 DB 클러스터 파라미터 그룹과 연결된 모든 DB 클러스터에 해당 변경 내용이 적용됩니다.

콘솔

DB 클러스터 파라미터 그룹을 수정하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.
3. 목록에서 수정할 파라미터 그룹을 선택합니다.
4. 파라미터 그룹 작업에서 편집을 선택합니다.
5. 수정하려는 파라미터의 값을 변경합니다. 대화 상자 오른쪽 위의 화살표 키를 사용하여 파라미터를 스크롤할 수 있습니다.

   기본 파라미터 그룹의 값은 변경할 수 없습니다.
6. [Save changes]를 선택합니다.
7. 클러스터의 재부팅하여 변경 사항을 적용합니다.

   클러스터를 재부팅하지 않으면 장애 조치 작업이 평소보다 오래 걸릴 수 있습니다.

AWS CLI

DB 클러스터 파라미터 그룹을 수정하려면 AWS CLI modify-db-cluster-parameter-group 명령을 다음 필수 파라미터와 함께 사용합니다.

- `--db-cluster-parameter-group-name`
- `--parameters`

다음 예에서는 mydbclusterparametergroup이라는 DB 클러스터 파라미터 그룹에서 `server_audit_logging` 및 `server_audit_logs_upload` 값을 수정합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbclusterparametergroup \
    --parameters
 "ParameterName=server_audit_logging,ParameterValue=1,ApplyMethod=immediate" \

 "ParameterName=server_audit_logs_upload,ParameterValue=1,ApplyMethod=immediate"
```

Windows의 경우:

```
aws rds modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbclusterparametergroup ^
    --parameters
 "ParameterName=server_audit_logging,ParameterValue=1,ApplyMethod=immediate" ^

 "ParameterName=server_audit_logs_upload,ParameterValue=1,ApplyMethod=immediate"
```

다음과 같은 출력이 생성됩니다.

```
DBCLUSTERPARAMETERGROUP  mydbclusterparametergroup
```

RDS API

DB 클러스터 파라미터 그룹을 수정하려면 RDS API ModifyDBClusterParameterGroup 명령을 다음 필수 파라미터와 함께 사용합니다.

- `DBClusterParameterGroupName`

- `Parameters`

DB 클러스터 파라미터 그룹의 파라미터 재설정

고객이 생성한 DB 클러스터 파라미터 그룹에서 파라미터를 기본값으로 수정할 수 있습니다. 고객이 생성한 DB 클러스터 파라미터 그룹의 파라미터를 변경하면 DB 클러스터 파라미터 그룹과 연결된 모든 DB 클러스터에 해당 변경 내용이 적용됩니다.

> **Note**
>
> 기본 DB 클러스터 파라미터 그룹에서 파라미터는 항상 기본값으로 설정됩니다.

콘솔

DB 클러스터 파라미터 그룹의 파라미터를 기본값으로 수정

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.
3. 목록에서 파라미터 그룹을 선택합니다.
4. 파라미터 그룹 작업에서 편집을 선택합니다.
5. 기본값으로 재설정할 파라미터를 선택합니다. 대화 상자 오른쪽 위의 화살표 키를 사용하여 파라미터를 스크롤할 수 있습니다.

   기본 파라미터 그룹의 값은 변경할 수 없습니다.
6. Reset을 선택한 다음 Reset parameters를 선택하여 확인합니다.
7. DB 클러스터의 를 재부팅하여 변경 사항을 DB 클러스터의 모든 DB 인스턴스에 적용합니다.

AWS CLI

DB 클러스터 파라미터 그룹의 파라미터를 기본값으로 수정하려면 AWS CLI `reset-db-cluster-parameter-group` 명령을 필수 옵션 `--db-cluster-parameter-group-name`과 함께 사용합니다.

DB 클러스터 파라미터 그룹의 모든 파라미터를 수정하려면 `--reset-all-parameters` 옵션을 지정합니다. 특정 파라미터를 수정하려면 `--parameters` 옵션을 지정합니다.

다음 예제에서는 mydbparametergroup이라는 DB 파라미터 그룹의 모든 파라미터를 기본값으로 수정합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds reset-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbparametergroup \
    --reset-all-parameters
```

Windows의 경우:

```
aws rds reset-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbparametergroup ^
    --reset-all-parameters
```

다음 예제에서는 mydbclusterparametergroup이라는 DB 클러스터 파라미터 그룹에서 `server_audit_logging` 및 `server_audit_logs_upload` 값을 수정합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds reset-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbclusterparametergroup \
    --parameters "ParameterName=server_audit_logging,ApplyMethod=immediate" \
                 "ParameterName=server_audit_logs_upload,ApplyMethod=immediate"
```

Windows의 경우:

```
aws rds reset-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbclusterparametergroup ^
    --parameters
 "ParameterName=server_audit_logging,ParameterValue=1,ApplyMethod=immediate" ^

 "ParameterName=server_audit_logs_upload,ParameterValue=1,ApplyMethod=immediate"
```

다음과 같은 출력이 생성됩니다.

```
DBClusterParameterGroupName  mydbclusterparametergroup
```

RDS API

DB 클러스터 파라미터 그룹의 파라미터를 기본값으로 재설정하려면 다음과 같은 필수 파라미터와 함께 RDS API ResetDBClusterParameterGroup 명령을 사용합니다: DBClusterParameterGroupName.

DB 클러스터 파라미터 그룹의 모든 파라미터를 재설정하려면 ResetAllParameters 파라미터를 true로 설정합니다. 특정 파라미터를 재설정하려면 Parameters 파라미터를 지정합니다.

DB 클러스터 파라미터 그룹 복사

생성하는 사용자 지정 DB 클러스터 파라미터 그룹을 복사할 수 있습니다. DB 클러스터 파라미터 그룹을 이미 생성했으며 해당 그룹의 사용자 지정 파라미터와 값의 대부분을 새 DB 클러스터 파라미터 그룹에 포함하려는 경우 파라미터 그룹을 복사하면 편리합니다. AWS CLIcopy-db-cluster-parameter-group 명령이나 RDS API CopyDBClusterParameterGroup 작업을 사용하여 DB 클러스터 파라미터 그룹을 복사할 수 있습니다.

DB 클러스터 파라미터 그룹을 복사한 후 5분 이상 기다렸다가 해당 DB 클러스터 파라미터 그룹을 기본 파라미터 그룹으로 사용하는 첫 번째 DB 클러스터를 복사합니다. 이렇게 하면 새 DB 클러스터에서 사용하기 전에 Amazon RDS에서 파라미터 그룹을 완전히 복사할 수 있습니다. Amazon RDS 콘솔의 파라미터 그룹 페이지 또는 describe-db-cluster-parameters 명령을 사용하여 DB 클러스터 파라미터 그룹이 생성되었는지 확인할 수 있습니다.

> **Note**
>
> 기본 파라미터 그룹은 복사할 수 없습니다. 하지만 기본 파라미터 그룹을 바탕으로 하는 새로운 파라미터 그룹을 만들 수 있습니다.
> DB 클러스터 파라미터 그룹을 다른 AWS 계정 또는 AWS 리전에 복사할 수 없습니다.

콘솔

DB 클러스터 파라미터 그룹을 복사하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.
3. 목록에서 복사할 사용자 지정 파라미터 그룹을 선택합니다.
4. 파라미터 그룹 작업에서 복사를 선택합니다.

5. 새로운 DB 파라미터 그룹 식별자에 새로운 파라미터 그룹의 이름을 입력합니다.
6. 설명에 새로운 파라미터 그룹에 대한 설명을 입력합니다.
7. [Copy]를 선택합니다.

AWS CLI

DB 클러스터 파라미터 그룹을 복사하려면 AWS CLI copy-db-cluster-parameter-group 명령을 다음 필수 파라미터와 함께 사용합니다.

- `--source-db-cluster-parameter-group-identifier`
- `--target-db-cluster-parameter-group-identifier`
- `--target-db-cluster-parameter-group-description`

다음 예에서는 DB 클러스터 파라미터 그룹 mygroup2을 복사하여 mygroup1라는 새 DB 클러스터 파라미터 그룹을 생성합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds copy-db-cluster-parameter-group \
    --source-db-cluster-parameter-group-identifier mygroup1 \
    --target-db-cluster-parameter-group-identifier mygroup2 \
    --target-db-cluster-parameter-group-description "DB parameter group 2"
```

Windows의 경우:

```
aws rds copy-db-cluster-parameter-group ^
    --source-db-cluster-parameter-group-identifier mygroup1 ^
    --target-db-cluster-parameter-group-identifier mygroup2 ^
    --target-db-cluster-parameter-group-description "DB parameter group 2"
```

RDS API

DB 클러스터 파라미터 그룹을 복사하려면 RDS API CopyDBClusterParameterGroup 작업을 다음 필수 파라미터와 함께 사용합니다.

- `SourceDBClusterParameterGroupIdentifier`

- `TargetDBClusterParameterGroupIdentifier`
- `TargetDBClusterParameterGroupDescription`

DB 클러스터 파라미터 그룹 나열

AWS 계정에 대해 생성한 DB 클러스터 파라미터 그룹을 나열할 수 있습니다.

> **Note**
>
> 특정 DB 엔진과 버전에 대한 DB 클러스터를 생성할 때 기존 파라미터 템플릿에서 기본 파라미터 그룹이 자동으로 생성됩니다. 이 기본 파라미터 그룹은 기본 파라미터 설정을 포함하며 수정할 수 없습니다. 사용자 지정 파라미터 그룹을 생성할 때 파라미터 설정을 수정할 수 있습니다.

콘솔

AWS 계정에 대한 모든 DB 클러스터 파라미터 그룹을 나열하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.

   DB 클러스터 파라미터 그룹은 목록에서 DB 클러스터 파라미터 그룹유형에 나타납니다.

AWS CLI

AWS 계정에 사용할 수 있는 모든 DB 클러스터 파라미터 그룹을 나열하려면 AWS CLI `describe-db-cluster-parameter-groups` 명령을 사용합니다.

Example

다음 예에서는 AWS 계정에 사용할 수 있는 모든 DB 클러스터 파라미터 그룹을 나열합니다.

```
aws rds describe-db-cluster-parameter-groups
```

다음은 mydbclusterparametergroup 파라미터 그룹을 설명하는 예제입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-cluster-parameter-groups \
    --db-cluster-parameter-group-name mydbclusterparametergroup
```

Windows의 경우:

```
aws rds describe-db-cluster-parameter-groups ^
    --db-cluster-parameter-group-name mydbclusterparametergroup
```

다음과 같은 응답이 반환됩니다.

```
{
    "DBClusterParameterGroups": [
        {
            "DBClusterParameterGroupName": "mydbclusterparametergroup2",
            "DBParameterGroupFamily": "mysql8.0",
            "Description": "My new cluster parameter group",
            "DBClusterParameterGroupArn": "arn:aws:rds:us-east-1:123456789012:cluster-
pg:mydbclusterparametergroup"
        }
    ]
}
```

RDS API

AWS 계정에 사용할 수 있는 모든 DB 클러스터 파라미터 그룹을 나열하려면 RDS API [DescribeDBClusterParameterGroups](#) 작업을 사용합니다.

DB 클러스터 파라미터 그룹의 파라미터 값 보기

DB 클러스터 파라미터 그룹의 모든 파라미터와 해당 값 목록을 가져올 수 있습니다.

콘솔

DB 클러스터 파라미터 그룹의 파라미터 값을 보려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.

   DB 클러스터 파라미터 그룹은 목록에서 DB 클러스터 파라미터 그룹유형에 나타납니다.

3. DB 클러스터 파라미터 그룹의 이름을 선택하여 파라미터 목록을 봅니다.

AWS CLI

DB 클러스터 파라미터 그룹의 파라미터 값을 보려면 AWS CLI [describe-db-cluster-parameters](#) 명령을 다음 필수 파라미터와 함께 사용합니다.

- `--db-cluster-parameter-group-name`

Example

다음 예에서는 mydbclusterparametergroup이라는 DB 클러스터 파라미터 그룹에 대한 파라미터와 파라미터 값을 JSON 형식으로 나열합니다.

다음과 같은 응답이 반환됩니다.

```
aws rds describe-db-cluster-parameters --db-cluster-parameter-group-
name mydbclusterparametergroup
```

```
{
    "Parameters": [
        {
            "ParameterName": "activate_all_roles_on_login",
            "ParameterValue": "0",
            "Description": "Automatically set all granted roles as active after the
 user has authenticated successfully.",
            "Source": "engine-default",
            "ApplyType": "dynamic",
            "DataType": "boolean",
            "AllowedValues": "0,1",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot",
            "SupportedEngineModes": [
                "provisioned"
            ]
        },
        {
            "ParameterName": "allow-suspicious-udfs",
            "Description": "Controls whether user-defined functions that have only an
 xxx symbol for the main function can be loaded",
            "Source": "engine-default",
```

```
            "ApplyType": "static",
            "DataType": "boolean",
            "AllowedValues": "0,1",
            "IsModifiable": false,
            "ApplyMethod": "pending-reboot",
            "SupportedEngineModes": [
                "provisioned"
            ]
        },
...
```

RDS API

DB 클러스터 파라미터 그룹의 파라미터 값을 보려면 RDS API DescribeDBClusterParameters 명령을 다음 필수 파라미터와 함께 사용하세요.

- DBClusterParameterGroupName

경우에 따라 파라미터에 허용된 값이 표시되지 않습니다. 이러한 파라미터는 항상 소스가 데이터베이스 엔진 기본값인 파라미터입니다.

이러한 파라미터의 값을 보려면 다음 SQL 문을 실행하면 됩니다.

- MySQL:

```
-- Show the value of a particular parameter
mysql$ SHOW VARIABLES LIKE '%parameter_name%';

-- Show the values of all parameters
mysql$ SHOW VARIABLES;
```

- PostgreSQL:

```
-- Show the value of a particular parameter
postgresql=> SHOW parameter_name;

-- Show the values of all parameters
postgresql=> SHOW ALL;
```

DB 클러스터 파라미터 그룹 삭제

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB 클러스터 파라미터 그룹을 삭제할 수 있습니다. DB 클러스터 파라미터 그룹 파라미터 그룹은 DB 클러스터와 연결되지 않은 경우에만 삭제할 수 있습니다.

콘솔

파라미터 그룹을 삭제하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.

   파라미터 그룹이 목록에 나타납니다.
3. 삭제할 DB 클러스터 파라미터 그룹의 이름을 선택합니다.
4. 작업을 선택한 후 삭제를 선택합니다.
5. 파라미터 그룹 이름을 검토한 다음 삭제를 선택합니다.

AWS CLI

DB 클러스터 파라미터 그룹을 삭제하려면 AWS CLI `delete-db-cluster-parameter-group` 명령을 다음 필수 파라미터와 함께 사용합니다.

- `--db-parameter-group-name`

Example

다음 예제에서는 mydbparametergroup이라는 DB 클러스터 파라미터 그룹을 삭제합니다.

```
aws rds delete-db-cluster-parameter-group --db-parameter-group-name mydbparametergroup
```

RDS API

DB 클러스터 파라미터 그룹을 삭제하려면 RDS API `DeleteDBClusterParameterGroup` 명령을 다음 필수 파라미터와 함께 사용합니다.

- `DBParameterGroupName`

DB 파라미터 그룹 비교

AWS Management Console을 사용하여 두 DB 파라미터 그룹 간 차이를 확인할 수 있습니다.

지정된 파라미터 그룹은 둘 다 DB 파라미터 그룹이거나 둘 다 DB 클러스터 파라미터 그룹이어야 합니다. DB 엔진과 버전이 같더라도 마찬가지입니다. 예를 들어 `aurora-mysql8.0`(Aurora MySQL 버전 3) DB 파라미터 그룹과 `aurora-mysql8.0` DB 클러스터 파라미터 그룹을 비교할 수 없습니다.

버전이 다르더라도 Aurora MySQL 파라미터 그룹과 RDS for MySQL DB 파라미터 그룹을 비교할 수 있지만 Aurora PostgreSQL DB 파라미터 그룹과 RDS for PostgreSQL DB 파라미터 그룹을 비교할 수는 없습니다.

두 DB 파라미터 그룹을 비교하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.
3. 목록에서 비교하려는 두 파라미터 그룹을 선택합니다.

   > **Note**
   >
   > 기본 파라미터 그룹을 사용자 지정 파라미터 그룹과 비교하려면 먼저 기본 탭에서 기본 파라미터 그룹을 선택한 다음 사용자 지정 탭에서 사용자 지정 파라미터 그룹을 선택합니다.

4. 작업 에서 비교를 선택합니다.

DB 파라미터 지정

DB 파라미터 유형은 다음과 같습니다.

- Integer
- 불
- String
- Long
- Double
- Timestamp

- 다른 정의된 데이터 형식의 객체
- 정수, 부울, 문자열, long, double, 타임스탬프 또는 객체 형식의 값 배열

표현식, 수식 및 함수를 사용하여 정수 및 부울 파라미터를 지정할 수도 있습니다.

Oracle 엔진의 경우 `DBInstanceClassHugePagesDefault` 수식 변수를 사용하여 부울 DB 파라미터를 지정할 수 있습니다. DB 파라미터 수식 변수 단원을 참조하십시오.

PostgreSQL 엔진의 경우 표현식을 사용하여 부울 DB 파라미터를 지정할 수 있습니다. 부울 DB 파라미터 표현식 단원을 참조하십시오.

목차

- DB 파라미터 수식
  - DB 파라미터 수식 변수
  - DB 파라미터 수식 연산자
- DB 파라미터 함수
- 부울 DB 파라미터 표현식
- DB 파라미터 로그 표현식
- DB 파라미터 값 예제

DB 파라미터 수식

DB 파라미터 수식은 정수 값 또는 부울 값으로 확인되는 표현식입니다. 이 표현식은 중괄호({})로 묶여 있습니다. DB 파라미터 값에 수식을 사용하거나 DB 파라미터 함수의 인수로 수식을 사용할 수 있습니다.

구문

```
{FormulaVariable}
{FormulaVariable*Integer}
{FormulaVariable*Integer/Integer}
{FormulaVariable/Integer}
```

DB 파라미터 수식 변수

각 수식 변수는 정수 또는 부울 값을 반환합니다. 변수 이름은 대소문자를 구분합니다.

AllocatedStorage

데이터 볼륨의 크기(바이트)를 나타내는 정수를 반환합니다.

DBInstanceClassHugePagesDefault

부울 값 반환 현재 Oracle 엔진에 대해서만 지원됩니다.

자세한 내용은 RDS for Oracle 인스턴스에 HugePages 활성화 섹션을 참조하세요.

DBInstanceClassMemory

데이터베이스 프로세스에 사용할 수 있는 메모리의 바이트 수에 대한 정수를 반환합니다. 이 숫자는 DB 인스턴스 클래스에 대한 총 메모리 양에서 시작하여 내부적으로 계산됩니다. 이 계산에서 인스턴스를 관리하는 RDS 프로세스와 운영 체제용으로 예약된 메모리가 차감됩니다. 따라서 이 숫자는 DB 인스턴스 클래스의 인스턴스 클래스 표에 표시된 메모리 수치보다 항상 다소 낮습니다. 정확한 값은 여러 요인의 조합에 따라 달라집니다. 여기에는 인스턴스 클래스, DB 엔진이 포함되며 RDS 인스턴스에 적용되는지 아니면 Aurora 클러스터의 일부인 인스턴스에 적용되는지에 따라 값이 달라집니다.

DBInstanceVCPU

Amazon RDS가 인스턴스를 관리하는 데 사용하는 vCPU(가상 중앙 처리 유닛)의 수를 나타내는 정수를 반환합니다.

EndPointPort

DB 인스턴스에 연결하는 데 사용되는 포트를 나타내는 정수를 반환합니다.

TrueIfReplica

DB 인스턴스가 읽기 전용 복제본이면 `1`, 아니면 `0`을 반환합니다. MySQL의 `read_only` 파라미터에 대한 기본값입니다.

DB 파라미터 수식 연산자

DB 파라미터 수식은 나눗셈과 곱셈의 두 가지 연산자를 지원합니다.

나눗셈 연산자: /

나뉨수를 나눗수로 나누어 정수 몫을 반환합니다. 몫의 소수는 잘리며 반올림되지 않습니다.

구문

```
dividend / divisor
```

나뉨수 및 나눗수 인수는 정수 식이어야 합니다.

곱셈 연산자: *

표현식을 곱하여 해당 표현식의 곱을 반환합니다. 표현식 소수는 잘리며 반올림되지 않습니다.

구문

```
expression * expression
```

두 식 모두 정수여야 합니다.

DB 파라미터 함수

DB 파라미터 함수의 인수를 정수 또는 수식으로 지정합니다. 함수마다 인수가 하나 이상 있어야 합니다. 여러 인수를 쉼표로 구분된 목록으로 지정합니다. 목록에 argument1, argument3과 같은 빈 멤버를 넣을 수 없습니다. 함수 이름은 대소문자를 구분하지 않습니다.

IF

인수를 반환합니다.

현재 Oracle 엔진에 대해서만 지원하고 지원하는 첫 인수는 `{DBInstanceClassHugePagesDefault}`입니다. 자세한 내용은 RDS for Oracle 인스턴스에 HugePages 활성화 섹션을 참조하세요.

구문

```
IF(argument1, argument2, argument3)
```

첫 번째 인수가 true이면 두 번째 인수를 반환합니다. 그렇지 않으면 세 번째 인수를 반환합니다.

GREATEST

정수 또는 파라미터 수식 목록에서 가장 큰 값을 반환합니다.

구문

```
GREATEST(argument1, argument2,...argumentn)
```

정수를 반환합니다.

LEAST

정수 또는 파라미터 수식 목록에서 가장 작은 값을 반환합니다.

구문

```
LEAST(argument1, argument2,...argumentn)
```

정수를 반환합니다.

SUM

지정된 정수나 파라미터 수식의 값을 더합니다.

구문

```
SUM(argument1, argument2,...argumentn)
```

정수를 반환합니다.

부울 DB 파라미터 표현식

부울 DB 파라미터 표현식은 1 또는 0의 부울 값으로 해석됩니다. 이 표현식은 따옴표로 묶여 있습니다.

> **Note**
> 부울 DB 파라미터 표현식은 PostgreSQL 엔진에 대해서만 지원됩니다.

구문

```
"expression operator expression"
```

두 표현식 모두 정수로 해석되어야 합니다. 표현식은 다음 형식일 수 있습니다.

- 정수 상수
- DB 파라미터 수식
- DB 파라미터 함수

- DB 파라미터 변수

부울 DB 파라미터 표현식은 다음과 같은 부등식 연산자를 지원합니다.

보다 큼 연산자: **>**

구문

```
"expression > expression"
```

보다 작음 연산자: **<**

구문

```
"expression < expression"
```

크거나 같음 연산자: **>=**, **=>**

구문

```
"expression >= expression"
"expression => expression"
```

작거나 같음 연산자: **<=**, **=<**

구문

```
"expression <= expression"
"expression =< expression"
```

Example 부울 DB 파라미터 표현식 사용

다음 부울 DB 파라미터 표현식 예시에서는 파라미터 수식의 결과를 정수와 비교합니다. PostgreSQL DB 인스턴스의 부울 DB 파라미터 `wal_compression`을 수정하기 위한 것입니다. 이 파라미터 표현식은 vCPU 수를 값 2와 비교합니다. vCPU 수가 2보다 크면 `wal_compression` DB 파라미터가 true로 설정됩니다.

```
aws rds modify-db-parameter-group --db-parameter-group-name group-name \
--parameters "ParameterName=wal_compression,ParameterValue=\"{DBInstanceVCPU} > 2\" "
```

DB 파라미터 로그 표현식

정수 DB 파라미터 값을 로그 표현식으로 설정할 수 있습니다. 이 표현식은 중괄호({})로 묶여 있습니다. 예:

```
{log(DBInstanceClassMemory/8187281418)*1000}
```

log 함수는 로그 밑 2를 나타냅니다. 또한 이 예제에서는 DBInstanceClassMemory 수식 변수를 사용합니다. DB 파라미터 수식 변수 단원을 참조하십시오.

> **Note**
>
> 현재, MySQL innodb_log_file_size 파라미터를 정수 이외의 값으로 지정할 수 없습니다.

DB 파라미터 값 예제

이 예에서는 DB 파라미터 값에 수식, 함수 및 표현식을 사용하는 방법을 보여 줍니다.

> **Warning**
>
> DB 파라미터 그룹에 파라미터를 잘못 설정하면 의도하지 않은 부작용이 있을 수 있습니다. 이러한 부작용에는 성능 저하, 시스템 불안정 등이 포함될 수 있습니다. 데이터베이스 파라미터를 수정할 때 주의하고 DB 파라미터 그룹을 수정하기 전에 데이터를 백업하세요. 파라미터 그룹 변경 내용을 프로덕션 DB 인스턴스에 적용하기 전에 특정 시점으로 복원을 사용하여 생성한 테스트 DB 인스턴스에 적용해 봐야 합니다.

Example DB 파라미터 함수 GREATEST 사용

Oracle 프로세스 파리미터에 GREATEST 함수를 지정할 수 있습니다. DBInstanceClassMemory를 9,868,951로 나눈 값과 80 중 더 큰 값으로 사용자 프로세스 수를 설정하려면 이 함수를 사용합니다.

```
GREATEST({DBInstanceClassMemory/9868951},80)
```

Example DB 파라미터 함수 LEAST 사용

MySQL LEAST 파라미터 값에 max_binlog_cache_size 함수를 지정할 수 있습니다. 트랜잭션이 MySQL 인스턴스에서 사용할 수 있는 최대 캐시 값을 1MB와 DBInstanceClass/256 중 더 작은 값으로 설정하려면 이 함수를 사용합니다.

```
LEAST({DBInstanceClassMemory/256},10485760)
```

Amazon RDS DB 인스턴스 설정을 사용하여 Amazon ElastiCache 캐시 생성

ElastiCache는 유연한 실시간 사용 사례를 지원하는 마이크로초 단위의 읽기 및 쓰기 지연 시간을 제공하는 완전 관리형 인 메모리 캐싱 서비스입니다. ElastiCache는 애플리케이션 및 데이터베이스 성능을 가속화하는 데 도움이 될 수 있습니다. ElastiCache를 게임 리더보드, 스트리밍, 데이터 분석 같은 데이터 내구성이 필요하지 않은 사용 사례의 기본 데이터 스토어로 사용할 수 있습니다. ElastiCache는 분산된 컴퓨팅 환경의 배포 및 관리와 관련된 복잡성을 해소하는 데 도움을 줍니다. 자세한 내용은 일반적인 ElastiCache 사용 사례 및 ElastiCache가 도움이 되는 방법(Memcached) 및 일반적인 ElastiCache 사용 사례 및 ElastiCache가 도움이 되는 방법(Redis OSS)을 참조하세요. Amazon RDS 콘솔을 사용하여 ElastiCache 캐시를 생성할 수 있습니다.

Amazon ElastiCache는 2가지 형식으로 운영할 수 있습니다. 서버리스 캐시로 시작하거나 자체 캐시 클러스터를 설계하도록 선택할 수 있습니다. 자체 캐시 클러스터를 설계하기로 선택한 경우 ElastiCache는 Redis OSS 엔진 및 Memcached 엔진 모두와 함께 작동합니다. 어떤 엔진을 사용해야 할지 잘 모르겠는 경우 Memcached와 Redis OSS 비교를 참조하세요. Amazon ElastiCache에 대한 자세한 내용은 Amazon ElastiCache 사용 설명서를 참조하세요.

주제

- RDS DB 인스턴스 설정을 사용하는 ElastiCache 캐시 생성 개요
- RDS DB 인스턴스의 설정을 사용하여 ElastiCache 캐시 생성

RDS DB 인스턴스 설정을 사용하는 ElastiCache 캐시 생성 개요

새로 생성한 또는 기존의 RDS DB 인스턴스와 동일한 구성 설정을 사용하여 Amazon RDS에서 ElastiCache 캐시를 생성할 수 있습니다.

ElastiCache 캐시를 DB 인스턴스와 연결하는 몇 가지 사용 사례:

- RDS에서만 실행하는 대신 ElastiCache를 RDS와 함께 사용하면 비용을 절감하고 성능을 개선할 수 있습니다.

  예를 들어, RDS for MySQL만 사용할 때와 비교하여 RDS for MySQL과 함께 ElastiCache를 사용하면 비용을 최대 55% 절감하고 최대 80배 더 빠른 읽기 성능을 얻을 수 있습니다.
- 데이터 내구성이 필요하지 않은 애플리케이션의 프라이머리 데이터 스토어로 ElastiCache 캐시를 사용할 수 있습니다. Redis OSS 또는 Memcached를 사용하는 애플리케이션은 거의 수정하지 않고 ElastiCache를 사용할 수 있습니다.

RDS에서 ElastiCache 캐시를 생성하는 경우, ElastiCache 캐시는 연결된 RDS DB 인스턴스에서 다음 설정을 상속합니다.

- ElastiCache 연결 설정
- ElastiCache 보안 설정

요구 사항에 따라 캐시 구성 설정을 설정할 수 있습니다.

애플리케이션에 ElastiCache 설정

ElastiCache 캐시를 활용하도록 애플리케이션을 설정해야 합니다. 또한 요구 사항에 따라 캐싱 전략을 사용하도록 애플리케이션을 설정하여 캐시 성능을 최적화하고 개선할 수 있습니다.

- ElastiCache 캐시에 액세스하여 시작하려면 ElastiCache 시작하기(Redis OSS) 및 ElastiCache 시작하기(Memcached)를 참조하세요.
- 캐싱 전략에 대한 자세한 내용은 캐싱 전략 및 모범 사례(Memcached) 및 캐싱 전략 및 모범 사례(Redis OSS)를 참조하세요.
- ElastiCache(Redis OSS) 클러스터의 고가용성에 대한 자세한 내용은 복제 그룹을 사용한 고가용성을 참조하세요.
- 백업 스토리지, 리전 내 또는 리전 간 데이터 전송 또는 AWS Outposts 사용과 관련된 비용이 발생할 수 있습니다 요금에 대한 자세한 내용은 Amazon ElastiCache 요금을 참조하세요.

RDS DB 인스턴스의 설정을 사용하여 ElastiCache 캐시 생성

DB 인스턴스에서 상속되는 설정을 사용하여 RDS DB 인스턴스용 ElastiCache 캐시를 생성할 수 있습니다.

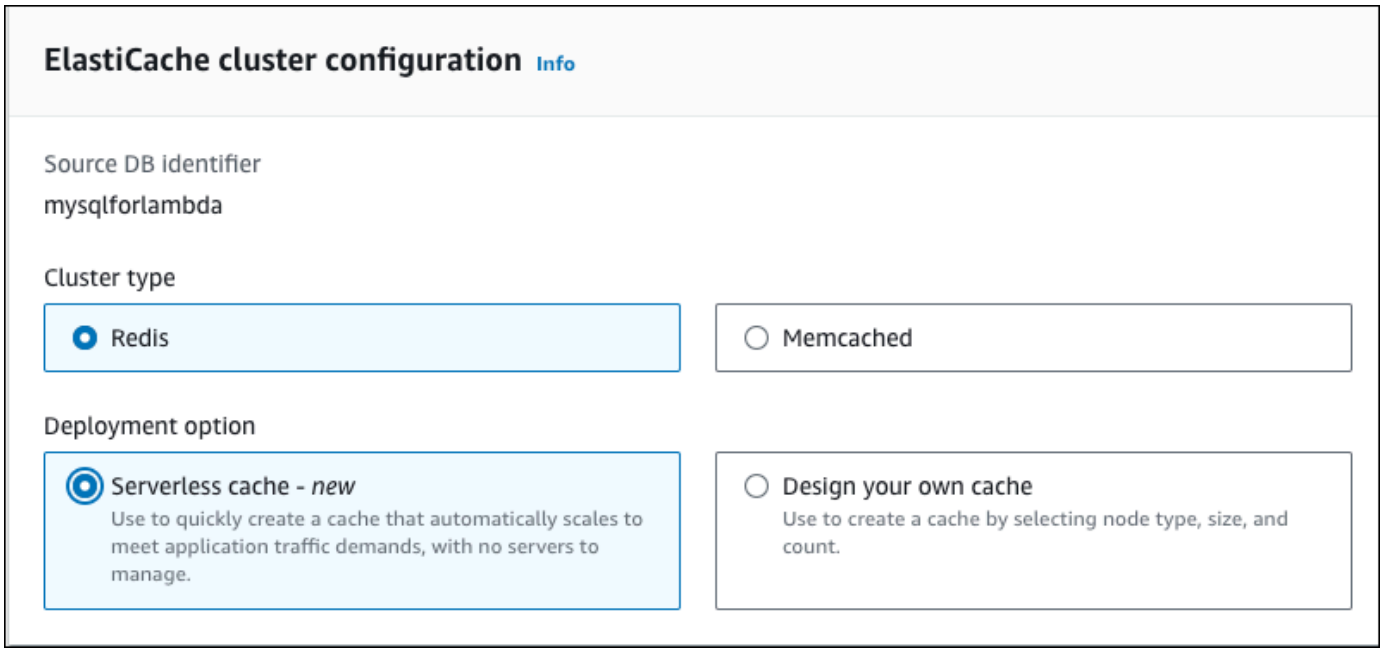
DB 인스턴스의 설정을 사용하여 ElastiCache 캐시 생성

1. DB 인스턴스를 생성하려면 Amazon RDS DB 인스턴스 생성의 지침을 따릅니다.
2. RDS DB 인스턴스를 생성하면 콘솔에 제안된 추가 기능 창이 표시됩니다. Create an ElastiCache cluster from RDS using your DB settings(DB 설정을 사용하여 RDS에서 ElastiCache 클러스터 생성)를 선택합니다.

   기존 데이터베이스의 경우 데이터베이스 페이지에서 필요한 DB 인스턴스를 선택합니다. 작업 드롭다운 메뉴에서 ElastiCache 클러스터 생성을 선택하여 기존 RDS DB 인스턴스와 동일한 설정을 가진 RDS에 ElastiCache 캐시를 생성합니다.

ElastiCache 구성 섹션의 소스 DB 식별자에 ElastiCache 캐시가 설정을 상속받는 DB 인스턴스가 표시됩니다.

3. Redis OSS 클러스터를 생성할지 또는 Memcached 클러스터를 생성할지 선택하세요. 자세한 내용은 Memcached와 Redis OSS 비교를 참조하세요.



4. 그런 다음 서버리스 캐시를 만들지 아니면 자체 캐시를 설계할지 선택합니다. 자세한 내용은 배포 옵션 간 선택을 참조하세요.

   서버리스 캐시를 선택하는 경우:

   a. 캐시 설정의 이름 및 설명에 값을 입력합니다.

   b. 기본 설정 보기에서 기본 설정을 그대로 두고 캐시와 DB 인스턴스 간의 연결을 설정합니다.

   c. 기본 설정 사용자 지정을 선택하여 기본 설정을 편집할 수도 있습니다. ElastiCache 연결 설정, ElastiCache 보안 설정, 최대 사용량 제한을 선택합니다.

5. 자체 캐시 설계를 선택하는 경우:

   a. Redis OSS 클러스터를 선택한 경우 클러스터 모드를 활성화 또는 비활성화 상태로 유지할지 선택합니다. 자세한 내용은 복제: Redis OSS(클러스터 모드 비활성화됨)와 Redis OSS(클러스터 모드 활성화됨) 비교를 참조하세요.

   b. 이름과 설명, 엔진 버전을 입력합니다.

      엔진 버전의 경우 권장 기본값은 최신 엔진 버전입니다. 요구 사항에 가장 적합한 ElastiCache 캐시용 엔진 버전을 선택할 수도 있습니다.

c. 노드 유형 옵션에서 노드 유형을 선택합니다. 자세한 내용은 노드 관리를 참조하세요.

클러스터 모드를 활성화됨으로 설정한 상태에서 Redis OSS 클러스터를 생성하려고 선택한 경우, 샤드 수 옵션에 샤드 수(파티션/노드 그룹 수)를 입력하세요.

복제본 개수에 각 샤드의 복제본 수를 입력합니다.

> **Note**
>
> 선택한 노드 유형, 샤드 수, 복제본 수는 모두 캐시 성능 및 리소스 비용에 영향을 미칩니다. 이러한 설정이 데이터베이스 요구 사항과 일치하는지 확인하세요. 요금에 대한 자세한 정보는 Amazon ElastiCache 요금을 참조하세요.

d. ElastiCache 연결 설정 및 ElastiCache 보안 설정을 선택합니다. 기본 설정을 유지하거나 요구 사항에 따라 이러한 설정을 사용자 지정할 수 있습니다.

6. ElastiCache 캐시의 기본 설정 및 상속된 설정을 확인합니다. 일부 설정은 생성 후에 변경할 수 없습니다.

> **Note**
>
> RDS는 최소 기간 요구 사항인 60분을 충족하도록 ElastiCache 캐시의 백업 기간을 조정할 수 있습니다. 소스 데이터베이스의 백업 기간은 동일하게 유지됩니다.

7. 준비가 되면 ElastiCache 캐시 생성을 선택합니다.

콘솔에는 ElastiCache 캐시 생성을 위한 확인 배너가 표시됩니다. 배너의 링크를 따라 ElastiCache 콘솔로 이동하면 캐시 세부 정보를 볼 수 있습니다. ElastiCache 콘솔에는 새로 생성된 ElastiCache 캐시가 표시됩니다.

AWS Database Migration Service를 사용하여 EC2 데이터베이스를 Amazon RDS로 자동 마이그레이션

RDS 콘솔을 사용하여 EC2 데이터베이스를 RDS로 마이그레이션할 수 있습니다. RDS는 AWS Database Migration Service(AWS DMS)를 사용하여 소스 EC2 데이터베이스를 마이그레이션합니다. AWS DMS를 사용하면 관계형 데이터베이스를 AWS 클라우드로 마이그레이션할 수 있습니다. AWS Database Migration Service에 대한 자세한 내용은 AWS Database Migration Service 사용 설명서의 AWS Database Migration Service이란 무엇인가요?를 참조하세요.

마이그레이션을 시작하려면 데이터를 마이그레이션하기 위해 상응하는 RDS DB 인스턴스를 만들어야 합니다. 대상 데이터베이스를 만든 후 EC2 데이터베이스를 가져올 수 있습니다. 1TiB보다 작은 소스 데이터베이스의 경우 이 마이그레이션 작업은 데이터를 RDS로 마이그레이션하는 데 필요한 시간과 리소스를 줄입니다.

개요

RDS 콘솔을 사용하면 EC2 데이터베이스를 상응하는 RDS 데이터베이스로 마이그레이션할 수 있습니다. 콘솔에서 마이그레이션을 활성화하려면 RDS 데이터베이스를 만들어야 합니다.

다음 데이터베이스 엔진의 EC2 데이터베이스를 마이그레이션할 수 있습니다.

- MySQL
- MariaDB
- PostgreSQL

마이그레이션 프로세스는 다음과 같은 단계로 구성됩니다.

- RDS에서 상응하는 데이터베이스를 만듭니다. 데이터베이스가 상응하려면 데이터베이스 엔진과 호환되는 엔진 버전이 동일해야 합니다. 또한 동일한 VPC에 있어야 합니다. 데이터베이스 만들기에 대한 지침은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
- 데이터베이스의 복제 유형을 선택합니다.
  - 전체 로드 마이그레이션 - RDS가 전체 소스 데이터베이스를 대상 데이터베이스에 복사하며 필요한 경우 대상에 새 테이블을 만듭니다.

> **Note**
>
> 이 옵션을 사용하면 RDS 데이터베이스가 중단됩니다.

- 전체 로드 및 변경 데이터 캡처(CDC) 마이그레이션 - 전체 로드 마이그레이션과 마찬가지로 이 옵션을 사용하면 RDS가 전체 소스 데이터베이스를 대상으로 복사합니다. 그러나 전체 로드 마이그레이션 후 RDS는 소스에서 캡처된 변경 사항을 대상 데이터베이스에 적용합니다. 변경 데이터 캡처는 데이터베이스 엔진의 기본 API를 사용하여 데이터베이스 로그에 대한 변경 사항을 수집합니다.

  > **Note**
  >
  > 이 옵션을 사용하면 RDS 데이터베이스가 중단됩니다.

- 변경 데이터 캡처(CDC) - 마이그레이션 과정 내내 대상 데이터베이스를 계속 사용할 수 있도록 하려면 이 옵션을 사용합니다. RDS는 소스 데이터베이스의 진행 중인 변경 사항을 대상 데이터베이스로 마이그레이션합니다.

- RDS는 마이그레이션을 용이하게 하는 데 필요한 네트워킹 리소스를 만듭니다. RDS가 필요한 리소스를 만들면 만들어진 리소스에 대해 사용자에게 알리고, 사용자는 데이터 전송을 시작할 수 있습니다.

  마이그레이션을 완료하는 데 필요한 시간은 복제 유형과 소스 데이터베이스 크기에 따라 다릅니다.

사전 조건

MySQL 및 MariaDB

소스 데이터베이스로 MySQL 또는 MariaDB 데이터베이스를 사용하기 전에 다음 사전 조건을 충족하는지 확인해야 합니다. 이러한 사전 요구 사항은 AWS 관리형 소스에 적용됩니다.

Replication Admin 역할이 할당된 AWS DMS용 계정이 있어야 합니다. 역할에는 다음 권한이 있어야 합니다.

- REPLICATION CLIENT - 이 권한은 변경 데이터 캡처(CDC) 작업에만 필요합니다. 즉, 전체 로드 전용 작업에서는 이 권한이 필요하지 않습니다.
- REPLICATION SLAVE - 이 권한은 변경 데이터 캡처(CDC) 작업에만 필요합니다. 즉, 전체 로드 전용 작업에서는 이 권한이 필요하지 않습니다.

또한, AWS DMS 사용자에게는 복제를 위해 지정된 원본 테이블에 대해 SELECT 권한이 있어야 합니다.

MySQL별 사전 마이그레이션 평가를 사용하는 경우 다음 권한을 부여합니다.

```
grant select on mysql.user to <dms_user>;
grant select on mysql.db to <dms_user>;
grant select on mysql.tables_priv to <dms_user>;
grant select on mysql.role_edges to <dms_user>  #only for MySQL version 8.0.11 and
 higher
```

PostgreSQL

AWS 관리형 PostgreSQL 소스 데이터베이스에서 데이터를 마이그레이션하기 전에 다음을 수행합니다.

- PostgreSQL DB 인스턴스에 필요한 최소 권한이 있는 AWS 사용자 계정을 AWS DMS의 PostgreSQL 소스 엔드포인트 사용자 계정으로 사용하는 것이 좋습니다. 마스터 계정을 사용하는 것은 권장되지 않습니다. 계정에는 `rds_superuser` 역할과 `rds_replication` 역할이 있어야 합니다. `rds_replication` 역할은 논리적 슬롯을 관리하고 논리적 슬롯을 사용하여 데이터를 스트리밍할 수 있는 권한을 부여합니다.

> **Note**
>
> 일부 AWS DMS 트랜잭션은 DMS 엔진에서 다시 사용하기 전까지 얼마 동안 유휴 상태입니다. PostgreSQL 버전 9.6 이상에서 `idle_in_transaction_session_timeout` 파라미터를 사용하면 유휴 트랜잭션이 시간 초과되어 실패하도록 할 수 있습니다.

제한 사항

자동 마이그레이션 프로세스에는 다음 제한 사항이 적용됩니다.

- 소스 데이터베이스 마이그레이션을 시작하려면 대상 데이터베이스 상태가 사용 가능이어야 합니다.
- MySQL 소스 데이터베이스에서 마이그레이션할 때 RDS 계정에 복제 관리자 역할이 있어야 합니다. 또한 해당 역할에 적절한 권한이 적용되어 있어야 합니다.
- EC2 인스턴스와 대상 데이터베이스는 동일한 VPC에 있어야 합니다.
- EC2 데이터베이스에서 데이터 마이그레이션 작업을 사용할 때는 EC2 데이터베이스를 다음 대상 데이터베이스로 마이그레이션할 수 없습니다.
  - 클러스터의 멤버인 데이터베이스
  - Oracle, SQL Server 및 Db2 데이터베이스

- MySQL 버전이 5.7 미만인 데이터베이스
- PostgreSQL 버전이 10.4 미만인 데이터베이스
- MariaDB 버전이 10.2 미만인 데이터베이스

동종 마이그레이션을 위한 IAM 리소스 만들기

RDS는 AWS DMS를 사용하여 데이터를 마이그레이션합니다. 데이터베이스에 액세스하고 데이터를 마이그레이션하기 위해 AWS DMS는 동종 데이터 마이그레이션을 위한 서버리스 환경을 생성합니다. 이 환경에서 AWS DMS는 VPC 피어링, 라우팅 테이블, 보안 그룹 및 기타 AWS 리소스에 대한 액세스가 필요합니다. 또한 AWS DMS는 Amazon CloudWatch에 각 데이터 마이그레이션에 대한 로그, 지표 및 진행 상황을 저장합니다. 데이터 마이그레이션 프로젝트를 생성하려면 AWS DMS는 이러한 서비스에 대한 액세스 권한이 필요합니다.

또한 AWS DMS에서는 소스 및 대상 연결에 대한 데이터베이스 연결을 인증하기 위해 사용자 자격 증명 세트를 나타내는 보안 암호에 대한 액세스 권한이 필요합니다.

> **Note**
>
> EC2 인스턴스에서 데이터 마이그레이션 작업을 사용하면 RDS 콘솔을 사용하여 이러한 IAM 리소스를 생성할 수 있습니다. 콘솔에서 생성한 IAM 리소스를 사용하는 경우 이 단계를 건너뜁니다.

이 프로세스에는 다음과 같은 IAM 리소스가 필요합니다.

주제

- 동종 데이터 마이그레이션을 위한 IAM 정책 만들기
- 동종 데이터 마이그레이션을 위한 IAM 역할 만들기
- 보안 암호 액세스 정책 및 역할 만들기
- AWS DMS에 대한 IAM 역할을 만들어 Amazon VPC 관리

동종 데이터 마이그레이션을 위한 IAM 정책 만들기

이 단계에서는 Amazon EC2 및 CloudWatch 리소스에 대한 액세스 권한을 AWS DMS에 부여하는 IAM 정책을 생성합니다. 그런 다음, IAM 역할을 생성하고 이 정책을 연결합니다.

동종 데이터 마이그레이션을 위한 IAM 정책을 만드는 방법

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 정책을 선택합니다.
3. 정책 생성을 선택합니다.
4. 정책 생성 페이지에서 JSON 탭을 선택합니다.
5. 다음 JSON을 편집기에 붙여넣습니다.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeVpcPeeringConnections",
                "ec2:DescribeVpcs",
                "ec2:DescribePrefixLists",
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "servicequotas:GetServiceQuota"
            ],
            "Resource": "arn:aws:servicequotas:*:*:vpc/L-0EA8095F"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:dms-data-migration-*"
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:dms-data-migration-*:log-
stream:dms-data-migration-*"
        },
        {
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateRoute",
                "ec2:DeleteRoute"
            ],
            "Resource": "arn:aws:ec2:*:*:route-table/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:security-group-rule/*",
                "arn:aws:ec2:*:*:route-table/*",
                "arn:aws:ec2:*:*:vpc-peering-connection/*",
                "arn:aws:ec2:*:*:vpc/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group-rule/*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:RevokeSecurityGroupIngress"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AcceptVpcPeeringConnection",
                "ec2:ModifyVpcPeeringConnectionOptions"
            ],
            "Resource": "arn:aws:ec2:*:*:vpc-peering-connection/*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:AcceptVpcPeeringConnection",
            "Resource": "arn:aws:ec2:*:*:vpc/*"
        }
    ]
}
```

6. 다음: 태그와 다음: 검토를 선택합니다.
7. 이름*의 경우, **HomogeneousDataMigrationsPolicy**를 입력한 후 정책 생성을 선택합니다.

동종 데이터 마이그레이션을 위한 IAM 역할 만들기

이 단계에서는 AWS Secrets Manager, Amazon EC2 및 CloudWatch에 대한 액세스 권한을 제공하는 IAM 역할을 만듭니다.

데이터 마이그레이션을 위한 IAM 역할을 만드는 방법

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할을 선택합니다.
3. 역할 생성을 선택합니다.
4. 신뢰할 수 있는 엔터티 선택 페이지의 신뢰할 수 있는 엔터티 유형 아래에서 AWS 서비스를 선택합니다. 기타 AWS 서비스 사용 사례에서 DMS를 선택합니다.

5. DMS 확인란을 선택하고 다음을 선택합니다.
6. 권한 추가 페이지에서 이전에 만든 HomogeneousDataMigrationsPolicy를 선택합니다. 다음을 선택합니다.
7. 이름 지정, 검토 및 생성 페이지에서 역할 이름에 **HomogeneousDataMigrationsRole**을 입력하고 역할 생성을 선택합니다.
8. 역할 페이지에서 역할 이름으로 **HomogeneousDataMigrationsRole**을 입력합니다. HomogeneousDataMigrationsRole을 선택합니다.
9. HomogeneousDataMigrationsRole 페이지에서 신뢰 관계 탭을 선택합니다. 신뢰 정책 편집을 선택합니다.
10. 신뢰 정책 편집 페이지에서 기존 텍스트를 대체하여 다음 JSON을 편집기에 붙여넣습니다.

    JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "dms-data-migrations.amazonaws.com",
                    "dms.your_region.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

    이전 예제에서 *your_region*을 AWS 리전의 이름으로 바꾸십시오.

    위의 리소스 기반 정책은 고객 관리형 HomogeneousDataMigrationsPolicy 정책에 따라 작업을 수행할 수 있는 권한을 AWS DMS 서비스 위탁자에게 부여합니다.
11. 정책 업데이트를 선택합니다.

보안 암호 액세스 정책 및 역할 만들기

아래 프로시저에 따라 DMS가 소스 및 대상 데이터베이스의 사용자 자격 증명에 액세스할 수 있도록 허용하는 보안 암호 액세스 정책 및 역할을 만듭니다.

Amazon RDS가 적절한 보안 암호에 액세스하기 위해 AWS Secrets Manager에 액세스하도록 허용하는 보안 암호 액세스 정책 및 역할을 만드는 방법

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 AWS Identity and Access Management(IAM) 콘솔을 엽니다.
2. 정책을 선택한 후 정책 생성을 선택합니다.
3. JSON을 선택하고 다음 정책을 입력하여 보안 암호에 대한 액세스 및 암호 해독을 활성화합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": secret_arn,
        },
        {
             "Effect": "Allow",
             "Action": [
                        "kms:Decrypt",
                        "kms:DescribeKey"
                      ],
             "Resource": kms_key_arn,
        }
     ]
}
```

여기서 *secret_arn*은 보안 암호의 ARN(상황에 따라 SecretsManagerSecretId에서 가져올 수 있음)이고, *kms_key_arn*은 다음 예시와 같이 보안 암호를 암호화하는 데 사용하는 AWS KMS 키의 ARN입니다.

JSON

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-2:123456789012:secret:MySQLTestSecret-qeHamH"
        },
        {
             "Effect": "Allow",
             "Action": [
                        "kms:Decrypt",
                        "kms:DescribeKey"
                      ],
             "Resource": "arn:aws:kms:us-
east-2:123456789012:key/761138dc-0542-4e58-947f-4a3a8458d0fd"
        }
     ]
}
```

> **Note**
>
> AWS Secrets Manager에서 생성한 기본 암호화 키를 사용하는 경우, *kms_key_arn*에 대한 AWS KMS 권한을 지정하지 않아도 됩니다.
> 정책에서 두 보안 암호에 대한 액세스를 제공하도록 하려면 다른 *secret_arn*에 대해 추가 JSON 리소스 객체를 지정하면 됩니다.

4. 기억하기 쉬운 이름과 설명(선택 사항)이 있는 정책을 검토하고 생성합니다.
5. 역할을 선택한 다음, 역할 생성을 선택합니다.
6. 신뢰할 수 있는 엔터티 유형으로서 AWS 서비스를 선택합니다.
7. 서비스 목록에서 DMS를 신뢰할 수 있는 서비스로 선택하고 다음: 권한을 선택합니다.
8. 4단계에서 생성한 정책을 찾아 연결한 다음, 계속 진행하여 태그를 추가하고 역할을 검토합니다. 이때 해당 역할의 신뢰 관계를 편집하여 Amazon RDS 리전 서비스 위탁자를 신뢰할 수 있는 엔터티로 사용합니다. 이 보안 주체의 형식은 다음과 같습니다.

```
dms.region-name.amazonaws.com
```

여기서 *region-name*은 리전의 이름입니다(예: `us-east-1`). 따라서 이 리전의 Amazon RDS 리전 서비스 위탁자는 다음과 같습니다.

```
dms.us-east-1.amazonaws.com
dms-data-migrations.amazonaws.com
```

AWS DMS에 대한 IAM 역할을 만들어 Amazon VPC 관리

리소스의 VPC 설정을 관리하려면 AWS DMS에 대한 IAM 역할을 만들어야 합니다. 마이그레이션에 성공하려면 이 역할을 사용할 수 있어야 합니다.

데이터베이스 마이그레이션을 위해 **dms-vpc-role** 만들기

<result>

이렇게 하면 DMS가 마이그레이션을 위한 VPC 설정을 관리하는 데 사용할 역할이 만들어집니다.
</result>

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 콘솔의 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.
3. 신뢰할 수 있는 엔터티 선택 옵션에서 AWS 서비스 옵션을 선택합니다.

   사용 사례에서 DMS를 선택합니다.
4. 권한 추가 단계에서 AmazonDMSVPCManagementRole을 선택하고 다음을 선택합니다.
5. 이름 지정, 검토 및 생성 페이지에서 역할 이름을 dms-vpc-role로 설정하고 역할 생성을 선택합니다.

EC2 데이터베이스에 대한 데이터 마이그레이션 설정

EC2 소스 데이터베이스에서 데이터 마이그레이션을 시작하려면 상승하는 RDS 데이터베이스를 만들어야 합니다. 데이터베이스 만들기에 대한 지침은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.

대상 데이터베이스를 만든 후 다음 단계에 따라 데이터 마이그레이션을 설정합니다.

데이터 마이그레이션 프로젝트 설정

1. RDS 콘솔의 데이터베이스 페이지에서 대상 데이터베이스를 선택합니다.

2. 작업 드롭다운을 선택하고 EC2 데이터베이스에서 데이터 마이그레이션 옵션을 선택합니다. 지원되는 대상 데이터베이스는 제한 사항 섹션을 참조하세요.
3. 소스 EC2 데이터베이스 선택 섹션에서 다음을 수행합니다.

    1. 엔진 유형을 확인하고 소스 데이터베이스와 동일한지 확인합니다.

        또한 엔진 버전이 호환되는지 확인합니다.
    2. EC2 인스턴스에서 소스 데이터베이스가 있는 EC2 인스턴스를 선택합니다.
    3. 포트에 소스 데이터베이스가 트래픽을 허용하는 포트를 입력합니다.
    4. 기존 보안 암호가 없는 경우 보안 암호에서 새 보안 암호를 만들고 사용하기를 선택합니다. 소스 데이터베이스의 사용자 이름과 암호를 입력합니다. 또한 보안 암호를 암호화할 KMS 키를 선택합니다.

        기존 보안 암호가 있는 경우 기존 보안 암호 사용을 선택한 후 드롭다운 메뉴에서 보안 암호를 선택합니다.
    5. 보안 암호의 IAM 역할의 경우 기존 IAM 역할이 있으면 기존 IAM 역할 사용을 선택하고 드롭다운에서 이전 단계의 보안 암호 ID에 액세스할 수 있는 IAM 역할을 선택합니다.

        기존 IAM 역할이 없는 경우 새 IAM 역할을 만들고 사용하기를 선택합니다. IAM 역할 이름에 새 역할의 이름을 입력합니다. 아래 링크에서 이 역할과 연결된 권한을 볼 수 있습니다.
4. 대상 RDS 데이터베이스 보기 섹션에서 다음을 수행합니다.

    1. 섹션 상단에서 대상 데이터베이스의 설정을 확인합니다.
    2. 대상 데이터베이스 자격 증명을 보유한 기존 보안 암호가 없는 경우 보안 암호에서 새 보안 암호를 만들고 사용하기를 선택합니다.

        기존 보안 암호가 있는 경우 드롭다운에서 보안 암호를 선택합니다.
    3. 보안 암호의 IAM 역할의 경우 이전 단계에서 보안 암호에 액세스할 수 있는 IAM 역할을 선택합니다. 기존 IAM 역할이 없는 경우 새 IAM 역할을 만들 수도 있습니다.

        드롭다운에 IAM 역할이 채워지지 않으면 IAM 역할 ARN을 `arn:aws:iam:`*`account_id`*`:role/`*`roleName`* 형식으로 지정합니다.
5. 데이터 마이그레이션 구성 섹션에서 다음을 수행합니다.

    1. 전체 로드, 전체 로드 및 변경 데이터 캡처(CDC) 또는 변경 데이터 캡처(CDC) 중에서 선택하여 데이터 마이그레이션 유형을 선택합니다. 이러한 옵션에 대한 자세한 내용은 개요 섹션을 참조하세요.

마이그레이션이 시작된 후에는 마이그레이션 유형을 수정할 수 없습니다.

2. 데이터 마이그레이션의 IAM 역할의 경우 기존 IAM 역할이 있는 경우 기존 IAM 역할 사용을 선택하고 드롭다운에서 마이그레이션에 필요한 리소스를 만들 수 있는 권한을 DMS에 부여하는 IAM 역할을 선택합니다. 기존 IAM 역할이 없는 경우 새 IAM 역할을 만들고 사용하기를 선택합니다.

6. 마이그레이션 설정 보기 탭에 데이터 마이그레이션을 성공적으로 설정하는 데 필요한 설정이 표시되는지 확인합니다.
7. 마이그레이션을 선택하여 마이그레이션 설정을 완료합니다.

이 단계를 완료한 후 콘솔의 진행률 배너에서 세부 정보 보기를 선택하여 데이터 마이그레이션을 위해 설정되고 있는 리소스를 확인할 수 있습니다. 필요한 리소스가 설정되면 마이그레이션이 자동으로 시작됩니다. 만드는 경우

여러 데이터베이스를 대상 데이터베이스로 마이그레이션하려면 새 EC2 데이터베이스에 대한 세부 정보를 사용하여 이 프로세스를 다시 시작합니다.

데이터 마이그레이션 관리

RDS 콘솔에서 EC2 데이터베이스에서 데이터 마이그레이션 작업을 사용하면 RDS가 마이그레이션을 자동으로 시작합니다.

AWS DMS 콘솔을 사용하여 마이그레이션 리소스를 만든 경우 마이그레이션 프로세스를 시작할 수 있습니다.

데이터 마이그레이션 시작

다음 단계에 따라 데이터 마이그레이션을 시작합니다.

데이터 마이그레이션 시작

1. RDS 콘솔의 데이터베이스 페이지에서 대상 데이터베이스를 선택합니다.
2. 데이터베이스 세부 정보 페이지에서 데이터 마이그레이션 탭을 선택합니다.
3. 데이터 마이그레이션 탭에서 연결된 데이터 마이그레이션에는 사용 가능한 데이터 마이그레이션이 나열됩니다.

   RDS 콘솔을 사용하여 설정된 마이그레이션은 필요한 리소스가 설정되면 자동으로 시작됩니다.

   DMS 콘솔을 사용하여 설정된 마이그레이션은 준비 완료로 설정됩니다.

이러한 마이그레이션을 시작하려면 작업 드롭다운을 선택하고 시작을 선택합니다.

4. 그러면 EC2 데이터베이스의 데이터 마이그레이션이 시작됩니다.

데이터 마이그레이션 중지

복제 유형이 전체 로드인 데이터 마이그레이션의 경우 마이그레이션을 중지하면 프로세스가 중지되고 재개할 수 없습니다. 중지하고 나면 마이그레이션을 다시 시작해야 합니다.

복제 유형이 변경 데이터 캡처(CDC) 또는 전체 로드 및 CDC로 설정된 마이그레이션의 경우 연속 복제 프로세스를 중지하고 나중에 프로세스를 재개할 수 있습니다.

데이터 마이그레이션 중지

1. RDS 콘솔의 데이터베이스 페이지에서 대상 데이터베이스를 선택합니다.
2. 데이터베이스 세부 정보 페이지에서 데이터 마이그레이션 탭을 선택합니다.
3. 데이터 마이그레이션 탭에서 연결된 데이터 마이그레이션에는 진행 중인 데이터 마이그레이션이 나열됩니다.

   마이그레이션을 중지하려면 데이터 마이그레이션을 선택하고 작업 드롭다운에서 중지를 선택합니다.
4. 그러면 EC2 데이터베이스의 데이터 마이그레이션이 중지됩니다.

데이터 마이그레이션 재개

복제 유형이 전체 로드 및 변경 데이터 캡처(CDC) 또는 변경 데이터 캡처(CDC) 마이그레이션인 데이터 마이그레이션의 경우 마지막 중지 지점에서 CDC 프로세스를 재개할 수 있습니다.

데이터 마이그레이션 재개

1. RDS 콘솔의 데이터베이스 페이지에서 대상 데이터베이스를 선택합니다.
2. 데이터베이스 세부 정보 페이지에서 데이터 마이그레이션 탭을 선택합니다.
3. 데이터 마이그레이션 탭에서 연결된 데이터 마이그레이션에는 중지된 데이터 마이그레이션이 나열됩니다.

   마이그레이션을 재개하려면 데이터 마이그레이션을 선택하고 작업 드롭다운에서 처리 재개를 선택합니다.

4. 그러면 EC2 데이터베이스의 데이터 마이그레이션이 재개됩니다.

데이터 마이그레이션 삭제

연결된 데이터 마이그레이션을 삭제하려면 다음 지침을 사용합니다.

데이터 마이그레이션 삭제

1. RDS 콘솔의 데이터베이스 페이지에서 대상 데이터베이스를 선택합니다.
2. 데이터베이스 세부 정보 페이지에서 데이터 마이그레이션 탭을 선택합니다.
3. 마이그레이션을 삭제하려면 데이터 마이그레이션을 선택하고 작업 드롭다운에서 삭제를 선택합니다.
4. 이렇게 하면 데이터 마이그레이션이 삭제됩니다.

진행 중이던 데이터 마이그레이션을 삭제해도 대상 데이터베이스에 이미 로드된 데이터는 영향을 받지 않습니다.

데이터 마이그레이션 다시 시작

CDC 시작 지점에서 연결된 데이터 마이그레이션을 다시 시작하려면 다음 지침을 사용합니다.

데이터 마이그레이션 다시 시작

1. RDS 콘솔의 데이터베이스 페이지에서 대상 데이터베이스를 선택합니다.
2. 데이터베이스 세부 정보 페이지에서 데이터 마이그레이션 탭을 선택합니다.
3. 마이그레이션을 다시 시작하려면 데이터 마이그레이션을 선택하고 작업 드롭다운에서 다시 시작을 선택합니다.
4. 그러면 CDC 시작 지점에서 데이터 마이그레이션이 다시 시작됩니다.

진행 중이던 데이터 마이그레이션을 다시 시작해도 대상 데이터베이스에 이미 로드된 데이터는 영향을 받지 않습니다.

데이터 마이그레이션 모니터링

데이터 마이그레이션을 시작한 후 상태 및 진행 상황을 모니터링할 수 있습니다. 대규모 데이터집합의 데이터 마이그레이션은 완료하는 데 몇 시간이 걸립니다. 데이터 마이그레이션의 안정성, 가용성 및 고성능을 유지하려면 데이터 마이그레이션 진행 상태를 정기적으로 모니터링하십시오.

데이터 마이그레이션의 상태 및 진행 상황을 확인하려면

1. RDS 콘솔의 데이터베이스 페이지에서 대상 데이터베이스를 선택합니다.
2. 데이터베이스 세부 정보 페이지에서 데이터 마이그레이션 탭을 선택합니다.
3. 연결된 데이터 마이그레이션 섹션에는 데이터 마이그레이션이 나열됩니다. 상태 열을 확인합니다.
4. 실행 중인 데이터 마이그레이션의 경우, 마이그레이션 진행 상황 열에 마이그레이션된 데이터의 비율(%)이 표시됩니다.
5. CloudWatch에서 프로세스를 모니터링하려면 CloudWatch 열에 있는 링크를 사용합니다.

마이그레이션 상태

실행하는 각 데이터 마이그레이션마다 RDS 콘솔에 상태가 표시됩니다. 다음 목록에 상태가 포함되어 있습니다.

- `Ready`: 데이터 마이그레이션을 시작할 준비가 되었습니다.
- `Starting`: RDS가 데이터 마이그레이션을 위한 서버리스 환경을 만드는 중입니다.
- `Load running`: RDS가 전체 로드 마이그레이션을 수행하는 중입니다.
- `Load complete, replication ongoing`: RDS가 전체 로드를 완료했으며 현재 진행 중인 변경 사항을 복제합니다. 이 상태는 전체 로드 및 CDC 유형 마이그레이션에만 적용됩니다.
- `Replication ongoing`: RDS가 진행 중인 변경 사항을 복제하고 있습니다. 이 상태는 CDC 유형 마이그레이션에만 적용됩니다.
- `Stopping`: RDS가 데이터 마이그레이션을 중지하고 있습니다. 이 상태는 작업 메뉴에서 데이터 마이그레이션을 중지하도록 선택할 때 적용됩니다.
- `Stopped`: RDS가 데이터 마이그레이션을 중지했습니다.
- `Failed`: 데이터 마이그레이션이 실패했습니다. 자세한 내용은 로그 파일을 참조하십시오.
- `Restarting`: 데이터 마이그레이션이 CDC 시작점에서 진행 중인 데이터 복제를 다시 시작했습니다.

자습서: 사용자 지정 파라미터 및 새로운 옵션 그룹을 사용하여 MySQL DB 인스턴스 만들기

이 자습서에서는 사용자 지정 파라미터와 새로운 옵션 그룹을 사용하여 MySQL DB 인스턴스 를 만듭니다. 사용자 지정 파라미터 및 옵션 그룹에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 및 옵션 그룹 작업. 의 내용을 참조하시기 바랍니다.

> **Important**
>
> AWS 계정 생성은 무료입니다. 그러나 이 자습서를 완료하면 사용하는 AWS 리소스에 대한 비용이 발생할 수 있습니다. 자습서가 더 이상 필요하지 않은 경우 자습서를 완료한 후에 이러한 리소스를 삭제할 수 있습니다.

사용자 지정 구성 및 설정이 있는 DB 인스턴스를 만들려면 사용자 지정 파라미터 및 새로운 옵션 그룹을 사용하면 됩니다. 사용자 지정 파라미터 및 새로운 옵션 그룹은 여러 데이터베이스를 사용하고 이에 대한 설정을 균일하게 구성하려는 경우 특히 유용합니다.

이 단계를 완료하면 다음을 학습하게 됩니다.

- Amazon RDS를 사용하여 사용자 지정 파라미터 및 새로운 옵션 그룹이 있는 MySQL DB 인스턴스를 만드는 방법입니다.
- MySQL DB 인스턴스에 대한 특정 파라미터 및 옵션을 사용하는 방법입니다.

이 자습서를 완료하려면 다음 작업을 수행하십시오.

1. MySQL 파라미터 `default_password_lifetime`및 `disconnect_on_expired_password`를 사용하여 사용자 지정 파라미터 그룹을 만듭니다.
2. MySQL 옵션 기능(`MariaDB Audit Plugin`)을 사용하여 새 옵션 그룹을 만듭니다. 옵션 그룹을 생성하는 단계는 옵션 그룹 작업 섹션을 참조하세요.
3. 사용자가 만든 사용자 지정 파라미터 그룹 및 새 옵션 그룹으로 MySQL DB 인스턴스를 만듭니다.

주제

- 사전 조건
- Amazon RDS 파라미터 그룹 만들기
- 사용자 지정 파라미터 그룹에서 파라미터 값 수정

- 새 Amazon RDS 옵션 그룹 만들기
- 새 옵션 그룹에 옵션 추가
- 사용자 지정 파라미터 및 새 옵션 그룹을 사용하여 MySQL DB 인스턴스 만들기

사전 조건

이 자습서에서는 AWS 계정 및 관리자 액세스 권한이 있는 사용자가 있어야 합니다. 아직 설정하지 않았다면 다음 섹션에 나와 있는 단계를 완료합니다.

- AWS 계정에 등록
- 관리자 액세스 권한이 있는 사용자 생성

Amazon RDS 파라미터 그룹 만들기

이 자습서에서는 콘솔에서 MySQL DB 인스턴스에 대해 default_password_lifetime 및 disconnect_on_expired_password를 사용하여 사용자 지정 파라미터 그룹을 만드는 방법을 알아봅니다. `default_password_lifetime` 파라미터는 클라이언트 암호가 자동으로 만료되기까지의 일수를 제어합니다. `disconnect_on_expired_password` 파라미터는 암호가 만료될 때 MySQL DB 인스턴스가 클라이언트 연결을 해제하는지 여부를 제어합니다. MySQL DB 인스턴스에 사용할 수 있는 기타 파라미터에 대한 자세한 내용은 MySQL 파라미터 섹션을 참조하시기 바랍니다.

파라미터 그룹을 생성하려면

1. Amazon RDS 콘솔을 열고 파라미터 그룹을 선택합니다.
2. 사용자 지정 파라미터 그룹에서 파라미터 그룹 생성을 선택합니다.
3. 파라미터 그룹 세부 정보를 설정합니다.
    1. 파라미터 그룹의 이름을 입력합니다.
    2. 파라미터 그룹에 대한 설명을 입력합니다.
    3. 엔진 유형에서 MySQL 커뮤니티를 선택합니다.
    4. 파라미터 그룹 패밀리에서 MySQL 8.0 을 선택합니다.
4. 생성(Create)을 선택합니다.

새 파라미터 그룹이 Amazon RDS 콘솔의 파라미터 그룹 페이지에 나타납니다. 다음 단계에서는 파라미터 값을 수정하여 파라미터 그룹을 사용자 지정하는 방법을 보여줍니다.

사용자 지정 파라미터 그룹에서 파라미터 값 수정

다음 단계에 따라 Amazon RDS 파라미터 그룹 만들기에서 만든 파라미터 그룹의 파라미터 값을 수정합니다.

파라미터 그룹에서 파라미터 값을 수정하려면 다음과 같이 합니다.

1. Amazon RDS 콘솔을 열고 파라미터 그룹을 선택합니다.
2. 사용자 지정 파라미터 그룹에서 사용자가 만든 파라미터 그룹의 이름을 선택합니다.
3. 편집을 선택합니다.
4. 파라미터 필터링 검색 상자에서 사용자 지정 파라미터(`default_password_lifetime`)를 검색합니다.
5. 파라미터 옆의 확인란을 선택하고 이 암호 수명 파라미터에 대해 설정할 일 수 값을 입력합니다.
6. 변경 사항 저장을 선택합니다.
7. 파라미터(`disconnect_on_expired_password`)에 대해 동일한 단계를 반복합니다. 이 파라미터를 선택하면 드롭다운 메뉴에서 0 또는 1의 값을 선택하라는 메시지가 표시됩니다. 만료된 암호에서 연결을 해제하려면 1을 선택합니다.

이제 사용자 지정 파라미터 그룹을 MySQL 8.0 DB 인스턴스에 대해 Amazon RDS와 연결할 수 있습니다. 다음으로, DB 인스턴스용 새로운 옵션 그룹을 만듭니다.

새 Amazon RDS 옵션 그룹 만들기

MariaDB 감사 플러그인 옵션을 사용하여 새 옵션 그룹을 만듭니다. 이 플러그인은 보안 및 규정 준수를 위해 서버 활동을 기록합니다. MySQL DB 인스턴스에 사용할 수 있는 다른 사용자 지정 옵션에 대한 자세한 내용은 MySQL DB 인스턴스 옵션 섹션을 참조하시기 바랍니다.

옵션 그룹을 만들려면 다음과 같이 합니다.

1. Amazon RDS 콘솔을 열고 옵션 그룹을 선택합니다.
2. 옵션 그룹에서 그룹 생성을 선택합니다.
3. 옵션 그룹 세부 정보를 설정합니다.
   - 옵션 그룹의 이름을 입력합니다.
   - 옵션 그룹에 대한 설명을 입력합니다.
   - 엔진에서 mysql을 선택합니다.

- 메이저 엔진 버전에서 8.0을 선택합니다.

4. 생성(Create)을 선택합니다.

새 옵션 그룹이 Amazon RDS 콘솔의 옵션 그룹 페이지에 표시됩니다. 다음 단계는 옵션을 옵션 그룹에 추가하는 방법을 보여줍니다.

새 옵션 그룹에 옵션 추가

새 Amazon RDS 옵션 그룹 만들기에서 만든 새 옵션 그룹에 옵션을 추가하려면 다음 단계를 사용합니다.

옵션 그룹에 옵션을 추가하려면 다음과 같이 합니다.

1. Amazon RDS 콘솔을 열고 옵션 그룹을 선택합니다.
2. 옵션 그룹에서 사용자가 만든 옵션 그룹의 이름을 선택합니다.
3. 옵션에서 옵션 추가를 선택합니다.
4. 옵션 그룹 세부 정보를 설정합니다.
   - 옵션 이름에서 MariaDB 감사 플러그인인 MARIADB_AUDIT_PLUGIN 옵션을 선택합니다.
   - 옵션 설정에서 기본 옵션을 모두 선택한 상태로 둡니다.
   - 즉시 적용에서 예를 선택합니다.
5. 생성 옵션을 선택합니다.

이제 모든 연결된 DB 인스턴스에서 새 옵션 그룹을 사용할 수 있습니다. 다음으로 사용자 지정 파라미터와 새 옵션 그룹을 사용하여 MySQL DB 인스턴스를 만듭니다.

사용자 지정 파라미터 및 새 옵션 그룹을 사용하여 MySQL DB 인스턴스 만들기

마지막으로 이전 단계에서 만든 사용자 지정 파라미터 및 새 옵션 그룹으로 MySQL DB 인스턴스를 만듭니다. 다음 단계는 사용자 지정 파라미터 및 새 옵션 그룹을 사용하여 MySQL DB 인스턴스를 만드는 방법을 보여줍니다.

사용자 지정 파라미터 및 새 옵션 그룹을 사용하여 DB 인스턴스를 만들려면 다음과 같이 합니다.

1. Amazon RDS 콘솔을 열고 데이터베이스를 선택합니다.

2. 데이터베이스 생성를 선택합니다.
3. 데이터베이스 생성 방법 선택에서 표준 생성을 선택합니다.
4. 엔진 옵션에서 MySQL 를 선택합니다.
5. 가용성 및 내구성에서 단일 DB 인스턴스를 선택합니다. 이 단계는 사용자 지정 파라미터 또는 새 옵션 그룹을 지원하는 데 필요합니다.
6. 추가 구성을 선택합니다.
   - 초기 데이터베이스 이름에서 DB 인스턴스의 이름을 선택합니다.
   - DB 파라미터 그룹 드롭다운에서 이전에 만든 사용자 지정 파라미터 그룹의 이름을 선택합니다.
   - 옵션 그룹 드롭다운에서 이전에 만든 새 옵션 그룹의 이름을 선택합니다.
7. 이 자습서에서는 다른 DB 설정의 기본 설정을 그대로 두거나 필요에 따라 수정할 수 있습니다.
8. 데이터베이스 생성를 선택합니다.

RDS는 사용자 지정 파라미터 그룹 및 새 옵션 그룹을 사용하여 새로운 MySQL DB 인스턴스를 만듭니다. 이 데이터베이스에 대한 자세한 내용은 Amazon RDS 콘솔의 데이터베이스 페이지를 참조하세요.

이 자습서에서는 사용자 지정 파라미터 및 새 옵션 그룹을 사용하여 맞춤형 설정으로 MySQL DB 인스턴스를 구성했습니다. 새로 만들어진 MySQL DB 인스턴스는 `default_password_lifetime` 파라미터로 사용자 암호 수명을 관리합니다. 또한 이 DB 인스턴스는 `disconnect_on_expired_password` 파라미터를 사용하여 만료된 암호로 연결하는 사용자의 연결을 해제합니다. 또한 `MariaDB Audit Plugin` 옵션을 사용하여 서버 활동을 추적합니다. 데이터베이스를 최적화하기 위해 사용자 지정 파라미터 그룹에 추가 설정을 적용하고 옵션을 추가할 수 있습니다.

사용자 지정된 DB 인스턴스 만들기를 완료한 후에는 원치 않는 비용이 발생하지 않도록 리소스를 삭제해야 합니다. DB 인스턴스를 삭제하려면 DB 인스턴스 삭제 섹션의 설명을 따릅니다.

Amazon RDS DB 인스턴스 관리

아래에서는 Amazon RDS DB 인스턴스의 관리 및 유지 관리에 대한 지침을 확인할 수 있습니다.

주제

- Amazon RDS DB 인스턴스의 일시적 중지
- 이전에 중지된 Amazon RDS DB 인스턴스 시작
- DB 인스턴스 재부팅
- EC2 인스턴스와 DB 인스턴스를 자동으로 연결
- Lambda 함수와 DB 인스턴스 자동 연결
- Amazon RDS DB 인스턴스 수정
- DB 인스턴스 유지 관리
- DB 인스턴스 엔진 버전 업그레이드
- DB 인스턴스 이름 변경
- DB 인스턴스 읽기 전용 복제본 작업
- Amazon RDS 리소스에 태그 지정
- Amazon RDS의 Amazon 리소스 이름(ARN)
- Amazon RDS DB 인스턴스 스토리지 작업
- DB 인스턴스 삭제
- 자습서: 개발에서 프로덕션까지 MySQL DB 인스턴스 환경 관리

Amazon RDS DB 인스턴스의 일시적 중지

임시 테스트 또는 하루 단위의 개발 작업을 위해 DB 인스턴스를 간헐적으로 최대 7일 연속 중지할 수 있습니다. 가장 일반적인 사용 사례는 비용 최적화입니다.

DB 인스턴스를 중지하는 데 걸리는 시간은 인스턴스 클래스, 네트워크 상태, DB 엔진 유형, 데이터베이스 상태 등의 요인에 따라 달라집니다. 프로세스에는 몇 분이 걸릴 수 있습니다. 서비스는 다음 작업을 수행해야 합니다.

- 데이터베이스 엔진 프로세스를 종료합니다.
- RDS 플랫폼 프로세스를 종료합니다.
- DB 인스턴스와 연결된 EBS 스토리지 볼륨을 분리합니다.
- 기본 Amazon EC2 인스턴스를 종료합니다.

> **Warning**
>
> DB 인스턴스를 시작하려면 인스턴스 복구가 필요하며, 몇 분에서 몇 시간까지 걸릴 수 있습니다. 따라서 인스턴스 가용성이 우려되는 경우 프로덕션 인스턴스를 일시적으로 중단하지 않도록 주의하세요. 자세한 내용은 이전에 중지된 Amazon RDS DB 인스턴스 시작 단원을 참조하십시오.

DB 인스턴스를 중지하고 동일한 작업으로 DB 인스턴스를 시작하려면 DB 인스턴스를 재부팅합니다. 자세한 내용은 DB 인스턴스 재부팅 단원을 참조하십시오.

주제

- DB 인스턴스 중지 사용 사례
- 지원되는 DB 엔진, 인스턴스 클래스, 리전
- 다중 AZ 배포에서 DB 인스턴스 중지
- DB 인스턴스 중지 작동 방식
- DB 인스턴스 중지 제한 사항
- 옵션 및 파라미터 그룹 고려 사항
- 퍼블릭 IP 주소 고려 사항
- DB 인스턴스의 일시적 중지: 기본 단계

DB 인스턴스 중지 사용 사례

DB 인스턴스를 중지하고 시작하는 것은 DB 스냅샷을 생성하고 DB 인스턴스를 삭제한 다음 인스턴스에 액세스할 때 스냅샷을 복원하는 것보다 빠릅니다. 인스턴스 중지의 일반적인 사용 사례는 다음과 같습니다.

- 비용 최적화 - 비프로덕션 데이터베이스의 경우 Amazon RDS DB 인스턴스를 일시적으로 중지하여 비용을 절약할 수 있습니다. 인스턴스가 중지된 동안에는 DB 인스턴스 시간에 대해 요금이 부과되지 않습니다.

  > **Important**
  >
  > DB 인스턴스가 중지되어 있는 동안 프로비저닝된 IOPS를 포함하여 프로비저닝된 스토리지에 대해 요금이 부과됩니다. 지정된 보존 기간 내의 수동 스냅샷 및 자동 백업을 포함하여 백업 스토리지에 대한 요금도 부과됩니다. 하지만 DB 인스턴스 시간에 대해서는 요금이 부과되지 않습니다. 자세한 내용은 결제 FAQ 단원을 참조하십시오.

- 일일 개발 - 개발 목적으로 DB 인스턴스를 유지 관리하는 경우, 필요할 때 인스턴스를 시작하고 필요하지 않을 때는 인스턴스를 종료할 수 있습니다.
- 테스트 - 백업 및 복구 절차, 마이그레이션, 애플리케이션 업그레이드 또는 관련 활동을 테스트하려면 임시 DB 인스턴스가 필요할 수 있습니다. 이러한 사용 사례에서는 필요하지 않을 때 DB 인스턴스를 중지할 수 있습니다.
- 교육 - RDS에서 교육을 진행하는 경우 교육 세션 중에 DB 인스턴스를 시작하고 그 후에 종료해야 할 수 있습니다.

지원되는 DB 엔진, 인스턴스 클래스, 리전

다음 DB 엔진을 실행하는 Amazon RDS DB 인스턴스를 시작 및 중지할 수 있습니다.

- Db2
- MariaDB
- Microsoft SQL Server(RDS Custom for SQL Server 포함)
- MySQL
- Oracle
- PostgreSQL

DB 인스턴스 중지 및 시작은 모든 DB 인스턴스 클래스 및 모든 AWS 리전에서 지원됩니다.

다중 AZ 배포에서 DB 인스턴스 중지

다중 AZ 배포에서 DB 인스턴스를 중지하고 시작할 수 있습니다. 다음과 같은 제한 사항이 있습니다.

- 데이터베이스 엔진이 지원하는 경우에만 다중 AZ 배포를 생성할 수 있습니다. 엔진 지원에 대한 자세한 내용은 Amazon RDS에서 다중 AZ DB 클러스터를 지원하는 리전 및 DB 엔진 섹션을 참조하세요.
- RDS for SQL Server는 다중 AZ 배포에서 RDS for SQL Server DB 인스턴스 중지를 지원하지 않습니다. 자세한 내용은 Microsoft SQL Server 다중 AZ 배포의 제한, 참고 및 권장 사항 단원을 참조하십시오.
- DB 인스턴스를 중지하는 데 시간이 많이 걸릴 수 있습니다. 이전 장애 조치 후 백업이 하나 이상 있는 경우 장애 조치 작업으로 재부팅을 수행하여 중지 작업의 속도를 높일 수 있습니다. 자세한 내용은 DB 인스턴스 재부팅 단원을 참조하십시오.

DB 인스턴스 중지 작동 방식

중지 작업은 다음 단계에서 수행됩니다.

1. DB 인스턴스가 정상 종료 프로세스를 시작합니다.

   DB 인스턴스 상태가 `stopping`으로 바뀝니다.
2. 최대 7일 연속으로 인스턴스 실행이 중지됩니다.

   DB 인스턴스 상태가 `stopped`로 바뀝니다.

중지된 DB 인스턴스의 특징

중지된 상태인 DB 인스턴스의 특징은 다음과 같습니다.

- 중지된 DB 인스턴스는 다음을 유지합니다.
  - 인스턴스 ID
  - 도메인 네임 서버(DNS) 엔드포인트
  - Parameter Group
  - 보안 그룹
  - 옵션 그룹

  - Amazon S3 트랜잭션 로그(특정 시점으로 복원에 필요)

  DB 인스턴스를 다시 시작하면 중지했을 때와 동일한 구성으로 시작됩니다.
- 모든 스토리지 볼륨이 DB 인스턴스에 연결된 상태로 유지되고 해당 데이터도 남습니다. RDS는 DB 인스턴스의 RAM에 저장된 모든 데이터를 삭제합니다.

  DB 인스턴스가 중지되어 있는 동안 프로비저닝된 IOPS를 포함하여 프로비저닝된 스토리지에 대해 요금이 부과됩니다. 지정된 보존 기간 내의 수동 스냅샷 및 자동 백업을 포함하여 백업 스토리지에 대한 요금도 부과됩니다.
- RDS는 DB 인스턴스의 옵션 그룹 또는 DB 파라미터 그룹에 대해 보류 중인 작업을 제외하고 예약된 유지 관리 업데이트를 비롯한 보류 중인 작업을 제거합니다.

> **Note**
>
> RDS for PostgreSQL DB 인스턴스가 완전히 종료되지 않는 경우가 있습니다. 이 경우 나중에 인스턴스를 다시 시작할 때 인스턴스가 복구 프로세스를 거치는 것을 확인할 수 있습니다. 이는 데이터베이스 무결성을 보호하기 위한 데이터베이스 엔진의 예상된 동작입니다. 일부 메모리 기반 통계 및 카운터는 기록을 유지하지 않고 앞으로의 운영 워크로드를 캡처하기 위해 재시작 후 다시 초기화됩니다.

중지된 DB 인스턴스의 자동 재시작

연속 7일이 지날 때까지 DB 인스턴스를 수동으로 시작하지 않으면 RDS는 DB 인스턴스를 자동으로 시작합니다. 이렇게 하면 인스턴스에 필요한 유지 보수 업데이트가 지연되지 않습니다. 일정에 따라 인스턴스를 중지하고 시작하는 방법을 알아보려면 How can I use Step Functions to stop an Amazon RDS instance for longer than 7 days?를 참조하세요.

DB 인스턴스 중지 제한 사항

다음은 작업 중지 시 몇 가지 제한 사항입니다.

- 읽기 전용 복제본을 포함한 또는 읽기 전용 복제본인 DB 인스턴스는 중지할 수 없습니다.
- 중지된 DB 인스턴스는 수정할 수 없습니다.
- 중지된 DB 인스턴스와 연결된 옵션 그룹은 삭제할 수 없습니다.
- 중지된 DB 인스턴스에 연결된 DB 파라미터 그룹을 삭제할 수 없습니다.
- 다중 AZ 배포에서는 다음 제한 사항에 유의하세요.

- RDS for SQL Server DB 인스턴스를 중지할 수 없습니다.
- DB 인스턴스를 시작한 후 기본 및 보조 가용 영역이 전환될 수 있습니다.

RDS Custom for SQL Server에는 추가 제한 사항이 적용됩니다. 자세한 내용은 RDS Custom for SQL Server DB 인스턴스 시작 및 중지 단원을 참조하십시오.

옵션 및 파라미터 그룹 고려 사항

옵션 그룹에 연결된 DB 인스턴스가 있는 경우 해당 옵션 그룹에서 지속적 옵션(영구 옵션 포함)을 제거할 수 없습니다. 이 기능은 상태가 `stopping`, `stopped` 또는 `starting`인 DB 인스턴스에서도 마찬가지입니다.

중지된 DB 인스턴스에 연결된 옵션 그룹 또는 DB 파라미터 그룹을 변경할 수 있습니다. 하지만 변경 사항은 다음에 DB 인스턴스를 시작할 때까지 적용되지 않습니다. 변경 사항을 즉시 적용하도록 선택한 경우에는 DB 인스턴스를 시작할 때 변경됩니다. 그렇지 않을 경우 DB 인스턴스가 시작된 후 다음 유지 관리 기간에 변경 사항이 적용됩니다.

퍼블릭 IP 주소 고려 사항

DB 인스턴스를 중지하더라도 DNS 엔드포인트는 유지됩니다. 퍼블릭 IP 주소가 있는 DB 인스턴스를 중지하는 경우 Amazon RDS는 퍼블릭 IP 주소를 릴리스합니다. DB 인스턴스가 다시 시작되면 다른 퍼블릭 IP 주소를 갖습니다.

> **Note**
>
> DB 인스턴스에는 항상 IP 주소가 아니라 DNS 엔드포인트를 사용하여 연결해야 합니다.

DB 인스턴스의 일시적 중지: 기본 단계

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB를 중지할 수 있습니다.

콘솔

DB 인스턴스를 중지하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.

2. 탐색 창에서 데이터베이스를 선택한 후 중지하려는 DB 인스턴스를 선택합니다.
3. Actions(작업)에서 Stop temporarily(일시적으로 중지)를 선택합니다.
4. Stop DB cluster temporarily(DB 클러스터 일시적으로 중지) 창에서 DB 인스턴스가 7일 후에 자동으로 다시 시작된다는 확인을 선택합니다.
5. (선택 사항) Save the DB instance in a snapshot(스냅샷에 DB 인스턴스 저장)을 선택하고 Snapshot name(스냅샷 이름)에 스냅샷 이름을 입력합니다. DB 인스턴스를 중단하기 전에 DB 인스턴스의 스냅샷을 생성하려면 이 옵션을 선택합니다.
6. Stop temporarily(일시적으로 중지)를 선택하여 DB 인스턴스를 중지하거나 Cancel(취소)을 선택하여 작업을 취소합니다.

AWS CLI

AWS CLI를 사용하여 DB 인스턴스를 중지하려면 다음 옵션과 함께 stop-db-instance 명령을 호출하십시오.

- `--db-instance-identifier` – DB 인스턴스의 이름입니다.

Example

```
aws rds stop-db-instance --db-instance-identifier mydbinstance
```

RDS API

Amazon RDS API를 사용하여 DB 인스턴스를 중지하려면 다음 파라미터와 함께 StopDBInstance 작업을 호출하십시오.

- `DBInstanceIdentifier` – DB 인스턴스의 이름입니다.

이전에 중지된 Amazon RDS DB 인스턴스 시작

이 주제에서는 이전에 중지된 Amazon RDS DB 인스턴스를 시작하는 방법을 설명하고, 데이터베이스 사용을 재개하는 데 필요한 단계와 주요 고려 사항을 간략하게 안내합니다.

Amazon RDS DB 인스턴스는 일시적으로 중지하여 비용을 절약할 수 있습니다. 중지한 DB 인스턴스는 재시작하여 다시 사용할 수 있습니다. DB 인스턴스 중지에 대한 자세한 내용은 Amazon RDS DB 인스턴스의 일시적 중지 섹션을 참조하세요.

이전에 중지한 DB 인스턴스를 시작하면 DB 인스턴스는 다음과 같은 정보를 유지합니다.

- 인스턴스 ID
- 도메인 네임 서버(DNS) 엔드포인트
- DB 파라미터 그룹
- VPC 보안 그룹
- DB 옵션 그룹

Amazon RDS는 데이터베이스 인스턴스 및 연결된 스토리지에 대해 1초 단위로 요금을 청구합니다. 인스턴스 시작 시 10분의 최소 요금이 있습니다.

Amazon RDS 서비스는 인스턴스를 시작하기 위해 다음과 같은 작업을 수행합니다.

- 기존 Amazon EC2 인스턴스 프로비저닝
- RDS 프로세스 시작
- 데이터베이스 엔진 프로세스 시작
- EBS 스토리지 볼륨 연결
- 이전에 활성화한 경우 성능 개선 도우미 활성화
- DB 인스턴스 복구(정상 종료 후에도 복구 수행)

DB 인스턴스를 시작하는 데 걸리는 시간은 인스턴스 클래스, 네트워크 상태, DB 엔진 유형, 데이터베이스 크기, 인스턴스가 종료된 시점의 데이터베이스 상태 등의 요인에 따라 달라집니다. 시작 프로세스에는 몇 분에서 몇 시간이 걸릴 수 있습니다. 가용성 계획을 세울 때는 시작 시간의 변동성을 고려하는 것이 좋습니다.

콘솔

DB 인스턴스를 시작하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 후 시작하려는 DB 인스턴스를 선택합니다.
3. Actions(작업)에는 Start(시작)을 선택합니다.

AWS CLI

AWS CLI를 사용하여 DB 인스턴스를 시작하려면 다음 옵션과 함께 start-db-instance 명령을 호출하십시오.

- `--db-instance-identifier` –DB 인스턴스의 이름입니다.

Example

```
aws rds start-db-instance --db-instance-identifier mydbinstance
```

RDS API

Amazon RDS API를 사용하여 DB 인스턴스를 시작하려면 다음 파라미터와 함께 StartDBInstance 작업을 호출하십시오.

- `DBInstanceIdentifier` –DB 인스턴스의 이름입니다.

DB 인스턴스 재부팅

재부팅이라는 단일 작업으로 RDS DB 인스턴스에서 데이터베이스 서비스를 중지하고 시작할 수 있습니다. 데이터베이스를 완전히 재시작하거나 마이그레이션하지 않고도 구성 변경 사항을 적용하거나, 사소한 문제를 해결하거나, 네트워크 문제를 해결하려면 재부팅이 필요할 수 있습니다.

> **Note**
>
> 이 주제는 단일 AZ 또는 다중 AZ DB 인스턴스를 재부팅하는 경우에만 적용됩니다. 다중 AZ DB 클러스터를 재부팅하는 지침은 the section called “다중 AZ DB 클러스터 재부팅” 섹션을 참조하세요.

주제

- DB 인스턴스 재부팅 사용 사례
- DB 인스턴스를 재부팅하는 방법
- 다중 AZ 배포에서 DB 인스턴스를 재부팅하는 방법
- DB 인스턴스 재부팅 시 고려 사항
- DB 인스턴스 재부팅 시 사전 조건
- DB 인스턴스 재부팅: 기본 단계

DB 인스턴스 재부팅 사용 사례

일반적으로 유지 관리 이유로 DB 인스턴스를 재부팅하여 변경 사항을 적용합니다. 일반적인 사용 사례는 다음과 같습니다.

- 새 DB 파라미터 그룹 연결 – 새 DB 파라미터 그룹을 DB 인스턴스와 연결하면 RDS는 DB 인스턴스가 재부팅된 후에만 수정된 정적 파라미터 및 동적 파라미터를 적용합니다. 그러나 DB 파라미터 그룹을 DB 인스턴스에 연결한 후 DB 파라미터 그룹에서 동적 파라미터를 수정하면 이러한 변경 사항이 재부팅 없이 즉시 적용됩니다. 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.
- 기존 DB 파라미터 그룹의 정적 파라미터에 변경 적용 - 정적 파라미터를 변경하고 DB 파라미터 그룹을 저장하면 콘솔에서 이 파라미터 그룹과 연결된 DB 인스턴스의 상태가 재부팅 보류 중으로 변경됩니다. 파라미터의 변경 사항은 연결된 DB 인스턴스를 재부팅한 후에만 적용됩니다. 기존 파라미터 그룹에서 동적 파라미터를 변경하면 재부팅할 필요 없이 기본적으로 변경 내용이 즉시 적용됩니다.

> **Note**
>
> 재부팅 보류 중 상태로 인해 다음번 유지 관리 기간에 자동 재부팅이 되지는 않습니다. DB 인스턴스에 최신 파라미터 변경 사항을 적용하려면 DB 인스턴스를 수동으로 재부팅합니다. 파라미터 그룹에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하세요.

- 문제 해결 - 재부팅이 필요한 성능 또는 기타 운영 문제가 발생할 수 있습니다. 예를 들어, DB 인스턴스가 응답하지 않을 수 있습니다.

DB 인스턴스를 재부팅하는 방법

Amazon RDS는 DB 인스턴스를 재부팅할 때 다음과 같은 순차적 작업을 수행합니다.

1. DB 인스턴스에 있는 데이터베이스 서비스를 중지합니다.
2. DB 인스턴스에서 데이터베이스 서비스를 시작합니다.

재부팅 프로세스로 인해 잠시 중단됩니다. 이 운영 중단 기간 동안에는 DB 인스턴스 상태가 재부팅 중으로 나타납니다. 단일 AZ 배포와 다중 AZ DB 인스턴스 배포 모두에서 중단이 발생하며, 장애 조치로 재부팅할 때도 마찬가지입니다.

다중 AZ 배포에서 DB 인스턴스를 재부팅하는 방법

Amazon RDS DB 인스턴스가 다중 AZ 배포에 있는 경우 장애 조치를 통해 재부팅할 수 있습니다. 이 작업은 DB 인스턴스 결함을 시뮬레이션하거나, 장애 조치 이후 원래 가용 영역으로 작업을 복구할 때 유용한 기능입니다.

장애 조치를 사용한 재부팅 중에 Amazon RDS는 다음을 수행합니다.

- 데이터베이스를 갑자기 중단합니다. DB 인스턴스와 해당 클라이언트 세션을 정상적으로 종료할 시간이 없을 수 있습니다.

> **Warning**
>
> 데이터 손실 가능성을 방지하려면 장애 조치로 재부팅하기 전에 DB 인스턴스에서 트랜잭션을 중지하는 것이 좋습니다.

- 필요한 경우 데이터베이스의 충돌 복구를 수행합니다.

- 다른 AZ의 대기 복제본으로 자동 전환합니다. AWS Management Console 및 AWS CLI와 RDS API 호출에 AZ 변경 사항이 몇 분 동안 반영되지 않을 수 있습니다.
- 대기 DB 인스턴스를 가리키도록 DB 인스턴스의 DNS 레코드를 업데이트합니다. 결과적으로 기존의 DB 인스턴스 연결을 모두 삭제한 후 재구성해야 합니다. 자세한 내용은 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 단원을 참조하십시오.
- 재부팅 후 Amazon RDS 이벤트를 생성합니다.

RDS for Microsoft SQL Server에서 장애 조치는 기본 DB 인스턴스만 재부팅합니다. 장애 조치 후에는 기본 DB 인스턴스가 새 보조 DB 인스턴스가 됩니다. 다중 AZ 인스턴스의 경우 파라미터가 업데이트되지 않을 수 있습니다. 장애 조치 없이 재부팅하는 경우 기본 DB 인스턴스와 보조 DB 인스턴스가 모두 재부팅되고 재부팅된 후 파라미터가 업데이트됩니다. DB 인스턴스가 응답하지 않는 경우 장애 조치 없이 재부팅하는 것이 좋습니다.

DB 인스턴스 재부팅 시 고려 사항

인스턴스를 재부팅하기 전에 다음 사항을 고려하세요.

- 읽기 전용 복제본이 있는 DB 인스턴스의 경우 원본 DB 인스턴스와 읽기 전용 복제본을 독립적으로 재부팅할 수 있습니다. 재부팅이 완료되면 복제가 자동으로 재개됩니다.
- 재부팅 시간은 충돌 복구 프로세스, 재부팅 시 데이터베이스 작업 및 특정 DB 엔진의 동작에 따라 달라집니다. 따라서 재부팅 시간을 단축하려면 재부팅 시 데이터베이스 작업을 최소화하는 것이 좋습니다. 이 방법을 사용하면 중간 트랜잭션의 롤백 작업이 줄어듭니다.

DB 인스턴스 재부팅 시 사전 조건

다음 사전 조건을 충족하는지 확인합니다.

- DB 인스턴스는 `available` 상태여야 합니다. 진행 중인 백업, 이전에 요청한 수정 또는 유지 관리 기간 작업과 같은 여러 가지 이유로 데이터베이스를 사용할 수 없습니다.
- 다른 AZ로 강제 장애 조치하는 경우 DB 인스턴스를 다중 AZ용으로 구성해야 합니다.
- 다른 AZ로 강제 장애 조치를 수행하는 경우 데이터 손실을 방지하기 위해 먼저 DB 인스턴스에서 트랜잭션을 중지하는 것이 좋습니다.

DB 인스턴스 재부팅: 기본 단계

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB 인스턴스를 재부팅할 수 있습니다.

콘솔

DB 인스턴스를 재부팅하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 다음 재부팅하려는 DB 인스턴스를 선택합니다.
3. 작업에서 재부팅을 선택합니다.

   DB 인스턴스 재부팅 페이지가 나타납니다.
4. (선택 사항) 한 AZ에서 다른 AZ로 장애 조치를 강제로 실행하려면 Reboot with failover(장애 조치로 재부팅하시겠습니까)?을 선택합니다.
5. DB 인스턴스를 재부팅하려면 [Reboot]를 선택합니다.

   또는 [Cancel]을 선택합니다.

AWS CLI

AWS CLI를 사용하여 DB 인스턴스를 재부팅하려면 `reboot-db-instance` 명령을 호출하십시오.

Example 간편한 재부팅

대상 LinuxmacOS, 또는Unix:

```
aws rds reboot-db-instance \
    --db-instance-identifier mydbinstance
```

Windows의 경우:

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

Example 장애 조치로 재부팅

다중 AZ DB 클러스터의 한 AZ에서 다른 AZ로 장애 조치를 강제로 수행하려면 `--force-failover` 파라미터를 사용합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds reboot-db-instance \
    --db-instance-identifier mydbinstance \
    --force-failover
```

Windows의 경우:

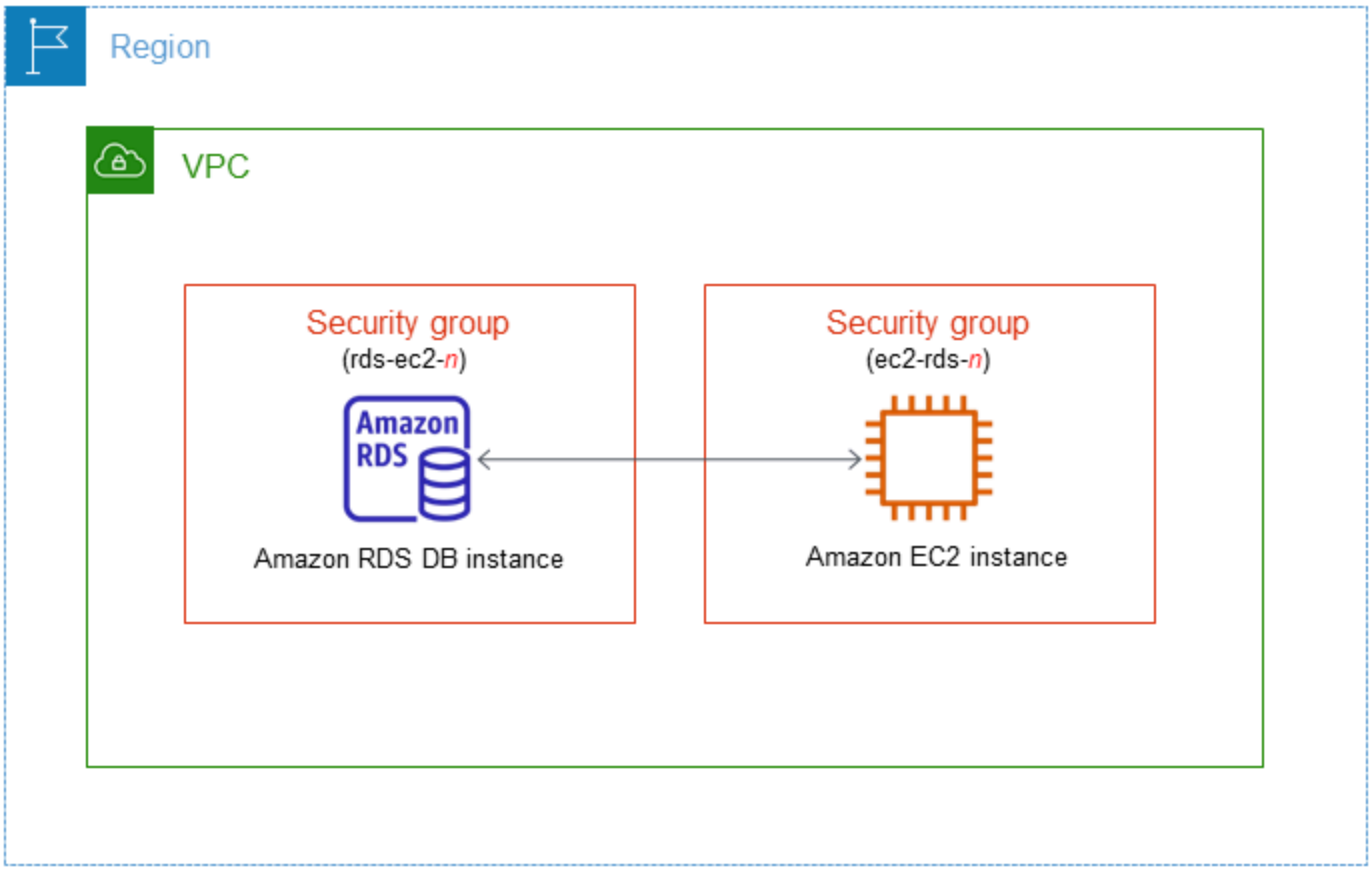
```
aws rds reboot-db-instance ^
    --db-instance-identifier mydbinstance ^
    --force-failover
```

RDS API

Amazon RDS API를 사용하여 DB 인스턴스를 재부팅하려면 `RebootDBInstance` 작업을 호출하세요.

EC2 인스턴스와 DB 인스턴스를 자동으로 연결

Amazon RDS 콘솔을 사용하여 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스와 DB 인스턴스 간의 연결 설정을 간소화할 수 있습니다. DB 인스턴스는 프라이빗 서브넷에 있고, EC2 인스턴스는 VPC 내의 퍼블릭 서브넷에 있는 경우가 많습니다. EC2 인스턴스의 SQL 클라이언트를 사용하여 DB 인스턴스에 연결할 수 있습니다. EC2 인스턴스는 프라이빗 DB 인스턴스에 액세스하는 웹 서버 또는 애플리케이션을 실행할 수도 있습니다. EC2 인스턴스와 다중 AZ DB 클러스터 간의 연결을 설정하는 방법에 대한 지침은 the section called “EC2 인스턴스와 다중 AZ DB 클러스터 연결” 섹션을 참조하세요.



DB 인스턴스와 동일한 VPC에 있지 않은 EC2 인스턴스에 연결하려는 경우 VPC에서 DB 인스턴스에 액세스하는 시나리오의 시나리오를 참조하세요.

주제

- EC2 인스턴스와의 자동 연결 개요
- EC2 인스턴스와 RDS 데이터베이스 자동 연결
- 연결된 컴퓨팅 리소스 보기

- 특정 DB 엔진을 실행하는 DB 인스턴스에 연결

EC2 인스턴스와의 자동 연결 개요

EC2 인스턴스와 RDS 데이터베이스 간의 연결을 설정하는 경우 Amazon RDS는 EC2 인스턴스 및 RDS 데이터베이스에 VPC 보안 그룹을 자동으로 구성합니다.

다음은 EC2 인스턴스를 RDS 데이터베이스와 연결하기 위한 요구 사항입니다.

- EC2 인스턴스는 RDS 데이터베이스와 동일한 VPC 있어야 합니다.

  EC2 인스턴스가 동일한 VPC에 없는 경우 콘솔은 EC2 인스턴스를 생성하기 위한 링크를 제공합니다.
- 연결을 설정하는 사용자는 다음 Amazon EC2 작업을 수행할 수 있는 권한이 있어야 합니다.
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeSecurityGroups`
  - `ec2:ModifyNetworkInterfaceAttribute`
  - `ec2:RevokeSecurityGroupEgress`

DB 인스턴스와 EC2 인스턴스가 서로 다른 가용 영역에 있는 경우 계정에 교차 가용 영역 비용이 발생할 가능성이 있습니다.

EC2 인스턴스에 대한 연결을 설정하면 Amazon RDS는 다음 테이블에 설명된 대로 RDS 데이터베이스 및 EC2 인스턴스와 연결된 보안 그룹의 현재 구성을 기반으로 조치를 취합니다.

현재 RDS 보안 그룹 구성	현재 EC2 보안 그룹 구성	RDS 작업
`rds-ec2-`*n*(*n*은 숫자를 나타냄) 패턴과 일치하는 이름을 가진 RDS 데이터베이스와 연결된 보안 그룹이 하나 이상 있습니다. 해당 패턴과 일치하는 보	`ec2-rds-`*n*(*n*은 숫자를 나타냄) 패턴과 일치하는 이름을 가진 EC2 인스턴스와 연결된 보안 그룹이 하나 이상 있습니다. 해당 패턴과 일치하는 보안 그	RDS는 아무 작업도 수행하지 않습니다. EC2 인스턴스와 RDS 데이터베이스 간의 연결이 이미 자동

현재 RDS 보안 그룹 구성	현재 EC2 보안 그룹 구성	RDS 작업
안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 EC2 인스턴스의 VPC 보안 그룹을 소스로 하는 인바운드 규칙이 하나만 포함됩니다.	룹이 수정되지 않았습니다. 이 보안 그룹에는 RDS 데이터베이스의 VPC 보안 그룹을 소스로 하는 아웃바운드 규칙이 하나만 포함됩니다.	으로 구성되었습니다. EC2 인스턴스와 RDS 데이터베이스 사이에 이미 연결이 존재하기 때문에 보안 그룹은 수정되지 않습니다.
다음 중 하나의 조건이 적용됩니다. • `rds-ec2-`*n* 패턴과 일치하는 이름을 가진 RDS 데이터베이스와 연결된 보안 그룹이 없습니다. • `rds-ec2-`*n* 패턴과 일치하는 이름을 가진 RDS 데이터베이스와 연결된 보안 그룹이 하나 이상 있습니다. 그러나 Amazon RDS는 EC2 인스턴스와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon RDS는 EC2 인스턴스의 VPC 보안 그룹을 소스로 하는 인바운드 규칙 하나가 포함되지 않은 보안 그룹을 사용할 수 없습니다. 또한 수정된 보안 그룹을 사용할 수 없습니다. 수정 사항의 예로는 규칙 추가 또는 기존 규칙의 포트 변경이 있습니다.	다음 중 하나의 조건이 적용됩니다. • `ec2-rds-`*n* 패턴과 일치하는 이름을 가진 EC2 인스턴스와 연결된 보안 그룹이 없습니다. • `ec2-rds-`*n* 패턴과 일치하는 이름을 가진 EC2 인스턴스와 연결된 보안 그룹이 하나 이상 있습니다. 그러나 Amazon RDS는 RDS 데이터베이스와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon RDS는 RDS 데이터베이스의 VPC 보안 그룹을 소스로 하는 아웃바운드 규칙 하나가 포함되지 않은 보안 그룹을 사용할 수 없습니다. 또한 수정된 보안 그룹을 사용할 수 없습니다.	RDS action: create new security groups

현재 RDS 보안 그룹 구성	현재 EC2 보안 그룹 구성	RDS 작업
`rds-ec2-`*n* 패턴과 일치하는 이름을 가진 RDS 데이터베이스와 연결된 보안 그룹이 하나 이상 있습니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 EC2 인스턴스의 VPC 보안 그룹을 소스로 하는 인바운드 규칙이 하나만 포함됩니다.	`ec2-rds-`*n* 패턴과 일치하는 이름을 가진 EC2 인스턴스와 연결된 보안 그룹이 하나 이상 있습니다. 그러나 Amazon RDS는 RDS 데이터베이스와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon RDS는 RDS 데이터베이스의 VPC 보안 그룹을 소스로 하는 아웃바운드 규칙 하나가 포함되지 않은 보안 그룹을 사용할 수 없습니다. 또한 수정된 보안 그룹을 사용할 수 없습니다.	RDS action: create new security groups
`rds-ec2-`*n* 패턴과 일치하는 이름을 가진 RDS 데이터베이스와 연결된 보안 그룹이 하나 이상 있습니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 EC2 인스턴스의 VPC 보안 그룹을 소스로 하는 인바운드 규칙이 하나만 포함됩니다.	연결에 유효한 EC2 보안 그룹이 있지만 EC2 인스턴스와 연결되어 있지 않습니다. 이 보안 그룹의 이름이 `ec2-rds-`*n* 패턴과 일치합니다. 수정되지 않았습니다. 여기에는 RDS 데이터베이스의 VPC 보안 그룹을 소스로 하는 아웃바운드 규칙이 하나만 포함됩니다.	RDS action: associate EC2 security group

현재 RDS 보안 그룹 구성	현재 EC2 보안 그룹 구성	RDS 작업
다음 중 하나의 조건이 적용됩니다. • `rds-ec2-`*n* 패턴과 일치하는 이름을 가진 RDS 데이터베이스와 연결된 보안 그룹이 없습니다. • `rds-ec2-`*n* 패턴과 일치하는 이름을 가진 RDS 데이터베이스와 연결된 보안 그룹이 하나 이상 있습니다. 그러나 Amazon RDS는 EC2 인스턴스와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon RDS는 EC2 인스턴스의 VPC 보안 그룹을 소스로 하는 인바운드 규칙 하나가 포함되지 않은 보안 그룹을 사용할 수 없습니다. 또한 수정된 보안 그룹을 사용할 수 없습니다.	`ec2-rds-`*n* 패턴과 일치하는 이름을 가진 EC2 인스턴스와 연결된 보안 그룹이 하나 이상 있습니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 RDS 데이터베이스의 VPC 보안 그룹을 소스로 하는 아웃바운드 규칙이 하나만 포함됩니다.	RDS action: create new security groups

RDS 작업: 새 보안 그룹 생성

Amazon RDS에서 다음 작업을 수행합니다.

- `rds-ec2-`*n* 패턴과 일치하는 새 보안 그룹을 생성합니다. 이 보안 그룹에는 EC2 인스턴스의 VPC 보안 그룹을 소스로 하는 인바운드 규칙이 하나 포함됩니다. 이 보안 그룹은 RDS 데이터베이스와 연결되어 있으며 EC2 인스턴스가 RDS 데이터베이스에 액세스하도록 허용합니다.
- `ec2-rds-`*n* 패턴과 일치하는 새 보안 그룹을 생성합니다. 이 보안 그룹에는 RDS 데이터베이스의 VPC 보안 그룹을 대상으로 하는 아웃바운드 규칙이 하나 포함됩니다. 이 보안 그룹은 EC2 인스턴스와 연결되어 있으며 EC2 인스턴스가 RDS 데이터베이스에 트래픽을 보내도록 허용합니다.

RDS 작업: EC2 보안 그룹 연결

Amazon RDS는 유효한 기존 EC2 보안 그룹을 EC2 인스턴스와 연결합니다. 이 보안 그룹은 EC2 인스턴스가 RDS 데이터베이스에 트래픽을 보내도록 허용합니다.

EC2 인스턴스와 RDS 데이터베이스 자동 연결

EC2 인스턴스와 RDS 데이터베이스 간의 연결을 설정하기 전에 EC2 인스턴스와의 자동 연결 개요에 설명된 요구 사항을 충족하는지 확인하세요.

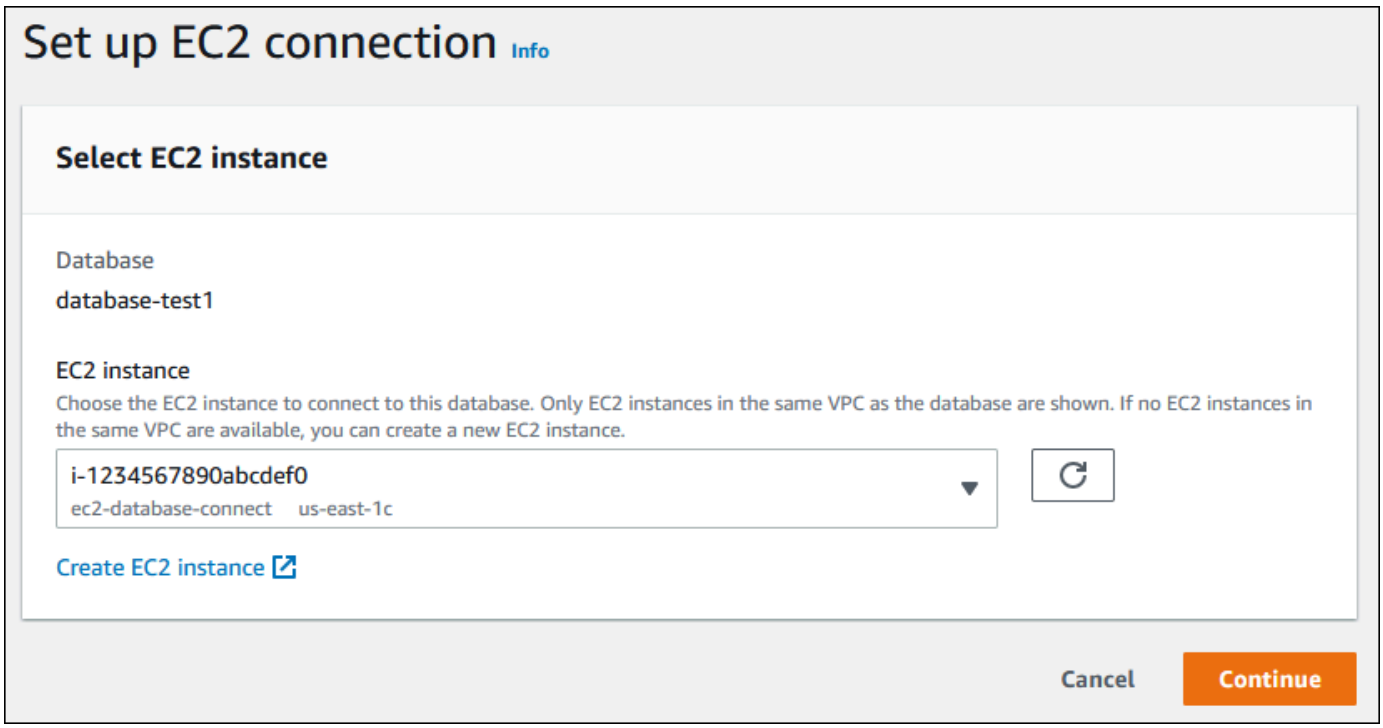
연결을 구성한 후에 보안 그룹을 변경할 경우 변경 사항이 EC2 인스턴스와 RDS 데이터베이스 간의 연결에 영향을 미칠 수 있습니다.

> **Note**
>
> AWS Management Console을 사용해야만 EC2 인스턴스와 RDS 데이터베이스 간의 연결을 자동으로 설정할 수 있습니다. AWS CLI 또는 RDS API로는 연결을 자동으로 설정할 수 없습니다.

EC2 인스턴스와 RDS 데이터베이스를 자동으로 연결하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택하고 RDS 데이터베이스를 선택합니다.
3. 작업에서 EC2 연결 설정을 선택합니다.

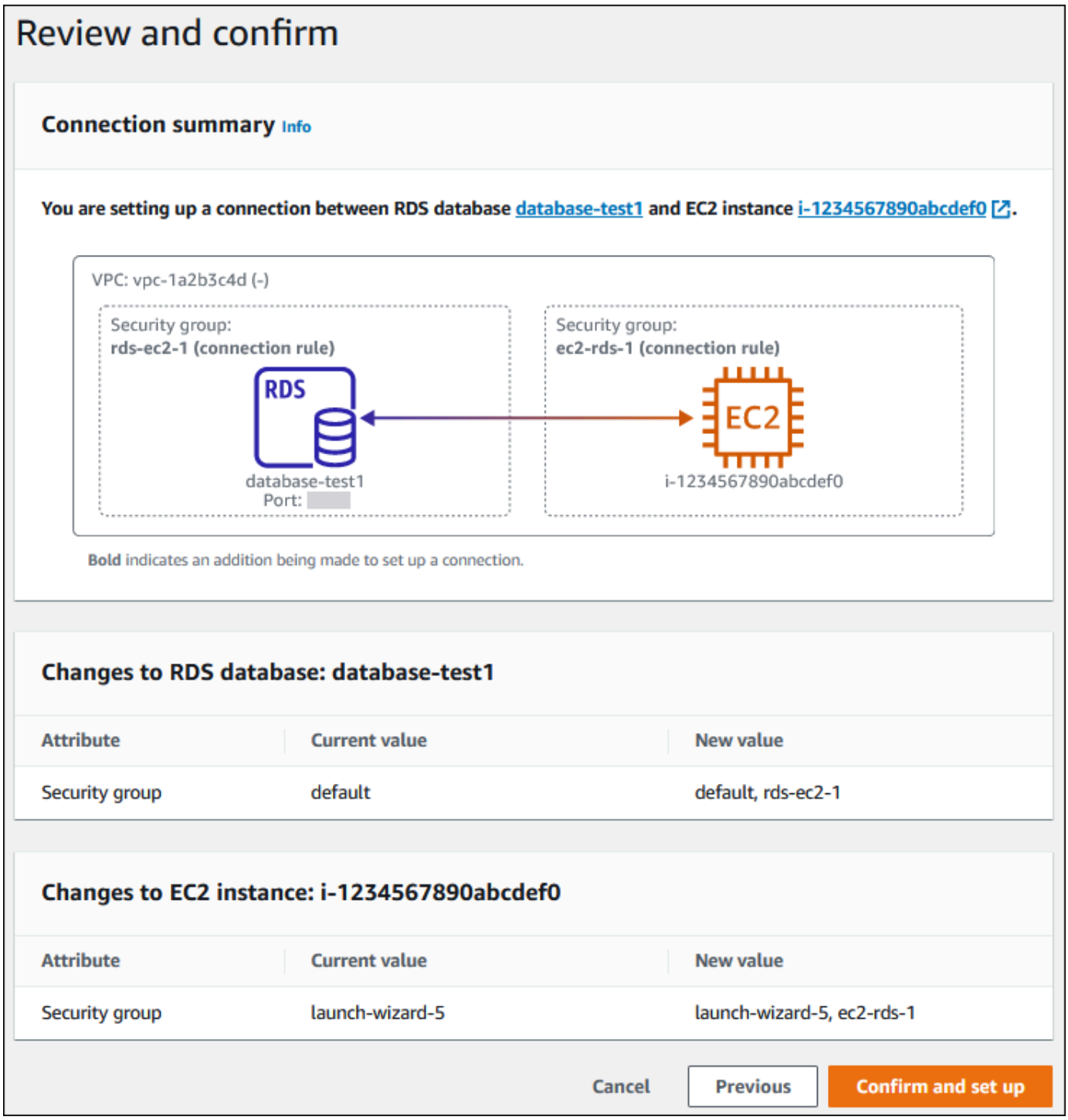
   EC2 연결 설정 페이지가 나타납니다.
4. EC2 연결 설정 페이지에서 EC2 인스턴스를 선택합니다.



동일한 VPC에 EC2 인스턴스가 없는 경우 EC2 인스턴스 생성을 선택하여 인스턴스를 생성합니다. 이 경우 새 EC2 인스턴스가 RDS 데이터베이스와 동일한 VPC 있어야 합니다.

5. 계속을 선택합니다.

   검토 및 확인 페이지가 나타납니다.



6. 검토 및 확인 페이지에서 RDS가 EC2 인스턴스와의 연결을 설정하기 위해 적용할 변경 사항을 검토합니다.

   변경 내용이 올바르면 확인 및 설정을 선택합니다.

   변경 내용이 올바르지 않으면 이전 또는 취소를 선택합니다.

연결된 컴퓨팅 리소스 보기

AWS Management Console을 사용하여 RDS 데이터베이스에 연결된 컴퓨팅 리소스를 볼 수 있습니다. 표시되는 리소스에는 자동으로 설정된 컴퓨팅 리소스 연결이 포함됩니다. 다음과 같은 방법으로 컴퓨팅 리소스와의 연결을 자동으로 설정할 수 있습니다.

- 데이터베이스를 생성할 때 컴퓨팅 리소스를 선택할 수 있습니다.

  자세한 내용은 Amazon RDS DB 인스턴스 생성 및 Amazon RDS용 다중 AZ DB 클러스터 생성 섹션을 참조하세요.
- 기존 데이터베이스와 컴퓨팅 리소스 간의 연결을 설정할 수 있습니다.

  자세한 내용은 EC2 인스턴스와 RDS 데이터베이스 자동 연결 단원을 참조하십시오.

나열된 컴퓨팅 리소스에는 데이터베이스에 수동으로 연결된 리소스는 포함되지 않습니다. 예를 들어 데이터베이스와 연결된 VPC 보안 그룹에 규칙을 추가하여 컴퓨팅 리소스가 데이터베이스에 수동으로 액세스하도록 허용할 수 있습니다.

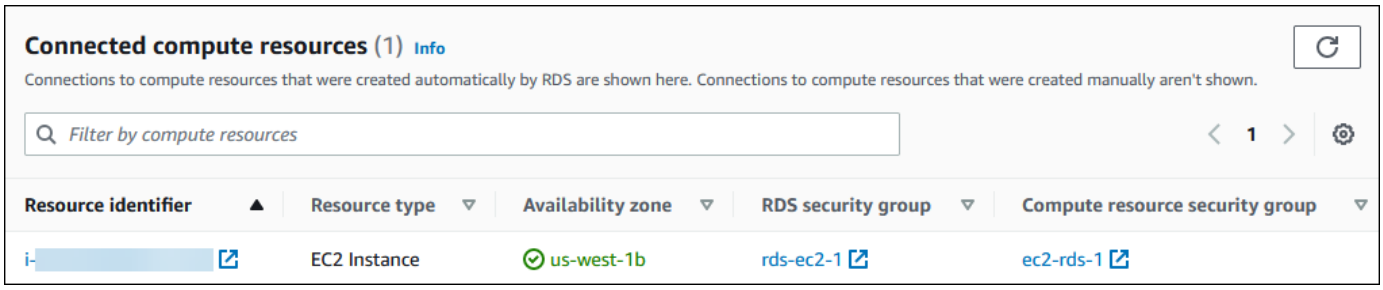
컴퓨팅 리소스가 나열되려면 다음 조건이 적용되어야 합니다.

- 컴퓨팅 리소스와 연결된 보안 그룹의 이름은 패턴 `ec2-rds-`*n*(여기서*n*은 숫자)과 일치합니다.
- 컴퓨팅 리소스와 연결된 보안 그룹에는 포트 범위가 RDS 데이터베이스에서 사용하는 포트로 설정된 아웃바운드 규칙이 있습니다.
- 컴퓨팅 리소스와 연결된 보안 그룹에는 소스가 RDS 데이터베이스에 연결된 보안 그룹으로 설정된 아웃바운드 규칙이 있습니다.
- RDS 데이터베이스와 연결된 보안 그룹의 이름은 패턴 `rds-ec2-`*n*(여기서 *n*은 숫자)과 일치합니다.
- RDS 데이터베이스와 연결된 보안 그룹에는 포트 범위가 RDS 데이터베이스에서 사용하는 포트로 설정된 인바운드 규칙이 있습니다.
- RDS 데이터베이스와 연결된 보안 그룹에는 소스가 컴퓨팅 리소스에 연결된 보안 그룹으로 설정된 인바운드 규칙이 있습니다.

RDS 데이터베이스에 연결된 컴퓨팅 리소스를 보는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 다음 RDS 데이터베이스의 이름을 선택합니다.

3. 연결 및 보안 탭의 연결된 컴퓨팅 리소스에서 컴퓨팅 리소스를 확인합니다.



특정 DB 엔진을 실행하는 DB 인스턴스에 연결

특정 DB 엔진을 실행하는 DB 인스턴스에 연결하는 방법에 대한 자세한 내용은 DB 엔진의 지침을 따르세요.

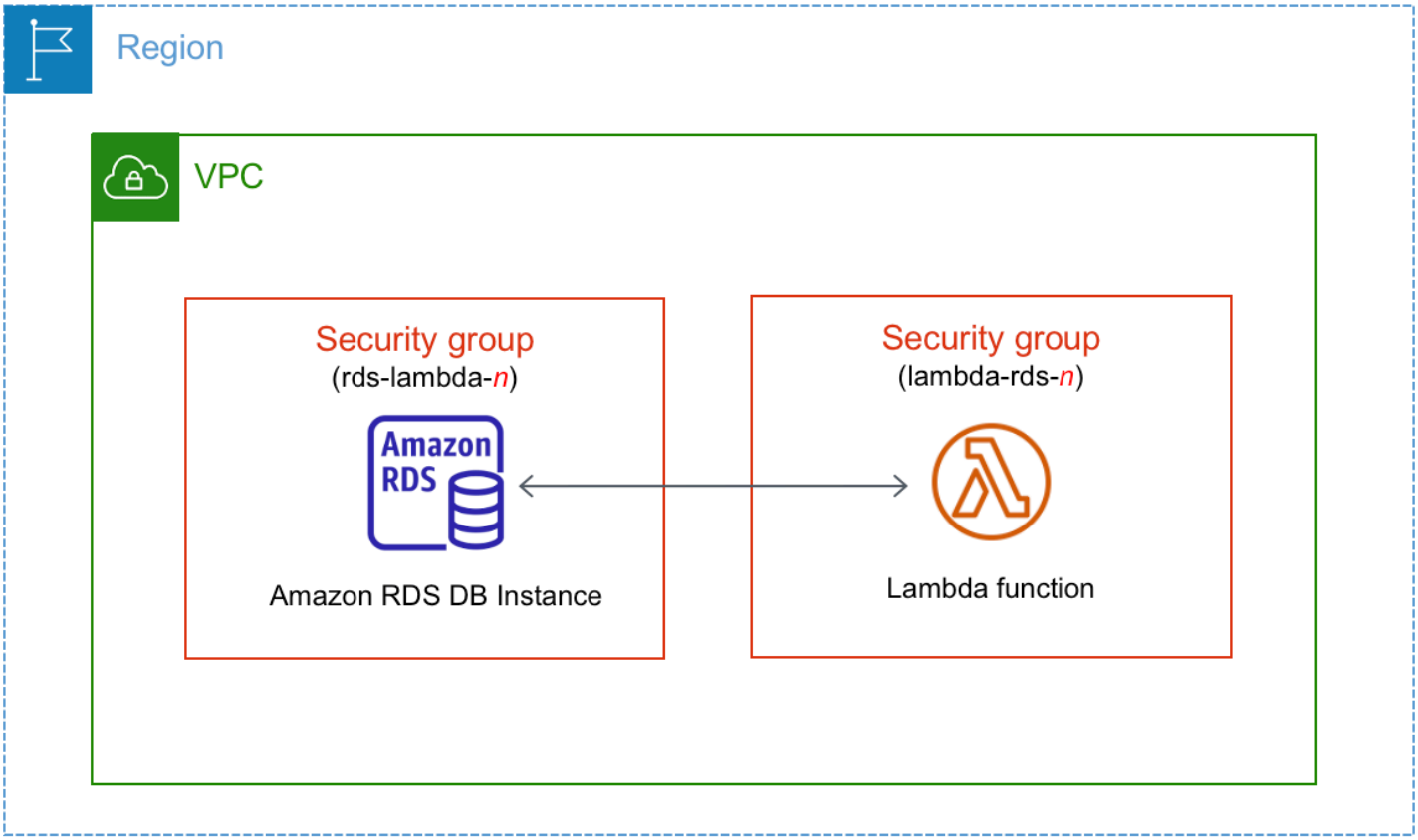
- MariaDB DB 인스턴스에 연결
- Microsoft SQL Server DB 인스턴스에 연결
- MySQL DB 인스턴스에 연결
- Oracle DB 인스턴스에 연결
- PostgreSQL 데이터베이스 엔진을 실행하는 DB 인스턴스에 연결

Lambda 함수와 DB 인스턴스 자동 연결

Amazon RDS 콘솔을 사용하여 Lambda 함수와 DB 인스턴스 간의 연결 설정을 간소화할 수 있습니다. DB 인스턴스가 VPC 내 프라이빗 서브넷에 있는 경우가 많습니다. 애플리케이션에서 프라이빗 DB 인스턴스에 액세스하는 데 Lambda 함수를 사용할 수 있습니다.

Lambda 함수와 다중 AZ DB 클러스터 간의 연결 설정에 대한 지침은 the section called “Lambda 함수와 다중 AZ DB 클러스터 연결” 섹션을 참조하세요.

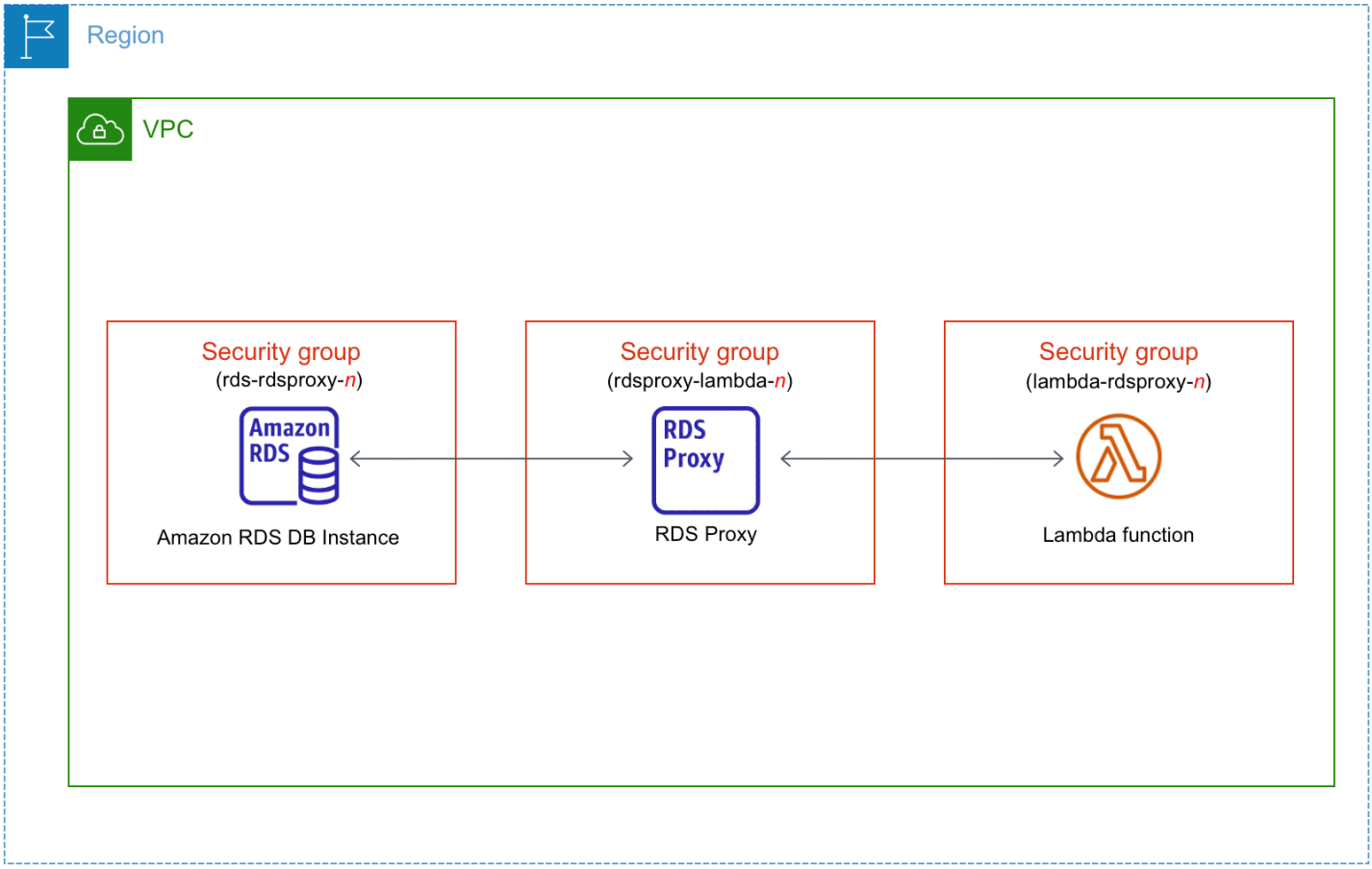
다음 이미지는 DB 인스턴스와 Lambda 함수 간의 직접 연결을 보여줍니다.



RDS 프록시를 통해 Lambda 함수와 DB 인스턴스 간의 연결을 설정하여 데이터베이스 성능과 복원력을 개선할 수 있습니다. Lambda 함수는 RDS 프록시가 제공하는 연결 풀링의 이점을 누릴 수 있는 단기 데이터베이스 연결을 자주 만드는 경우가 많습니다. Lambda 애플리케이션 코드에서 데이터베이스 보안 인증 정보를 관리하는 대신 Lambda 함수에 대해 이미 가지고 있는 AWS Identity and Access Management(IAM) 인증을 활용할 수 있습니다. 자세한 내용은 Amazon RDS Proxy 단원을 참조하십시오.

콘솔을 사용하여 기존 프록시에 연결하면 Amazon RDS가 DB 인스턴스와 Lambda 함수 간의 연결을 허용하도록 프록시 보안 그룹을 업데이트합니다.

동일한 콘솔 페이지에서 새 프록시를 만들 수도 있습니다. 콘솔에서 프록시를 만들 때 DB 인스턴스에 액세스하려면 데이터베이스 보안 인증 정보를 입력하거나 AWS Secrets Manager 보안 암호를 선택해야 합니다.



Tip

Lambda 함수를 DB 인스턴스에 빠르게 연결하려면 콘솔 내 안내 마법사를 사용할 수도 있습니다. 마법사에서 다음을 수행합니다.

1. Lambda 콘솔의 함수 페이지를 엽니다.
2. 데이터베이스를 연결할 함수를 선택합니다.
3. 구성 탭에서 RDS 데이터베이스를 선택합니다.
4. RDS 데이터베이스에 연결을 선택합니다.

함수를 데이터베이스에 연결한 후 프록시 추가를 선택하여 프록시를 생성할 수 있습니다.

주제

- Lambda 함수와의 자동 연결 개요
- Lambda 함수와 RDS 데이터베이스 자동 연결
- 연결된 컴퓨팅 리소스 보기

Lambda 함수와의 자동 연결 개요

다음은 Lambda 함수를 RDS DB 인스턴스와 연결하기 위한 요구 사항입니다.

- Lambda 함수가 DB 인스턴스와 같은 VPC에 있어야 합니다.
- 연결을 설정하는 사용자는 다음과 같은 Amazon RDS, Amazon EC2, Lambda, Secrets Manager 및 IAM 작업을 수행할 권한이 있어야 합니다.
  - Amazon RDS
    - `rds:CreateDBProxies`
    - `rds:DescribeDBInstances`
    - `rds:DescribeDBProxies`
    - `rds:ModifyDBInstance`
    - `rds:ModifyDBProxy`
    - `rds:RegisterProxyTargets`
  - Amazon EC2
    - `ec2:AuthorizeSecurityGroupEgress`
    - `ec2:AuthorizeSecurityGroupIngress`
    - `ec2:CreateSecurityGroup`
    - `ec2:DeleteSecurityGroup`
    - `ec2:DescribeSecurityGroups`
    - `ec2:RevokeSecurityGroupEgress`
    - `ec2:RevokeSecurityGroupIngress`
  - Lambda
    - `lambda:CreateFunctions`
    - `lambda:ListFunctions`
    - `lambda:UpdateFunctionConfiguration`
  - Secrets Manager

    - `secretsmanager:CreateSecret`
    - `secretsmanager:DescribeSecret`
- IAM
    - `iam:AttachPolicy`
    - `iam:CreateRole`
    - `iam:CreatePolicy`
- AWS KMS
    - `kms:describeKey`

> **Note**
>
> DB 인스턴스 및 Lambda 함수가 서로 다른 가용 영역에 있는 경우 계정에 교차 가용 영역 비용이 발생할 수 있습니다.

Lambda 함수와 RDS 데이터베이스 간의 연결을 자동으로 설정할 때 RDS는 함수 및 DB 인스턴스에 대한 VPC 보안 그룹을 구성합니다. RDS 프록시를 사용하는 경우 Amazon RDS는 프록시에 대한 VPC 보안 그룹도 구성합니다. Amazon RDS는 다음 테이블에 설명된 것과 같이 DB 인스턴스, Lambda 함수 및 프록시와 관련된 보안 그룹의 현재 구성에 따라 작동합니다.

현재 RDS 보안 그룹 구성	현재 Lambda 보안 그룹 구성	현재 프록시 보안 그룹 구성	RDS 작업
이름이 `rds-lambda-` *n* 패턴과 일치하며 DB 인스턴스와 연결된 보안 그룹이 하나 이상 있습니다. 또는 프록시가 이미 DB 인스턴스에 연결되어 있는 경우, RDS는 연결된 프록시의 `TargetHealth` 가 `AVAILABLE` 인지 확인합니다.	이름이 `lambda-rds-` *n* 또는 `lambda-rdsproxy-` *n*(*n*은 숫자를 나타냄) 패턴과 일치하며 Lambda 함수와 연결된 보안 그룹이 하나 이상 있습니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안	이름이 `rdsproxy-lambda-` *n*(*n*은 숫자를 나타냄) 패턴과 일치하며 프록시와 연결된 보안 그룹이 하나 이상 있습니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 Lambda 함수 및 DB 인스턴스의	Amazon RDS는 아무런 조치도 취하지 않습니다. Lambda 함수, 프록시(선택 사항) 및 DB 인스턴스 간에 연결이 이미 자동으로 구성되었습니다. 함수, 프록시, 데이터베이스 사이에 이미 연결이 존재하기

현재 RDS 보안 그룹 구성	현재 Lambda 보안 그룹 구성	현재 프록시 보안 그룹 구성	RDS 작업
해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 Lambda 함수 또는 프록시의 VPC 보안 그룹을 소스로 하는 인바운드 규칙이 하나만 있습니다.	그룹에는 DB 인스턴스 또는 프록시의 VPC 보안 그룹을 대상으로 하는 아웃바운드 규칙이 하나만 있습니다.	VPC 보안 그룹이 포함된 인바운드 및 아웃바운드 규칙이 있습니다.	때문에 보안 그룹은 수정되지 않습니다.

현재 RDS 보안 그룹 구성	현재 Lambda 보안 그룹 구성	현재 프록시 보안 그룹 구성	RDS 작업
다음 중 하나의 조건이 적용됩니다. • 이름이 `rds-lambda- n` 패턴과 일치하며 DB 인스턴스와 연결된 보안 그룹이 없거나 연결된 프록시의 `TargetHealth` 가 `AVAILABLE` 입니다. • 이름이 `rds-lambda- n` 패턴과 일치하며 DB 인스턴스와 연결된 보안 그룹이 없거나 연결된 프록시의 `TargetHealth` 가 `AVAILABLE` 입니다. 하지만 이러한 보안 그룹 중 어느 것도 Lambda 함수와의 연결에 사용할 수 없습니다. Amazon RDS는 Lambda 함수 또는 프록시의 VPC 보안 그룹을 소스로 하는 인바운드 규칙 하나가 포함되지 않은 보안 그룹을 사용할 수 없습니다. 또한 수정된 보안 그룹을 사용할 수 없습니	다음 중 하나의 조건이 적용됩니다. • 이름이 `lambda-rds- n` 또는 `lambda-rdsproxy- n` 패턴과 일치하며 Lambda 함수와 연결된 보안 그룹이 없습니다. • 이름이 `lambda-rds- n` 또는 `lambda-rdsproxy- n` 패턴과 일치하며 Lambda 함수와 연결된 보안 그룹이 하나 이상 있습니다. 그러나 Amazon RDS는 DB 인스턴스와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon RDS는 DB 인스턴스 또는 프록시의 VPC 보안 그룹을 대상으로 하는 아웃바운드 규칙 하나가 포함되지 않은 보안 그룹을 사용할 수 없습니다. 또한 수정된 보안 그	다음 중 하나의 조건이 적용됩니다. • 이름이 `rdsproxy-lambda- n` 패턴과 일치하며 프록시와 연결된 보안 그룹이 없습니다. • 이름이 `rdsproxy-lambda- n` 패턴과 일치하며 프록시와 연결된 보안 그룹이 하나 이상 있습니다. 그러나 Amazon RDS는 DB 인스턴스 또는 Lambda 함수와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon RDS는 DB 인스턴스 및 Lambda 함수의 VPC 보안 그룹을 대상으로 하는 인바운드 및 아웃바운드 규칙이 포함되지 않은 보안 그룹을 사용할 수 없습니다. 또한 수정된 보안 그룹을 사용할 수 없습니다.	RDS action: create new security groups

현재 RDS 보안 그룹 구성	현재 Lambda 보안 그룹 구성	현재 프록시 보안 그룹 구성	RDS 작업
다. 수정 사항의 예로는 규칙 추가 또는 기존 규칙의 포트 변경이 있습니다.	룹을 사용할 수 없습니다.		
이름이 `rds-lambda-` *n* 패턴과 일치하며 DB 인스턴스와 연결된 보안 그룹이 없거나 연결된 프록시의 `TargetHealth` 가 `AVAILABLE` 입니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 Lambda 함수 또는 프록시의 VPC 보안 그룹을 소스로 하는 인바운드 규칙이 하나만 있습니다.	이름이 `lambda-rds-` *n* 또는 `lambda-rdsproxy-` *n* 패턴과 일치하며 Lambda 함수와 연결된 보안 그룹이 하나 이상 있습니다. 그러나 Amazon RDS는 DB 인스턴스와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon RDS는 DB 인스턴스 또는 프록시의 VPC 보안 그룹을 대상으로 하는 아웃바운드 규칙 하나가 포함되지 않은 보안 그룹을 사용할 수 없습니다. 또한 수정된 보안 그룹을 사용할 수 없습니다.	이름이 `rdsproxy-lambda-` *n* 패턴과 일치하며 프록시와 연결된 보안 그룹이 하나 이상 있습니다. 그러나 Amazon RDS는 DB 인스턴스 또는 Lambda 함수와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon RDS는 DB 인스턴스 및 Lambda 함수의 VPC 보안 그룹을 대상으로 하는 인바운드 및 아웃바운드 규칙이 포함되지 않은 보안 그룹을 사용할 수 없습니다. 또한 수정된 보안 그룹을 사용할 수 없습니다.	RDS action: create new security groups

현재 RDS 보안 그룹 구성	현재 Lambda 보안 그룹 구성	현재 프록시 보안 그룹 구성	RDS 작업
이름이 `rds-lambda-` *n* 패턴과 일치하며 DB 인스턴스와 연결된 보안 그룹이 없거나 연결된 프록시의 `TargetHealth` 가 `AVAILABLE` 입니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 Lambda 함수 또는 프록시의 VPC 보안 그룹을 소스로 하는 인바운드 규칙이 하나만 있습니다.	연결에 유효한 Lambda 보안 그룹이 존재하지만 Lambda 함수와 연결되어 있지 않습니다. 이 보안 그룹의 이름이 `lambda-rds-` *n* 또는 `lambda-rdsproxy-` *n* 패턴과 일치합니다. 수정되지 않았습니다. DB 인스턴스 또는 프록시의 VPC 보안 그룹을 대상으로 하는 아웃바운드 규칙이 하나만 있습니다.	연결에 유효한 프록시 보안 그룹이 있지만 프록시와 연결되어 있지 않습니다. 이 보안 그룹의 이름이 `rdsproxy-lambda-` *n* 패턴과 일치합니다. 수정되지 않았습니다. DB 인스턴스와 Lambda 함수의 VPC 보안 그룹이 포함된 인바운드 및 아웃바운드 규칙이 있습니다.	RDS action: associate Lambda security group

현재 RDS 보안 그룹 구성	현재 Lambda 보안 그룹 구성	현재 프록시 보안 그룹 구성	RDS 작업
다음 중 하나의 조건이 적용됩니다. • 이름이 `rds-lambda-` *n* 패턴과 일치하며 DB 인스턴스와 연결된 보안 그룹이 없거나 연결된 프록시의 `TargetHealth` 가 `AVAILABLE` 입니다. • 이름이 `rds-lambda-` *n* 패턴과 일치하며 DB 인스턴스와 연결된 보안 그룹이 없거나 연결된 프록시의 `TargetHealth` 가 `AVAILABLE` 입니다. 그러나 Amazon RDS는 Lambda 함수 또는 프록시와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon RDS는 Lambda 함수 또는 프록시의 VPC 보안 그룹을 소스로 하는 인바운드 규칙 하나가 포함되지 않은 보안 그룹을 사용할 수 없습니다. 또한 수정된 보안 그	이름이 `lambda-rds-` *n* 또는 `lambda-rdsproxy-` *n* 패턴과 일치하며 Lambda 함수와 연결된 보안 그룹이 하나 이상 있습니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 DB 인스턴스 또는 프록시의 VPC 보안 그룹을 대상으로 하는 아웃바운드 규칙이 하나만 있습니다.	이름이 `rdsproxy-lambda-` *n* 패턴과 일치하며 프록시와 연결된 보안 그룹이 하나 이상 있습니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 DB 인스턴스 및 Lambda 함수의 VPC 보안 그룹이 포함된 인바운드 및 아웃바운드 규칙이 있습니다.	RDS action: create new security groups

현재 RDS 보안 그룹 구성	현재 Lambda 보안 그룹 구성	현재 프록시 보안 그룹 구성	RDS 작업
룹을 사용할 수 없습니다.			

현재 RDS 보안 그룹 구성	현재 Lambda 보안 그룹 구성	현재 프록시 보안 그룹 구성	RDS 작업
다음 중 하나의 조건이 적용됩니다. • 이름이 `rds-lambda- n` 패턴과 일치하며 DB 인스턴스와 연결된 보안 그룹이 없거나 연결된 프록시의 `TargetHealth` 가 `AVAILABLE` 입니다. • 이름이 `rds-lambda- n` 패턴과 일치하며 DB 인스턴스와 연결된 보안 그룹이 없거나 연결된 프록시의 `TargetHealth` 가 `AVAILABLE` 입니다. 그러나 Amazon RDS는 Lambda 함수 또는 프록시와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon RDS는 Lambda 함수 또는 프록시의 VPC 보안 그룹을 소스로 하는 인바운드 규칙 하나가 포함되지 않은 보안 그룹을 사용할 수 없습니다. 또한 수정된 보안 그	다음 중 하나의 조건이 적용됩니다. • 이름이 `lambda-rds- n` 또는 `lambda-rdsproxy- n` 패턴과 일치하며 Lambda 함수와 연결된 보안 그룹이 없습니다. • 이름이 `lambda-rds- n` 또는 `lambda-rdsproxy- n` 패턴과 일치하며 Lambda 함수와 연결된 보안 그룹이 하나 이상 있습니다. 그러나 Amazon RDS는 DB 인스턴스와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon RDS는 DB 인스턴스 또는 프록시의 VPC 보안 그룹을 소스로 하는 아웃바운드 규칙 하나가 포함되지 않은 보안 그룹을 사용할 수 없습니다. 또한 수정된 보안 그	다음 중 하나의 조건이 적용됩니다. • 이름이 `rdsproxy-lambda- n` 패턴과 일치하며 프록시와 연결된 보안 그룹이 없습니다. • 이름이 `rdsproxy-lambda- n` 패턴과 일치하며 프록시와 연결된 보안 그룹이 하나 이상 있습니다. 그러나 Amazon RDS는 DB 인스턴스 또는 Lambda 함수와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon RDS는 DB 인스턴스 및 Lambda 함수의 VPC 보안 그룹을 대상으로 하는 인바운드 및 아웃바운드 규칙이 포함되지 않은 보안 그룹을 사용할 수 없습니다. 또한 수정된 보안 그룹을 사용할 수 없습니다.	RDS action: create new security groups

현재 RDS 보안 그룹 구성	현재 Lambda 보안 그룹 구성	현재 프록시 보안 그룹 구성	RDS 작업
룹을 사용할 수 없습니다.	룹을 사용할 수 없습니다.		

RDS 작업: 새 보안 그룹 생성

Amazon RDS에서 다음 작업을 수행합니다.

- RDS 프록시를 사용하도록 선택하는 경우, `rds-lambda-`*n* 또는 `rds-rdsproxy-`*n* 패턴과 일치하는 새 보안 그룹을 생성합니다. 이 보안 그룹에는 Lambda 함수 또는 프록시의 VPC 보안 그룹을 소스로 하는 인바운드 규칙이 하나 있습니다. 이 보안 그룹은 DB 인스턴스와 연결되어 있으며, 함수가 DB 인스턴스에 액세스하도록 허용합니다.
- `lambda-rds-`*n* 또는 `lambda-rdsproxy-`*n* 패턴과 일치하는 새 보안 그룹을 생성합니다. 이 보안 그룹에는 DB 인스턴스 또는 프록시의 VPC 보안 그룹을 대상으로 하는 아웃바운드 규칙이 하나 있습니다. 이 보안 그룹은 Lambda 함수와 연결되어 있으며, 함수가 트래픽을 DB 인스턴스로 보내거나 프록시를 통해 보내도록 허용합니다.
- `rdsproxy-lambda-`*n* 패턴과 일치하는 새 보안 그룹을 생성합니다. 이 보안 그룹에는 DB 인스턴스 및 Lambda 함수의 VPC 보안 그룹이 포함된 인바운드 및 아웃바운드 규칙이 있습니다.

RDS 작업: Lambda 보안 그룹 연결

Amazon RDS는 유효한 기존 Lambda 보안 그룹을 Lambda 함수와 연결합니다. 이 보안 그룹은 함수가 트래픽을 DB 인스턴스로 보내거나 프록시를 통해 보내도록 허용합니다.

Lambda 함수와 RDS 데이터베이스 자동 연결

Amazon RDS 콘솔을 사용하여 Lambda 함수를 DB 인스턴스에 자동으로 연결할 수 있습니다. 이렇게 하면 이러한 리소스 간의 연결을 설정하는 프로세스가 간소화됩니다.

RDS 프록시를 사용하여 연결에 프록시를 포함할 수도 있습니다. Lambda 함수는 RDS 프록시가 제공하는 연결 풀링의 이점을 누릴 수 있는 단기 데이터베이스 연결을 자주 만듭니다. Lambda 애플리케이션 코드에서 데이터베이스 보안 인증 정보를 관리하는 대신 Lambda 함수에 대해 이미 설정한 IAM 인증을 사용할 수도 있습니다.

Lambda 연결 설정 페이지를 사용하여 기존 DB 인스턴스를 신규 및 기존 Lambda 함수에 연결할 수 있습니다. 이 설정 프로세스는 필요한 보안 그룹을 자동으로 설정합니다.

Lambda 함수와 DB 인스턴스 간의 연결을 설정하기 전에 다음 요구 사항을 충족해야 합니다.

- Lambda 함수와 DB 인스턴스가 동일한 VPC에 있습니다.
- 사용자 계정에 대한 올바른 권한이 있습니다. 요구 사항에 대한 자세한 내용은 Lambda 함수와의 자동 연결 개요 섹션을 참조하세요.

연결을 구성한 후에 보안 그룹을 변경할 경우 변경 사항이 Lambda 함수와 DB 인스턴스 간의 연결에 영향을 미칠 수 있습니다.

> **Note**
>
> AWS Management Console에서만 DB 인스턴스와 Lambda 함수의 연결을 자동으로 설정할 수 있습니다. Lambda 함수를 연결하려면 DB 인스턴스가 사용 가능 상태여야 합니다.

Lambda 함수와 DB 인스턴스를 자동으로 연결하는 방법

<result>

설정을 확인하면 Amazon RDS가 Lambda 함수, RDS 프록시(프록시를 사용한 경우) 및 DB 인스턴스 연결 프로세스를 시작합니다. 콘솔에 연결 세부 정보 대화 상자가 표시되며, 여기에 리소스 간 연결을 허용하는 보안 그룹 변경 사항이 나열됩니다.
</result>

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 다음, Lambda 함수에 연결할 DB 인스턴스를 선택합니다.
3. 작업에서 Lambda 연결 설정을 선택합니다.
4. Lambda 연결 설정 페이지의 Lambda 함수 선택에서 다음 중 하나를 수행합니다.
   - DB 인스턴스와 동일한 VPC에 기존 Lambda 함수가 있는 경우 기존 함수 선택을 선택하고 해당 함수를 선택합니다.
   - 동일한 VPC에 Lambda 함수가 없는 경우 새 함수 생성을 선택한 다음 함수 이름을 입력합니다. 기본 런타임은 Nodejs.18로 설정되어 있습니다. 연결 설정을 완료한 후 Lambda 콘솔에서 새 Lambda 함수의 설정을 수정할 수 있습니다.
5. (선택 사항) RDS 프록시에서 RDS 프록시를 사용하여 연결을 선택하고 다음 중 하나를 수행합니다.

- 사용하려는 기존 프록시가 있는 경우 기존 프록시 선택을 선택하고 해당 프록시를 선택합니다.
- 프록시가 없고 Amazon RDS에서 자동으로 프록시를 생성하도록 하려면 새 프록시 생성을 선택합니다. 그런 다음, 데이터베이스 보안 인증 정보에서 다음 중 하나를 수행합니다.

  a. 데이터베이스 사용자 이름 및 암호를 선택한 다음, DB 인스턴스의 사용자 이름과 암호를 입력합니다.

  b. Secrets Manager 보안 암호를 선택합니다. 그런 다음, 보안 암호 선택에서 AWS Secrets Manager 보안 암호를 선택합니다. Secrets Manager 보안 암호가 없다면 새 Secrets Manager 보안 암호 생성을 선택하여 새 보안 암호를 생성합니다. 보안 암호를 생성한 후 보안 암호 선택에서 새 보안 암호를 선택합니다.

  새 프록시를 생성한 후 기존 프록시 선택을 선택하고 해당 프록시를 선택합니다. 프록시를 연결에 사용하려면 다소 시간이 걸릴 수 있습니다.

6. (선택 사항) 연결 요약을 확장하고 리소스에서 강조 표시된 업데이트를 확인합니다.
7. 설정을 선택합니다.

연결된 컴퓨팅 리소스 보기

AWS Management Console을 사용하여 DB 인스턴스에 연결된 Lambda 함수를 볼 수 있습니다. 표시되는 리소스에는 Amazon RDS가 자동으로 설정한 컴퓨팅 리소스 연결이 포함됩니다.

나열된 컴퓨팅 리소스에는 DB 인스턴스에 수동으로 연결된 컴퓨팅 리소스가 포함되지 않습니다. 예를 들어 데이터베이스와 연결된 VPC 보안 그룹에 규칙을 추가하여 컴퓨팅 리소스가 DB 인스턴스에 수동으로 액세스하도록 허용할 수 있습니다.

콘솔에 Lambda 함수를 나열하려면 다음 조건을 적용해야 합니다.

- 컴퓨팅 리소스와 연결된 보안 그룹의 이름이 `lambda-rds-`*n* 또는 `lambda-rdsproxy-`*n*(*n*은 숫자를 나타냄) 패턴과 일치합니다.
- 컴퓨팅 리소스와 연결된 보안 그룹에 포트 범위가 DB 인스턴스 또는 프록시의 포트로 설정된 아웃바운드 규칙이 있습니다. 아웃바운드 규칙의 대상이 DB 인스턴스 또는 관련 프록시와 연결된 보안 그룹으로 설정되어 있어야 합니다.
- 구성에 프록시가 포함된 경우 데이터베이스와 연결된 프록시에 연결된 보안 그룹의 이름이 `rdsproxy-lambda-`*n*(*n*은 숫자를 나타냄) 패턴과 일치합니다.

- 함수와 연결된 보안 그룹에 포트가 DB 인스턴스 또는 관련 프록시가 사용하는 포트로 설정된 아웃바운드 규칙이 있습니다. 대상이 DB 인스턴스 또는 관련 프록시와 연결된 보안 그룹으로 설정되어 있어야 합니다.

DB 인스턴스에 자동으로 연결된 컴퓨팅 리소스를 보는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택하고 DB 인스턴스를 선택합니다.
3. 연결 및 보안 탭의 연결된 컴퓨팅 리소스에서 컴퓨팅 리소스를 확인합니다.

Amazon RDS DB 인스턴스 수정

추가 스토리지를 더하거나 DB 인스턴스 클래스를 변경하는 것과 같은 작업을 완수하기 위해 DB 인스턴스의 설정을 변경할 수 있습니다. 이 주제에서는 Amazon RDS DB 인스턴스를 수정하는 방법과 DB 인스턴스의 설정에 대해 알아볼 수 있습니다.

프로덕션 인스턴스를 변경하기 전에 테스트 인스턴스에서 변경 사항을 테스트하는 것이 좋습니다. 이렇게 하면 각 변경 사항의 영향을 완전히 이해하는 데 도움이 됩니다. 테스트는 특히 데이터베이스 버전을 업그레이드할 때 중요합니다.

DB 인스턴스에 대한 대부분의 수정 사항은 즉시 적용하거나 다음 유지 관리 기간까지 연기할 수 있습니다. 파라미터 그룹 변경 등의 수정 사항을 적용하려면 DB 인스턴스를 수동으로 재부팅해야 합니다.

> **Important**
>
> 일부 수정 사항을 적용하려면 Amazon RDS가 DB 인스턴스를 재부팅해야 하므로 가동 중지 시간이 발생할 수 있습니다. DB 인스턴스 설정을 수정하기 전에 데이터베이스 및 애플리케이션에 미치는 영향을 확인하십시오.

콘솔

DB 인스턴스를 수정하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 다음 변경하려는 DB 인스턴스를 선택합니다.
3. 수정을 선택합니다. Modify DB instance(DB 인스턴스 수정) 페이지가 나타납니다.
4. 원하는 설정을 모두 변경합니다. 각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하십시오.
5. 원하는 대로 모두 변경되었으면 [Continue]를 선택하고 수정 사항 요약을 확인합니다.
6. (선택 사항) 즉시 적용을 선택하여 변경 내용을 즉시 적용합니다. 일부의 경우 이 옵션을 선택하면 가동 중지 시간이 발생할 수 있습니다. 자세한 내용은 수정 예약 설정 사용 섹션을 참조하세요.
7. 확인 페이지에서 변경 내용을 검토합니다. 변경 내용이 정확할 경우 DB 인스턴스 수정을 선택하여 변경 내용을 저장합니다.

   또는 뒤로를 선택하여 변경 내용을 편집하거나 취소를 선택하여 변경 내용을 취소합니다.

AWS CLI

AWS CLI를 사용하여 DB 인스턴스를 수정하려면 modify-db-instance 명령을 호출합니다. DB 인스턴스 식별자와 수정하려는 옵션 값을 지정합니다. 각 옵션에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하십시오.

Example

다음 코드는 백업 보존 기간을 1주(7일)로 설정하여 mydbinstance를 수정합니다. 이 코드는 --deletion-protection을 사용하여 삭제 방지를 활성화합니다. 삭제 방지를 비활성화하려면 --no-deletion-protection을 사용합니다. 변경 사항은 --no-apply-immediately를 사용하여 다음 유지 관리 기간에 적용됩니다. 변경 사항을 바로 적용하려면 --apply-immediately를 사용합니다. 자세한 내용은 수정 예약 설정 사용 단원을 참조하십시오.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --backup-retention-period 7 \
    --deletion-protection \
    --no-apply-immediately
```

Windows의 경우:

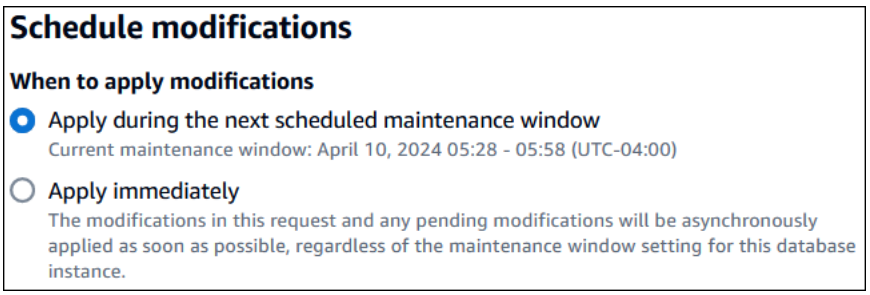
```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --backup-retention-period 7 ^
    --deletion-protection ^
    --no-apply-immediately
```

RDS API

Amazon RDS API를 사용하여 DB 인스턴스를 수정하려면 ModifyDBInstance 작업을 호출합니다. DB 인스턴스 식별자와 수정하려는 설정의 파라미터를 지정합니다. 각 파라미터에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하십시오.

수정 예약 설정 사용

DB 인스턴스를 수정하는 경우 수정 시기를 결정합니다.



다음 유지 관리 기간이 아니라 즉시 변경 사항을 적용하려면 AWS Management Console에서 즉시 적용 옵션을 선택합니다. 또는 AWS CLI를 호출할 때 `--apply-immediately` 파라미터를 사용하거나 Amazon RDS API를 사용할 때 `ApplyImmediately` 파라미터를 `true`로 설정합니다.

변경 사항을 즉시 적용하지 않기로 선택하면 RDS는 보류 중 수정 사항 대기열로 변경 사항을 보냅니다. 다음 유지 관리 기간에 RDS가 대기열에 있는 보류 중 변경 사항을 적용합니다. 변경 사항을 즉시 적용하기로 선택하면 새로운 변경 사항과 보류 중인 수정 사항 대기열에 있는 모든 변경 사항이 적용됩니다.

다음 유지 관리 기간에 보류 중인 수정 사항을 보려면 describe-db-instances AWS CLI 명령을 사용하고 `PendingModifiedValues` 필드를 확인합니다.

> **Important**
>
> 보류 중 수정 사항이 하나라도 적용하려면 DB 인스턴스를 일시적으로 사용 불능 상태(가동 중지 시간)로 만들어야 하는 경우 즉시 적용 옵션을 선택하면 예기치 못한 가동 중지가 발생할 수 있습니다.
>
> 변경 사항을 즉시 적용하도록 선택하면, 보류 중인 수정 사항도 다음 유지 관리 기간이 아니라 즉시 적용됩니다.
>
> 보류 중인 변경 사항을 다음 유지 관리 기간에 적용하지 않으려면 변경 사항을 되돌리도록 DB 인스턴스를 수정하면 됩니다. AWS CLI를 사용하고 `--apply-immediately` 옵션을 지정하여 이 작업을 수행할 수 있습니다.

변경을 지연하기로 선택하더라도 일부 데이터베이스 설정 변경 사항은 즉시 적용됩니다. 다른 데이터베이스 설정이 즉시 적용 설정과 상호 작용하는 방식을 보려면 DB 인스턴스에 대한 설정 단원을 참조하십시오.

DB 인스턴스에 대한 설정

아래 테이블에서 수정할 수 있는 설정과 수정할 수 없는 설정에 대한 세부 정보를 확인할 수 있습니다. 변경 사항을 적용할 수 있는 시기와 변경으로 인해 DB 인스턴스 가동 중지 시간이 발생하는지도 확인

할 수 있습니다. 다중 AZ와 같은 Amazon RDS 기능을 사용하면 나중에 DB 인스턴스를 수정할 때 가동 중지 시간을 최소화할 수 있습니다. 자세한 내용은 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 섹션을 참조하세요.

콘솔, `modify-db-instance` CLI 명령 또는 `ModifyDBInstance` RDS API 작업을 사용하여 DB 인스턴스를 수정할 수 있습니다.

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

할당된 스토리지
 	
DB 인스턴스에 할당할 스토리지(단위: GiB). 할당된 스토리지만 늘릴 수 있습니다. 할당된 스토리지를 줄일 수 없습니다.
 	
일부 이전 DB 인스턴스의 스토리지 및 이전 DB 스냅샷에서 복원된 DB 인스턴스는 수정할 수 없습니다. DB 인스턴스를 사용할 수 없는 경우 콘솔에서 할당된 스토리지 설정이 비활성화됩니다. CLI 명령 describe-valid-db-instance-modifications를 사용하여 더 많은 스토리지를 할당할 수 있는지 여부를 확인할 수 있습니다. 이 명령은 DB 인스턴스에 대해 유효한 스토리지 옵션을 반환합니다.
 	
DB 인스턴스 상태가 storage-optimization인 경우 할당된 스토리지를 수정할 수 없습니다. 지난 6시간 이내에 수정된 경우에도 DB 인스턴스에 할당된 스토리지를 수정할 수 없습니다.
	CLI 옵션:
	
	`--allocated-storage`
	
	RDS API 파라미터:
	
	`AllocatedStorage`
		변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
		
		변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.
			이 변경 도중에는 가동 중지 시간이 발생하지 않습니다. 변경 도중 성능이 저하될 수 있습니다.	모든 DB 엔진

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진
허용되는 최대 스토리지는 DB 엔진과 스토리지 유형에 따라 다릅니다. 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 섹션을 참조하세요.				

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진
아키텍처 구성 여러 테넌트 데이터베이스가 DB 인스턴스에 상주하도록 하는 구성입니다. 현재는 RDS for Oracle 컨테이너 데이터베이스(CDB)만 이 설정을 지원합니다. CDB가 단일 테넌트 구성인 경우 다중 테넌트 구성을 사용하도록 CDB를 수정할 수 있습니다. 이 구성에서는 데이터베이스 에디션 및 필요한 옵션 라이선스에 따라 RDS API를 사용하여 1~30개의 테넌트 데이터베이스를 만들 수 있습니다. 애플리케이션 PDB 및 프록시 PDB는 지원되지 않습니다. 다중 테넌트 구성은 영구적이므로 나중에 CDB를 다시 단일 테넌트 구성으로 변환할 수 없습니다. **Note** Amazon RDS 구성은 Oracle DB 엔진뿐만 아니라 RDS 플랫폼의 기능이기 때문에 '멀티테넌트'가 아닌 '다중 테넌트'라고 합니다. 마찬가지로 RDS 용어 '테넌트'는 Oracle PDB 뿐만 아니라 RDS 구성의	CLI 옵션: `--multi-tenant` (CDB 아키텍처의 다중 테넌트 구성) `--no-multi-tenant`(CDB 아키텍처의 단일 테넌트 구성) API 파라미터: `MultiTenant`	변경 사항이 즉시 적용됩니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.	Oracle

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진
모든 테넌트를 나타냅니다. RDS 설명서에서 하이픈이 포함되지 않은 용어 'Oracle 멀티테넌트'는 온프레미스 및 RDS 배포 모두와 호환되는 Oracle 데이터베이스 CDB 아키텍처만을 의미합니다. 자세한 내용은 RDS for Oracle CDB 개요 섹션을 참조하세요.				

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진
아키텍처 설정 Oracle 데이터베이스의 아키텍처: CDB 또는 비CDB입니다. Oracle 멀티테넌트 아키텍처를 선택하면 RDS for Oracle이 비CDB를 단일 테넌트 구성을 사용하는 CDB로 변환합니다. 이 설정은 데이터베이스가 2021년 4월 이상 RU에서 Oracle Database 19c를 실행하는 비CDB인 경우에만 지원됩니다. 변환 후에는 CDB에 최초의 플러그형 데이터베이스(PDB) 하나가 포함됩니다. 아키텍처 변경은 영구적이므로 CDB를 다시 비CDB로 변환할 수 없습니다. **Note** 단일 테넌트 구성의 CDB를 다중 테넌트 구성으로 변환하려면 CDB 인스턴스를 다시 수정하고 아키텍처 구성으로 다중 테넌트 구성을 선택하세요. 자세한 내용은 CDB 아키텍처의 단일 테넌트 구성 섹션을 참조하세요.	CLI 옵션: `--engine oracle-ee-cdb`(Oracle 멀티테넌트) `--engine oracle-se2-cdb` (Oracle 멀티테넌트) API 파라미터: `Engine`	변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다. 변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.	이 변경 도중에는 가동 중지 시간이 발생합니다.	Oracle

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

자동 마이너 버전 업그레이드
 	
DB 엔진의 기본 마이너 버전 업그레이드가 제공되면 DB 인스턴스가 자동으로 이를 받을 수 있게 하려면 마이너 버전 자동 업그레이드 사용을 선택합니다. 이는 기본 설정 동작입니다. Amazon RDS는 유지 관리 기간에 자동 마이너 버전 업그레이드를 수행합니다. 마이너 버전 자동 업그레이드 사용을 선택하지 않으면 새 마이너 버전이 출시될 때 DB 인스턴스가 자동으로 업그레이드되지 않습니다.
 	
자세한 내용은 마이너 엔진 버전 자동 업그레이드 섹션을 참조하세요.
	CLI 옵션:
	
	`--auto-minor-version-upgrade | --no-auto-minor-version-upgrade`
	
	RDS API 파라미터:
	
	`AutoMinorVersionUpgrade`
		변경 사항이 즉시 적용됩니다. 이 설정은 즉시 적용 설정을 무시합니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.	모든 DB 엔진

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진
백업 복제 재해 복구를 위해 추가 리전에 백업을 생성하려면 다른 AWS 리전으로의 복제 활성화(Enable replication to another Region)를 선택합니다. 그런 다음 추가 백업을 위한 대상 리전을 선택합니다.	DB 인스턴스를 수정할 때는 사용할 수 없습니다. AWS CLI 또는 RDS API를 사용하여 교차 리전 백업을 활성화하는 방법에 대한 자세한 내용은 교차 리전 자동 백업 활성화 섹션을 참조하세요.	비동기 방식이지만 최대한 빠른 시간 내에 변경 사항이 적용됩니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.	모든 DB 엔진

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

백업 보관 기간
 	
자동 백업을 보존할 일수. 자동 백업을 비활성화하기 위해 백업 보존 기간을 0으로 설정합니다.
 	
자세한 내용은 백업 소개 섹션을 참조하세요.
 	
Note
AWS Backup을 사용하여 백업을 관리하는 경우 이 옵션이 적용되지 않습니다. AWS Backup에 대한 자세한 내용은 AWS 백업 개발자 안내서를 참조하세요.
	CLI 옵션:
	
	`--backup-retention-period`
	
	RDS API 파라미터:
	
	`BackupRetentionPeriod`
		변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
		
		변경 사항을 즉시 적용하도록 선택하지 않은 경우 0이 아닌 값에서 다른 0이 아닌 값으로 변경하면 변경 사항이 최대한 빠른 시간 내에 비동기식으로 적용됩니다. 이때 적용되지 않을 경우, 다음 유지 관리 기간에 변경 사항이 적용됩니다.
			백업 보존 기간을 0에서 0이 아닌 값으로 또는 0이 아닌 값에서 0으로 변경할 경우 가동 중지 시간이 발생합니다.
			
			이는 단일 AZ 및 다중 AZ DB 인스턴스 모두에 적용됩니다.
				모든 DB 엔진

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

백업 기간
 	
데이터베이스의 자동 백업이 실행되는 기간. 백업 기간은 국제 표준시(UTC)의 시작 시간과 시간 단위의 지속 기간으로 구성됩니다.
 	
자세한 내용은 백업 소개 섹션을 참조하세요.
 	
Note
AWS Backup을 사용하여 백업을 관리하는 경우 이 옵션이 표시되지 않습니다. AWS Backup에 대한 자세한 내용은 AWS Backup 개발자 안내서를 참조하세요.
	CLI 옵션:
	
	`--preferred-backup-window`
	
	RDS API 파라미터:
	
	`PreferredBackupWindow`
		비동기 방식이지만 최대한 빠른 시간 내에 변경 사항이 적용됩니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.	모든 DB 엔진

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

인증 기관
 	
DB 인스턴스에서 사용하는 서버 인증서의 CA(인증 기관)입니다.
 	
자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 섹션을 참조하세요.
	CLI 옵션:
	
	`--ca-certificate-identifier`
	
	RDS API 파라미터:
	
	`CACertificateIdentifier`
		변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
		
		변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.
			가동 중단 시간은 DB 엔진이 재시작 없는 교체를 지원하지 않는 경우에만 발생합니다. describe-db-engine-versions AWS CLI 명령을 사용하여 DB 엔진이 재시작 없는 교체를 지원하는지 여부를 확인할 수 있습니다.	모든 DB 엔진

스냅샷으로 태그 복사
 	
DB 인스턴스 태그가 있는 경우 이 옵션을 사용하면 DB 스냅샷을 만들 때 해당 태그를 복사할 수 있습니다.
 	
자세한 내용은 Amazon RDS 리소스에 태그 지정 섹션을 참조하세요.
	CLI 옵션:
	
	`--copy-tags-to-snapshot` 또는 `--no-copy-tags-to-snapshot`
	
	RDS API 파라미터:
	
	`CopyTagsToSnapshot`
		변경 사항이 즉시 적용됩니다. 이 설정은 즉시 적용 설정을 무시합니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.	모든 DB 엔진

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

데이터베이스 포트
 	
DB 인스턴스에 액세스하는 데 사용할 포트입니다.
 	
포트 값은 DB 인스턴스와 연결된 옵션 그룹에서 옵션에 대해 지정한 포트 값과 일치하지 않아야 합니다.
 	
자세한 내용은 Amazon RDS DB 인스턴스에 연결 섹션을 참조하세요.
	CLI 옵션:
	
	`--db-port-number`
	
	RDS API 파라미터:
	
	`DBPortNumber`
		변경 사항이 즉시 적용됩니다. 이 설정은 즉시 적용 설정을 무시합니다.	DB 인스턴스가 즉시 재부팅됩니다.	모든 DB 엔진

DB 엔진 버전
 	
사용하려는 DB 엔진의 버전. 프로덕션 DB 인스턴스를 업그레이드하려면 먼저 테스트 DB 인스턴스에서 업그레이드 프로세스를 테스트하는 것이 좋습니다. 이렇게 하면 기간을 확인하고 애플리케이션을 검증하는 데 도움이 됩니다.
 	
자세한 내용은 DB 인스턴스 엔진 버전 업그레이드 섹션을 참조하세요.
	CLI 옵션:
	
	`--engine-version`
	
	RDS API 파라미터:
	
	`EngineVersion`
		변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
		
		변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.
			이 변경 도중에는 가동 중지 시간이 발생합니다.	모든 DB 엔진

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

DB 인스턴스 클래스
 	
사용할 DB 인스턴스 클래스.
 	
자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요.
	CLI 옵션:
	
	`--db-instance-class`
	
	RDS API 파라미터:
	
	`DBInstanceClass`
		변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
		
		변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.
			이 변경 도중에는 가동 중지 시간이 발생합니다.	모든 DB 엔진

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

DB 인스턴스 식별자
 	
새 DB 인스턴스 식별자입니다. 이 값은 소문자 문자열로 저장됩니다.
 	
DB 인스턴스 이름 바꾸기의 영향에 대한 자세한 내용은 DB 인스턴스 이름 변경 단원을 참조하십시오.
	CLI 옵션:
	
	`--new-db-instance-identifier`
	
	RDS API 파라미터:
	
	`NewDBInstanceIdentifier`
		변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
		
		변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.
			DB 엔진 버전이 동적 SSL 업로드를 지원하지 않으면 이 변경 중에 가동 중지 시간이 발생합니다. 사용 중인 버전에 재시작이 필요한지 확인하려면 다음 AWS CLI 명령을 실행하세요.
			
			`aws rds describe-db-engine-versions \ --default-only \ --engine your-db-engine \ --query 'DBEngineVersions[*].SupportsCertificateRotationWithoutRestart'`
				모든 DB 엔진

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

DB 파라미터 그룹
 	
DB 인스턴스와 연결하려는 DB 파라미터 그룹.
 	
자세한 내용은 Amazon RDS의 파라미터 그룹 섹션을 참조하세요.
	CLI 옵션:
	
	`--db-parameter-group-name`
	
	RDS API 파라미터:
	
	`DBParameterGroupName`
		새 DB 파라미터 그룹을 DB 인스턴스와 연결하는 작업은 즉시 이루어집니다.	새 DB 파라미터 그룹을 DB 인스턴스에 연결하면 가동 중지가 발생하지 않습니다.
			
			DB 파라미터 그룹의 연결은 파라미터 그룹 내에서 파라미터 변경을 적용하는 것과는 다릅니다. RDS는 DB 인스턴스를 수동으로 재부팅한 후에만 새로 연결된 그룹에 수정된 정적 및 동적 파라미터 설정을 적용합니다. 그러나 DB 파라미터 그룹을 DB 인스턴스에 연결한 후 DB 파라미터 그룹에서 동적 파라미터를 수정하면 이러한 파라미터 설정이 재부팅
				모든 DB 엔진

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진
			없이 즉시 적용됩니다. 자세한 내용은 Amazon RDS의 파라미터 그룹 및 DB 인스턴스 재부팅 단원을 참조하십시오.	

전용 로그 볼륨
 	
전용 로그 볼륨(DLV)을 사용하여 데이터베이스 테이블이 들어 있는 볼륨과 분리된 스토리지 볼륨에 데이터베이스 트랜잭션 로그를 저장합니다.
 	
자세한 내용은 전용 로그 볼륨(DLV) 사용 섹션을 참조하세요.
	CLI 옵션:
	
	`-dedicated-log-volume`
	
	RDS API 파라미터:
	
	`DedicatedLogVolume`
		DB 인스턴스가 재부팅되면 변경 내용이 적용됩니다.	DB 인스턴스가 재부팅되는 동안 가동 중지가 발생합니다.	MariaDB, MySQL, PostgreSQL

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

삭제 방지
 	
DB 인스턴스가 삭제되지 않도록 방지하려면, 삭제 방지를 활성화합니다.
 	
자세한 내용은 DB 인스턴스 삭제 섹션을 참조하세요.
	CLI 옵션:
	
	`--deletion-protection|--no-deletion-protection`
	
	RDS API 파라미터:
	
	`DeletionProtection`
		변경 사항이 즉시 적용됩니다. 이 설정은 즉시 적용 설정을 무시합니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.	모든 DB 엔진

확장 모니터링
 	
DB 인스턴스가 실행되는 운영 체제에 대한 실시간 측정치 수집을 활성화하려면 확장 모니터링 활성화을 선택합니다.
 	
자세한 내용은 Enhanced Monitoring을 사용하여 OS 지표 모니터링 섹션을 참조하세요.
	CLI 옵션:
	
	`--monitoring-interval` , , 및 `--monitoring-role-arn`
	
	RDS API 파라미터:
	
	`MonitoringInterval` , , 및 `MonitoringRoleArn`
		변경 사항이 즉시 적용됩니다. 이 설정은 즉시 적용 설정을 무시합니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.	모든 DB 엔진

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

IAM DB 인증
 	
IAM DB 인증을 활성화하여 사용자 및 역할을 통해 데이터베이스 사용자를 인증합니다.
 	
자세한 내용은 MariaDB, MySQL 및 PostgreSQL IAM 데이터베이스 인증 섹션을 참조하세요.
	CLI 옵션:
	
	`--enable-iam-database-authentication|--no-enable-iam-database-authentication`
	
	RDS API 파라미터:
	
	`EnableIAMDatabaseAuthentication`
		변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
		
		변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.
			이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.	MariaDB, MySQL, PostgreSQL만

Kerberos 인증
 	
DB 인스턴스를 이동할 Active Directory를 선택합니다. 디렉터리는 이 작업 이전에 존재해야 합니다. 이미 디렉터리를 선택한 경우 없음을 지정하여 현재 디렉터리에서 DB 인스턴스를 제거할 수 있습니다.
 	
자세한 내용은 Kerberos 인증 섹션을 참조하세요.
	CLI 옵션:
	
	`--domain`, , 및 `--domain-iam-role-name`
	
	RDS API 파라미터:
	
	`Domain`, , 및 `DomainIAMRoleName`
		변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
		
		변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.
			이 변경 도중에는 짧은 가동 중지 시간이 발생합니다.	Microsoft SQL Server, MySQL, Oracle 및 PostgreSQL에만 해당

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

라이선스 모델
 	
Db2 및 Oracle용 라이선스를 사용하려면 기존 보유 라이선스 사용을 선택합니다.
 	
라이선스 포함을 선택하여 Microsoft SQL Server 또는 Oracle에 대한 일반 라이선스 계약을 사용합니다.
 	
자세한 내용은 Amazon RDS for Db2 라이선스 옵션, Amazon RDS의 Microsoft SQL Server 라이선싱, RDS for Oracle 라이선스 옵션 섹션을 참조하세요.
	CLI 옵션:
	
	`--license-model`
	
	RDS API 파라미터:
	
	`LicenseModel`
		변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
		
		변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.
			이 변경 도중에는 가동 중지 시간이 발생합니다.	Microsoft SQL Server 및 Oracle에만 해당

로그 내보내기
 	
Amazon CloudWatch Logs에 게시할 데이터베이스 로그 파일의 유형입니다.
 	
자세한 내용은 Amazon CloudWatch Logs에 데이터베이스 로그 게시 섹션을 참조하세요.
	CLI 옵션:
	
	`--cloudwatch-logs-export-configuration`
	
	RDS API 파라미터:
	
	`CloudwatchLogsExportConfiguration`
		변경 사항이 즉시 적용됩니다. 이 설정은 즉시 적용 설정을 무시합니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.	모든 DB 엔진

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

유지보수 윈도우
 	
시스템 유지 관리를 실행하는 기간. 시스템 유지 관리는 업그레이드를 포함합니다(해당할 경우). 유지 관리 기간은 국제 표준시(UTC)의 시작 시간과 시간 단위의 지속 기간으로 구성됩니다.
 	
이 기간을 현재 시간으로 설정하는 경우 현재 시간과 기간 종료 시간 사이에 최소 30분의 간격이 필요합니다. 이 간격을 두면 보류 중인 모든 변경 사항이 적용되는 데 도움이 됩니다.
 	
자세한 내용은 Amazon RDS 유지 관리 기간 섹션을 참조하세요.
	CLI 옵션:
	
	`--preferred-maintenance-window`
	
	RDS API 파라미터:
	
	`PreferredMaintenanceWindow`
		변경 사항이 즉시 적용됩니다. 이 설정은 즉시 적용 설정을 무시합니다.	가동 중지를 유발할 수 있는 작업이 하나 이상 보류 중이고, 유지 관리 기간이 현재 시간을 포함하여 변경된 경우 보류 중인 작업이 즉시 적용되고 가동 중지 시간이 발생합니다.	모든 DB 엔진

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

AWS Secrets Manager에서 마스터 자격 증명 관리
 	
Secrets Manager에서 보안 암호의 마스터 사용자 암호를 관리하려면 AWS Secrets Manager에서 마스터 자격 증명 관리를 선택합니다.
 	
원하는 경우 보안 암호를 보호하는 데 사용할 KMS 키를 선택합니다. 자신의 계정에서 KMS 키를 선택하거나 다른 계정의 키를 입력할 수 있습니다.
 	
RDS에서 이미 DB 인스턴스의 마스터 사용자 암호를 관리하고 있는 경우 보안 암호 즉시 교체를 선택하여 마스터 사용자 암호를 교체할 수 있습니다.
 	
자세한 내용은 Amazon RDS 및 AWS Secrets Manager를 통한 암호 관리 섹션을 참조하세요.
	CLI 옵션:
	
	`--manage-master-user-password | --no-manage-master-user-password`
	
	`--master-user-secret-kms-key-id`
	
	`--rotate-master-user-password | --no-rotate-master-user-password`
	
	RDS API 파라미터:
	
	`ManageMasterUserPassword`
		자동 마스터 사용자 암호 관리를 켜거나 해제하면 변경 내용이 즉시 적용됩니다. 이 변경은 즉시 적용 설정을 무시합니다.
		
		마스터 사용자 암호를 교체하는 경우 변경 내용이 즉시 적용되도록 지정해야 합니다.
			이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.	모든 DB 엔진

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

	`MasterUserSecretKmsKeyId`
	
	`RotateMasterUserPassword`			

다중 AZ 배포
 	
다중 가용 영역에 DB 인스턴스를 배포하려면 예를 선택합니다. 그렇지 않으면 아니요입니다.
 	
자세한 내용은 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 섹션을 참조하세요.
	CLI 옵션:
	
	`--multi-az|--no-multi-az`
	
	RDS API 파라미터:
	
	`MultiAZ`
		변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
		
		변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.
			이 변경 도중에는 가동 중지 시간이 발생하지 않습니다. 그러나 성능에 영향을 줄 수 있습니다. 자세한 내용은 DB 인스턴스를 Amazon RDS용 다중 AZ 배포로 변환 단원을 참조하십시오.	모든 DB 엔진

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

네트워크 유형
 	
DB 인스턴스에서 지원하는 IP 주소 지정 프로토콜.
 	
IPv4 리소스가 인터넷 프로토콜 버전 4 (IPv4) 주소 지정 프로토콜을 통해서만 DB 인스턴스와 통신할 수 있도록 지정합니다.
 	
듀얼 스택 모드리소스가 IPv4, 인터넷 프로토콜 버전 6 (IPv6) 또는 둘 다를 통해 DB 인스턴스와 통신할 수 있도록 지정합니다. IPv6 주소 지정 프로토콜을 통해 DB 인스턴스와 통신해야 하는 리소스가 있는 경우 이중 스택 모드를 사용합니다. 또한 IPv6 CIDR 블록을 지정한 DB 서브넷 그룹의 모든 서브넷과 연결해야 합니다.
 	
자세한 내용은 Amazon RDS IP 주소 지정 단원을 참조하십시오.
	CLI 옵션:
	
	`--network-type`
	
	RDS API 파라미터:
	
	`NetworkType`
		변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
		
		변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.
			이 변경 도중 가동 중지 시간이 발생할 수 있습니다.	모든 DB 엔진

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

새 마스터 암호
 	
마스터 사용자의 암호. 암호는 8–41자의 영숫자 문자로 구성되어야 합니다.
	CLI 옵션:
	
	`--master-user-password`
	
	RDS API 파라미터:
	
	`MasterUserPassword`
		비동기 방식이지만 최대한 빠른 시간 내에 변경 사항이 적용됩니다. 이 설정은 즉시 적용 설정을 무시합니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.	모든 DB 엔진

옵션 그룹
 	
DB 인스턴스와 연결할 옵션 그룹.
 	
자세한 내용은 옵션 그룹 작업 섹션을 참조하세요.
	CLI 옵션:
	
	`--option-group-name`
	
	RDS API 파라미터:
	
	`OptionGroupName`
		변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
		
		변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.
			이 변경 도중에는 가동 중지 시간이 발생하지 않습니다. 한 가지 예외는 RDS for MariaDB 또는 RDS for MySQL DB 인스턴스에 MariaDB Audit 플러그인을 추가하는 것입니다. 이로 인해 운영 중단이 발생할 수 있습니다.	모든 DB 엔진

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

성능 개선 도우미
 	
데이터베이스 성능을 분석하고 문제를 해결할 수 있도록 DB 인스턴스 로드를 모니터링하려면 Performance Insights 활성화를 선택합니다.
 	
일부 DB 엔진 버전 및 DB 인스턴스 클래스에서는 성능 개선 도우미를 사용할 수 없습니다. DB 인스턴스에 사용할 수 없는 경우 성능 개선 도우미 섹션이 콘솔에 나타나지 않습니다.
 	
자세한 내용은 성능 개선 도우미를 통한 Amazon RDS 모니터링 및 Amazon RDS DB 엔진, 리전 및 인스턴스 클래스 Performance Insights 지원 단원을 참조하십시오.
	CLI 옵션:
	
	`--enable-performance-insights| --no-enable-performance-insights`
	
	RDS API 파라미터:
	
	`EnablePerformanceInsights`
		변경 사항이 즉시 적용됩니다. 이 설정은 즉시 적용 설정을 무시합니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.	Db2를 제외한 모든 항목

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

성능 개선 도우미 AWS KMS key
 	
성능 개선 도우미 데이터의 암호화를 위한 AWS KMS key의 AWS KMS 키 식별자입니다. 키 식별자는 Amazon 리소스 이름(ARN), AWS KMS 키 식별자 또는 KMS 키의 키 별칭입니다.
 	
자세한 내용은 Amazon RDS의 성능 개선 도우미 설정 및 해제 단원을 참조하십시오.
	CLI 옵션:
	
	`--performance-insights-kms-key-id`
	
	RDS API 파라미터:
	
	`PerformanceInsightsKMSKeyId`
		변경 사항이 즉시 적용됩니다. 이 설정은 즉시 적용 설정을 무시합니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.	Db2를 제외한 모든 항목

성능 개선 도우미 보존 기간
 	
성능 개선 도우미 데이터를 보관하는 시간(일). 프리 티어의 보존 설정은 기본값(7일)입니다. 성능 데이터를 더 오래 보존하려면 1~24개월을 지정하십시오. 보존 기간에 대한 자세한 내용은 성능 개선 도우미의 요금 및 데이터 보존 단원을 참조하십시오.
 	
자세한 내용은 Amazon RDS의 성능 개선 도우미 설정 및 해제 단원을 참조하십시오.
	CLI 옵션:
	
	`--performance-insights-retention-period`
	
	RDS API 파라미터:
	
	`PerformanceInsightsRetentionPeriod`
		변경 사항이 즉시 적용됩니다. 이 설정은 즉시 적용 설정을 무시합니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.	Db2를 제외한 모든 항목

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

프로세서 기능
 	
DB 인스턴스의 DB 인스턴스 클래스에 대한 CPU 코어 수와 코어당 스레드 수입니다.
 	
자세한 내용은 RDS for Oracle에서 DB 인스턴스 클래스의 프로세서 구성 섹션을 참조하세요.
	CLI 옵션:
	
	`--processor-features` , , 및 `--use-default-processor-features` | `--no-use-default-processor-features`
	
	RDS API 파라미터:
	
	`ProcessorFeatures` , , 및 `UseDefaultProcessorFeatures`
		변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
		
		변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.
			이 변경 도중에는 가동 중지 시간이 발생합니다.	Oracle에만 해당

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

프로비저닝된 IOPS
 	
DB 인스턴스의 프로비저닝된 IOPS(초당 I/O 작업 수) 값입니다. 이 설정은 스토리지 유형에서 다음 중 하나를 선택한 경우에만 사용할 수 있습니다.
 	
• 범용 SSD(gp3)
• 프로비저닝된 IOPS SSD(io1)
• 프로비저닝된 IOPS SSD(io2)
 	
자세한 내용은 the section called “프로비저닝된 IOPS 스토리지” 및 the section called “gp3 스토리지(권장)” 단원을 참조하십시오.
	CLI 옵션:
	
	`--iops`
	
	RDS API 파라미터:
	
	`Iops`
		변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
		
		변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.
			이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.	모든 DB 엔진

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

공개 액세스(Public access)
 	
DB 인스턴스에 퍼블릭 IP 주소를 부여하려면(즉 VPC 외부에서 액세스할 수 있음) 공개적으로 액세스할 수 있음(Publicly accessible)을 선택합니다. 공개적으로 액세스가 가능하려면 DB 인스턴스도 VPC의 퍼블릭 서브넷에 있어야 합니다.
 	
VPC 내부에서만 DB 인스턴스에 액세스할 수 있게 하려면 공개적으로 액세스할 수 없음(Not publicly accessible)을 선택합니다.
 	
자세한 내용은 VPC에 있는 DB 인스턴스를 인터넷에서 숨기기 섹션을 참조하세요.
 	
VPC 외부에서 DB 인스턴스에 연결하려면 DB 인스턴스에 공개적으로 액세스할 수 있어야 합니다. 또한 DB 인스턴스 보안 그룹의 인바운드 규칙을 사용하여 액세스 권한을 부여해야 하며, 다른 요구 사항도 충족해야 합니다. 자세한 내용은 Amazon RDS DB 인스턴스에 연결할 수 없음 섹션을 참조하세요.
	CLI 옵션:
	
	`--publicly-accessible | --no-publicly-accessible`
	
	RDS API 파라미터:
	
	`PubliclyAccessible`
		변경 사항이 즉시 적용됩니다. 이 설정은 즉시 적용 설정을 무시합니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.	모든 DB 엔진

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진
DB 인스턴스에 공개적으로 액세스할 수 없는 경우 AWS Site-to-Site VPN 연결 또는 AWS Direct Connect 연결을 사용하여 프라이빗 네트워크에서 액세스할 수도 있습니다. 자세한 내용은 인터네트워크 트래픽 개인 정보 섹션을 참조하세요.				

보안 그룹
 	
DB 인스턴스와 연결할 VPC 보안 그룹입니다.
 	
자세한 내용은 보안 그룹을 통한 액세스 제어 섹션을 참조하세요.
	CLI 옵션:
	
	`--vpc-security-group-ids`
	
	RDS API 파라미터:
	
	`VpcSecurityGroupIds`
		비동기 방식이지만 최대한 빠른 시간 내에 변경 사항이 적용됩니다. 이 설정은 즉시 적용 설정을 무시합니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.	모든 DB 엔진

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

Storage autoscaling(스토리지 Autoscaling)
 	
Enable storage autoscaling(스토리지 Autoscaling 활성화)를 사용하면 필요할 경우 Amazon RDS가 스토리지를 자동으로 확장하여 DB 인스턴스의 스토리지 공간이 부족해지는 문제를 방지할 수 있습니다.
 	
Maximum storage threshold(최대 스토리지 임계값)를 사용하여 Amazon RDS가 DB 인스턴스의 스토리지를 자동으로 확장할 수 있는 상한선을 설정합니다. 기본값은 1,000GiB입니다.
 	
자세한 내용은 Amazon RDS 스토리지 Autoscaling을 사용한 용량 자동 관리 섹션을 참조하세요.
	CLI 옵션:
	
	`--max-allocated-storage`
	
	RDS API 파라미터:
	
	`MaxAllocatedStorage`
		변경 사항이 즉시 적용됩니다. 이 설정은 즉시 적용 설정을 무시합니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.	모든 DB 엔진

메시지 처리량(throughput)
 	
DB 인스턴스의 새 스토리지 처리량(throughput) 값입니다. 이 설정은 스토리지 유형으로 범용 SSD(gp3)를 선택한 경우에만 사용할 수 있습니다.
 	
자세한 내용은 the section called “gp3 스토리지(권장)” 섹션을 참조하세요.
	CLI 옵션:
	
	`--storage-throughput`
	
	RDS API 파라미터:
	
	`StorageThroughput`
		변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
		
		변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.
			이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.	모든 DB 엔진

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

스토리지 유형
 	
사용할 스토리지 유형.
 	
범용 SSD(gp3)를 선택하면 고급 설정에서 프로비저닝된 IOPS 및 스토리지 처리량(throughput)을 추가로 프로비저닝할 수 있습니다.
 	
프로비저닝된 IOPS SSD(io1) 또는 프로비저닝된 IOPS SSD(io2)를 선택하는 경우 프로비저닝된 IOPS 값을 입력합니다.
 	
Amazon RDS가 DB 인스턴스를 수정하여 스토리지 크기나 유형을 변경하면 이후 6시간 동안은 스토리지 크기, 성능 또는 유형을 변경하는 또 다른 요청을 제출할 수 없습니다.
 	
자세한 내용은 Amazon RDS 스토리지 유형 섹션을 참조하세요.
	CLI 옵션:
	
	`--storage-type`
	
	RDS API 파라미터:
	
	`StorageType`
		변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
		
		변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.
			다음과 같이 변경하면 프로세스가 시작되는 동안 짧은 가동 중지 시간이 발생합니다. 그 이후에 변경 사항이 적용되는 동안 데이터베이스를 정상적으로 사용할 수 있습니다.
			
			• 범용(SSD) 또는 프로비저닝된 IOPS(SSD)에서 Magnetic으로 내보냅니다.
			• Magnetic 또는 범용(SSD)에서 프로비저닝된 IOPS(SSD)로 내보냅니다.
				모든 DB 엔진

콘솔 설정 및 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항	지원되는 DB 엔진

DB 서브넷 그룹
 	
DB 인스턴스에 대한 서브넷 그룹입니다. 이 설정을 사용하여 DB 인스턴스를 다른 VPC로 이동할 수 있습니다.
 	
자세한 내용은 Amazon VPC 및 Amazon RDS 단원을 참조하십시오.
	CLI 옵션:
	
	`--db-subnet-group-name`
	
	RDS API 파라미터:
	
	`DBSubnetGroupName`
		변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
		
		변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.
			이 변경 도중에는 가동 중지 시간이 발생합니다.	모든 DB 엔진

DB 인스턴스 유지 관리

Amazon RDS는 Amazon RDS 리소스를 정기적으로 유지 관리합니다.

주제

- DB 인스턴스 유지 관리 업데이트 개요
- 보류 중인 유지 관리 업데이트 보기
- DB 인스턴스에 업데이트 적용
- 다중 AZ 배포 유지
- Amazon RDS 유지 관리 기간
- 기본 DB 인스턴스 유지 관리 기간 조정
- RDS DB 인스턴스의 운영 체제 업데이트

DB 인스턴스 유지 관리 업데이트 개요

유지 관리에는 주로 DB 인스턴스의 다음 리소스에 대한 업데이트가 포함됩니다.

- 기본 하드웨어
- 기본 운영 체제(OS)
- 데이터베이스 엔진 버전

운영 체제 업데이트는 보안상 가장 빈번하게 발생하며 가능한 한 빨리 수행하는 것이 좋습니다. 운영 체제 업데이트에 대한 자세한 내용은 DB 인스턴스에 업데이트 적용 섹션을 참조하세요.

주제

- 유지 관리 업데이트 중 오프라인 리소스
- 연기된 DB 인스턴스 수정 사항
- DescribePendingMaintenanceActions API의 최종 일관성

유지 관리 업데이트 중 오프라인 리소스

일부 유지 관리 항목을 사용하려면 Amazon RDS에서 DB 인스턴스를 잠시 동안 오프라인 상태로 전환해야 합니다. 리소스가 오프라인 상태에 있어야 하는 유지 관리 항목에는 필수 운영 체제 또는 데이터베이스 패칭이 포함됩니다. 이때 보안 및 인스턴스 안정성과 관련된 패치에 한해 필수 패치 작업으로

자동 예약됩니다. 이러한 패치 작업은 드물게 발생하며 일반적인 빈도는 몇 개월에 한 번입니다. 대부분 유지 관리 기간의 일부만 필요합니다.

연기된 DB 인스턴스 수정 사항

즉시 적용되지 않도록 연기한 DB 인스턴스 수정은 유지 관리 기간에 적용됩니다. 예를 들어 DB 인스턴스 클래스 또는 파라미터 그룹을 유지 관리 기간에 변경하도록 선택할 수 있습니다. 대기 중인 재부팅 설정을 사용하여 지정한 수정 사항은 대기 중인 유지 관리 목록에 표시되지 않습니다. DB 인스턴스 수정에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

다음 유지 관리 기간에 보류 중인 수정 사항을 보려면 describe-db-instances AWS CLI 명령을 사용하고 `PendingModifiedValues` 필드를 확인합니다.

DescribePendingMaintenanceActions API의 최종 일관성

Amazon RDS `DescribePendingMaintenanceActions` API는 최종 일관성 모델을 따릅니다. 즉, `DescribePendingMaintenanceActions` 명령의 결과가 모든 후속 RDS 명령에 즉시 표시되지 않을 수 있습니다. 이전 API 명령을 사용한 즉시 `DescribePendingMaintenanceActions`를 사용할 때는 이 점을 염두에 두세요.

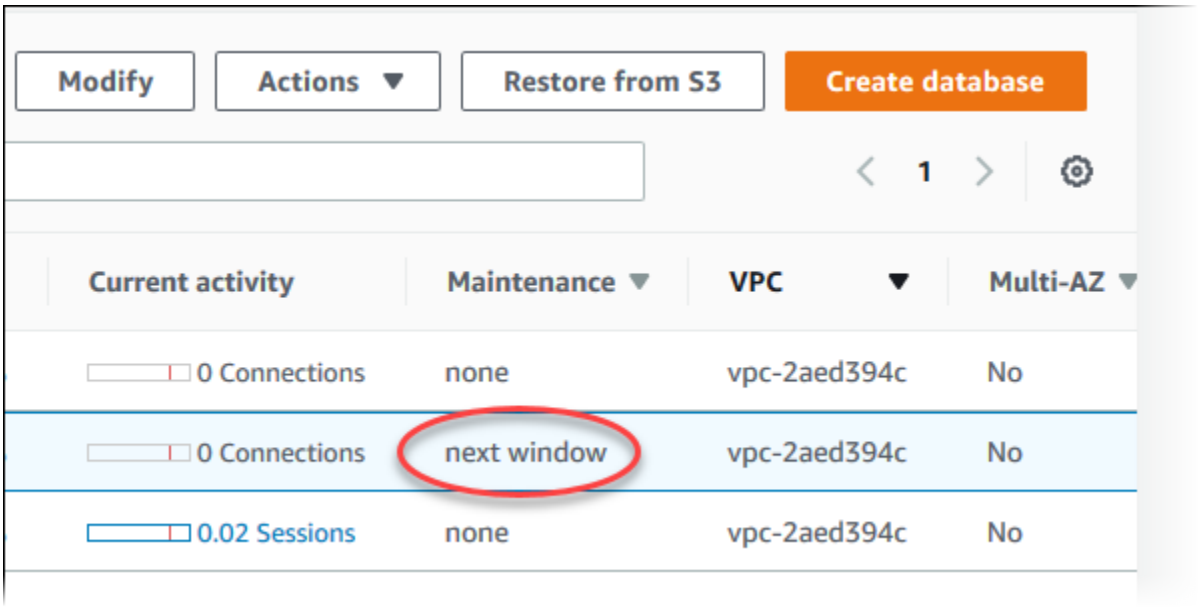
최종 일관성은 유지 관리 업데이트를 관리하는 방식에 영향을 미칠 수 있습니다. 예를 들어 `ApplyPendingMaintenanceActions` 명령을 실행하여 DB 인스턴스의 데이터베이스 엔진 버전을 업데이트하면 `DescribePendingMaintenanceActions`가 결국에는 해당 버전을 볼 수 있게 됩니다. 이 시나리오에서 `DescribePendingMaintenanceActions`는 유지 관리 작업이 적용되었음에도 불구하고 적용되지 않았다고 표시할 수 있습니다.

최종 일관성을 관리하려면 다음을 수행할 수 있습니다.

- 수정 명령을 실행하기 전에 DB 인스턴스의 상태를 확인합니다. 이전 명령이 시스템에 전파될 시간이 충분하도록 지수 백오프 알고리즘을 사용하여 적절한 `DescribePendingMaintenanceActions` 명령을 실행합니다. 이 작업을 수행하려면 몇 초의 대기 시간부터 시작하여 대기 시간을 5분까지 점진적으로 늘려가며 `DescribePendingMaintenanceActions` 명령을 반복적으로 실행합니다.
- `DescribePendingMaintenanceActions` 명령이 정확한 응답을 반환하더라도 후속 명령 사이에 대기 시간을 추가합니다. 몇 초의 대기 시간으로 시작하는 지수 백오프 알고리즘을 적용하고 대기 시간을 약 5분까지 점진적으로 늘립니다.

보류 중인 유지 관리 업데이트 보기

RDS 콘솔, AWS CLI 또는 RDS API를 사용하여 DB 인스턴스에 대해 유지 관리 업데이트를 사용할 수 있는지 확인합니다. 업데이트가 있는 경우에는 다음과 같이 Amazon RDS 콘솔에서 DB 인스턴스의 유지 관리 열에 사용 가능 여부가 표시됩니다.



DB 인스턴스에 대해 유지 관리 업데이트가 제공되지 않는 경우 그에 대한 열 값은 없음입니다.

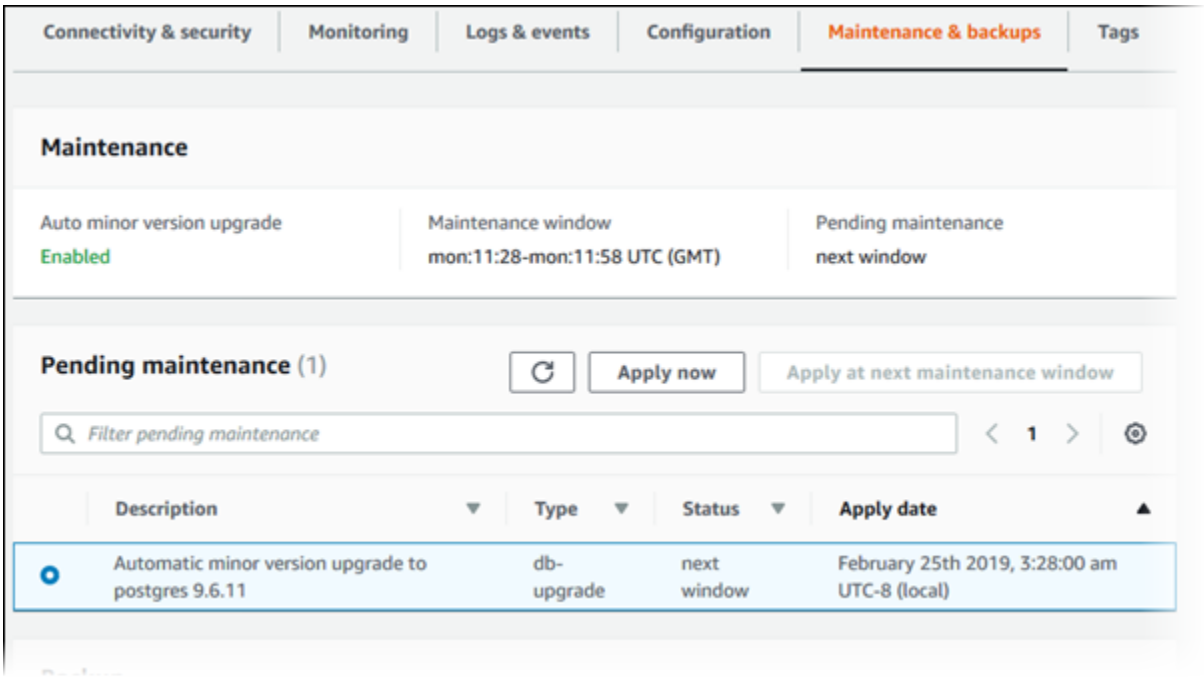
DB 인스턴스에 대해 유지 관리 업데이트가 제공되는 경우 다음과 같은 열 값이 가능합니다.

- 필수 – 유지 관리 작업은 리소스에 적용되며 무기한 보류할 수 없습니다.
- 사용 가능 – 유지 관리 작업을 사용할 수 있습니다. 그러나 리소스에 자동으로 적용되지 않고 수동으로 적용할 수 있습니다.
- 다음 기간 – 유지 관리 작업은 다음 유지 관리 기간 중에 리소스에 적용됩니다.
- 진행 중 – 유지 관리 작업이 리소스에 적용되고 있는 중입니다.

업데이트가 있을 경우에는 다음 테이블의 작업 중 하나를 실행할 수 있습니다.

- 유지 관리 값이 다음 기간인 경우 작업에서 업그레이드 보류를 선택하여 유지 관리 항목을 보류하십시오. 유지 관리 작업이 이미 시작된 경우에는 보류할 수 없습니다.
- 유지 관리 항목을 즉시 적용합니다.
- 다음 유지 관리 기간 중 시작할 유지 관리 항목을 예약합니다.
- 작업이 없습니다.

조치를 취하려면 DB 인스턴스 를 선택하여 세부 정보를 표시한 후 Maintenance & backups(유지 관리 및 백업)을 선택하십시오. 그러면 보류 중인 유지 관리 항목이 표시됩니다.



유지 관리 기간에 따라 대기 중인 작업의 시작 시기가 결정되지만 이러한 작업의 전체 실행 시간이 줄어들지는 않습니다. 유지 관리 기간에 끝나기 전에 반드시 유지 관리 작업이 끝나도록 되어 있는 것은 아니고, 특정 종료 시각을 지나 계속 진행될 수 있습니다. 자세한 내용은 Amazon RDS 유지 관리 기간 섹션을 참조하세요.

`describe-pending-maintenance-actions` AWS CLI 명령을 실행하여 DB 인스턴스에 유지 관리 업데이트를 사용할 수 있는지 여부를 확인할 수도 있습니다.

유지 관리 업데이트 적용에 대한 내용은 DB 인스턴스에 업데이트 적용 섹션을 참조하세요.

DB 인스턴스에 업데이트 적용

Amazon RDS를 사용하여 유지 관리 작업을 적용하는 시기를 선택할 수 있습니다. AWS Management Console, AWS CLI 또는 RDS API를 사용하여 Amazon RDS에서 업데이트를 적용하는 시기를 결정할 수 있습니다.

> **Note**
>
> RDS for SQL Server의 경우 DB 인스턴스를 중지했다가 다시 시작하거나 DB 인스턴스 클래스를 확장다가 다시 축소하여 기본 운영 체제에 대한 업데이트를 적용할 수 있습니다.

콘솔

DB 인스턴스의 업데이트를 관리하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 필수 업데이트가 포함된 DB 인스턴스를 선택합니다.
4. 작업에서 다음 중 하나를 선택합니다.
   - 지금 패치
   - 다음 기간에 패치

   > **Note**
   >
   > 다음 기간에 패치를 선택한 후 나중에 업데이트를 연기하려면 업그레이드 연기를 선택합니다. 유지 관리 작업이 이미 시작된 경우에는 보류할 수 없습니다.
   > 유지 관리 작업을 취소하려면 DB 인스턴스를 수정하고 마이너 버전 자동 업그레이드를 비활성화합니다.

AWS CLI

대기 중인 업데이트를 DB 인스턴스에 적용하려면 apply-pending-maintenance-action AWS CLI 명령을 사용합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds apply-pending-maintenance-action \
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db \
    --apply-action system-update \
    --opt-in-type immediate
```

Windows의 경우:

```
aws rds apply-pending-maintenance-action ^
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db ^
    --apply-action system-update ^
    --opt-in-type immediate
```

> **Note**
>
> 유지 관리 작업을 연기하려면 undo-opt-in에 `--opt-in-type`을 지정합니다. 유지 관리 작업이 이미 시작된 경우 undo-opt-in에 `--opt-in-type`을 지정할 수 없습니다.
> 유지 관리 작업을 취소하려면 modify-db-instance AWS CLI 명령을 실행하고 `--no-auto-minor-version-upgrade`을 지정합니다.

하나 이상의 대기 중인 업데이트가 있는 리소스 목록을 반환하려면, describe-pending-maintenance-actions AWS CLI 명령을 사용합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-pending-maintenance-actions \
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db
```

Windows의 경우:

```
aws rds describe-pending-maintenance-actions ^
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db
```

`describe-pending-maintenance-actions` AWS CLI 명령의 `--filters` 파라미터를 지정하여 DB 인스턴스에 대한 리소스 목록을 반환할 수도 있습니다. `--filters` 명령의 형식은 `Name=`*`filter-name`*`,Value=`*`resource-id`*`,...`입니다.

필터의 `Name` 파라미터에 대해 허용되는 값은 다음과 같습니다.

- `db-instance-id` – DB 인스턴스 식별자 또는 Amazon 리소스 이름(ARN) 목록을 허용합니다. 반환되는 목록에는 이러한 식별자 또는 ARN으로 식별된 DB 인스턴스에 대해 보류 중인 유지 관리 작업만 포함됩니다.
- `db-cluster-id` – Amazon Aurora의 DB 클러스터 식별자 또는 ARN 목록을 허용합니다. 반환되는 목록에는 이러한 식별자 또는 ARN으로 식별된 DB 클러스터에 대해 보류 중인 유지 관리 작업만 포함됩니다.

예를 들어 다음 예에서는 `sample-instance1` 및 `sample-instance2` DB 인스턴스에 대해 보류 중인 유지 관리 작업을 반환합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-pending-maintenance-actions \
 --filters Name=db-instance-id,Values=sample-instance1,sample-instance2
```

Windows의 경우:

```
aws rds describe-pending-maintenance-actions ^
 --filters Name=db-instance-id,Values=sample-instance1,sample-instance2
```

RDS API

업데이트를 DB 인스턴스에 적용하려면 Amazon RDS API `ApplyPendingMaintenanceAction` 작업을 호출합니다.

하나 이상의 대기 중인 업데이트가 있는 리소스 목록을 반환하려면 Amazon RDS API `DescribePendingMaintenanceActions` 작업을 호출합니다.

다중 AZ 배포 유지

DB 인스턴스를 다중 AZ 배포로 실행하면 유지 관리 이벤트의 영향을 더 줄일 수 있습니다. 이는 Amazon RDS가 다음 단계에 따라 운영 체제 업데이트를 적용하기 때문입니다.

1. 대기 목록의 유지 관리를 실행합니다.
2. 대기 목록을 기본 목록으로 승격시킵니다.
3. 이전에 기본 목록이었지만 현재는 새로운 대기 목록인 유지 관리를 실행합니다.

다중 AZ 배포에서 DB 인스턴스에 대한 데이터베이스 엔진을 업그레이드하면 Amazon RDS가 프라이머리 및 보조 DB 인스턴스를 모두 동시에 수정합니다. 이러한 경우 업그레이드하는 동안 다중 AZ 배포에 있는 프라이머리 및 보조 DB 인스턴스 모두를 사용할 수 없습니다. 이 작업을 수행하면 업그레이드가 완료될 때까지 가동 중지가 발생합니다. 다운타임 시간은 DB 인스턴스의 크기에 따라 다릅니다.

적용해야 하는 기본 운영 체제 패치가 있는 경우 기본 DB 인스턴스에 패치를 적용하려면 짧은 다중 AZ 장애 조치가 필요합니다. 이 장애 조치는 일반적으로 1분 미만입니다.

DB 인스턴스가 RDS for MySQL, RDS for PostgreSQL 또는 RDS for MariaDB를 실행하는 경우 블루/그린 배포를 사용하여 업그레이드에 필요한 가동 중지를 최소화할 수 있습니다. 자세한 내용은 데이터베이스 업데이트에 Amazon RDS 블루/그린 배포 사용 단원을 참조하십시오. 다중 AZ 배포에서 RDS for SQL Server 또는 RDS Custom for SQL Server DB 인스턴스를 업그레이드하면 Amazon RDS가 롤링 업그레이드를 수행하므로 장애 조치 기간 동안만 중단이 발생합니다. 자세한 내용은 다중 AZ 고려 사항 단원을 참조하십시오.

DB 인스턴스가 다중 AZ 배포에서 RDS for SQL Server를 실행하는 경우 다음 방법 중 하나를 사용하여 기본 운영 체제에 업데이트를 적용할 수 있습니다.

- DB 인스턴스 클래스를 다른 크기로 수정한 다음 다시 원래 크기로 수정합니다.
- DB 인스턴스 크기를 확장했다가 다시 원래 크기로 축소합니다.
- DB 인스턴스를 다중 AZ에서 단일 AZ로 수정하고, DB 인스턴스를 중지했다가 시작한 다음, 인스턴스를 다시 다중 AZ로 변경합니다.

다중 AZ 배포에 대한 자세한 정보는 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 단원을 참조하세요.

Amazon RDS 유지 관리 기간

유지 관리 기간은 시스템 변경 내용이 적용되는 주 단위의 기간입니다. 모든 DB 인스턴스에는 주간 유지 관리 기간이 있습니다. 유지 관리 기간은 수정 사항 및 소프트웨어 패치 적용 시점을 조절할 수 있는 기회입니다. 유지 관리 기간 조정에 대한 자세한 내용은 기본 DB 인스턴스 유지 관리 기간 조정 섹션을 참조하세요.

유지 관리가 적용되는 동안 RDS에서 사용자의 DB 인스턴스 에 있는 리소스 중 일부를 사용합니다. 이에 따라 성능에 미미한 영향이 있을 수 있습니다. DB 인스턴스의 경우 드물지만, 유지 관리 업데이트를 완료하려면 다중 AZ 장애 조치가 필요한 경우가 있을 수 있습니다.

유지 관리 이벤트가 특정 주에 예정되어 있는 경우 사용자가 지정하는 30분의 유지 관리 기간 중에 해당 이벤트가 시작됩니다. 또한 대부분의 유지 관리 이벤트가 30분의 유지 관리 기간 중에 완료됩니다. 단, 대규모 유지 관리 이벤트는 완료하는 데 30분이 넘게 걸릴 수 있습니다. DB 인스턴스가 중지되면 유지 관리 기간이 일시 중지됩니다.

지역별로 8시간 블록 시간 중에서 30분 유지 관리 시간이 임의로 선택됩니다. DB 인스턴스 생성 시 유지 관리 기간을 지정하지 않으면 RDS에서 임의로 선택한 요일에 30분 유지 관리 기간을 배정합니다.

다음에서 기본 유지 관리 기간이 할당된 리전별 시간 블록을 확인할 수 있습니다.

리전 이름	리전	시간 블록
미국 동부(오하이오)	us-east-2	03:00~11:00 UTC
미국 동부(버지니아 북부)	us-east-1	03:00~11:00 UTC
미국 서부(캘리포니아 북부 지역)	us-west-1	06:00~14:00 UTC
미국 서부(오리건)	us-west-2	06:00~14:00 UTC
Africa (Cape Town)	af-south-1	03:00~11:00 UTC
Asia Pacific (Hong Kong)	ap-east-1	06:00~14:00 UTC
아시아 태평양(하이데라바드)	ap-south-2	06:30~14:30 UTC
아시아 태평양(자카르타)	ap-southeast-3	08:00~16:00 UTC
아시아 태평양(멜버른)	ap-southeast-4	11:00~19:00 UTC

리전 이름	리전	시간 블록
아시아 태평양(뭄바이)	ap-south-1	06:00~14:00 UTC
Asia Pacific (Osaka)	ap-northeast-3	22:00~23:59 UTC
Asia Pacific (Seoul)	ap-northeast-2	13:00~21:00 UTC
아시아 태평양(싱가포르)	ap-southeast-1	14:00~22:00 UTC
아시아 태평양(시드니)	ap-southeast-2	12:00~20:00 UTC
아시아 태평양(도쿄)	ap-northeast-1	13:00~21:00 UTC
Canada (Central)	ca-central-1	03:00~11:00 UTC
캐나다 서부(캘거리)	ca-west-1	18:00~02:00 UTC
중국(베이징)	cn-north-1	06:00~14:00 UTC
China (Ningxia)	cn-northwest-1	06:00~14:00 UTC
Europe (Frankfurt)	eu-central-1	21:00–05:00 UTC
유럽(아일랜드)	eu-west-1	22:00~06:00 UTC
Europe (London)	eu-west-2	22:00~06:00 UTC
유럽(밀라노)	eu-south-1	02:00~10:00 UTC
유럽(파리)	eu-west-3	23:59~07:29 UTC
유럽(스페인)	eu-south-2	02:00~10:00 UTC
Europe (Stockholm)	eu-north-1	23:00~07:00 UTC
유럽(취리히)	eu-central-2	02:00~10:00 UTC
이스라엘(텔아비브)	il-central-1	03:00~11:00 UTC

리전 이름	리전	시간 블록
중동(바레인)	me-south-1	06:00~14:00 UTC
중동(UAE)	me-central-1	05:00~13:00 UTC
남아메리카(상파울루)	sa-east-1	00:00~08:00 UTC
AWS GovCloud(미국 동부)	us-gov-east-1	17:00~01:00 UTC
AWS GovCloud(미국 서부)	us-gov-west-1	06:00~14:00 UTC

기본 DB 인스턴스 유지 관리 기간 조정

유지 관리 기간은 사용률이 가장 낮은 시간에 할당되어야 하므로 수시로 수정되어야 할 수 있습니다. 시스템 변경 사항(DB 인스턴스 클래스 변경)을 적용 중이고 가동 중단이 필요한 경우에만 이 기간 동안 DB 인스턴스를 사용할 수 없습니다. DB 인스턴스는 필수 변경 사항을 적용하는 데 필요한 최소 시간 동안만 사용이 불가능합니다.

다음 예에서는 DB 인스턴스에 대한 기본 유지 관리 기간을 조정합니다.

이 예에서는 mydbinstance라는 DB 인스턴스가 있으며 기본 유지 관리 기간이 "Sun:05:00-Sun:06:00" UTC라고 가정하겠습니다.

콘솔

기본 유지 관리 기간을 조정하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 다음 변경하려는 DB 인스턴스를 선택합니다.
3. 수정을 선택합니다. Modify DB instance(DB 인스턴스 수정) 페이지가 나타납니다.
4. 유지 관리 섹션에서 유지 관리 기간을 업데이트합니다.

> **Note**
>
> DB 인스턴스에 대한 유지 관리 기간 및 백업 기간은 겹칠 수 없습니다. 백업 기간과 겹치는 유지 관리 기간의 값을 입력하면 오류 메시지가 나타납니다.

5. [Continue]를 선택합니다.

   확인 페이지에서 변경 내용을 검토합니다.

6. 유지 관리 기간에 변경 사항을 즉시 적용하려면 즉시 적용을 선택합니다.
7. DB 인스턴스 수정을 선택하여 변경 사항을 저장합니다.

   그렇지 않으면 [Back]을 선택하여 변경 내용을 편집하거나 [Cancel]을 선택하여 변경 내용을 취소합니다.

AWS CLI

기본 유지 관리 기간을 조정하려면 AWS CLI `modify-db-instance` 명령을 다음 파라미터와 함께 사용합니다.

- `--db-instance-identifier`
- `--preferred-maintenance-window`

Example

다음은 유지 관리 기간을 화요일 오전 4:00-4:30(UTC)로 설정하는 코드 예제입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
--db-instance-identifier mydbinstance \
--preferred-maintenance-window Tue:04:00-Tue:04:30
```

Windows의 경우:

```
aws rds modify-db-instance ^
--db-instance-identifier mydbinstance ^
--preferred-maintenance-window Tue:04:00-Tue:04:30
```

RDS API

기본 유지 관리 기간을 조정하려면 Amazon RDS API `ModifyDBInstance` 작업을 다음 파라미터와 함께 사용합니다.

- `DBInstanceIdentifier`
- `PreferredMaintenanceWindow`

RDS DB 인스턴스의 운영 체제 업데이트

RDS for Db2, RDS for MariaDB, RDS for MySQL, RDS for PostgreSQL, RDS for Oracle DB 인스턴스는 가끔 운영 체제를 업데이트해야 합니다. Amazon RDS는 운영 체제를 최신 버전으로 업그레이드하여 데이터베이스 성능과 고객의 전반적인 보안 태세를 개선합니다. 일반적으로 업데이트에는 10분 정도 걸립니다. 운영 체제 업데이트는 DB 인스턴스의 DB 엔진 버전이나 DB 인스턴스 클래스를 변경하지 않습니다.

운영 체제 업데이트는 선택 사항일 수도, 필수일 수도 있습니다.

- 선택적 업데이트는 언제든지 적용할 수 있습니다. 이러한 업데이트는 선택 사항이지만, 정기적으로 적용하여 RDS 플릿을 최신 상태로 유지하는 것이 좋습니다. RDS는 이러한 업데이트를 자동으로 적용하지 않습니다.

  새로운 선택적 운영 체제 시스템 패치가 제공될 때 알림을 받으려면 보안 패치 이벤트 범주에서 RDS-EVENT-0230 구독을 신청하면 됩니다. RDS 이벤트 구독에 대한 자세한 내용은 Amazon RDS 이벤트 알림 구독 단원을 참조하십시오.

  > **Note**
  >
  > `RDS-EVENT-0230`은 운영 체제 배포 업그레이드에는 적용되지 않습니다.

  > **Note**
  >
  > RDS for SQL Server DB 인스턴스에서 `RDS-EVENT-0230`을 수신한 경우 `apply-pending-maintenance` 작업을 사용해 OS 업데이트를 적용할 수 없습니다. 자세한 내용은 DB 인스턴스에 업데이트 적용 단원을 참조하십시오.

- 필수 업데이트는 필수이며, 적용 날짜가 정해져 있습니다. 이 적용 날짜 이전에 업데이트 일정을 계획하세요. 지정된 적용 날짜 이후 Amazon RDS는 할당된 유지 관리 기간 중 하나에 DB 인스턴스의 운영 체제를 최신 버전으로 자동 업그레이드합니다.

> **Note**
>
> 여러 규정 준수 의무를 충족하려면 모든 선택 및 필수 업데이트를 적용하여 최신 상태를 유지해야 할 수 있습니다. 유지 관리 기간 동안 RDS에서 제공하는 모든 업데이트를 정기적으로 적용하는 것이 좋습니다.

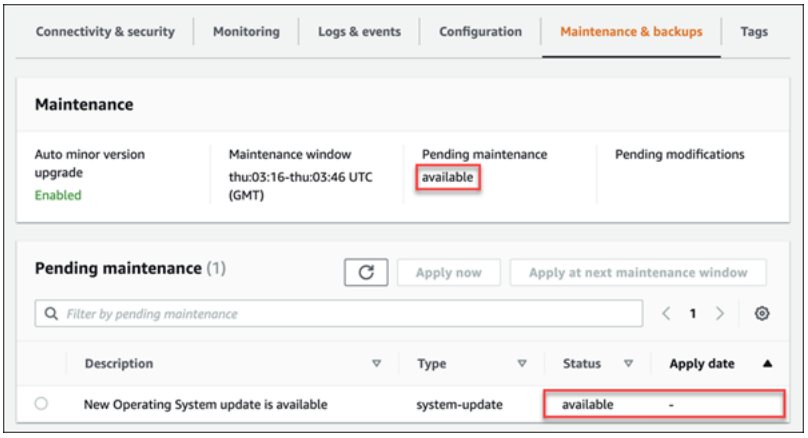
AWS Management Console 또는 AWS CLI를 사용하여 운영 체제 업그레이드 유형에 대한 정보를 얻을 수 있습니다.

콘솔

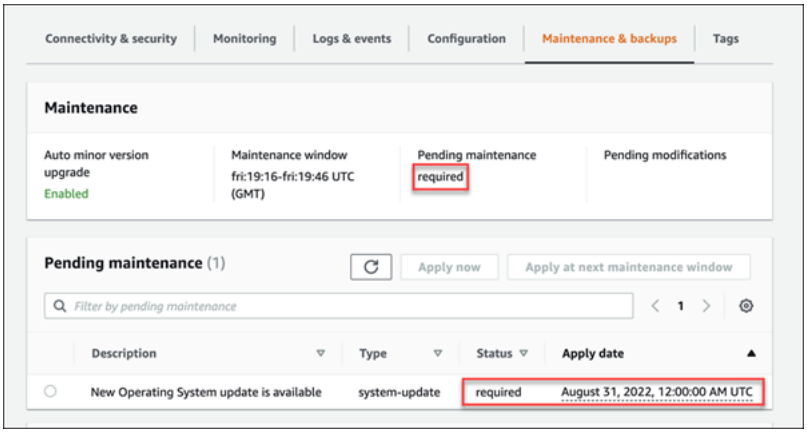
AWS Management Console을 사용하여 업데이트 정보를 얻는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스(Databases)를 선택한 다음 DB 인스턴스를 선택합니다.
3. 유지 관리 및 백업을 선택합니다.
4. 보류 중인 유지 관리(Pending maintenance) 섹션에서 운영 체제 업데이트를 찾은 다음 상태(Status) 값을 확인합니다.

AWS Management Console에서는 다음 이미지와 같이 선택적 업데이트의 유지 관리 상태가 사용 가능 상태로 설정되고 적용 날짜가 없습니다.



필수 업데이트는 다음 이미지와 같이 유지 관리 상태가 필수로 설정되고 적용 날짜가 정해져 있습니다.



AWS CLI

AWS CLI에서 업데이트 정보를 가져오려면 describe-pending-maintenance-actions 명령을 사용합니다.

```
aws rds describe-pending-maintenance-actions
```

필수 운영 체제 업데이트에는 `AutoAppliedAfterDate` 값과 `CurrentApplyDate` 값이 포함됩니다. 선택적 운영 체제 업데이트에는 두 값이 포함되지 않습니다.

다음 출력은 필수 운영 체제 업데이트를 보여줍니다.

```
{
  "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:db:mydb1",
  "PendingMaintenanceActionDetails": [
    {
      "Action": "system-update",
      "AutoAppliedAfterDate": "2022-08-31T00:00:00+00:00",
      "CurrentApplyDate": "2022-08-31T00:00:00+00:00",
      "Description": "New Operating System update is available"
    }
  ]
}
```

다음 출력은 선택적 운영 체제 업데이트를 보여줍니다.

```
{
  "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:db:mydb2",
  "PendingMaintenanceActionDetails": [
```

```
    {
      "Action": "system-update",
      "Description": "New Operating System update is available"
    }
  ]
}
```

운영 체제 업데이트 가용성

운영 체제 업데이트는 DB 엔진 버전 및 DB 인스턴스 클래스에 따라 다릅니다. 따라서 DB 인스턴스는 서로 다른 시점에 업데이트를 받거나 이를 요구합니다. 엔진 버전 및 인스턴스 클래스에 따라 DB 인스턴스에 운영 체제 업데이트가 지원되는 경우 업데이트가 콘솔에 표시됩니다. describe-pending-maintenance-actions AWS CLI 명령을 실행하거나 DescribePendingMaintenanceActions RDS API 작업을 직접 호출하여 확인할 수도 있습니다. 인스턴스에 대한 업데이트가 지원되는 경우 DB 인스턴스에 업데이트 적용의 지침에 따라 운영 체제를 업데이트할 수 있습니다.

DB 인스턴스 엔진 버전 업그레이드

Amazon RDS는 지원되는 각 데이터베이스 엔진의 최신 버전을 제공하여 DB 인스턴스를 최신 상태로 유지합니다. 최신 버전에는 데이터베이스 엔진의 버그 수정, 보안 강화 및 기타 개선 사항이 포함될 수 있습니다. Amazon RDS가 새로운 버전의 데이터베이스 엔진을 지원하는 경우, 데이터베이스 DB 인스턴스를 업그레이드할 방법과 시기를 선택할 수 있습니다.

메이저 버전 업그레이드와 마이너 버전 업그레이드라는 두 가지 업그레이드가 있습니다. 일반적으로 메이저 엔진 버전 업그레이드로 기존 애플리케이션과 호환되지 않는 변경 사항이 도입될 수 있습니다. 이와 대조적으로 마이너 버전 업그레이드에는 기존 애플리케이션과 역호환되는 변경 사항만 포함됩니다.

다중 AZ DB 클러스터의 경우 RDS for PostgreSQL에만 메이저 버전 업그레이드가 지원됩니다. 다중 AZ DB 클러스터를 지원하는 모든 엔진에 마이너 버전 업그레이드가 지원됩니다. 자세한 내용은 the section called “다중 AZ DB 클러스터 업그레이드” 섹션을 참조하세요.

버전 번호 순서는 각 데이터베이스 엔진마다 다릅니다. 예를 들어 RDS for MySQL 5.7 및 8.0은 메이저 엔진 버전이고, 5.7 버전에서 8.0 버전으로의 업그레이드는 메이저 버전 업그레이드입니다. RDS for MySQL 버전 5.7.22 및 5.7.23은 마이너 버전이고, 5.7.22에서 5.7.23으로의 업그레이드는 마이너 버전 업그레이드입니다.

> **Important**
>
> 업그레이드 중에는 DB 인스턴스를 수정할 수 없습니다. 업그레이드 중에 DB 인스턴스 상태는 `upgrading`입니다.

특정 DB 엔진의 메이저 및 마이너 버전 업그레이드에 대한 자세한 내용은 다음 DB 엔진 설명서를 참조하십시오.

- MariaDB DB 엔진 업그레이드
- Microsoft SQL Server DB 엔진 업그레이드
- RDS for MySQL DB 엔진 업그레이드
- RDS for Oracle DB 엔진 업그레이드
- RDS for PostgreSQL DB 엔진 업그레이드

메이저 버전 업그레이드를 위해서는 AWS Management Console, AWS CLI 또는 RDS API를 통해 DB 엔진 버전을 수동으로 수정해야 합니다. 마이너 버전 업그레이드의 경우, 엔진 버전을 수동으로 수정하거나 마이너 버전 자동 업그레이드 옵션을 활성화할 수 있습니다.

> **Note**
>
> 데이터베이스 엔진 업그레이드에는 다운타임이 필요합니다. 블루/그린 배포를 사용하면 DB 인스턴스 업그레이드에 필요한 다운타임을 최소화할 수 있습니다. 자세한 내용은 데이터베이스 업데이트에 Amazon RDS 블루/그린 배포 사용 섹션을 참조하세요.

주제

- 엔진 버전 수동 업그레이드
- 마이너 엔진 버전 자동 업그레이드

엔진 버전 수동 업그레이드

DB 인스턴스의 엔진 버전을 수동으로 업그레이드하려면 AWS Management Console, AWS CLI 또는 RDS API를 사용할 수 있습니다.

콘솔

콘솔을 사용하여 DB 인스턴스의 엔진 버전을 업그레이드하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 다음 업그레이드하려는 DB 인스턴스를 선택합니다.
3. 수정을 선택합니다. Modify DB instance(DB 인스턴스 수정) 페이지가 나타납니다.
4. DB 엔진 버전에서 새 버전을 선택합니다.
5. 계속해서 수정 사항을 요약한 내용을 확인합니다.
6. 다음과 같이 업그레이드 일정을 결정합니다.
   - 변경 사항을 보류 중인 수정 대기열에 배치하려면 예약된 다음 유지 관리 기간에 적용을 선택합니다. 다음 유지 관리 기간에 RDS가 대기열에 있는 보류 중 변경 사항을 적용합니다.
   - 변경 사항을 즉시 적용하려면 즉시 적용을 선택합니다. 일부의 경우 이 옵션을 선택하면 중단이 발생할 수 있습니다. 자세한 내용은 수정 예약 설정 사용 섹션을 참조하세요.

7. 확인 페이지에서 변경 내용을 검토합니다. 변경 내용이 정확할 경우 DB 인스턴스 수정을 선택하여 변경 내용을 저장합니다.

   그렇지 않으면 [Back]을 선택하여 변경 내용을 편집하거나 [Cancel]을 선택하여 변경 내용을 취소합니다.

AWS CLI

DB 인스턴스의 엔진 버전을 업그레이드하려면 CLI modify-db-instance 명령을 사용합니다. 다음 파라미터를 지정합니다.

- `--db-instance-identifier` – DB 인스턴스의 이름입니다.
- `--engine-version` – 업그레이드할 데이터베이스 엔진의 버전 번호입니다.

  유효한 엔진 버전에 대한 정보를 보려면 AWS CLI describe-db-engine-versions 명령을 사용합니다.
- `--allow-major-version-upgrade` – 메이저 버전을 업그레이드합니다.
- `--no-apply-immediately` – 변경 사항이 다음 유지 관리 기간에 적용됩니다. 변경 사항을 바로 적용하려면 `--apply-immediately`를 사용합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --engine-version new_version \
    --allow-major-version-upgrade \
    --no-apply-immediately
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --engine-version new_version ^
    --allow-major-version-upgrade ^
    --no-apply-immediately
```

RDS API

DB 인스턴스의 엔진 버전을 업그레이드하려면 ModifyDBInstance 작업을 사용합니다. 다음 파라미터를 지정합니다.

- `DBInstanceIdentifier` – DB 인스턴스의 이름입니다(예:*`mydbinstance`*).
- `EngineVersion` – 업그레이드할 데이터베이스 엔진의 버전 번호입니다. 유효한 엔진 버전에 대한 정보를 보려면 DescribeDBEngineVersions 작업을 사용합니다.
- `AllowMajorVersionUpgrade` – 메이저 버전 업그레이드를 허용하는지 여부입니다. 그렇게 하려면 값을 `true`로 설정합니다.
- `ApplyImmediately` – 변경 사항을 즉시 적용하거나 다음 유지 관리 기간에 적용합니다. 변경 사항을 바로 적용하려면 값을 `true`로 설정합니다. 변경 사항을 다음 유지 관리 기간에 적용하려면 값을 `false`로 설정합니다.

마이너 엔진 버전 자동 업그레이드

마이너 버전 자동 업그레이드는 데이터베이스를 최신 데이터베이스 엔진 버전으로 주기적으로 업데이트합니다. 그러나 업그레이드에 항상 최신 데이터베이스 엔진 버전이 포함되는 것은 아닙니다. 특정 시간에 데이터베이스를 특정 버전으로 유지해야 하는 경우 필요한 일정에 따라 필요한 데이터베이스 버전으로 수동으로 업그레이드하는 것이 좋습니다. 중요한 보안 문제가 발생하거나 버전이 지원 종료에 도달하면 마이너 버전 자동 업그레이드 옵션을 활성화하지 않은 경우에도 Amazon RDS에서 마이너 버전 업그레이드가 적용될 수 있습니다. 자세한 내용은 특정 데이터베이스 엔진에 대한 업그레이드 설명서를 참조하세요.

- RDS for PostgreSQL 마이너 버전 자동 업그레이드
- RDS for MySQL 마이너 버전 자동 업그레이드
- RDS for MariaDB 마이너 버전 자동 업그레이드
- Oracle 마이너 버전 업그레이드
- Microsoft SQL Server DB 엔진 업그레이드
- Amazon RDS의 Db2 버전

주제

- 마이너 버전 자동 업그레이드가 작동하는 방식
- 마이너 버전 자동 업그레이드 활성화
- 유지 관리 업데이트 사용 가능 여부 확인

- 자동 마이너 버전 업그레이드 대상 찾기

마이너 버전 자동 업그레이드가 작동하는 방식

업그레이드 대상은 Amazon RDS가 데이터베이스를 업그레이드하는 대상 DB 엔진 버전입니다. 마이너 엔진 버전은 다음 조건이 충족될 때 업그레이드 대상으로 지정됩니다.

- 데이터베이스가 대상 마이너 엔진 버전보다 낮은 DB 엔진의 마이너 버전을 실행하고 있습니다.

  데이터베이스 세부 정보 페이지의 구성 탭을 살펴보거나 `describe-db-instances` CLI 명령을 실행하여 DB 인스턴스의 현재 엔진 버전을 확인할 수 있습니다.
- 데이터베이스에 마이너 버전 자동 업그레이드가 사용 설정되어 있습니다.

RDS는 유지 관리 기간에 자동으로 업그레이드가 실행되도록 일정을 예약합니다. 업그레이드 중에 RDS는 다음을 수행합니다.

1. 시스템 사전 확인을 실행하여 데이터베이스가 정상이고 업그레이드할 준비가 되었는지 확인합니다.
2. DB 엔진을 대상 마이너 엔진 버전으로 업그레이드
3. 업그레이드 후 확인 실행
4. 데이터베이스 업그레이드를 완료로 표시

자동 업그레이드 시 가동 중지가 발생합니다. 가동 중지 시간은 DB 엔진 유형, 데이터베이스 크기 등 다양한 요인에 따라 달라집니다.

마이너 버전 자동 업그레이드 활성화

다음 작업을 수행할 때 DB 인스턴스에서 마이너 버전 자동 업그레이드를 활성화할지 여부를 제어할 수 있습니다.

- DB 인스턴스 생성
- DB 인스턴스 수정
- 읽기 전용 복제본 생성
- 스냅샷에서 DB 인스턴스 복원
- DB 인스턴스를 특정 시간으로 복원
- Amazon S3에서 DB 인스턴스 가져오기(Amazon S3에 MySQL 백업의 경우)

이러한 작업을 수행할 때 다음과 같은 방법으로 DB 인스턴스에서 마이너 버전 자동 업그레이드를 활성화할지 여부를 제어할 수 있습니다.

- 콘솔을 사용하여 마이너 버전 자동 업그레이드 옵션을 설정합니다.
- AWS CLI를 사용하여 `--auto-minor-version-upgrade|--no-auto-minor-version-upgrade` 옵션을 설정합니다.
- RDS API를 사용하여 `AutoMinorVersionUpgrade` 파라미터를 설정합니다.

유지 관리 업데이트 사용 가능 여부 확인

DB 인스턴스에 DB 엔진 버전 업그레이드 등의 유지 관리 업데이트를 사용할 수 있는지 여부를 확인하려면 콘솔, AWS CLI 또는 RDS API를 사용하면 됩니다. DB 엔진 버전을 수동으로 업그레이드하고 유지 관리 기간을 조정할 수도 있습니다. 자세한 내용은 DB 인스턴스 유지 관리 섹션을 참조하세요.

자동 마이너 버전 업그레이드 대상 찾기

다음 AWS CLI 명령을 사용하여 특정 AWS 리전 리전의 지정된 마이너 DB 엔진 버전에 대한 현재의 자동 마이너 업그레이드 대상 버전을 확인할 수 있습니다. CreateDBInstance의 `Engine` 파라미터에 대한 설명에서 이 명령에 대해 가능한 `--engine` 값을 찾을 수 있습니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-engine-versions \
--engine engine \
--engine-version minor-version \
--region region \
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" \
--output text
```

Windows의 경우:

```
aws rds describe-db-engine-versions ^
--engine engine ^
--engine-version minor-version ^
--region region ^
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" ^
--output text
```

예를 들어, 다음 AWS CLI 명령은 미국 동부(오하이오) AWS 리전(us-east-2)의 MySQL 마이너 버전 8.0.11에 대한 자동 마이너 업그레이드 대상을 안내합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-engine-versions \
--engine mysql \
--engine-version 8.0.11 \
--region us-east-2 \
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" \
--output table
```

Windows의 경우:

```
aws rds describe-db-engine-versions ^
--engine mysql ^
--engine-version 8.0.11 ^
--region us-east-2 ^
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" ^
--output table
```

다음과 같은 출력이 표시됩니다.

```
----------------------------------
|    DescribeDBEngineVersions    |
+--------------+-----------------+
|  AutoUpgrade |  EngineVersion  |
+--------------+-----------------+
|  False       |  8.0.15         |
|  False       |  8.0.16         |
|  False       |  8.0.17         |
|  False       |  8.0.19         |
|  False       |  8.0.20         |
|  False       |  8.0.21         |
|  True        |  8.0.23         |
|  False       |  8.0.25         |
+--------------+-----------------+
```

이 예제에서 `AutoUpgrade` 값은 MySQL 버전 8.0.23의 경우 `True`입니다. 따라서 자동 마이너 업그레이드 대상은 출력에서 강조 표시된 MySQL 버전 8.0.23입니다.

Important

곧 RDS for PostgreSQL DB 인스턴스를 Aurora PostgreSQL DB 클러스터로 마이그레이션하려는 경우 계획 단계 초기에 DB 인스턴스의 자동 마이너 버전 업그레이드를 비활성화하는 것이 좋습니다. Aurora PostgreSQL에서 해당 RDS for PostgreSQL 버전이 아직 지원되지 않는 경우 Aurora PostgreSQL로의 마이그레이션이 지연될 수 있습니다. Aurora PostgreSQL 버전에 대한 자세한 내용은 Amazon Aurora PostgreSQL의 엔진 버전을 참조하세요.

DB 인스턴스 이름 변경

AWS Management Console, AWS CLI `modify-db-instance` 명령 또는 Amazon RDS API `ModifyDBInstance` 작업을 사용하여 DB 인스턴스의 이름을 바꿀 수 있습니다. DB 인스턴스 이름을 변경하면 커다란 영향을 끼칠 수 있습니다. 다음은 DB 인스턴스의 이름을 바꾸기 전에 고려해야 할 사항의 목록입니다.

- DB 인스턴스 이름을 변경하면 DB 인스턴스의 엔드포인트도 변경됩니다. URL에는 DB 인스턴스에 할당한 이름이 포함되어 있기 때문입니다. 트래픽은 항상 이전 URL에서 새 URL로 리디렉션해야 합니다.
- DB 인스턴스 이름을 변경하면 DB 인스턴스에서 이전에 사용된 DNS 이름은 바로 삭제되지만 캐시는 몇 분 더 남을 수도 있습니다. 이름이 바뀐 DB 인스턴스의 새로운 DNS 이름은 약 10분 후부터 적용됩니다. 이름이 바뀐 DB 인스턴스를 사용하려면 새로운 이름이 적용될 때까지 기다려야 합니다.
- 인스턴스 이름이 바뀌면 기존 DB 인스턴스 이름은 사용할 수 없습니다.
- DB 인스턴스와 연동되어 있던 읽기 전용 복제본은 이름이 바뀐 후에도 모두 인스턴스와 연동된 상태를 유지합니다. 예를 들어 프로덕션 데이터베이스 역할을 하는 DB 인스턴스에 읽기 전용 복제본이 여러 개 연동되어 있다고 가정하겠습니다. 이때 DB 인스턴스 이름을 변경한 후 프로덕션 환경에서 DB 스냅샷으로 교체하더라도 이름을 바꾼 DB 인스턴스는 읽기 전용 복제본이 그대로 연동되어 있습니다.
- DB 인스턴스 이름을 재사용하면 DB 인스턴스 이름과 연동되어 있는 측정치와 이벤트가 유지됩니다. 예를 들어, 읽기 전용 복제본을 승격하여 이전 기본 DB 인스턴스의 이름으로 변경하는 경우 기본 DB 인스턴스와 연결된 이벤트와 지표가 이름이 바뀐 인스턴스와 연결됩니다.
- DB 인스턴스 태그는 이름 변경 여부에 상관없이 DB 인스턴스에 그대로 남습니다.
- DB 스냅샷은 바뀐 이름의 DB 인스턴스로 유지됩니다.

> **Note**
>
> DB 인스턴스는 클라우드에서 실행하는 격리된 데이터베이스 환경입니다. 한 DB 인스턴스에서 여러 개의 데이터베이스 또는 여러 스키마를 포함하는 단일 Oracle 데이터베이스를 호스팅할 수 있습니다. 데이터베이스 이름 변경에 대한 자세한 내용은 DB 엔진 설명서를 참조하세요.

기존 DB 인스턴스 교체를 위한 이름 바꾸기

DB 인스턴스 이름을 바꾸는 가장 일반적인 이유는 읽기 전용 복제본을 승격하거나 DB 스냅샷 또는 특정 시점으로 복구(PITR)에서 데이터를 복구해야 하기 때문입니다. 데이터베이스 이름을 변경하면 애플리케이션 코드를 변경하여 DB 인스턴스를 참조하지 않아도 DB 인스턴스를 교체할 수 있습니다. 이러한 경우 방법은 다음과 같습니다.

1. 기본 DB 인스턴스로 전송되는 트래픽을 모두 차단합니다. 여기에는 DB 인스턴스의 데이터베이스에서 수신되는 트래픽을 리디렉션하거나 그 밖에 트래픽이 DB 인스턴스의 데이터베이스에 액세스하지 못하도록 차단할 수 있는 방법도 사용됩니다.
2. 기본 DB 인스턴스의 이름을 이 주제의 뒷부분에 설명된 대로 더 이상 기본 DB 인스턴스가 아님을 나타내는 이름으로 바꿉니다.
3. DB 스냅샷에서 복구하거나 읽기 전용 복제본을 승격하여 새로운 기본 DB 인스턴스를 생성한 다음 새로운 인스턴스를 이전 기본 DB 인스턴스 이름으로 명명합니다.
4. 읽기 전용 복제본을 새로운 기본 DB 인스턴스와 연결합니다.

이전 기본 DB 인스턴스를 삭제할 때 기존 기본 DB 인스턴스에서 원하지 않는 DB 스냅샷까지 삭제되는 경우 사용자 본인에게 책임이 있습니다.

읽기 전용 복제본 승격에 대한 자세한 내용은 읽기 전용 복제본을 독립 DB 인스턴스로 승격 단원을 참조하십시오.

> **Important**
>
> 이름이 변경되면 DB 인스턴스가 재부팅됩니다. AlwaysOn 또는 미러링 옵션이 활성화된 RDS for SQL Server 다중 AZ 인스턴스의 경우 이름 바꾸기 작업 후 인스턴스를 재부팅하면 장애 조치가 수행될 수 있습니다.

콘솔

DB 인스턴스 이름 변경 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 이름을 바꿀 DB 인스턴스를 선택합니다.

4. 수정을 선택합니다.
5. 설정에서 DB 인스턴스 식별자에 새 이름을 입력합니다.
6. [Continue]를 선택합니다.
7. 변경 사항을 즉시 적용하려면 즉시 적용을 선택합니다. 일부의 경우 이 옵션을 선택하면 중단이 발생할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.
8. 확인 페이지에서 변경 내용을 검토합니다. 변경 내용이 정확할 경우 [DB 인스턴스 수정(Modify DB instance)]를 선택하여 변경 내용을 저장합니다.

   그렇지 않으면 [Back]을 선택하여 변경 내용을 편집하거나 [Cancel]을 선택하여 변경 내용을 취소합니다.

AWS CLI

DB 인스턴스의 이름을 바꾸려면 AWS CLI 명령 `modify-db-instance`를 사용합니다. 현재 `--db-instance-identifier` 값 및 `--new-db-instance-identifier` 파라미터를 DB 인스턴스의 새 이름과 함께 제공합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier DBInstanceIdentifier \
    --new-db-instance-identifier NewDBInstanceIdentifier
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier DBInstanceIdentifier ^
    --new-db-instance-identifier NewDBInstanceIdentifier
```

RDS API

DB 인스턴스의 이름을 변경하려면 다음 파라미터를 사용하여 Amazon RDS API 작업 `ModifyDBInstance`를 호출합니다.

- `DBInstanceIdentifier` — 인스턴스의 기존 이름
- `NewDBInstanceIdentifier` — 인스턴스의 새 이름

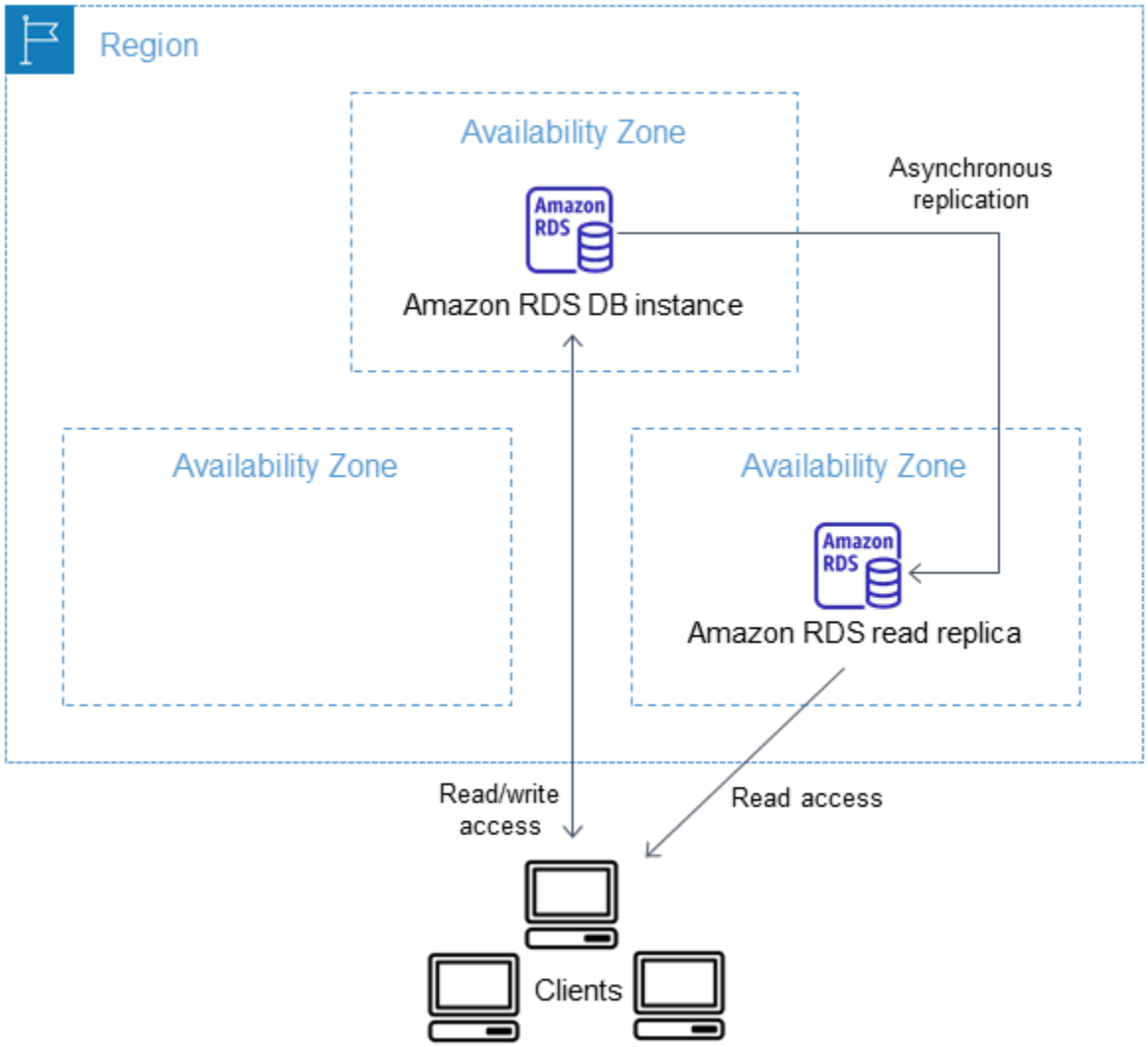
DB 인스턴스 읽기 전용 복제본 작업

읽기 전용 복제본은 DB 인스턴스의 읽기 전용 사본입니다. 애플리케이션에서 읽기 전용 복제본으로 쿼리를 라우팅하여 프라이머리 DB 인스턴스의 로드를 줄일 수 있습니다. 이렇게 하면 읽기 중심의 데이터베이스 워크로드에 대한 단일 DB 인스턴스의 용량 제한에서 벗어나 탄력적으로 스케일 아웃할 수 있습니다.

Amazon RDS는 소스 DB 인스턴스의 읽기 전용 복제본을 생성하기 위해 DB 엔진의 기본 복제 기능을 사용합니다. 특정 엔진에서 읽기 전용 복제본을 사용하는 것에 대한 자세한 내용은 다음 단원을 참조하십시오.

- Amazon RDS for Db2의 복제본 작업
- MariaDB 읽기 전용 복제본 작업
- Amazon RDS에서 Microsoft SQL Server용 읽기 전용 복제본 작업
- MySQL 읽기 전용 복제본 작업
- Amazon RDS의 Oracle의 읽기 전용 복제본 작업
- Amazon RDS for PostgreSQL의 읽기 전용 복제본 작업

소스 DB 인스턴스에서 읽기 전용 복제본을 생성하면 해당 소스가 프라이머리 DB 인스턴스가 됩니다. 프라이머리 DB 인스턴스를 업데이트하면 Amazon RDS가 읽기 전용 복제본에 비동기식으로 복사합니다. 다음 다이어그램은 다른 가용 영역(AZ)의 읽기 전용 복제본으로 복제되는 소스 DB 인스턴스를 보여 줍니다. 클라이언트는 기본 DB 인스턴스에 대한 읽기/쓰기 액세스 권한과 복제본에 대한 읽기 전용 액세스 권한을 가집니다.



읽기 전용 복제본에는 복제본에 사용된 DB 인스턴스 클래스와 동일한 요금이 표준 DB 인스턴스로 청구됩니다. 동일한 AWS 리전 내에서 원본 DB 인스턴스와 읽기 전용 복제본 간의 데이터 복제로 발생한 데이터 전송에 대해서는 요금이 부과되지 않습니다. 자세한 내용은 리전 간 복제 비용 및 Amazon RDS에 대한 DB 인스턴스 결제(을)를 참조하세요.

주제

- Amazon RDS 읽기 전용 복제본의 개요
- DB 엔진별 읽기 전용 복제본의 차이점
- 읽기 전용 복제본 생성
- 읽기 전용 복제본을 독립 DB 인스턴스로 승격
- 읽기 전용 복제본 모니터링
- 다른 AWS 리전에서 읽기 전용 복제본 생성

Amazon RDS 읽기 전용 복제본의 개요

다음 섹션에서는 DB 인스턴스 읽기 전용 복제본에 대해 설명합니다. 다중 AZ DB 클러스터 읽기 전용 복제본에 대한 자세한 내용은 the section called “다중 AZ DB 클러스터 읽기 전용 복제본 사용” 섹션을 참조하세요.

주제

- 읽기 전용 복제본에 대한 사용 사례
- 읽기 전용 복제본의 작동 방식
- 다중 AZ 배포의 읽기 전용 복제본
- 리전 간 읽기 전용 복제본
- 읽기 전용 복제본 스토리지 유형
- 복제본에서 복제본을 생성할 때의 제한 사항
- 복제본 삭제 시 고려 사항

읽기 전용 복제본에 대한 사용 사례

지정된 원본 DB 인스턴스에 하나 이상의 읽기 전용 복제본을 배포하면 다음을 비롯해 다양한 시나리오에서 적용될 수 있습니다.

- 읽기 중심의 데이터베이스 워크로드를 위해 단일 DB 인스턴스의 컴퓨팅 파워 또는 I/O 용량을 확장합니다. 이 과도한 읽기 트래픽을 하나 이상의 읽기 전용 복제본으로 이동할 수 있습니다.
- 원본 DB 인스턴스를 사용할 수 없는 동안 읽기 트래픽을 처리합니다. 경우에 따라 백업 또는 예약된 유지 관리를 위한 I/O 일시 중단 등으로 인해 원본 DB 인스턴스가 I/O 요청을 처리하지 못할 수 있습니다. 이러한 경우 읽기 트래픽을 읽기 전용 복제본으로 리디렉션할 수 있습니다. 이 사용 사례의 경우 원본 DB 인스턴스를 사용할 수 없으므로 읽기 전용 복제본의 데이터가 "무효"일 수 있다는 점에 유의하십시오.
- 비즈니스 보고 또는 데이터 웨어하우징 시나리오에서는 프로덕션 DB 인스턴스가 아닌 읽기 전용 복제본에 대한 비즈니스 보고 쿼리를 실행할 수 있습니다.
- 재해 복구 구현 기본 DB 인스턴스가 실패할 경우 재해 복구 솔루션으로 읽기 전용 복제본을 독립형 인스턴스로 승격할 수 있습니다.

읽기 전용 복제본의 작동 방식

읽기 전용 복제본을 만들면 기존 DB 인스턴스를 소스로 지정합니다. 그런 다음 Amazon RDS가 소스 인스턴스의 스냅샷을 생성하여 해당 스냅샷에서 읽기 전용 인스턴스를 생성합니다. 그러면 Amazon RDS가 기본 DB 인스턴스를 변경할 때마다 DB 엔진에 비동기식 복제 방법을 사용하여 읽기 전용 복제본을 업데이트합니다.

읽기 전용 복제본은 읽기 전용 연결만을 허용하는 DB 인스턴스로 작동합니다. RDS for Db2 및 RDS for Oracle DB 엔진은 예외로, 각각 대기 모드 및 탑재 모드에서 복제본 데이터베이스를 지원합니다. 대기 복제본과 탑재된 복제본은 사용자 연결을 허용하지 않으므로 읽기 전용 워크로드를 처리할 수 없습니다. 대기 복제본과 탑재된 복제본의 주된 용도는 교차 리전 재해 복구입니다. 자세한 내용은 Amazon RDS for Db2의 복제본 작업 및 Amazon RDS의 Oracle의 읽기 전용 복제본 작업(을)를 참조하세요.

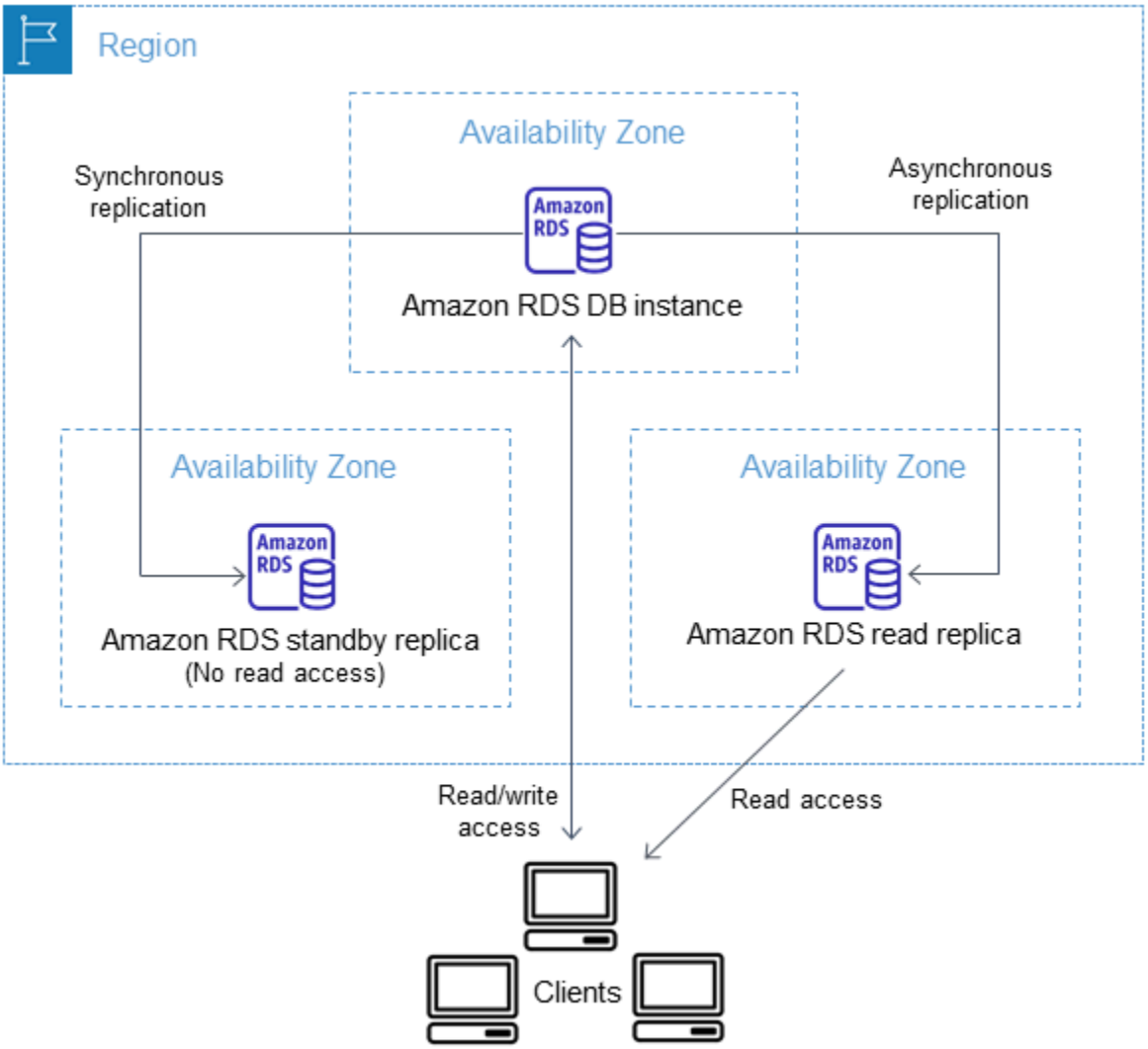
애플리케이션은 DB 인스턴스에 연결되는 방식과 동일하게 읽기 전용 복제본에 연결됩니다. Amazon RDS는 소스 DB 인스턴스에서 모든 데이터베이스를 복제합니다.

읽기 전용 복제본을 수동으로 만들어야 합니다. RDS는 읽기 수요 변경 사항에 따라 읽기 전용 복제본을 자동으로 추가하거나 제거하는 읽기 전용 복제본의 자동 크기 조정을 지원하지 않습니다.

다중 AZ 배포의 읽기 전용 복제본

다중 AZ 배포의 고가용성을 위해 구성된 대기 복제본이 있는 DB 인스턴스에 대해 읽기 전용 복제본을 구성할 수 있습니다. 대기 복제본을 사용한 복제는 동기식입니다. 읽기 전용 복제본과 다르게 대기 복제본은 읽기 트래픽을 처리할 수 없습니다.

다음 시나리오에서 클라이언트는 한 AZ의 프라이머리 DB 인스턴스에 대한 읽기/쓰기 액세스 권한을 가집니다. 프라이머리 인스턴스는 업데이트를 두 번째 AZ의 읽기 전용 복제본에 비동기적으로 복사하고 세 번째 AZ의 대기 복제본에도 동기적으로 복사합니다. 클라이언트는 읽기 전용 복제본에 대한 읽기 액세스 권한만 가집니다.



고가용성 및 대기 복제본에 대한 자세한 내용은 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 섹션을 참조하세요.

리전 간 읽기 전용 복제본

경우에 따라 읽기 전용 복제본이 프라이머리 DB 인스턴스와 다른 AWS 리전에 있습니다. 이러한 경우 Amazon RDS는 기본 DB 인스턴스와 읽기 전용 복제본 사이에 보안 통신 채널을 설정합니다. Amazon RDS는 보안 그룹 항목 추가와 같이 보안 채널을 활성화하는 데 필요한 모든 AWS 보안 구성을 설정합니다. 리전 간 읽기 전용 복제본에 대한 자세한 내용은 다른 AWS 리전에서 읽기 전용 복제본 생성 섹션을 참조하세요.

이 장의 정보는 소스 DB 인스턴스와 동일한 AWS 리전이나 별도의 AWS 리전에 Amazon RDS 읽기 전용 복제본을 생성할 때도 똑같이 적용됩니다. 하지만 Amazon EC2 인스턴스에서 실행되거나 온프레미스인 인스턴스로 복제를 설정할 때는 적용되지 않습니다.

읽기 전용 복제본 스토리지 유형

기본적으로 읽기 전용 복제본은 원본 DB 인스턴스와 동일한 스토리지 유형과 함께 생성됩니다. 하지만 다음 표에 나와 있는 옵션에 따라 원본 DB 인스턴스와 다른 스토리지 유형을 가진 읽기 전용 복제본을 생성할 수도 있습니다.

원본 DB 인스턴스 스토리지 유형	원본 DB 인스턴스 스토리지 할당	읽기 전용 복제본 스토리지 유형 옵션
프로비저닝된 IOPS	100GiB–64TiB	프로비저닝된 IOPS, 범용, 마그네틱
범용	100GiB–64TiB	프로비저닝된 IOPS, 범용, 마그네틱
범용	<100GiB	범용, 마그네틱
Magnetic	100GiB–6TiB	프로비저닝된 IOPS, 범용, 마그네틱
Magnetic	<100GiB	범용, 마그네틱

> **Note**
>
> 읽기 복제본의 할당된 스토리지를 늘릴 때는 10% 이상이어야 합니다. 이 값을 10% 미만으로 늘리면 오류가 발생합니다.

복제본에서 복제본을 생성할 때의 제한 사항

Amazon RDS는 순환 복제를 지원하지 않습니다. DB 인스턴스를 기존 DB 인스턴스의 복제 소스로 구성할 수 없습니다. 기존 DB 인스턴스에서 새 읽기 전용 복제본만 생성할 수 있습니다. 예를 들어, **`MySourceDBInstance`**을 **`ReadReplica1`**으로 복제하는 경우, **`ReadReplica1`**을 **`MySourceDBInstance`**로 다시 복제하도록 구성할 수 없습니다.

RDS for MariaDB 및 RDS for MySQL과 특정 RDS for PostgreSQL 버전의 경우 기존 읽기 전용 복제본에서 읽기 전용 복제본을 생성할 수 있습니다. 예를 들어 기존 복제본 **`ReadReplica1`**에서 새로운 읽

기 전용 복제본 **ReadReplica2**를 생성할 수 있습니다. RDS for Db2, RDS for Oracle 및 RDS for SQL Server의 경우 기존 읽기 전용 복제본에서 읽기 전용 복제본을 생성할 수 없습니다.

복제본 삭제 시 고려 사항

RDS는 읽기 전용 복제본의 자동 크기 조정을 지원하지 않습니다. 따라서 RDS는 수요가 증가할 때 복제본의 수를 늘리지 않고, 수요가 감소할 때 복제본의 수를 줄이지 않습니다. 읽기 전용 복제본이 더 이상 필요하지 않은 경우, DB 인스턴스를 삭제하는 동일한 메커니즘을 사용하여 읽기 전용 복제본을 수동으로 삭제할 수 있습니다. 동일한 AWS 리전에서 읽기 전용 복제본을 삭제하지 않고 소스 DB 인스턴스를 삭제하면 각 복제본이 독립 실행형 DB 인스턴스로 승격됩니다.

DB 인스턴스 삭제에 대한 자세한 내용은 DB 인스턴스 삭제 단원을 참조하세요. 읽기 전용 복제본 승격에 대한 자세한 내용은 읽기 전용 복제본을 독립 DB 인스턴스로 승격 단원을 참조하십시오. 교차 리전 읽기 전용 복제본이 있는 경우, 소스 DB 인스턴스 삭제와 관련된 자세한 내용은 리전 간 복제 시 고려 사항 섹션을 참조하시기 바랍니다.

DB 엔진별 읽기 전용 복제본의 차이점

Amazon RDS DB 엔진은 복제를 다르게 구현하기 때문에 몇 가지 중요한 차이점을 알아야 합니다.

Db2

RDS for Db2의 복제본에는 다음과 같은 기능과 동작이 있습니다.

- 복제 방법 - 물리적 복제
- 트랜잭션 로그 제거 - RDS for Db2는 다음 조건이 충족되면 기본 DB 인스턴스에서 로그를 제거합니다.
  - 로그가 최소 2시간 이상 경과됨
  - 아카이브 로그 보존 시간 설정이 초과됨
  - RDS for Db2가 모든 복제본 DB 인스턴스에 로그를 성공적으로 복제함

  이는 동일한 AWS 리전의 DB 인스턴스와 교차 리전 DB 인스턴스에 적용됩니다. 아카이브 로그 보존 시간 설정에 대한 자세한 내용은 rdsadmin.set_archive_log_retention 단원을 참조하세요.
- 쓰기 가능한 복제본 - Db2 복제본은 물리적 복사본이며 Db2는 복제본에 쓰기를 허용하지 않습니다. 복제본을 쓰기 가능하도록 승격할 수 있습니다. 승격된 복제본에는 승격 요청이 이루어진 시점까지 복제된 데이터가 있습니다.
- 백업 - RDS for Db2 복제본에서는 자동 백업과 수동 스냅샷이 지원됩니다.

- 병렬 복제 - 아카이브 로그 데이터는 항상 기본 데이터베이스에서 모든 복제본으로 병렬로 전송됩니다.
- 대기 상태 - 대기 복제본의 주된 용도는 리전 간 재해 복구입니다. 자세한 내용은 Amazon RDS for Db2의 복제본 작업 섹션을 참조하세요.

MariaDB 및 MySQL

RDS for MariaDB 및 RDS for MySQL의 읽기 전용 복제본에는 다음과 같은 기능과 동작이 있습니다.

- 복제 방법 - 논리적 복제
- 트랜잭션 로그 제거 - RDS for MariaDB와 RDS for MySQL은 적용되지 않은 이진 로그를 유지합니다.
- 쓰기 가능한 복제본 - MariaDB 또는 MySQL 읽기 전용 복제본은 쓰기가 가능하도록 활성화할 수 있습니다.
- 백업 - RDS for MariaDB 또는 RDS for MySQL 읽기 전용 복제본에서는 자동 백업과 수동 스냅샷이 지원됩니다.
- 병렬 복제 - 지원되는 모든 MariaDB 및 MySQL 버전은 병렬 복제 스레드가 가능합니다.
- 탑재된 상태 - 지원되지 않습니다.

Oracle

RDS for Oracle의 읽기 전용 복제본에는 다음과 같은 기능과 동작이 있습니다.

- 복제 방법 - 물리적 복제
- 트랜잭션 로그 제거 - 기본 DB 인스턴스에 리전 간 읽기 전용 복제본이 없는 경우 Amazon RDS for Oracle이 소스 DB 인스턴스에 대한 최소 2시간의 트랜잭션 로그를 유지합니다. 로그는 아카이브 로그 보존 시간 설정이 통과된 후 또는 두 시간 후 중에서 더 긴 시간이 경과한 후 원본 DB 인스턴스에서 제거됩니다. 설정이 데이터베이스에 성공적으로 적용된 경우에만 아카이브 로그 보존 시간 설정 통과 후 로그가 읽기 전용 복제본에서 제거됩니다.

  경우에 따라 기본 DB 인스턴스에 하나 이상의 리전 간 읽기 전용 복제본이 있을 수 있습니다. 이 경우 Amazon RDS for Oracle은 원본 DB 인스턴스에 대한 트랜잭션 로그가 전송되어 모든 리전 간 읽기 전용 복제본에 적용될 때까지 이 로그를 유지합니다.

  아카이브 로그 보존 시간 설정에 대한 자세한 내용은 보관된 다시 실행 로그 보존 단원을 참조하세요.

- 쓰기 가능한 복제본 - Oracle 읽기 전용 복제본은 물리적 복사본이며 Oracle은 읽기 전용 복제본에 쓰기를 허용하지 않습니다. 읽기 전용 복제본을 쓰기 가능하도록 승격할 수 있습니다. 승격된 읽기 전용 복제본에는 승격 요청이 이루어진 시점까지 복제된 데이터가 있습니다.
- 백업 - RDS for Oracle 읽기 전용 복제본에서는 자동 백업과 수동 스냅샷이 지원됩니다.
- 병렬 복제 - 다시 실행 로그 데이터는 항상 기본 데이터베이스에서 모든 읽기 전용 복제본으로 병렬로 전송됩니다.
- 탑재된 상태 - 탑재된 복제본의 주된 용도는 리전 간 재해 복구입니다. 탑재된 복제본에는 Active Data Guard 라이선스가 필요하지 않습니다. 자세한 내용은 Amazon RDS의 Oracle의 읽기 전용 복제본 작업 섹션을 참조하세요.

PostgreSQL

RDS for PostgreSQL의 읽기 전용 복제본에는 다음과 같은 기능과 동작이 있습니다.

- 복제 방법 - 물리적 복제
- 트랜잭션 로그 제거 - PostgreSQL에는 데이터를 읽기 전용 복제본으로 보낼 때 유지할 Write Ahead Log(WAL) 파일 수를 결정하는 파라미터인 `wal_keep_segments`가 있습니다. 이 파라미터 값에 따라 유지할 로그 수가 결정됩니다.
- 쓰기 가능한 복제본 - PostgreSQL 읽기 전용 복제본은 물리적 복사본이며 PostgreSQL은 읽기 전용 복제본에 쓰기를 허용하지 않습니다.
- 백업 - RDS for PostgreSQL 읽기 전용 복제본에 수동 스냅샷이 지원됩니다. 읽기 전용 복제본에 대한 자동 백업은 RDS for PostgreSQL 14.1 이상 버전에서만 지원됩니다. RDS for PostgreSQL 14.1 이전 버전의 PostgreSQL 읽기 전용 복제본에 대해서는 자동 백업을 설정할 수 없습니다. RDS for PostgreSQL 13 이전 버전의 경우 백업하려면 읽기 전용 복제본에서 스냅샷을 생성하면 됩니다.
- 병렬 복제 - PostgreSQL은 복제를 처리하는 단일 프로세스가 있습니다.
- 탑재된 상태 - 지원되지 않습니다.

SQL Server

RDS for SQL Server의 읽기 전용 복제본에는 다음과 같은 기능과 동작이 있습니다.

- 복제 방법 - 물리적 복제
- 트랜잭션 로그 제거 - 보조 복제본에 더 이상 필요하지 않은 경우 기본 복제본에 있는 트랜잭션 로그 파일의 가상 로그 파일(VLF)이 잘릴 수 있습니다.

VLF는 복제본에서 로그 레코드가 강화된 경우에만 비활성으로 표시할 수 있습니다. 기본 복제본에 있는 디스크 하위 시스템의 속도에 관계없이 트랜잭션 로그는 가장 느린 복제본이 VLF를 강화할 때까지 VLF를 유지합니다.

- 쓰기 가능한 복제본 - SQL Server 읽기 전용 복제본은 물리적 복사본이며 쓰기도 허용하지 않습니다. 읽기 전용 복제본을 쓰기 가능하도록 승격할 수 있습니다. 승격된 읽기 전용 복제본에는 승격 요청이 이루어진 시점까지 복제된 데이터가 있습니다.
- 백업 - RDS for SQL Server 읽기 전용 복제본에서는 자동 백업과 수동 스냅샷이 지원되지 않습니다.
- 병렬 복제 - 다시 실행 로그 데이터는 항상 기본 데이터베이스에서 모든 읽기 전용 복제본으로 병렬로 전송됩니다.
- 탑재된 상태 - 지원되지 않습니다.

읽기 전용 복제본 생성

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 기존 DB 인스턴스에서 읽기 전용 복제본을 생성할 수 있습니다. 또한 `SourceDBInstanceIdentifier`를 지정하여 읽기 전용 복제본을 생성할 수도 있습니다. 이 코드는 복제하려는 원본 DB 인스턴스 식별자입니다.

읽기 전용 복제본을 생성하면 Amazon RDS가 원본 DB 인스턴스의 DB 스냅샷을 캡처하고 복제를 시작합니다. DB 스냅샷 작업이 시작될 때 원본 DB 인스턴스에서 매우 짧은 I/O 중단이 발생합니다. 이러한 I/O 중단은 일반적으로 1초 정도 지속됩니다. 원본 DB 인스턴스가 다중 AZ 배포인 경우에는 I/O 중단을 방지할 수 있습니다. 이 경우에는 보조 DB 인스턴스에서 스냅샷을 생성하기 때문입니다.

활성 상태의 장기 실행 트랜잭션은 읽기 전용 복제본 생성 프로세스를 늦출 수 있습니다. 읽기 전용 복제본을 생성하기 전에 장기 실행 트랜잭션이 완료되기를 기다리는 것이 좋습니다. 동일한 원본 DB 인스턴스에서 다수의 읽기 전용 복제본을 병렬 방식으로 생성하는 경우에는 Amazon RDS가 첫 번째 생성 작업을 시작하면서 한 번만 스냅샷을 캡처합니다.

읽기 전용 복제본을 생성할 때는 몇 가지 고려할 사항이 있습니다. 첫째, 백업 보존 기간을 0이 아닌 다른 값으로 설정하여 원본 DB 인스턴스의 자동 백업을 활성화해야 합니다. 이 요구 사항은 다른 읽기 전용 복제본의 원본 DB 인스턴스인 읽기 전용 복제본에도 적용됩니다. RDS for MySQL 읽기 전용 복제본에서 자동 백업을 활성화하려면 먼저 읽기 전용 복제본을 생성한 다음 읽기 전용 복제본을 수정하여 자동 백업을 활성화합니다.

> **Note**
>
> AWS 리전 내에서 모든 읽기 전용 복제본은 소스 DB 인스턴스와 동일한 Amazon VPC 기반의 Virtual Private Cloud(VPC)에 생성하는 것이 좋습니다. 원본 DB 인스턴스와 다른 VPC에 읽기

전용 복제본을 생성하는 경우 CIDR(Classless Inter-Domain Routing) 범위가 복제본과 RDS 시스템 간에 겹칠 수 있습니다. CIDR이 겹치면 복제본이 불안정해져 복제본에 연결하는 애플리케이션에 부정적인 영향을 줄 수 있습니다. 읽기 전용 복제본을 생성할 때 오류가 발생하면 다른 대상 DB 서브넷 그룹을 선택합니다. 자세한 내용은 VPC에서 DB 인스턴스를 사용한 작업 섹션을 참조하세요.
콘솔이나 AWS CLI를 사용하여 다른 AWS 계정에서 읽기 전용 복제본을 직접 만들 수 있는 방법은 없습니다.

콘솔

소스 DB 인스턴스에서 읽기 복제본을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 읽기 전용 복제본의 소스로 사용할 DB 인스턴스를 선택합니다.
4. 작업에서 읽기 전용 복제본 만들기를 선택합니다.
5. DB 인스턴스 식별자에 읽기 전용 복제본의 이름을 입력합니다.
6. 인스턴스 구성을 선택합니다. 읽기 전용 복제본의 소스 DB 인스턴스보다 크거나 같은 DB 인스턴스 클래스와 스토리지 유형을 사용하는 것이 좋습니다.
7. AWS 리전에서 읽기 전용 복제본의 대상 리전을 지정합니다.
8. 스토리지에서는 할당된 스토리지 크기 및 스토리지 AutoScaling 사용 여부 등을 지정합니다.

   소스 DB 인스턴스가 최신 스토리지 구성을 사용하지 않는 경우 스토리지 파일 시스템 구성 업그레이드 옵션을 사용할 수 있습니다. 이 설정을 활성화하여 읽기 전용 복제본의 스토리지 파일 시스템을 원하는 구성으로 업그레이드할 수 있습니다. 자세한 내용은 the section called “스토리지 파일 시스템 업그레이드” 섹션을 참조하세요.
9. 사용 가능에서 복제본에 대한 장애 조치 지원을 위해 다른 가용 영역에 예비 복제본을 생성할 것인지 선택합니다.

> **Note**
>
> 읽기 전용 복제본을 다중 AZ DB 인스턴스로 생성하는 작업은 원본 데이터베이스가 다중 AZ DB 인스턴스인지 여부와는 무관합니다.

10. 다른 DB 인스턴스 설정을 지정합니다. 각 사용 가능 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하세요.
11. 암호화된 읽기 전용 복제본을 생성하려면 추가 구성을 펼치고 다음 설정을 지정합니다.

    a. 암호화 활성을 선택합니다.

    b. AWS KMS key의 경우 KMS 키의 AWS KMS key 식별자를 선택합니다.

> **Note**
>
> 원본 DB 인스턴스를 암호화해야 합니다. 원본 DB 인스턴스를 암호화하는 방법에 대해 자세히 알아보려면 Amazon RDS 리소스 암호화 단원을 참조하세요.

12. 읽기 전용 복제본 생성을 선택합니다.

읽기 전용 복제본이 생성되면 RDS 콘솔의 [데이터베이스(Databases)] 페이지에서 확인할 수 있습니다. [역할(Role)] 열에 [복제본(Replica)]이 표시됩니다.

AWS CLI

소스 DB 인스턴스에서 읽기 복제본을 생성하려면 AWS CLI 명령 create-db-instance-read-replica를 사용합니다. 또한 이 예에서는 할당된 스토리지 크기를 설정하고 스토리지 AutoScaling을 활성화하고 파일 시스템을 원하는 구성으로 업그레이드합니다.

다른 설정을 지정할 수 있습니다. 각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하세요.

> **Note**
>
> RDS for Db2 대기 복제본을 생성하려면 다음 AWS CLI 명령에 `--replica-mode mounted`를 포함합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance-read-replica \
    --db-instance-identifier myreadreplica \
```

```
    --source-db-instance-identifier mydbinstance \
    --allocated-storage 100 \
    --max-allocated-storage 1000 \
    --upgrade-storage-config
```

Windows의 경우:

```
aws rds create-db-instance-read-replica ^
    --db-instance-identifier myreadreplica ^
    --source-db-instance-identifier mydbinstance ^
    --allocated-storage 100 ^
    --max-allocated-storage 1000 ^
    --upgrade-storage-config
```

RDS API

소스 MySQL, MariaDB, Oracle, PostgreSQL 또는 SQL Server DB 인스턴스에서 읽기 전용 복제본을 생성하려면 다음 필수 파라미터를 사용하여 Amazon RDS API CreateDBInstanceReadReplica 작업을 직접 호출합니다.

- `DBInstanceIdentifier`
- `SourceDBInstanceIdentifier`

> **Note**
>
> RDS for Db2 대기 복제본을 생성하려면 선택 사항인 `ReplicaMode` 작업을 `mounted`로 설정합니다.

읽기 전용 복제본을 독립 DB 인스턴스로 승격

읽기 전용 복제본을 독립 실행형 DB 인스턴스로 승격할 수 있습니다. 원본 DB 인스턴스에 읽기 전용 복제본이 다수 있을 경우 읽기 전용 복제본 중 하나를 DB 인스턴스로 승격하더라도 나머지 복제본에는 아무런 영향도 끼치지 않습니다.

읽기 전용 복제본을 승격할 때 RDS는 DB 인스턴스를 사용할 수 있도록 하기 전에 DB 인스턴스를 재부팅합니다. 승격 프로세스는 읽기 전용 복제본의 크기에 따라 완료하는 데 몇 분 또는 더 오래도 걸릴 수 있습니다.



읽기 전용 복제본 승격 사용 사례

다음과 같은 이유로 읽기 전용 복제본을 독립형 DB 인스턴스로 승격할 수 있습니다.

- 장애 복구 실행(Implementing failure recovery) – 기본 DB 인스턴스에 장애가 발생할 경우 읽기 전용 복제본을 데이터 복구 체계로 사용할 수 있습니다. 이 방법은 동기식 복제, 장애 자동 감지 및 장애 조치를 보완합니다.

비동기식 복제의 영향이나 한계에 대해 알고 있더라도 데이터 복구를 위해 읽기 전용 복제본 승격을 사용할 수 있습니다. 이를 위해 먼저 읽기 전용 복제본을 생성한 다음 기본 DB 인스턴스의 장애 여부를 모니터링해야 합니다. 그 결과 장애가 발견된 경우에는 다음과 같이 실행합니다.

1. 읽기 전용 복제본을 승격합니다.
2. 데이터베이스 트래픽을 승격된 DB 인스턴스로 유도합니다.
3. 승격된 DB 인스턴스를 원본으로 하는 교체용 읽기 전용 복제본을 생성합니다.

- 스토리지 구성 업그레이드 -소스 DB 인스턴스가 원하는 스토리지 구성이 아닌 경우, 인스턴스의 읽기 전용 복제본을 만들고 스토리지 파일 시스템 구성을 업그레이드할 수 있습니다. 이 옵션은 읽기 전용 복제본의 파일 시스템을 원하는 구성으로 마이그레이션합니다. 그런 다음 읽기 전용 복제본을 독립 실행형 인스턴스로 승격할 수 있습니다.

  이 옵션을 사용하면 이전 32비트 파일 시스템의 스토리지 및 파일 크기 조정 제한을 극복할 수 있습니다. 자세한 내용은 the section called “스토리지 파일 시스템 업그레이드” 섹션을 참조하세요.

  이 옵션은 소스 DB 인스턴스가 최신 스토리지 구성을 사용하지 않거나 동일한 요청으로 DB 인스턴스 클래스를 수정하는 경우에만 사용할 수 있습니다.

- 샤딩 – 샤딩이란 "무공유(share-nothing)" 아키텍처를 구현함으로써 기본적으로 대규모 데이터베이스를 다수의 소규모 데이터베이스로 분할하는 기술입니다. 데이터베이스 분할은 동일한 쿼리에 조인되지 않은 테이블을 다른 호스트로 분할하는 방법이 일반적입니다. 그 밖에 테이블을 여러 호스트로 복사한 후 해싱 알고리즘을 사용해 어떤 호스트를 업데이트할지 결정하는 방법도 있습니다. 각 샤드(소규모 데이터베이스)에 해당하는 읽기 전용 복제본을 생성한 후 독립된 샤드로 변환하기로 결정 시 이러한 복제본을 승격할 수 있습니다. 그런 다음 각 샤드마다 요건에 따라 키 범위(행을 분할한 경우)나 배포된 테이블을 얻을 수 있습니다.
- DDL 작업 실행(MySQL 및 MariaDB에 한함) – 인덱스를 생성하거나 리빌드하는 등의 DDL 작업은 시간이 걸릴 뿐만 아니라 DB 인스턴스에 상당한 성능 저하를 초래할 수 있습니다. MySQL 또는 MariaDB 읽기 전용 복제본에서 이러한 작업을 수행하려면 먼저 읽기 전용 복제본이 기본 DB 인스턴스와 동기화되어 있어야 합니다. 그런 다음 읽기 전용 복제본을 승격해야 애플리케이션이 승격된 인스턴스를 사용하도록 유도할 수 있습니다.

> **Note**
>
> 읽기 전용 복제본이 RDS for Oracle DB 인스턴스인 경우 승격 대신 전환을 수행할 수 있습니다. 전환 시 소스 DB 인스턴스는 새 복제본이 되고 복제본은 새 소스 DB 인스턴스가 됩니다. 자세한 내용은 Oracle Data Guard 전환 수행 섹션을 참조하세요.

승격된 읽기 전용 복제본의 특징

일단 읽기 전용 복제본을 승격하고 나면 읽기 전용 복제본은 더 이상 읽기 전용 복제본으로 기능하지 않고 독립형 DB 인스턴스가 됩니다. 새 독립형 DB 인스턴스의 특성은 다음과 같습니다.

- 독립형 DB 인스턴스는 승격 전 읽기 전용 복제본의 옵션 그룹 및 파라미터 그룹을 그대로 보존합니다.
- 독립형 DB 인스턴스에서 읽기 전용 복제본을 생성하고 특정 시점 복원 작업을 수행할 수 있습니다.
- DB 인스턴스는 더 이상 읽기 전용 복제본이 아니므로, 복제 대상으로 사용할 수 없습니다.

읽기 전용 복제본을 승격하기 위한 사전 조건

읽기 전용 복제본을 승격하려면 먼저 다음을 수행하세요.

- 백업 전략을 검토합니다.
  - 백업을 활성화하고 하나 이상의 백업을 완성하는 것이 좋습니다. 백업 기간은 이전 백업 이후 데이터베이스에 대해 이루어진 변경 횟수를 지정하는 기능입니다.
  - 읽기 전용 복제본에 대해 백업을 활성화한 경우 일일 백업이 읽기 전용 복제본 승격을 방해하지 않도록 자동 백업 기간을 구성하십시오.
  - 읽기 전용 복제본이 `backing-up` 상태가 아닌지 확인합니다. 이 상태에서는 읽기 전용 복제본을 승격할 수 없습니다.
- 트랜잭션이 기본 DB 인스턴스에 기록되지 않도록 한 다음 RDS가 모든 업데이트를 읽기 전용 복제본에 적용할 때까지 기다립니다.

  읽기 전용 복제본에서 수행된 데이터베이스 업데이트는 기본 DB 인스턴스의 업데이트가 끝난 후에 이어집니다. 복제 지연은 상당히 다양하게 나타날 수 있습니다. `Replica Lag` 지표를 사용하여 읽기 전용 복제본의 업데이트가 모두 완료되는 시간을 측정합니다.
- (MySQL 및 MariaDB에 한함) MySQL 또는 MariaDB 읽기 전용 복제본을 승격하기 전에 변경하려면 읽기 전용 복제본의 DB 파라미터 그룹에서 `read_only` 파라미터를 `0`으로 설정합니다. 그래야만 인덱스 생성 등 필요한 DDL 작업을 모두 읽기 전용 복제본에서 실행할 수 있기 때문입니다. 읽기 전용 복제본에서 실행하는 작업은 기본 DB 인스턴스의 성능에 아무런 영향도 끼치지 않습니다.

읽기 전용 복제본 승격: 기본 단계

다음 단계는 읽기 전용 복제본을 DB 인스턴스로 승격하기 위한 일반적인 프로세스입니다.

1. Amazon RDS 콘솔의 승격(Promote) 옵션, AWS CLI 명령 `promote-read-replica` 또는 `PromoteReadReplica` Amazon RDS API 작업을 사용하여 읽기 전용 복제본을 승격합니다.

   > **Note**
   >
   > 승격 프로세스는 완료할 때까지 몇 분 걸립니다. 읽기 전용 복제본을 승격하면 RDS가 복제를 중지하고 읽기 전용 복제본을 재부팅합니다. 재부팅이 완료되면 읽기 전용 복제본을 새 DB 인스턴스로 사용할 수 있습니다.

2. (선택 사항) 다중 AZ 배포가 되도록 새 DB 인스턴스를 수정합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 및 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 단원을 참조하세요.

콘솔

읽기 전용 복제본을 독립 실행형 DB 인스턴스로 승격하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 콘솔에서 데이터베이스를 선택합니다.

   데이터베이스 창이 표시됩니다. 각 읽기 전용 복제본은 역할 열에 복제본이라고 표시됩니다.
3. 승격시키려는 읽기 전용 복제본을 선택합니다.
4. 작업에서 Promote(승격)를 선택합니다.
5. 읽기 전용 복제본 승격(Promote Read Replica) 페이지에서 새롭게 승격된 DB 인스턴스의 백업 보존 기간과 백업 기간을 입력합니다.
6. 원하는 대로 설정되었으면 [Continue]를 선택합니다.
7. 승인 페이지에서 [Promote Read Replica]를 선택합니다.

AWS CLI

읽기 전용 복제본을 독립 실행형 DB 인스턴스로 승격하려면 AWS CLI `promote-read-replica` 명령을 사용합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds promote-read-replica \
```

```
    --db-instance-identifier myreadreplica
```

Windows의 경우:

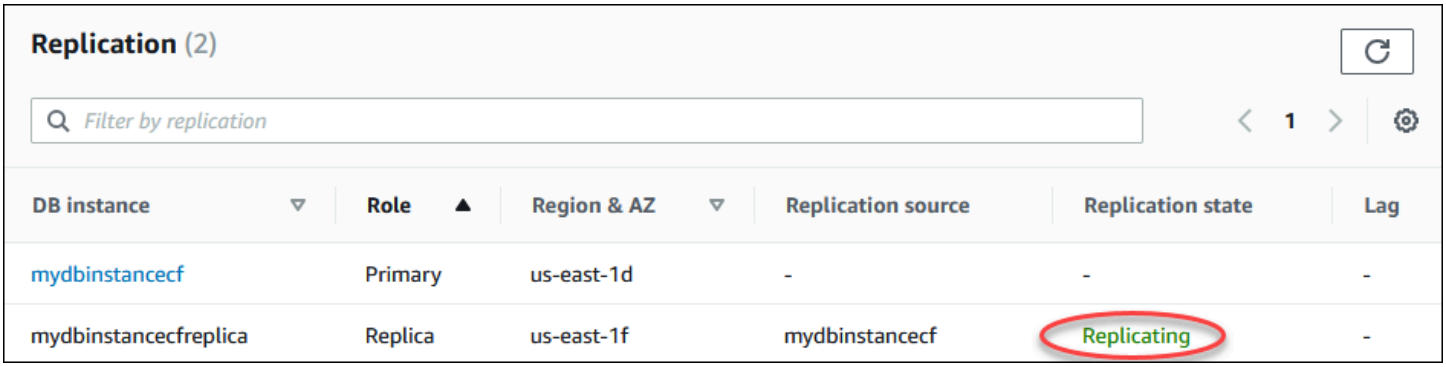
```
aws rds promote-read-replica ^
    --db-instance-identifier myreadreplica
```

RDS API

읽기 전용 복제본을 독립형 DB 인스턴스로 승격하려면 필수 파라미터 `DBInstanceIdentifier`를 사용하여 Amazon RDS API `PromoteReadReplica` 작업을 호출합니다.

읽기 전용 복제본 모니터링

읽기 전용 복제본의 상태는 몇 가지 방법으로 모니터링할 수 있습니다. Amazon RDS 콘솔의 읽기 전용 복제본 세부 정보에 있는 연결 및 보안(Connectivity & security) 탭의 복제(Replication) 섹션에 읽기 전용 복제본의 상태가 표시됩니다. 읽기 전용 복제본의 세부 정보를 보려면 Amazon RDS 콘솔의 DB 인스턴스 목록에서 읽기 전용 복제본의 이름을 선택합니다.



AWS CLI `describe-db-instances` 명령 또는 Amazon RDS API `DescribeDBInstances` 작업을 사용하여 읽기 전용 복제본의 상태도 볼 수 있습니다.

읽기 전용 복제본의 상태는 다음 중 한 가지가 될 수 있습니다.

- 복제 중 – 읽기 전용 복제본이 성공적으로 복제되고 있습니다.
- 복제 성능 저하됨(SQL Server 및 PostgreSQL만 해당) – 복제본이 기본 인스턴스에서 데이터를 수신하지만, 하나 이상의 데이터베이스가 업데이트를 가져오지 못할 수 있습니다. 예를 들어, 복제본이 새로 생성된 데이터베이스를 설정하는 동안 이러한 문제가 발생할 수 있습니다. 블루/그린 배포의 블루 환경에서 지원되지 않는 DDL이나 대규모 객체 변경이 이루어진 경우에도 발생할 수 있습니다.

  성능이 저하된 상태에서 오류가 발생하지 않는 한 상태가 `replication degraded`에서 `error`로 전환되지 않습니다.

- 오류 – 복제 중에 오류가 발생했습니다. Amazon RDS 콘솔의 [Replication Error] 필드나 이벤트 로그를 검사하여 정확한 오류 원인을 찾아내야 합니다. 복제 오류의 문제 해결에 대한 자세한 내용은 MySQL 읽기 전용 복제본의 문제 해결 단원을 참조하십시오.
- 종료됨(MariaDB, MySQL 또는 PostgreSQL만 해당) – 복제가 종료됩니다. 수동으로 또는 복제 오류로 인해 연속하여 30일 이상 복제가 중지된 경우 이 오류가 발생합니다. 이 경우 Amazon RDS는 기본 DB 인스턴스와 모든 읽기 전용 복제본 간의 복제를 종료합니다. Amazon RDS는 소스 DB 인스턴스에 대한 스토리지 요건 강화 및 장애 조치 장기화를 방지하기 위해 이 작업을 수행합니다.

  복제가 중단되면 대용량의 오류 메시지를 로그에 써야 하기 때문에 로그 크기와 수가 증가하면서 스토리지에 영향을 끼칠 수 있습니다. 또한 Amazon RDS가 복구 단계에서 다수의 로그를 유지 및 처리하는 데 필요한 시간 때문에 장애 복구에 영향을 끼칠 수도 있습니다.
- 종료됨(Oracle만 해당) - 복제가 종료되었습니다. 읽기 전용 복제본에 남아 있는 스토리지가 충분하지 않아 복제가 8시간 이상 중지된 경우 이 오류가 발생합니다. 이 경우 Amazon RDS는 기본 DB 인스턴스와 영향을 받는 읽기 전용 복제본 간의 복제를 종료합니다. 이 상태는 터미널 상태이며, 읽기 전용 복제본을 다시 만들어야 합니다.
- 중지됨(MariaDB 또는 MySQL만 해당) – 고객이 시작한 요청으로 복제가 중지되었습니다.
- replication stop point set(복제 중지 지점 설정됨)(MySQL만 해당) – 고객이 시작한 중지 지점이 mysql.rds_start_replication_until 저장 프로시저를 사용하여 설정되었으며 복제가 진행 중입니다.
- replication stop point reached(복제 중지 지점 도달됨)(MySQL만 해당) – 고객이 시작한 중지 지점이 mysql.rds_start_replication_until 저장 프로시저를 사용하여 설정되었으며 중지 지점에 도달했기 때문에 복제가 중지됩니다.

DB 인스턴스가 복제되는 위치를 확인할 수 있으며, DB 인스턴스가 복제되는 경우 복제 상태를 확인할 수 있습니다. RDS 콘솔의 [데이터베이스(Databases)] 페이지에서 [역할(Role)] 열에 [기본(Primary)]이 표시됩니다. DB 인스턴스 이름을 선택합니다. 세부 정보 페이지의 [연결 및 보안(Connectivity & security)] 탭에서 복제 상태는 [복제(Replication)] 아래에 있습니다.

복제 모니터링 지연 시간

Amazon CloudWatch에서 Amazon RDS `ReplicaLag` 지표를 보고 복제 지연을 모니터링할 수 있습니다.

Db2의 경우 `ReplicaLag` 지표는 지연된 데이터베이스의 최대 지연 시간(초)입니다. 예를 들어, 두 개의 데이터베이스가 각각 5초와 10초 지연되는 경우 `ReplicaLag`는 10초입니다. 사용 가능한 고가용성 재해 복구(HADR) 상태가 없는 데이터베이스는 계산에 포함되지 않습니다.

MySQL 및 MariaDB의 경우 ReplicaLag 지표가 Seconds_Behind_Master 명령의 SHOW REPLICA STATUS 필드 값을 보고합니다. 이렇게 MySQL 및 MariaDB에서 복제 지연이 발생하는 공통 원인은 다음과 같습니다.

- 네트워크 중단.
- 읽기 전용 복제본의 인덱스가 있는 테이블에 쓰기 작업 중일 때. 읽기 전용 복제본에서 read_only 파라미터가 0으로 설정되어 있지 않으면 복제가 중단될 수 있습니다.
- MyISAM과 같은 비트랜잭션 스토리지 엔진 사용. 복제는 MySQL용 InnoDB 스토리지 엔진 및 MariaDB용 XtraDB 스토리지 엔진에 대해서만 지원됩니다.

> **Note**
>
> 이전 버전의 MariaDB에는 SHOW SLAVE STATUS 대신 SHOW REPLICA STATUS가 사용되었습니다. 10.5 이하의 MariaDB 버전을 사용하는 경우 SHOW SLAVE STATUS를 사용합니다.

ReplicaLag 지표가 0에 도달하면 복제본이 기본 DB 인스턴스를 따라잡은 것입니다. ReplicaLag 지표가 -1을 반환하는 경우 복제가 현재 활성 상태가 아닙니다. ReplicaLag = -1은 Seconds_Behind_Master = NULL과 동등합니다.

Oracle의 경우, ReplicaLag 지표는 Apply Lag 값과 현재 시간과 적용 지연의 DATUM_TIME 값 차이를 합한 값입니다. DATUM_TIME 값은 읽기 전용 복제본이 마지막으로 원본 DB 인스턴스에서 데이터를 수신한 시간입니다. 자세한 내용은 Oracle 설명서에서 V$DATAGUARD_STATS를 참조하십시오.

SQL Server의 경우 ReplicaLag 지표는 지연된 데이터베이스의 최대 지연 시간(초)입니다. 예를 들어, 각각 5초와 10초가 지연되는 두 개의 데이터베이스가 각각 있는 경우 ReplicaLag는 10초입니다. ReplicaLag 지표는 다음 쿼리의 값을 반환합니다.

```
SELECT MAX(secondary_lag_seconds) max_lag FROM sys.dm_hadr_database_replica_states;
```

자세한 내용은 Microsoft 설명서의 secondary_lag_seconds를 참조하십시오.

ReplicaLag는 복제본 설정 중 또는 읽기 전용 복제본이 -1 상태에 있는 경우와 같이 RDS가 지연 시간을 확인할 수 없는 경우 error을 반환합니다.

> **Note**
>
> 새 데이터베이스는 읽기 전용 복제본에서 액세스할 수 있을 때까지 지연 시간 계산에 포함되지 않습니다.

PostgreSQL의 경우 `ReplicaLag` 지표는 다음 쿼리의 값을 반환합니다.

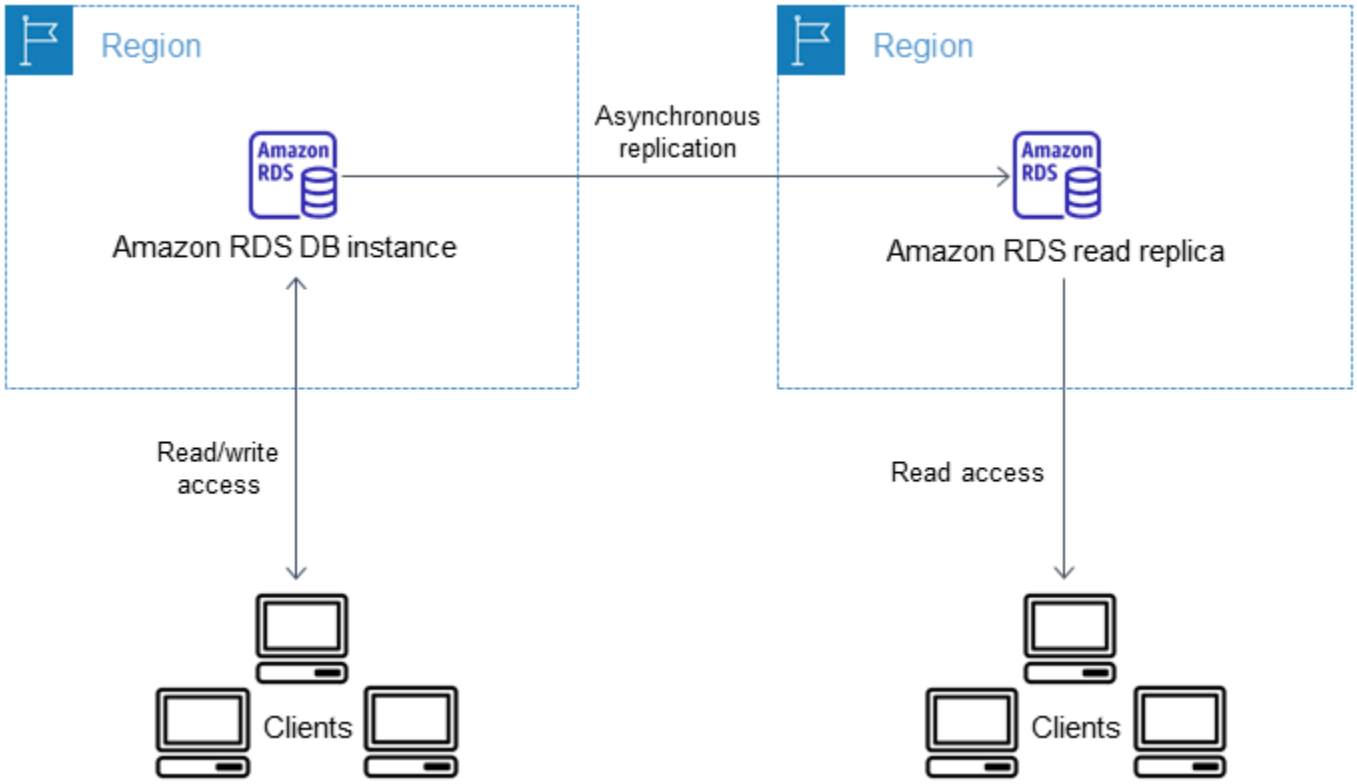
```
SELECT extract(epoch from now() - pg_last_xact_replay_timestamp()) AS reader_lag
```

PostgreSQL 버전 9.5.2 이상은 물리적인 복제 슬롯을 사용하여 원본 인스턴스에서 Write Ahead Log(WAL) 보존을 관리합니다. Amazon RDS가 리전 간 읽기 전용 복제본 인스턴스마다 물리적 복제 슬롯을 생성하여 인스턴스와 연동시킵니다. 2개의 Amazon CloudWatch 지표인 `Oldest Replication Slot Lag`와 `Transaction Logs Disk Usage`는 수신되는 WAL 데이터와 관련하여 가장 지체된 복제본이 얼마나 오래되었는지, 그리고 현재 WAL 데이터에 얼마나 많은 스토리지가 사용되고 있는지 나타냅니다. `Transaction Logs Disk Usage` 값은 리전 간 읽기 전용 복제본이 많이 지체될수록 크게 증가합니다.

CloudWatch를 통한 DB 인스턴스 모니터링에 대한 자세한 내용은 Amazon CloudWatch로 Amazon RDS 지표 모니터링 단원을 참조하세요.

다른 AWS 리전에서 읽기 전용 복제본 생성

Amazon RDS를 사용하면 소스 DB 인스턴스와 다른 AWS 리전에서 읽기 전용 복제본을 생성할 수 있습니다.



다른 AWS 리전에서 읽기 전용 복제본을 생성하여 다음을 수행합니다.

- 재해 복구 기능을 향상시킵니다.
- 읽기 작업을 사용자에 더 가까운 AWS 리전으로 조정합니다.
- 한 AWS 리전의 데이터 센터에서 다른 AWS 리전의 데이터 센터로 마이그레이션하는 작업을 더 용이하게 합니다.

소스 인스턴스와 다른 AWS 리전에서 읽기 전용 복제본을 생성하는 방법은 동일한 AWS 리전에서 복제본을 생성하는 것과 유사합니다. AWS Management Console을 사용하거나 `create-db-instance-read-replica` 명령을 실행하거나, `CreateDBInstanceReadReplica` API 작업을 호출할 수 있습니다.

> **Note**
>
> 소스 DB 인스턴스와 다른 AWS 리전에서 암호화된 읽기 전용 복제본을 생성하려면 소스 DB 인스턴스를 암호화해야 합니다.

주제

- 리전 및 버전 사용 가능 여부
- 리전 간 읽기 전용 복제본 생성
- Amazon RDS의 리전 간 복제 방법
- 리전 간 복제 시 고려 사항
- 리전 간 복제 비용

리전 및 버전 사용 가능 여부

기능 가용성 및 해당 지원은 각 데이터베이스 엔진의 특정 버전 및 AWS 리전에 따라 다릅니다. 교차 리전 복제를 통한 버전 및 리전 가용성에 관한 자세한 내용은 Amazon RDS에서 크로스 리전 읽기 전용 복제본을 지원하는 리전 및 DB 엔진 단원을 참조하세요.

리전 간 읽기 전용 복제본 생성

다음 절차에서는 다른 AWS 리전의 소스 MariaDB, Microsoft SQL Server, MySQL, Oracle 또는 PostgreSQL DB 인스턴스에서 읽기 전용 복제본을 생성하는 방법을 보여줍니다.

콘솔

AWS Management Console을 사용하여 여러 AWS 리전에 걸쳐 읽기 전용 복제본을 생성할 수 있습니다.

콘솔로 여러 AWS 리전에서 읽기 전용 복제본을 생성하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
3. 읽기 전용 복제본의 원본으로 사용할 MariaDB, Microsoft SQL Server, MySQL, Oracle 또는 PostgreSQL DB 인스턴스를 선택합니다.
4. 작업에서 읽기 전용 복제본 만들기를 선택합니다.
5. DB 인스턴스 식별자에 읽기 전용 복제본의 이름을 입력합니다.
6. 대상 리전(Destination Region)을 선택합니다.
7. 사용할 인스턴스 사양을 선택합니다. 읽기 전용 복제본에는 동일하거나 큰 DB 인스턴스 클래스와 스토리지 유형을 사용하는 것이 좋습니다.
8. 다른 AWS 리전에서 암호화된 읽기 전용 복제본을 생성하는 방법

a. 암호화 활성을 선택합니다.

b. AWS KMS key의 경우 대상 AWS 리전에 있는 KMS 키의 AWS KMS key 식별자를 선택합니다.

> **Note**
>
> 암호화된 읽기 전용 복제본을 생성하려면 원본 DB 인스턴스를 암호화해야 합니다. 원본 DB 인스턴스를 암호화하는 방법에 대해 자세히 알아보려면 Amazon RDS 리소스 암호화 단원을 참조하십시오.

9. 스토리지 Auto Scaling과 같은 다른 옵션을 선택합니다.
10. [Create read replica]를 선택합니다.

AWS CLI

다른 AWS 리전의 소스 MariaDB, Microsoft SQL Server, MySQL, Oracle 또는 PostgreSQL DB 인스턴스에서 읽기 전용 복제본을 생성하려면 `create-db-instance-read-replica` 명령을 사용할 수 있습니다. 이 경우 읽기 전용 복제본을 원하는 AWS 리전(대상 리전)에서 `create-db-instance-read-replica`를 사용하고 소스 DB 인스턴스의 Amazon 리소스 이름(ARN)을 지정하면 됩니다. ARN은 Amazon Web Services에서 생성된 리소스를 고유하게 식별합니다.

예를 들어, 원본 DB 인스턴스가 US East (N. Virginia) 리전에 있는 경우 ARN은 다음과 유사합니다.

```
arn:aws:rds:us-east-1:123456789012:db:mydbinstance
```

ARN에 대한 자세한 내용은 Amazon RDS의 Amazon 리소스 이름(ARN) 단원을 참조하세요.

소스 DB 인스턴스와 다른 AWS 리전에서 읽기 전용 복제본을 생성하려면 대상 AWS 리전에서 AWS CLI `create-db-instance-read-replica` 명령을 사용할 수 있습니다. 다른 AWS 리전에 읽기 전용 복제본을 생성하려면 다음 파라미터가 필요합니다.

- `--region` - 읽기 전용 복제본이 생성되는 대상 AWS 리전입니다.
- `--source-db-instance-identifier` – 소스 DB 인스턴스의 DB 인스턴스 식별자입니다. 이 식별자는 소스 AWS 리전용 ARN 형식으로 되어 있어야 합니다.
- `--db-instance-identifier` - 대상 AWS 리전에 있는 읽기 전용 복제본의 식별자입니다.

Example 리전 간 읽기 전용 복제본

다음 코드는 US East (N. Virginia) 리전의 소스 DB 인스턴스에서 미국 서부(오레곤) 리전에 읽기 전용 복제본을 생성합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance-read-replica \
    --db-instance-identifier myreadreplica \
    --region us-west-2 \
    --source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:mydbinstance
```

Windows의 경우:

```
aws rds create-db-instance-read-replica ^
    --db-instance-identifier myreadreplica ^
    --region us-west-2 ^
    --source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:mydbinstance
```

다른 AWS 리전에 암호화된 읽기 전용 복제본을 생성하려면 다음 파라미터도 필요합니다.

- `--kms-key-id` – 대상 AWS 리전에서 읽기 전용 복제본을 암호화하는 데 사용할 KMS 키의 AWS KMS key 식별자입니다.

Example 암호화된 리전 간 읽기 전용 복제본

다음 코드는 US East (N. Virginia) 리전의 소스 DB 인스턴스에서 미국 서부(오레곤) 리전에 암호화된 읽기 전용 복제본을 생성합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance-read-replica \
    --db-instance-identifier myreadreplica \
    --region us-west-2 \
    --source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:mydbinstance
 \
    --kms-key-id my-us-west-2-key
```

Windows의 경우:

```
aws rds create-db-instance-read-replica ^
    --db-instance-identifier myreadreplica ^
    --region us-west-2 ^
    --source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:mydbinstance
 ^
    --kms-key-id my-us-west-2-key
```

이 `--source-region` 옵션은 AWS GovCloud(미국 동부) 및 AWS GovCloud(미국 서부) 리전 간에 암호화된 읽기 전용 복제본을 생성할 때 필요합니다. `--source-region`의 경우 소스 DB 인스턴스의 AWS 리전을 지정합니다.

`--source-region`이 지정되지 않은 경우에는 `--pre-signed-url` 값을 지정합니다. 미리 서명된 URL은 소스 AWS 리전에서 호출되는 `create-db-instance-read-replica` 명령에 대한 서명 버전 4의 서명된 요청이 포함된 URL입니다. `pre-signed-url` 옵션에 대한 자세한 정보는 AWS CLI 명령 참조에서 create-db-instance-read-replica를 참조하세요.

RDS API

다른 AWS 리전의 소스 MariaDB, Microsoft SQL Server, MySQL, Oracle 또는 PostgreSQL DB 인스턴스에서 읽기 전용 복제본을 생성하려면 Amazon RDS API 작업 CreateDBInstanceReadReplica를 직접 호출하면 됩니다. 이 경우 읽기 전용 복제본을 원하는 AWS 리전(대상 리전)에서 CreateDBInstanceReadReplica를 호출하고 소스 DB 인스턴스의 Amazon 리소스 이름(ARN)을 지정합니다. ARN은 Amazon Web Services에서 생성된 리소스를 고유하게 식별합니다.

소스 DB 인스턴스와 다른 AWS 리전에서 암호화된 읽기 전용 복제본을 생성하려면 대상 AWS 리전에서 Amazon RDS API `CreateDBInstanceReadReplica` 작업을 사용할 수 있습니다. 다른 AWS 리전에서 암호화된 읽기 전용 복제본을 생성하려면 `PreSignedURL`의 값을 지정해야 합니다. `PreSignedURL`에는 읽기 전용 복제본이 생성된 소스 AWS 리전에서 호출할 `CreateDBInstanceReadReplica` 작업에 대한 요청이 포함되어야 합니다. `PreSignedUrl`에 대한 자세한 내용은 `CreateDBInstanceReadReplica`를 참조하세요.

예를 들어, 원본 DB 인스턴스가 US East (N. Virginia) 리전에 있는 경우, ARN은 다음과 유사한 모양을 띕니다.

```
arn:aws:rds:us-east-1:123456789012:db:mydbinstance
```

ARN에 대한 자세한 내용은 Amazon RDS의 Amazon 리소스 이름(ARN) 단원을 참조하십시오.

Example

```
https://us-west-2.rds.amazonaws.com/
    ?Action=CreateDBInstanceReadReplica
    &KmsKeyId=my-us-east-1-key
    &PreSignedUrl=https%253A%252F%252Frds.us-west-2.amazonaws.com%252F
         %253FAction%253DCreateDBInstanceReadReplica
         %2526DestinationRegion%253Dus-east-1
         %2526KmsKeyId%253Dmy-us-east-1-key
         %2526SourceDBInstanceIdentifier%253Darn%25253Aaws%25253Ards%25253Aus-
west-2%123456789012%25253Adb%25253Amydbinstance
         %2526SignatureMethod%253DHmacSHA256
         %2526SignatureVersion%253D4%2526SourceDBInstanceIdentifier%253Darn%25253Aaws
%25253Ards%25253Aus-west-2%25253A123456789012%25253Ainstance%25253Amydbinstance
         %2526Version%253D2014-10-31
         %2526X-Amz-Algorithm%253DAWS4-HMAC-SHA256
         %2526X-Amz-Credential%253DAKIADQKE4SARGYLE%252F20161117%252Fus-west-2%252Frds
%252Faws4_request
         %2526X-Amz-Date%253D20161117T215409Z
         %2526X-Amz-Expires%253D3600
         %2526X-Amz-SignedHeaders%253Dcontent-type%253Bhost%253Buser-agent%253Bx-amz-
content-sha256%253Bx-amz-date
         %2526X-Amz-Signature
%253D255a0f17b4e717d3b67fad163c3ec26573b882c03a65523522cf890a67fca613
    &DBInstanceIdentifier=myreadreplica
    &SourceDBInstanceIdentifier=&region-arn;rds:us-east-1:123456789012:db:mydbinstance
    &Version=2012-01-15
    &SignatureVersion=2
    &SignatureMethod=HmacSHA256
    &Timestamp=2012-01-20T22%3A06%3A23.624Z
    &AWSAccessKeyId=<&AWS; Access Key ID>
    &Signature=<Signature>
```

Amazon RDS의 리전 간 복제 방법

Amazon RDS는 다음 프로세스를 사용하여 리전 간 읽기 전용 복제본을 생성합니다. 이 프로세스는 관련된 AWS 리전과 데이터베이스의 데이터 양에 따라 완료하는 데 몇 시간이 걸리기도 합니다. 이 정보를 사용하여 리전 간 읽기 전용 복제본을 생성할 때 프로세스가 얼마나 진행되었는지 확인할 수 있습니다.

1. Amazon RDS가 원본 DB 인스턴스를 복제 원본으로 구성하기 시작하면서 상태를 modifying으로 설정합니다.

2. Amazon RDS가 대상 AWS 리전에 지정한 읽기 전용 복제본 설정을 시작하고 상태를 생성 중(creating)으로 설정합니다.
3. Amazon RDS가 소스 DB 인스턴스의 자동 DB 스냅샷을 소스 AWS 리전에 생성합니다. DB 스냅샷 이름은 `rds:<InstanceID>-<timestamp>`와 같은 형식을 갖습니다. 여기에서 `<InstanceID>`는 원본 인스턴스의 식별자이고, `<timestamp>`는 복제 시작일과 시간을 나타냅니다. 예를 들어 `rds:mysourceinstance-2013-11-14-09-24`는 인스턴스 `mysourceinstance`에서 `2013-11-14-09-24`에 생성되었다는 것을 의미합니다. 자동 DB 스냅샷의 생성 단계에서 원본 DB 인스턴스 상태는 수정 중으로 유지되고, 읽기 전용 복제본 상태는 생성 중으로 유지되고, DB 스냅샷 상태는 생성 중입니다. 콘솔의 [DB Snapshots] 페이지에서 [Progress] 열을 보면 DB 스냅샷 생성이 얼마나 진행되었는지 알 수 있습니다. DB 스냅샷이 완료되면 DB 스냅샷과 원본 DB 인스턴스의 상태가 모두 available로 설정됩니다.
4. Amazon RDS가 리전 간 스냅샷 복사를 시작하면서 첫 번째 데이터 전송이 이루어집니다. 스냅샷 사본은 대상 AWS 리전에 생성 중(creating) 상태와 함께 자동 스냅샷으로 등록됩니다. 이름은 원본 DB 스냅샷과 동일합니다. [DB Snapshots] 페이지에서 [Progress] 열을 보면 복사가 얼마나 진행되었는지 알 수 있습니다. 복사가 완료되면 DB 스냅샷 사본 상태가 available로 설정됩니다.
5. 그런 다음 Amazon RDS가 복사된 DB 스냅샷을 사용하여 읽기 전용 복제본에 처음으로 데이터를 로드하기 시작합니다. 이 단계에서 읽기 전용 복제본이 대상 DB 인스턴스 목록에 포함되며, 상태는 생성 중입니다. 로드가 완료되면 읽기 전용 복제본 상태가 사용 가능으로 설정되고, DB 스냅샷 복사본이 삭제됩니다.
6. 읽기 전용 복제본이 사용 가능 상태에 이르면 Amazon RDS가 읽기 전용 복제본 생성 작업을 시작한 이후 원본 인스턴스에 발생한 변경 사항을 복제하기 시작합니다. 이 단계에서 읽기 전용 복제본의 복제 지연 시간은 0보다 큽니다.

   복제 지연 시간에 대한 자세한 내용은 읽기 전용 복제본 모니터링 단원을 참조하십시오.

리전 간 복제 시 고려 사항

한 AWS 리전 내에서 복제할 때 고려해야 할 모든 사항이 교차 리전 복제 시에도 그대로 적용됩니다. 이 외에도 AWS 리전 간 복제 시 다음과 같은 사항을 추가로 고려해야 합니다.

- 소스 DB 인스턴스는 다수의 AWS 리전에서 교차 리전 읽기 전용 복제본을 가질 수 있습니다. 소스 VPC에 대한 ACL(액세스 제어 목록) 항목 수 제한으로 인해 RDS는 교차 리전 읽기 복제본 DB 인스턴스를 5개 이상 보장할 수 없습니다.
- GovCloud(US-East) 리전과 GovCloud(US-West) 리전 간에 복제할 수 있지만 GovCloud(US) 내부 또는 외부로 복제할 수는 없습니다.

- 다음 RDS 엔진의 경우, 교차 리전 Amazon RDS 읽기 전용 복제본은 다른 Amazon RDS DB 인스턴스의 읽기 전용 복제본이 아닌 경우에만 소스 Amazon RDS DB 인스턴스에서 생성할 수 있습니다.
  - RDS for SQL Server
  - RDS for Oracle
  - RDS for PostgreSQL 버전 14.1 미만

  이 제한은 RDS for PostgreSQL 버전 14.1 이상, RDS for MariaDB 및 RDS for MySQL을 실행하는 DB 인스턴스에는 적용되지 않습니다.
- 소스 인스턴스와 다른 AWS 리전에 읽기 전용 복제본을 생성할 때는 지연 시간이 증가할 수 있다는 점을 감안해야 합니다. 지역 데이터 센터 간 네트워크 채널이 긴 경우에 이러한 지연 시간이 발생합니다.
- 리전 간 읽기 전용 복제본의 경우 `--db-subnet-group-name` 파라미터를 지정하는 읽기 전용 복제본 생성 명령은 모두 DB 서브넷 그룹을 동일한 VPC에서 지정해야 합니다.
- 대부분의 경우 읽기 전용 복제본은 지정된 DB 엔진에 대해 기본 DB 파라미터 그룹 및 DB 옵션 그룹을 사용합니다.

  Db2 DB 엔진의 경우 복제본에 대한 사용자 지정 파라미터 그룹을 지정해야 합니다. BYOL 모델을 사용하는 복제본의 경우 이 사용자 지정 파라미터 그룹에 IBM Site ID 및 IBM Customer ID가 포함되어야 합니다. AWS Management Console, AWS CLI 또는 RDS API를 사용하여 복제본에 대해 이 사용자 지정 파라미터 그룹을 지정할 수 있습니다. 자세한 내용은 Db2에 기존 보유 라이선스 사용을 위한 IBM ID 섹션을 참조하세요.

  MySQL 및 Oracle DB 엔진의 경우 AWS CLI 명령 create-db-instance-read-replica의 `--db-parameter-group-name` 옵션으로 읽기 복제본에 대해 사용자 지정 파라미터 그룹을 지정할 수 있습니다. AWS Management Console을 사용할 때는 사용자 지정 파라미터 그룹을 지정할 수 없습니다.
- 읽기 전용 복제본은 기본 보안 그룹을 사용합니다.
- MariaDB, Microsoft SQL Server, MySQL 및 Oracle DB 인스턴스의 경우 교차 리전 읽기 전용 복제본의 소스 DB 인스턴스가 삭제되면 읽기 전용 복제본이 승격됩니다.
- PostgreSQL DB 인스턴스의 경우 교차 리전 읽기 전용 복제본의 소스 DB 인스턴스가 삭제되면 읽기 전용 복제본의 복제 상태가 `terminated`로 설정됩니다. 읽기 전용 복제본은 승격되지 않습니다.

  읽기 전용 복제본을 승격하거나 삭제해야 합니다.

리전 간 읽기 전용 복제본 요청

소스 리전과 통신하여 리전 간 읽기 전용 복제본 생성을 요청하려면 요청자(IAM 역할 또는 IAM 사용자)가 소스 DB 인스턴스와 소스 리전에 액세스할 수 있어야 합니다.

요청자의 IAM 정책에 있는 특정 조건으로 인해 요청이 실패할 수 있습니다. 다음 예에서는 소스 DB 인스턴스가 미국 동부(오하이오)에 있고 읽기 전용 복제본이 US East (N. Virginia)에 생성되는 것으로 가정합니다. 이 예에서는 요청을 실패하게 하는 요청자의 IAM 정책에 있는 조건을 보여 줍니다.

- 요청자의 정책에 `aws:RequestedRegion`에 대한 조건이 있습니다.

```
...
"Effect": "Allow",
"Action": "rds:CreateDBInstanceReadReplica",
"Resource": "*",
"Condition": {
    "StringEquals": {
        "aws:RequestedRegion": "us-east-1"
    }
}
```

  정책이 소스 리전에 대한 액세스를 허용하지 않기 때문에 요청이 실패합니다. 요청이 성공하려면 소스 리전과 대상 리전을 모두 지정합니다.

```
...
"Effect": "Allow",
"Action": "rds:CreateDBInstanceReadReplica",
"Resource": "*",
"Condition": {
    "StringEquals": {
        "aws:RequestedRegion": [
            "us-east-1",
            "us-east-2"
        ]
    }
}
```

- 요청자의 정책이 소스 DB 인스턴스에 대한 액세스를 허용하지 않습니다.

```
...
"Effect": "Allow",
"Action": "rds:CreateDBInstanceReadReplica",
```

```
"Resource": "arn:aws:rds:us-east-1:123456789012:db:myreadreplica"
...
```

요청이 성공하려면 소스 인스턴스와 복제본을 모두 지정합니다.

```
...
"Effect": "Allow",
"Action": "rds:CreateDBInstanceReadReplica",
"Resource": [
    "arn:aws:rds:us-east-1:123456789012:db:myreadreplica",
    "arn:aws:rds:us-east-2:123456789012:db:mydbinstance"
]
...
```

- 요청자의 정책이 `aws:ViaAWSService`를 거부합니다.

```
...
"Effect": "Allow",
"Action": "rds:CreateDBInstanceReadReplica",
"Resource": "*",
"Condition": {
    "Bool": {"aws:ViaAWSService": "false"}
}
```

소스 리전과의 통신은 요청자를 대신하여 RDS가 수행합니다. 요청이 성공하려면 AWS 서비스로부터의 호출을 거부하지 않아야 합니다.

- 요청자의 정책에 `aws:SourceVpc` 또는 `aws:SourceVpce`에 대한 조건이 있습니다.

RDS가 원격 리전을 호출할 때 지정된 VPC 또는 VPC 엔드포인트에서 수신된 요청이 아니기 때문에 요청이 실패할 수 있습니다.

요청을 실패하게 하는 이전 조건 중 하나를 사용해야 하는 경우 `aws:CalledVia`를 사용한 두 번째 문을 포함하여 요청이 성공하도록 할 수 있습니다. 예를 들어 다음과 같이 `aws:CalledVia`와 함께 `aws:SourceVpce`를 사용할 수 있습니다.

```
...
"Effect": "Allow",
"Action": "rds:CreateDBInstanceReadReplica",
"Resource": "*",
"Condition": {
```

```
    "Condition" : {
        "ForAnyValue:StringEquals" : {
          "aws:SourceVpce": "vpce-1a2b3c4d"
        }
     }
},
{
    "Effect": "Allow",
    "Action": [
        "rds:CreateDBInstanceReadReplica"
    ],
    "Resource": "*",
    "Condition": {
        "ForAnyValue:StringEquals": {
            "aws:CalledVia": [
                "rds.amazonaws.com"
            ]
        }
    }
}
```

자세한 내용은 IAM 사용 설명서에서 IAM의 정책 및 권한을 참조하세요.

읽기 전용 복제본 승인

리전 간 DB 읽기 전용 복제본 생성 요청에서 success가 반환되면 RDS가 백그라운드에서 복제본 생성을 시작합니다. RDS가 소스 DB 인스턴스에 액세스할 수 있는 권한이 생성됩니다. 이 승인은 소스 DB 인스턴스를 읽기 전용 복제본에 연결하고 RDS가 지정된 읽기 전용 복제본에만 복사할 수 있도록 합니다.

승인은 서비스 연결 IAM 역할에서 rds:CrossRegionCommunication 권한을 사용하여 RDS에 의해 확인됩니다. 복제본이 승인되면 RDS가 소스 리전과 통신하고 복제본 생성을 완료합니다.

RDS는 이전에 CreateDBInstanceReadReplica 요청에 의해 승인되지 않은 DB 인스턴스에 액세스할 권한이 없습니다. 읽기 전용 복제본 생성이 완료되면 승인이 취소됩니다.

RDS는 서비스 연결 역할을 사용하여 소스 리전에서 승인을 확인합니다. 복제 생성 프로세스 중에 서비스 연결 역할을 삭제하면 생성이 실패합니다.

자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 사용을 참조하십시오.

AWS Security Token Service 보안 인증 사용

전역 AWS Security Token Service(AWS STS) 엔드포인트의 세션 토큰은 기본적으로 활성화된 AWS 리전(상용 리전)에서만 유효합니다. `assumeRole`에서 AWS STS API 작업을 통해 얻은 자격 증명을 사용하는 경우 소스 리전이 옵트인 리전이면 리전별 엔드포인트를 사용합니다. 그렇지 않으면 요청이 실패합니다. 이는 자격 증명이 두 리전 모두에서 유효해야 하기 때문입니다. 옵트인 리전의 경우 리전별 AWS STS 엔드포인트를 사용할 때만 두 리전 모두에서 유효합니다.

전역 엔드포인트를 사용하려면 작업의 두 리전 모두에 대해 전역 엔드포인트를 사용하도록 설정되어 있어야 합니다. `Valid in all AWS ##` 계정 설정에서 AWS STS에 대해 전역 엔드포인트를 설정합니다.

미리 서명된 URL 파라미터의 보안 인증에도 동일한 규칙이 적용됩니다.

자세한 내용은 IAM 사용 설명서의 AWS 리전에서 AWS STS 관리를 참조하세요.

리전 간 복제 비용

리전 간 복제를 위해 데이터를 전송할 때는 Amazon RDS 데이터 전송 요금이 부과됩니다. 이러한 교차 리전 복제 작업에서 요금이 발생하는 이유는 다음과 같이 소스 AWS 리전을 벗어나 데이터를 전송하기 때문입니다.

- 읽기 전용 복제본을 생성할 때는 Amazon RDS가 소스 인스턴스의 스냅샷을 캡처하여 읽기 전용 복제본 AWS 리전으로 전송합니다.
- 소스 데이터베이스에서 데이터를 변경할 때마다 Amazon RDS가 소스 AWS 리전에서 읽기 전용 복제본 AWS 리전으로 데이터를 전송합니다.

데이터 전송 요금에 대한 자세한 내용은 Amazon RDS 요금을 참조하세요.

MySQL 및 MariaDB 인스턴스의 경우 리전 간 읽기 전용 복제본의 생성 수를 줄이면 데이터 전송 요금을 절감할 수 있습니다. 예를 들어 AWS 리전 하나에 1개의 소스 DB 인스턴스가 있으며, 다른 AWS 리전에 3개의 읽기 전용 복제본을 두기를 바란다고 가정해봅니다. 이 경우 원본 DB 인스턴스에서 읽기 전용 복제본 중 한 개만 생성합니다. 그런 다음 다른 두 개의 복제본은 원본 DB 인스턴스가 아닌 첫 번째 읽기 전용 복제본에서 생성합니다.

예를 들어 한 AWS 리전에 `source-instance-1`이 있는 경우 다음 작업을 수행할 수 있습니다.

- 새로운 AWS 리전에 `read-replica-1`을 생성한 후 `source-instance-1`을 소스로 지정합니다.
- `read-replica-2`를 `read-replica-1`에서 생성합니다.

- read-replica-3를 read-replica-1에서 생성합니다.

위 예에서 요금은 source-instance-1에서 read-replica-1로 데이터를 전송할 때만 발생합니다. read-replica-1에서 다른 두 복제본으로 데이터를 전송할 때는 동일한 AWS 리전에 모두 있기 때문에 요금이 부과되지 않습니다. 복제본 3개를 전부 다른 AWS 리전의 source-instance-1에서 직접 생성한다면 복제본 3개 모두에 대한 데이터 전송 요금이 발생합니다.

Amazon RDS 리소스에 태그 지정

Amazon RDS 태그는 사용자가 정의하고 DB 인스턴스 또는 DB 스냅샷과 같은 Amazon RDS 리소스와 연결하는 이름-값 페어입니다. 이 이름을 키라고 합니다. 원하는 경우 키에 값을 입력할 수 있습니다.

AWS Management Console, AWS CLI 또는 Amazon RDS API를 사용하여 Amazon RDS 리소스에서 태그를 추가, 나열 및 삭제할 수 있습니다. CLI 또는 API를 사용할 때 사용할 RDS 리소스에 대한 Amazon 리소스 이름(ARN)을 제공해야 합니다. ARN 생성에 대한 자세한 내용은 Amazon RDS의 ARN 구성 주제단원을 참조하십시오.

태그를 사용하여 Amazon RDS 리소스에 메타데이터를 추가할 수 있습니다. 태그를 사용하여 데이터베이스 인스턴스, 스냅샷, Aurora 클러스터 등에 대한 고유한 표기법을 추가할 수 있습니다. 이렇게 하면 Amazon RDS 리소스를 문서화하는 데 도움이 될 수 있습니다. 자동화된 유지 관리 절차와 함께 태그를 사용할 수도 있습니다.

특히 이러한 태그를 IAM 정책에 사용할 수 있습니다. 태그를 사용하여 Amazon RDS 리소스에 대한 액세스를 관리하고 해당 리소스에 적용 가능한 작업을 제어할 수 있습니다. 또한 비슷하게 태그가 지정된 리소스에 대한 비용을 그룹화하여 이러한 태그로 비용을 추적할 수 있습니다.

다음 Amazon RDS 리소스에 태그를 지정할 수 있습니다.

- DB 인스턴스
- DB 클러스터
- Aurora 글로벌 클러스터
- DB 클러스터 엔드포인트
- 읽기 전용 복제본
- DB 스냅샷
- DB 클러스터 스냅샷
- 예약 DB 인스턴스
- 이벤트 구독
- DB 옵션 그룹
- DB 파라미터 그룹
- DB 클러스터 파라미터 그룹
- DB 서브넷 그룹
- RDS 프록시

- RDS Proxy 엔드포인트
- 블루/그린 배포
- 제로 ETL 통합

> **Note**
>
> 현재는 AWS Management Console을 사용하여 RDS 프록시 및 RDS 프록시 엔드포인트에 태그를 지정할 수 없습니다.

주제

- Amazon RDS 리소스 태그를 사용하는 이유는 무엇인가요?
- Amazon RDS 리소스 태그 작동 방식
- Amazon RDS 리소스에 태그를 지정하는 모범 사례
- DB 스냅샷에 태그 복사
- Amazon RDS에서 태그 추가 및 삭제
- 자습서: 태그를 사용하여 중지할 DB 인스턴스를 지정합니다.

Amazon RDS 리소스 태그를 사용하는 이유는 무엇인가요?

태그를 사용하여 다음을 수행할 수 있습니다.

- 애플리케이션, 프로젝트, 부서, 환경 등을 기준으로 RDS 리소스를 분류하세요. 예를 들어 태그 키를 사용하여 범주를 정의하고 태그 값은 이 범주의 항목으로 정의할 수 있습니다. `environment=prod` 태그를 생성할 수도 있습니다. 또는 `project`의 태그 키와 `Salix`의 태그 값을 정의하여 Amazon RDS 리소스가 Salix 프로젝트에 할당되었음을 나타낼 수 있습니다.
- 리소스 관리 작업을 자동화하세요. 예를 들어, `environment=prod` 태그가 지정된 인스턴스에 대한 유지 보수 창을 `environment=test` 태그가 지정된 인스턴스에 대한 창과 다르게 만들 수 있습니다. `environment=prod` 태그가 지정된 인스턴스에 대한 자동 DB 스냅샷을 구성할 수도 있습니다.
- IAM 정책 내에서 RDS 리소스에 대한 액세스를 제어하세요. 전역 `aws:ResourceTag/`*`tag-key`* 조건 키를 사용하면 됩니다. 예를 들어, 정책에서 `DBAdmin` 그룹의 사용자만 `environment=prod` 태그가 지정된 DB 인스턴스를 수정하도록 허용할 수 있습니다. IAM 정책으로 태그가 지정된 리소스에 대한 액세스를 관리하는 방법에 대한 자세한 내용은 AWS ID 및 액세스 관리 사용 설명서에서 Amazon RDS의 자격 증명 및 액세스 관리 및 AWS에 대한 액세스 제어를 참조하세요.

- 태그를 기반으로 리소스를 모니터링하세요. 예를 들어, environment=prod 태그가 지정된 DB 인스턴스에 대한 Amazon CloudWatch 대시보드를 만들 수 있습니다.
- 유사하게 태그가 지정된 리소스에 대한 비용을 그룹화하여 비용을 추적하세요. 예를 들어, Salix 프로젝트와 관련된 RDS 리소스에 project=Salix 태그를 지정하면 이 프로젝트에 대한 비용 보고서를 생성하고 비용을 할당할 수 있습니다. 자세한 내용은 Amazon RDS에서 태그와 함께 AWS 결제가 작동하는 방식 단원을 참조하십시오.

Amazon RDS 리소스 태그 작동 방식

AWS은(는) 태그에 의미론적 의미를 적용하지 않습니다. 태그는 엄격히 문자열로 해석됩니다.

주제

- Amazon RDS의 태그 세트
- Amazon RDS의 태그 구조
- 태그 지정이 가능한 Amazon RDS 리소스
- Amazon RDS에서 태그와 함께 AWS 결제가 작동하는 방식

Amazon RDS의 태그 세트

모든 Amazon RDS 리소스에는 태그 세트라는 컨테이너가 있습니다. 컨테이너에는 리소스에 할당된 모든 태그가 포함됩니다. 리소스에는 정확히 하나의 태그 세트가 있습니다.

태그 세트 하나에는 0~50개의 태그가 포함됩니다. 기존 리소스 태그와 키가 동일한 태그를 RDS 리소스에 추가하면 새 값이 이전 값을 덮어씁니다.

Amazon RDS의 태그 구조

RDS 태그의 구조는 다음과 같습니다

태그 키

키는 태그의 필수 이름입니다. 문자열 값은 1~128자의 유니코드 문자 길이여야 하며 앞에 aws: 또는 rds: 접두사를 붙일 수 없습니다. 문자열에는 유니코드 문자, 숫자, 공백, _, ., :, /, =, +, - 및 @의 세트만 포함할 수 있습니다. Java 정규식은 "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-@]*)$"입니다. 태그 키는 대/소문자를 구분합니다. 따라서 project와 Project는 구별됩니다.

키는 태그 세트에 고유합니다. 예를 들어, 태그 세트에 project=Trinity와 project=Xanadu처럼 키는 같지만 값은 다른 키-페어가 있을 수 없습니다.

태그 값

값은 태그의 선택적 문자열 값입니다. 문자열 값은 1~256자의 유니코드 문자여야 합니다. 문자열에는 유니코드 문자, 숫자, 공백, _, ., :, /, =, +, - 및 @의 세트만 포함할 수 있습니다. Java 정규식은 "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-@]*)$"입니다. 태그 값은 대소문자를 구분합니다. 따라서 prod 값과 Prod은 구별됩니다.

태그 세트에서 값이 고유할 필요는 없으며 null일 수도 있습니다. 예를 들어 project=Trinity 및 cost-center=Trinity의 태그 세트에 키-값 페어가 있을 수 있습니다.

태그 지정이 가능한 Amazon RDS 리소스

다음 Amazon RDS 리소스에 태그를 지정할 수 있습니다.

- DB 인스턴스
- DB 클러스터
- DB 클러스터 엔드포인트
- 읽기 전용 복제본
- DB 스냅샷
- DB 클러스터 스냅샷
- 예약 DB 인스턴스
- 이벤트 구독
- DB 옵션 그룹
- DB 파라미터 그룹
- DB 클러스터 파라미터 그룹
- DB 서브넷 그룹
- RDS 프록시
- RDS Proxy 엔드포인트

> **Note**
>
> 현재는 AWS Management Console을 사용하여 RDS 프록시 및 RDS 프록시 엔드포인트에 태그를 지정할 수 없습니다.

- 블루/그린 배포
- 제로 ETL 통합(미리 보기)

Amazon RDS에서 태그와 함께 AWS 결제가 작동하는 방식

태그를 사용하여 비용 구조를 반영하도록 AWS 청구서를 구성합니다. 이렇게 하려면 가입하여 태그 키 값이 포함된 AWS 계정 청구서를 가져옵니다. 그런 다음 같은 태그 키 값을 가진 리소스에 따라 결제 정보를 구성하여 리소스 비용의 합을 볼 수 있습니다. 예를 들어, 특정 애플리케이션 이름으로 여러 리소스에 태그를 지정한 다음 결제 정보를 구성하여 여러 서비스에 걸친 해당 애플리케이션의 총 비용을 볼 수 있습니다. 자세한 내용은 AWS Billing 사용 설명서의 비용 할당 태그 사용을 참조하십시오.

비용 할당 태그가 DB 스냅샷에서 작동하는 방식

DB 스냅샷에 태그를 추가할 수 있습니다. 그러나 청구서에는 이러한 그룹화가 반영되지 않습니다. 비용 할당 태그를 DB 스냅샷에 적용하려면 다음 조건을 충족해야 합니다.

- 태그가 상위 DB 인스턴스에 연결되어 있어야 합니다.
- 상위 DB 인스턴스가 DB 스냅샷과 동일한 AWS 계정 계정에 존재해야 합니다.
- 상위 DB 인스턴스가 DB 스냅샷과 동일한 AWS 리전 계정에 존재해야 합니다.

DB 스냅샷이 상위 DB 인스턴스와 동일한 리전에 존재하지 않거나 상위 DB 인스턴스가 삭제된 경우 분리된 것으로 간주됩니다. 분리된 DB 스냅샷은 비용 할당 태그를 지원하지 않습니다. 분리된 스냅샷에 대한 비용은 태그가 지정되지 않은 품목에 집계됩니다. 다음 조건을 충족하는 경우 크로스 계정 DB 스냅샷은 분리된 것으로 간주되지 않습니다.

- 상위 DB 인스턴스와 동일한 리전에 있어야 합니다.
- 상위 DB 인스턴스는 소스 계정에서 소유합니다.

> **Note**
>
> 상위 DB 인스턴스가 다른 계정에서 소유하고 있는 경우, 대상 계정의 크로스 계정 스냅샷에는 비용 할당 태그가 적용되지 않습니다.

Amazon RDS 리소스에 태그를 지정하는 모범 사례

태그를 사용할 때는 다음 모범 사례를 준수하는 것이 좋습니다.

- 조직의 모든 팀이 따르는 태그 사용 규칙을 문서화하세요. 특히 이름을 설명적이면서도 일관성 있게 지어야 합니다. 예를 들어 일부 리소스에 `env:production`으로 태그를 지정하는 대신 `environment:prod` 형식으로 표준화하세요.

  > **Important**
  >
  > 개인 식별 정보(PII)나 기타 기밀 정보 또는 민감한 정보를 태그에 저장하지 마세요.

- 태그 지정을 자동화하여 일관성을 보장하세요. 예를 들어 다음과 같은 기술을 사용할 수 있습니다.
  - AWS CloudFormation 템플릿에 태그를 포함하세요. 템플릿으로 리소스를 만들면 리소스에 자동으로 태그가 지정됩니다.
  - AWS Lambda 함수를 사용하여 태그를 정의하고 적용하세요.
  - RDS 리소스에 태그를 추가하는 단계가 포함된 SSM 문서를 생성하세요.
- 필요한 경우에만 태그를 사용하세요. 단일 RDS 리소스에 대해 최대 50개의 태그를 추가할 수 있지만, 불필요한 태그의 증가와 복잡성을 피하는 것이 가장 좋습니다.
- 태그의 관련성과 정확성을 정기적으로 검토하세요. 필요에 따라 오래된 태그를 제거하거나 수정하세요.
- AWS Management Console의 AWS Tag Editor를 사용하여 태그를 생성해 보세요. Tag Editor를 사용하여 RDS 리소스를 포함하여 지원되는 여러 AWS 리소스에 동시에 태그를 추가할 수 있습니다. 자세한 내용은 AWS 리소스 그룹 사용 설명서의 Tag Editor를 참조하세요.

DB 스냅샷에 태그 복사

DB 인스턴스를 만들거나 복원할 경우 DB 인스턴스의 태그가 DB 인스턴스의 스냅샷으로 복사되도록 지정할 수 있습니다. 태그를 복사하는 것은 DB 스냅샷의 메타데이터가 원본 DB 인스턴스의 메타데이터와 일치하도록 보장합니다. 또한 DB 스냅샷의 액세스 정책이 원본 DB 인스턴스의 액세스 정책과 일치하도록 보장합니다.

다음 작업 시 DB 스냅샷으로 태그를 복사하도록 지정할 수 있습니다.

- DB 인스턴스 생성
- DB 인스턴스 복원
- 읽기 전용 복제본 생성
- DB 스냅샷 복사

이전 작업에 대한 태그를 복사하려면 AWS Management Console에서 스냅샷에 태그 복사를 선택하거나 AWS CLI에서 `--copy-tags-to-snapshot`을 지정합니다.

대부분의 경우 태그는 기본적으로 복사되지 않습니다. 그러나 DB 스냅샷에서 DB 인스턴스를 복원할 때 RDS는 새 태그를 지정하는지 여부를 확인합니다. 그렇다면 새 태그가 복원된 DB 인스턴스에 추가됩니다. 새 태그가 없는 경우 RDS는 스냅샷 생성 시 원본 DB 인스턴스의 태그를 복원된 DB 인스턴스에 추가합니다.

원본 DB 인스턴스의 태그가 복원된 DB 인스턴스에 추가되지 않도록 하려면 DB 인스턴스를 복원할 때 새 태그를 지정하는 것이 좋습니다.

> **Note**
>
> 경우에 따라 create-db-snapshot AWS CLI 명령의 `--tags` 파라미터 값을 포함할 수 있습니다. 또는 CreateDBSnapshot API 작업에 하나 이상의 태그를 제공할 수도 있습니다. 이러한 경우 RDS는 원본 DB 인스턴스에서 새 DB 스냅샷으로 태그를 복사하지 않습니다. 이 기능은 원본 DB 인스턴스에 `--copy-tags-to-snapshot`(CopyTagsToSnapshot) 옵션이 활성화되어 있어도 적용됩니다.
>
> 이 방법을 사용할 경우 DB 스냅샷에서 DB 인스턴스의 복사본을 생성할 수 있습니다. 이 접근 방식을 사용하면 새 DB 인스턴스에 적용되지 않는 태그를 추가하는 것을 피할 수 있습니다. AWS CLI `create-db-snapshot` 명령(또는 `CreateDBSnapshot` RDS API 작업)을 사용하여 DB 스냅샷을 생성합니다. DB 스냅샷을 생성한 후 이 주제의 뒷부분에서 설명된 대로 태그를 추가할 수 있습니다.

Amazon RDS에서 태그 추가 및 삭제

다음을 수행할 수 있습니다.

- 리소스를 생성할 때 태그를 생성합니다(예: AWS CLI 명령 `create-db-instance`를 실행할 때).
- `add-tags-to-resource` 명령을 사용하여 기존 리소스에 태그를 추가합니다.
- `list-tags-for-resource` 명령을 사용하여 특정 리소스와 연결된 태그를 나열합니다.
- `add-tags-to-resource` 명령을 사용하여 태그를 업데이트합니다.
- `remove-tags-from-resource` 명령을 사용하여 리소스에서 태그를 제거합니다.

다음 절차에서는 DB 인스턴스 와 관련된 리소스에 대해 일반적인 태그 지정 작업을 수행하는 방법을 설명합니다. 권한 부여 목적으로 태그가 캐시됩니다. 이러한 이유로 Amazon RDS 리소스에서 태그를 추가하거나 업데이트할 때 수정 사항을 사용할 수 있기까지 몇 분이 걸릴 수 있습니다.

콘솔

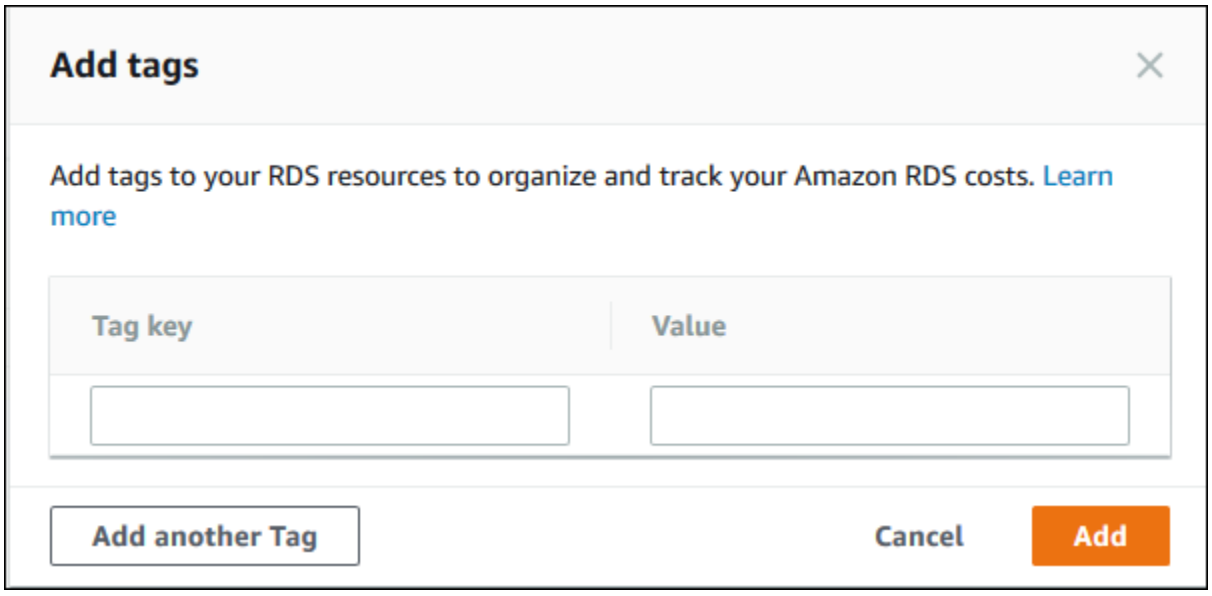
Amazon RDS 리소스에 태그를 지정하는 프로세스는 모든 리소스에서 비슷합니다. 다음 절차에서는 Amazon RDS DB 인스턴스에 태그를 지정하는 방법을 보여줍니다.

DB 인스턴스에 태그를 추가하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.

> **Note**
>
> DB 인스턴스 목록을 필터링하려면 데이터베이스 창의 Filter databases(데이터베이스 필터링)에 텍스트 문자열을 입력합니다. 해당 문자열을 포함하는 DB 인스턴스만 표시됩니다.

3. 세부 정보를 보기 위해 태그 지정하려는 DB 인스턴스의 이름을 선택합니다.
4. 세부 정보 섹션에서 아래에 있는 태그 섹션으로 스크롤합니다.
5. [추가]를 선택합니다. 태그 추가 창이 나타납니다.



6. 태그 키와 값에 값을 입력합니다.

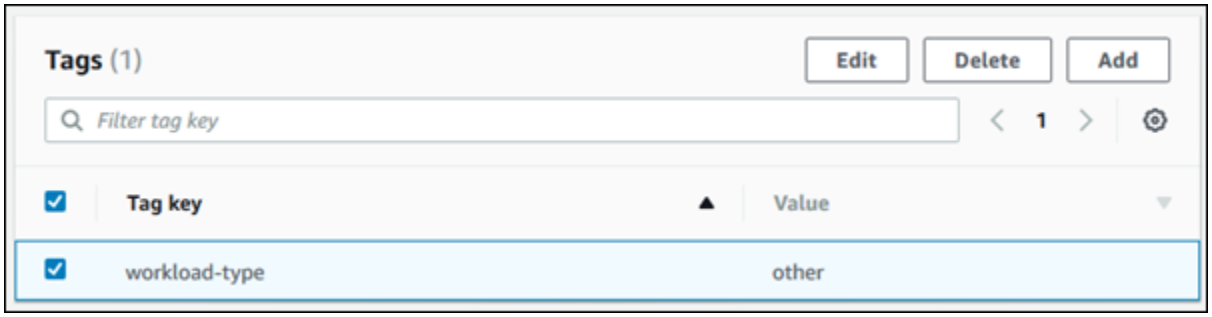
7. 다른 태그를 추가하려면 다른 태그 추가를 선택하고 태그 키와 값에 값을 입력합니다.

   이 단계를 필요한 만큼 반복합니다.
8. [추가]를 선택합니다.

DB 인스턴스에서 태그를 삭제하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.

   > **Note**
   >
   > DB 인스턴스 목록을 필터링하려면 데이터베이스 창의 Filter databases(데이터베이스 필터링) 상자에 텍스트 문자열을 입력합니다. 해당 문자열을 포함하는 DB 인스턴스만 표시됩니다.
3. 세부 정보를 표시할 DB 인스턴스의 이름을 선택합니다.
4. 세부 정보 섹션에서 아래에 있는 태그 섹션으로 스크롤합니다.
5. 삭제하려는 태그를 선택합니다.



6. Delete(삭제)를 선택한 다음 Delete tags(삭제 태그)창에서 Delete(삭제)를 선택합니다.

AWS CLI

AWS CLI 사용을 통해 DB 인스턴스에 대한 태그를 추가, 나열 또는 제거할 수 있습니다.

- Amazon RDS 리소스에 하나 이상의 태그를 추가하려면 AWS CLI 명령 `add-tags-to-resource`를 사용합니다.
- Amazon RDS 리소스의 태그를 나열하려면 AWS CLI 명령 `list-tags-for-resource`를 사용합니다.

- Amazon RDS 리소스에서 하나 이상의 태그를 삭제하려면 AWS CLI 명령 `remove-tags-from-resource`를 사용합니다.

필수 ARN을 생성하는 방법에 대해 자세히 알아보려면 Amazon RDS의 ARN 구성 단원을 참조하십시오.

RDS API

Amazon RDS API를 사용하여 DB 인스턴스에 대한 태그를 추가, 나열 또는 제거할 수 있습니다.

- Amazon RDS 리소스에 태그를 추가하려면 `AddTagsToResource` 작업을 사용합니다.
- Amazon RDS 리소스에 배정된 태그를 나열하려면 `ListTagsForResource`를 사용합니다.
- Amazon RDS 리소스에서 태그를 제거하려면 `RemoveTagsFromResource` 작업을 사용합니다.

필수 ARN을 생성하는 방법에 대해 자세히 알아보려면 Amazon RDS의 ARN 구성 단원을 참조하십시오.

Amazon RDS API를 사용한 XML 작업 시 다음 스키마를 사용하십시오.

```
<Tagging>
      <TagSet>
          <Tag>
              <Key>Project</Key>
              <Value>Trinity</Value>
          </Tag>
          <Tag>
              <Key>User</Key>
              <Value>Jones</Value>
          </Tag>
      </TagSet>
  </Tagging>
```

다음 표에는 허용되는 XML 태그와 해당 특성의 목록이 나와 있습니다. `Key`와 `Value`은 대/소문자를 구분합니다. 예를 들어 `project=Trinity`와 `PROJECT=Trinity`는 서로 다른 태그입니다.

태그 지정 요소	설명
TagSet	태그 세트에는 Amazon RDS 리소스에 배정된 모든 태그가 포함됩니다. 리소스당 하나의 태그 세트만 있을 수 있습니다. Amazon RDS API를 통해서만 TagSet로 작업합니다.
Tag	태그는 사용자가 정의하는 키-값 페어입니다. 태그 세트에 1~50개의 태그가 있을 수 있습니다.
키	키는 태그의 필수 이름입니다. 제한 사항에 대한 자세한 내용은 Amazon RDS의 태그 구조 단원을 참조하세요. 문자열 값은 길이가 1~128자인 유니코드 문자이며 `aws:` 또는 `rds:`를 접두사로 사용할 수 없습니다. 문자열에는 유니코드 문자, 숫자, 공백, '_', '.', '/', '=', '+', '-'(Java regex: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$")만 포함될 수 있습니다. 키는 태그 집합에 대해 고유해야 합니다. 예를 들어, 태그 세트에 project/Trinity와 project/Xanadu처럼 키는 같지만 값은 다른 키-페어가 있을 수 없습니다.
값	값은 태그의 선택적 값입니다. 제한 사항은 을 참조하십시오Amazon RDS의 태그 구조 단원을 참조하세요. 문자열 값은 길이가 1~256자인 유니코드 문자이며 `aws:` 또는 `rds:`를 접두사로 사용할 수 없습니다. 문자열에는 유니코드 문자, 숫자, 공백, '_', '.', '/', '=', '+', '-'(Java regex: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$")만 포함될 수 있습니다. 값은 태그 세트에서 고유할 필요는 없으며 null일 수 있습니다. 예를 들어, project/Trinity 및 cost-center/Trinity의 태그 세트에 키-값 페어가 있을 수 있습니다.

자습서: 태그를 사용하여 중지할 DB 인스턴스를 지정합니다.

이 자습서에서는 개발 또는 테스트 환경에 여러 개의 DB 인스턴스가 있다고 가정합니다. 며칠 동안 이러한 DB 인스턴스를 모두 유지해야 합니다. 일부 DB 인스턴스는 밤새 테스트를 실행하는 반면, 다른 인스턴스는 밤새 테스트를 중지했다가 다음 날 다시 시작할 수 있습니다.

다음 자습서에서는 밤새 중지하기에 적합한 DB 인스턴스에 태그를 할당하는 방법을 설명합니다. 이 자습서에서는 스크립트가 어떤 DB 인스턴스에 태그가 있는지 감지한 다음 태그가 지정된 DB 인스턴스

를 중지하는 방법을 보여 줍니다. 이 예제에서 키-값 쌍의 값 부분은 중요하지 않습니다. `stoppable` 태그가 있다는 것은 DB 인스턴스에 이 사용자 정의 속성이 있음을 의미합니다.

다음 자습서에서는 태그 지정을 위한 명령과 API가 ARN을 사용하여 작동하므로, RDS가 AWS 리전, AWS 계정 및 동일한 짧은 이름을 가질 수 있는 여러 유형의 리소스에서 원활하게 작동할 수 있습니다. DB 인스턴스에서 작동하는 CLI 명령에서 DB 인스턴스 ID 대신 ARN을 지정할 수 있습니다.

중지할 DB 인스턴스를 지정하려면

1. 중지 가능으로 지정하려는 DB 인스턴스의 ARN을 결정합니다.

   다음 예제에서는 *dev-test-db-instance*를 자체 DB 인스턴스의 이름으로 대체합니다. ARN 파라미터를 사용하는 후속 명령에서, 사용자 DB 인스턴스의 ARN을 대체합니다. ARN에는 사용자의 AWS 계정 ID와 DB 인스턴스가 위치한 AWS 리전의 이름이 포함되어 있습니다.

   ```
   $ aws rds describe-db-instances --db-instance-identifier dev-test-db-instance \
     --query "*[].{DBInstance:DBInstanceArn}" --output text
   arn:aws:rds:us-east-1:123456789102:db:dev-test-db-instance
   ```

2. 이 DB 인스턴스에 `stoppable` 태그를 추가합니다.

   이 태그의 이름은 사용자가 선택합니다. 이 예제에서는 태그를 있거나 없는 속성으로 취급하기 때문에 `Value=` 파라미터의 `--tags` 일부를 생략합니다. 이 방법을 사용하면 모든 관련 정보를 이름에 인코딩하는 명명 규칙을 만드는 것을 피할 수 있습니다. 이러한 규칙 하에서는 DB 인스턴스 이름 또는 다른 리소스의 이름에 정보를 인코딩할 수 있습니다.

   ```
   $ aws rds add-tags-to-resource \
     --resource-name arn:aws:rds:us-east-1:123456789102:db:dev-test-db-instance \
     --tags Key=stoppable
   ```

3. 태그가 DB 인스턴스에 있는지 확인합니다.

   다음 명령은 DB 인스턴스에 대한 태그 정보를 JSON 형식 및 탭으로 구분된 일반 텍스트로 검색합니다.

   ```
   $ aws rds list-tags-for-resource \
     --resource-name arn:aws:rds:us-east-1:123456789102:db:dev-test-db-instance
   {
       "TagList": [
           {
               "Key": "stoppable",
   ```

```
            "Value": ""

        }
    ]
}
aws rds list-tags-for-resource \
  --resource-name arn:aws:rds:us-east-1:123456789102:db:dev-test-db-instance --
output text
TAGLIST stoppable
```

4. stoppable로 지정된 모든 DB 인스턴스를 중지합니다.

   다음 예제에서는 모든 DB 인스턴스를 나열하는 텍스트 파일을 만듭니다. 셸 명령은 목록을 반복하여 각 DB 인스턴스에 관련 속성이 태그되어 있는지 확인한 후 각 DB 인스턴스에 대해 aws rds stop-db-instance 명령을 실행합니다.

```
$ aws rds describe-db-instances --query "*[].[DBInstanceArn]" --output text >/tmp/
db_instance_arns.lst
$ for arn in $(cat /tmp/db_instance_arns.lst)
do
  match="$(aws rds list-tags-for-resource --resource-name $arn --output text | grep
 stoppable)"
  if [[ ! -z "$match" ]]
  then
      echo "DB instance $arn is tagged as stoppable. Stopping it now."
# Note that you need to get the DB instance identifier from the ARN.
      dbid=$(echo $arn | sed -e 's/.*://')
      aws rds stop-db-instance --db-instance-identifier $dbid
  fi
done

DB instance arn:arn:aws:rds:us-east-1:123456789102:db:dev-test-db-instance is
 tagged as stoppable. Stopping it now.
{
    "DBInstance": {
        "DBInstanceIdentifier": "dev-test-db-instance",
        "DBInstanceClass": "db.t3.medium",
        ...
```

매일 일과가 끝날 때마다 위와 같은 스크립트를 실행하여 불필요한 DB 인스턴스가 중지되도록 할 수 있습니다. 매일 밤 이러한 검사를 수행하기 위해 cron 등의 유틸리티를 사용하여 작업을 예약할 수도

있습니다. 예를 들어, 일부 DB 인스턴스가 실수로 실행 중으로 남아 있지 않도록 하려면 검사를 예약하면 됩니다. 이때 검사할 DB 인스턴스 목록을 준비하는 명령을 세부 조정할 수 있습니다.

다음 명령은 DB 인스턴스 목록을 생성하지만 `available` 상태에 있는 인스턴스만 생성합니다. 스크립트는 이미 중지된 DB 인스턴스를 무시할 수 있습니다. 이러한 인스턴스는 `stopped` 또는 `stopping` 등 상태 값이 다르기 때문입니다.

```
$ aws rds describe-db-instances \
  --query '*[].{DBInstanceArn:DBInstanceArn,DBInstanceStatus:DBInstanceStatus}|[?
DBInstanceStatus == `available`]|[].{DBInstanceArn:DBInstanceArn}' \
  --output text
arn:aws:rds:us-east-1:123456789102:db:db-instance-2447
arn:aws:rds:us-east-1:123456789102:db:db-instance-3395
arn:aws:rds:us-east-1:123456789102:db:dev-test-db-instance
arn:aws:rds:us-east-1:123456789102:db:pg2-db-instance
```

> **Tip**
>
> 태그를 할당하고 해당 태그가 있는 DB 인스턴스를 찾는 기능을 사용하여 다른 방법으로 비용을 줄일 수 있습니다. 개발 및 테스트에 사용되는 DB 인스턴스의 시나리오를 예로 들어 보겠습니다. 이 경우 일과가 끝날 때 일부 DB 인스턴스를 삭제하도록 지정할 수 있습니다. 또는 사용량이 낮을 것으로 예상되는 시기에 DB 인스턴스를 소규모 DB 인스턴스 클래스로 변경하도록 지정할 수 있습니다.

Amazon RDS의 Amazon 리소스 이름(ARN)

Amazon Web Services에 생성되는 리소스는 각기 고유한 Amazon 리소스 이름(ARN)으로 식별됩니다. 특정 Amazon RDS 작업에서는 ARN을 지정하여 Amazon RDS 리소스를 고유한 이름으로 식별해야 합니다. 예를 들어, RDS DB 인스턴스 읽기 전용 복제본을 생성할 때 원본 DB 인스턴스에 대한 ARN을 제공해야 합니다.

ARN 구성 및 기존 ARN 가져오기에 대한 자세한 내용은 아래의 주제를 참조하세요.

주제

- Amazon RDS의 ARN 구성
- Amazon RDS의 기존 ARN 가져오기

Amazon RDS의 ARN 구성

Amazon Web Services에 생성되는 리소스는 각기 고유한 Amazon 리소스 이름(ARN)으로 식별됩니다. 다음 구문을 사용하여 Amazon RDS 리소스에 대한 ARN을 생성할 수 있습니다.

```
arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

리전 이름	지역	엔드포인트	프로토콜	
미국 동부(오하이오)	us-east-2	rds.us-east-2.amazonaws.com	HTTPS	
		rds-fips.us-east-2.api.aws	HTTPS	
		rds.us-east-2.api.aws	HTTPS	
		rds-fips.us-east-2.amazonaws.com	HTTPS	
미국 동부(버지니아 북부)	us-east-1	rds.us-east-1.amazonaws.com	HTTPS	
		rds-fips.us-east-1.api.aws	HTTPS	
		rds-fips.us-east-1.amazonaws.com	HTTPS	
		rds.us-east-1.api.aws	HTTPS	

리전 이름	지역	엔드포인트	프로토콜
미국 서부(캘리포니아 북부)	us-west-1	rds.us-west-1.amazonaws.com	HTTPS
		rds.us-west-1.api.aws	HTTPS
		rds-fips.us-west-1.amazonaws.com	HTTPS
		rds-fips.us-west-1.api.aws	HTTPS
미국 서부(오레곤)	us-west-2	rds.us-west-2.amazonaws.com	HTTPS
		rds-fips.us-west-2.amazonaws.com	HTTPS
		rds.us-west-2.api.aws	HTTPS
		rds-fips.us-west-2.api.aws	HTTPS
아프리카(케이프타운)	af-south-1	rds.af-south-1.amazonaws.com	HTTPS
		rds.af-south-1.api.aws	HTTPS
아시아 태평양(홍콩)	ap-east-1	rds.ap-east-1.amazonaws.com	HTTPS
		rds.ap-east-1.api.aws	HTTPS
아시아 태평양(하이데라바드)	ap-south-2	rds.ap-south-2.amazonaws.com	HTTPS
		rds.ap-south-2.api.aws	HTTPS
아시아 태평양(자카르타)	ap-southeast-3	rds.ap-southeast-3.amazonaws.com	HTTPS
		rds.ap-southeast-3.api.aws	HTTPS
아시아 태평양(말레이시아)	ap-southeast-5	rds.ap-southeast-5.amazonaws.com	HTTPS

리전 이름	지역	엔드포인트	프로토콜
아시아 태평양(멜버른)	ap-southeast-4	rds.ap-southeast-4.amazonaws.com rds.ap-southeast-4.api.aws	HTTPS HTTPS
아시아 태평양(뭄바이)	ap-south-1	rds.ap-south-1.amazonaws.com rds.ap-south-1.api.aws	HTTPS HTTPS
아시아 태평양(오사카)	ap-northeast-3	rds.ap-northeast-3.amazonaws.com rds.ap-northeast-3.api.aws	HTTPS HTTPS
아시아 태평양(서울)	ap-northeast-2	rds.ap-northeast-2.amazonaws.com rds.ap-northeast-2.api.aws	HTTPS HTTPS
아시아 태평양(싱가포르)	ap-southeast-1	rds.ap-southeast-1.amazonaws.com rds.ap-southeast-1.api.aws	HTTPS HTTPS
아시아 태평양(시드니)	ap-southeast-2	rds.ap-southeast-2.amazonaws.com rds.ap-southeast-2.api.aws	HTTPS HTTPS
아시아 태평양(타이베이)	ap-east-2	rds.ap-east-2.amazonaws.com	HTTPS
아시아 태평양(태국)	ap-southeast-7	rds.ap-southeast-7.amazonaws.com	HTTPS
아시아 태평양(도쿄)	ap-northeast-1	rds.ap-northeast-1.amazonaws.com rds.ap-northeast-1.api.aws	HTTPS HTTPS

리전 이름	지역	엔드포인트	프로토콜
캐나다(중부)	ca-central-1	rds.ca-central-1.amazonaws.com	HTTPS
		rds.ca-central-1.api.aws	HTTPS
		rds-fips.ca-central-1.api.aws	HTTPS
		rds-fips.ca-central-1.amazonaws.com	HTTPS
캐나다 서부(캘거리)	ca-west-1	rds.ca-west-1.amazonaws.com	HTTPS
		rds-fips.ca-west-1.amazonaws.com	HTTPS
유럽(프랑크푸르트)	eu-central-1	rds.eu-central-1.amazonaws.com	HTTPS
		rds.eu-central-1.api.aws	HTTPS
유럽(아일랜드)	eu-west-1	rds.eu-west-1.amazonaws.com	HTTPS
		rds.eu-west-1.api.aws	HTTPS
유럽(런던)	eu-west-2	rds.eu-west-2.amazonaws.com	HTTPS
		rds.eu-west-2.api.aws	HTTPS
유럽(밀라노)	eu-south-1	rds.eu-south-1.amazonaws.com	HTTPS
		rds.eu-south-1.api.aws	HTTPS
유럽(파리)	eu-west-3	rds.eu-west-3.amazonaws.com	HTTPS
		rds.eu-west-3.api.aws	HTTPS
유럽(스페인)	eu-south-2	rds.eu-south-2.amazonaws.com	HTTPS
		rds.eu-south-2.api.aws	HTTPS
유럽(스톡홀름)	eu-north-1	rds.eu-north-1.amazonaws.com	HTTPS
		rds.eu-north-1.api.aws	HTTPS

리전 이름	지역	엔드포인트	프로토콜
유럽(취리히)	eu-central-2	rds.eu-central-2.amazonaws.com rds.eu-central-2.api.aws	HTTPS HTTPS
이스라엘(텔아비브)	il-central-1	rds.il-central-1.amazonaws.com rds.il-central-1.api.aws	HTTPS HTTPS
멕시코(중부)	mx-central-1	rds.mx-central-1.amazonaws.com	HTTPS
중동(바레인)	me-south-1	rds.me-south-1.amazonaws.com rds.me-south-1.api.aws	HTTPS HTTPS
중동(UAE)	me-central-1	rds.me-central-1.amazonaws.com rds.me-central-1.api.aws	HTTPS HTTPS
남아메리카(상파울루)	sa-east-1	rds.sa-east-1.amazonaws.com rds.sa-east-1.api.aws	HTTPS HTTPS
AWS GovCloud(미국 동부)	us-gov-east-1	rds.us-gov-east-1.amazonaws.com rds.us-gov-east-1.api.aws	HTTPS HTTPS
AWS GovCloud(미국 서부)	us-gov-west-1	rds.us-gov-west-1.amazonaws.com rds.us-gov-west-1.api.aws	HTTPS HTTPS

다음 표에는 특정 Amazon RDS 리소스 유형에 대한 ARN 생성 시 사용해야 하는 형식이 나와 있습니다.

리소스 유형	ARN 형식
DB 인스턴스	arn:aws:rds:*<region>*:*<account>* :db:*<name>* 예: `arn:aws:rds: us-east-2 :123456789012 :db:my-mysql-instance-1`
DB 클러스터	arn:aws:rds:*<region>*:*<account>* :cluster:*<name>* 예: `arn:aws:rds: us-east-2 :123456789012 :cluster: my-aurora-cluster-1`
이벤트 구독	arn:aws:rds:*<region>*:*<account>* :es:*<name>* 예: `arn:aws:rds: us-east-2 :123456789012 :es:my-subscription`
DB 옵션 그룹	arn:aws:rds:*<region>*:*<account>* :og:*<name>* 예: `arn:aws:rds: us-east-2 :123456789012 :og:my-og`
DB 파라미터 그룹	arn:aws:rds:*<region>*:*<account>* :pg:*<name>* 예: `arn:aws:rds: us-east-2 :123456789012 :pg:my-param-enable-logs`
DB 클러스터 파라미터 그룹	arn:aws:rds:*<region>*:*<account>* :cluster-pg:*<name>* 예:

리소스 유형	ARN 형식
	`arn:aws:rds:` *us-east-2* `:`*123456789012* `:cluster-pg:` *my-cluster-param-timezone*
예약 DB 인스턴스	arn:aws:rds:*<region>*:*<account>* :ri:*<name>* 예: `arn:aws:rds:` *us-east-2* `:`*123456789012* `:ri:`*my-reserved-postgresql*
DB 보안 그룹	arn:aws:rds:*<region>*:*<account>* :secgrp:*<name>* 예: `arn:aws:rds:` *us-east-2* `:`*123456789012* `:secgrp:`*my-public*
자동화된 DB 스냅샷	arn:aws:rds:*<region>*:*<account>* :snapshot:rds:*<name>* 예: `arn:aws:rds:` *us-east-2* `:`*123456789012* `:snapshot:rds:` *my-mysql-db-2019-07-22-07-23*
자동화된 DB 클러스터 스냅샷	arn:aws:rds:*<region>*:*<account>* :cluster-snapshot:rds:*<name>* 예: `arn:aws:rds:` *us-east-2* `:`*123456789012* `:cluster-snapshot:rds:` *my-aurora-cluster-2019-07-22-16-16*

리소스 유형	ARN 형식
수동 DB 스냅샷	arn:aws:rds:*<region>*:*<account>* :snapshot:*<name>* 예: `arn:aws:rds: us-east-2 :123456789012 :snapshot: my-mysql-db-snap`
수동 DB 클러스터 스냅샷	arn:aws:rds:*<region>*:*<account>* :cluster-snapshot:*<name>* 예: `arn:aws:rds: us-east-2 :123456789012 :cluster-snapshot: my-aurora-cluster-snap`
DB 서브넷 그룹	arn:aws:rds:*<region>*:*<account>* :subgrp:*<name>* 예시: `arn:aws:rds: us-east-2 :123456789012 :subgrp:my-subnet-10`

Amazon RDS의 기존 ARN 가져오기

AWS Management Console, AWS Command Line Interface(AWS CLI) 또는 RDS API를 사용해 RDS 리소스에 대한 ARN을 가져올 수 있습니다.

콘솔

AWS Management Console에서 ARN을 확인하려면 ARN을 보려는 리소스를 탐색하거나, 해당 리소스의 세부 정보를 확인합니다.

예를 들어, DB 인스턴스 세부 정보의 구성 탭에서 DB 인스턴스의 ARN을 가져올 수 있습니다.

AWS CLI

AWS CLI에서 특정 RDS 리소스에 대한 ARN을 가져오려면 해당 리소스에 `describe` 명령을 사용합니다. 다음 표에서는 각 AWS CLI ARN 명령, 그리고 ARN을 가져오기 위해 명령과 함께 사용하는 ARN 속성을 보여 줍니다.

AWS CLI 명령	ARN 속성
describe-event-subscriptions	EventSubscriptionArn
describe-certificates	CertificateArn
describe-db-parameter-groups	DBParameterGroupArn
describe-db-cluster-parameter-groups	DBClusterParameterGroupArn
describe-db-instances	DBInstanceArn
describe-db-security-groups	DBSecurityGroupArn
describe-db-snapshots	DBSnapshotArn
describe-events	SourceArn
describe-reserved-db-instances	ReservedDBInstanceArn
describe-db-subnet-groups	DBSubnetGroupArn
describe-option-groups	OptionGroupArn
describe-db-clusters	DBClusterArn
describe-db-cluster-snapshots	DBClusterSnapshotArn

예를 들어, 다음 AWS CLI 명령은 DB 인스턴스에 대한 ARN을 가져옵니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-instances \
--db-instance-identifier DBInstanceIdentifier \
--region us-west-2 \
--query "*[].{DBInstanceIdentifier:DBInstanceIdentifier,DBInstanceArn:DBInstanceArn}"
```

Windows의 경우:

```
aws rds describe-db-instances ^
--db-instance-identifier DBInstanceIdentifier ^
--region us-west-2 ^
--query "*[].{DBInstanceIdentifier:DBInstanceIdentifier,DBInstanceArn:DBInstanceArn}"
```

이 명령의 출력은 다음과 같습니다.

```
[
    {
        "DBInstanceArn": "arn:aws:rds:us-west-2:account_id:db:instance_id",
        "DBInstanceIdentifier": "instance_id"
    }
]
```

RDS API

특정 RDS 리소스에 대한 ARN을 확인하기 위해 다음과 같이 RDS API 작업을 호출하고 ARN 속성을 사용할 수 있습니다.

RDS API 작업	ARN 속성
DescribeEventSubscriptions	EventSubscriptionArn
DescribeCertificates	CertificateArn
DescribeDBParameterGroups	DBParameterGroupArn
DescribeDBClusterParameterGroups	DBClusterParameterGroupArn
DescribeDBInstances	DBInstanceArn

RDS API 작업	ARN 속성
DescribeDBSecurityGroups	DBSecurityGroupArn
DescribeDBSnapshots	DBSnapshotArn
DescribeEvents	SourceArn
DescribeReservedDBInstances	ReservedDBInstanceArn
DescribeDBSubnetGroups	DBSubnetGroupArn
DescribeOptionGroups	OptionGroupArn
DescribeDBClusters	DBClusterArn
DescribeDBClusterSnapshots	DBClusterSnapshotArn

Amazon RDS DB 인스턴스 스토리지 작업

Amazon RDS에 데이터를 저장하는 방식을 지정하려면 DB 인스턴스를 만들거나 수정할 때 스토리지 유형을 선택하고 스토리지 크기를 입력합니다. 추후에는 DB 인스턴스를 수정하여 스토리지의 양을 늘리거나 스토리지의 유형을 변경할 수 있습니다. 워크로드를 사용할 때 어떤 스토르지 유형을 사용할지에 대한 자세한 정보는 Amazon RDS 스토리지 유형 단원을 참조하십시오.

주제

- DB 인스턴스 스토리지 용량 증가
- Amazon RDS 스토리지 Autoscaling을 사용한 용량 자동 관리
- DB 인스턴스의 스토리지 파일 시스템 업그레이드
- 프로비저닝된 IOPS SSD 스토리지 설정 수정
- I/O 집약적 스토리지 수정
- 범용 SSD(gp3) 스토리지 설정 수정
- 전용 로그 볼륨(DLV) 사용

DB 인스턴스 스토리지 용량 증가

추가 데이터에 대한 공간이 필요할 경우, 기존 DB 인스턴스의 스토리지를 확장할 수 있습니다. Amazon RDS 관리 콘솔, Amazon RDS API 또는 AWS Command Line Interface(AWS CLI)를 사용하면 가능합니다. 스토리지 제한에 대한 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 단원을 참조하십시오.

> **Note**
>
> 스토리지가 할당된 후에 DB 인스턴스의 스토리지 양을 줄일 수 없습니다. 할당된 스토리지를 늘릴 경우, 10% 이상 늘려야 합니다. 이 값을 10% 미만으로 늘리면 오류가 발생합니다. RDS for SQL Server DB 인스턴스의 스토리지 확장 기능은 범용 SSD 또는 프로비저닝된 IOPS SSD 스토리지 유형에서만 지원됩니다.

필요한 경우 대응할 수 있도록 DB 인스턴스의 여유 스토리지 크기를 모니터링하려면 Amazon CloudWatch 경보를 생성하는 것이 좋습니다. CloudWatch 경보 설정에 대한 자세한 내용은 CloudWatch 경보 사용을 참조하세요.

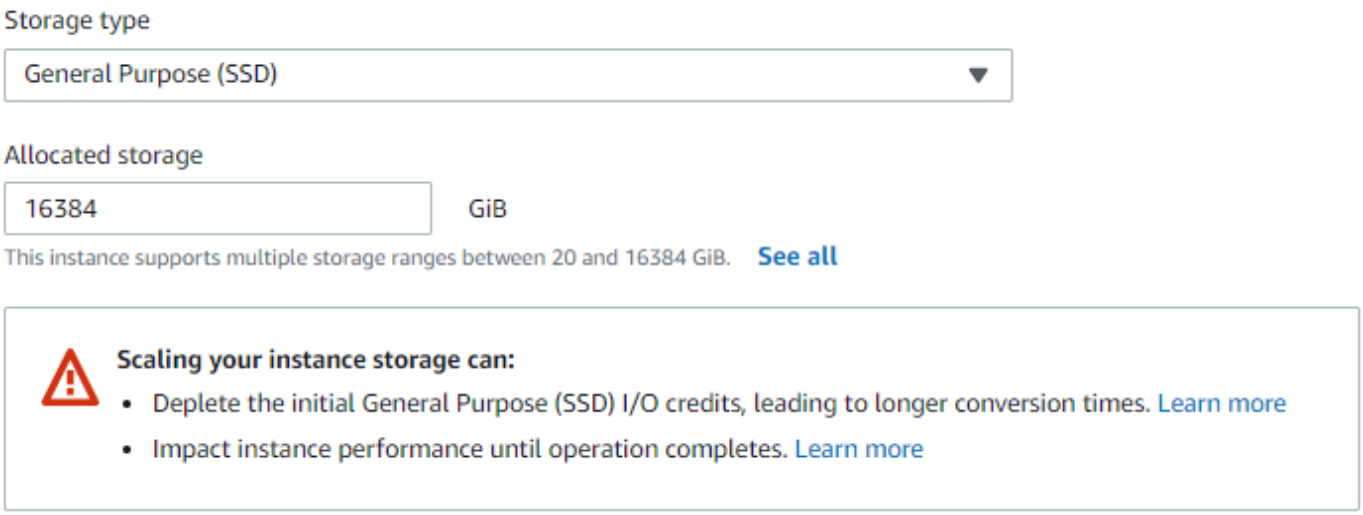
스토리지 크기 조정은 일반적으로 DB 인스턴스의 중단이나 성능 저하를 일으키지 않습니다. DB 인스턴스에 대한 스토리지 크기를 수정하면 DB 인스턴스의 상태가 스토리지 최적화로 됩니다.

> **Note**
>
> 스토리지 최적화는 몇 시간이 걸릴 수 있습니다. 6시간 또는 인스턴스에서 스토리지 최적화가 완료된 시간 둘 중 더 긴 시간 동안은 스토리지를 추가로 수정할 수 없습니다. AWS Management Console에서 또는 describe-db-instances AWS CLI 명령을 사용하여 스토리지 최적화 진행 상황을 확인할 수 있습니다.

콘솔

DB 인스턴스 스토리지 증가

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 수정하려는 DB 인스턴스를 선택합니다.
4. 수정을 선택합니다.
5. 할당된 스토리지에 새로운 값을 입력합니다. 현재 값보다 커야 합니다.



6. [Continue]를 선택하여 다음 화면으로 이동합니다.
7. Scheduling of modifications(수정 사항 예약) 섹션에서 즉시 적용 확인란을 선택하여 스토리지 변경 사항을 DB 인스턴스에 즉시 적용합니다.

또는 Apply during the next scheduled maintenance window(예약된 다음 유지 관리 기간에 적용)를 선택하여 다음 유지 관리 기간에 변경 사항을 적용합니다.

8. 원하는 대로 설정이 되었으면 [Modify DB instance]를 선택합니다.

AWS CLI

DB 인스턴스 스토리지를 늘리려면 AWS CLI 명령 `modify-db-instance`를 사용합니다. 다음 파라미터를 설정합니다.

- `--allocated-storage` - DB 인스턴스에 할당할 스토리지 크기(GiB)입니다.
- `--apply-immediately` – `--apply-immediately`을(를) 사용하여 스토리지 변경 사항을 바로 적용합니다.

  그 밖에 다음 유지 관리 기간에 스토리지 변경 사항을 적용하려면 `--no-apply-immediately`(기본값)을(를) 사용합니다. 변경 사항이 적용되면 즉시 중단됩니다.

스토리지에 대한 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 단원을 참조하십시오.

RDS API

DB 인스턴스 스토리지를 확장하려면 Amazon RDS API 작업 `ModifyDBInstance`를 사용합니다. 다음 파라미터를 설정합니다.

- `AllocatedStorage` - DB 인스턴스에 할당할 스토리지 크기(GiB)입니다.
- `ApplyImmediately` - 이 옵션을 `True`(으)로 설정하면 스토리지 변경 사항이 즉시 적용됩니다. 다음 유지 관리 기간에 변경 사항을 적용하려면 이 옵션을 `False`(기본값)(으)로 설정합니다. 변경 사항이 적용되면 즉시 중단됩니다.

스토리지에 대한 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 단원을 참조하십시오.

Amazon RDS 스토리지 Autoscaling을 사용한 용량 자동 관리

워크로드가 예측할 수 없는 경우에는 Amazon RDS DB 인스턴스에서 스토리지 Autoscaling을 활성화할 수 있습니다. Amazon RDS 콘솔, Amazon RDS API 또는 AWS CLI를 사용하면 가능합니다.

예를 들어 사용자들의 도입 속도가 빠른 모바일 게임 애플리케이션에서 이러한 기능을 사용할 수 있습니다. 이러한 경우에 급증하는 워크로드가 사용 가능한 데이터베이스 스토리지를 초과할 수 있기 때문

입니다. 데이터베이스 스토리지를 수동으로 확장하지 않으려면 Amazon RDS 스토리지 Autoscaling을 사용할 수 있습니다.

스토리지 자동 크기 조정이 활성화된 상태에서 Amazon RDS가 데이터베이스의 여유 공간이 부족한 것을 감지하면 자동으로 스토리지를 확장합니다. Amazon RDS는 다음과 같은 요인이 적용될 때 자동 크기 조정이 활성화된 DB 인스턴스에서 스토리지 수정을 시작합니다.

- 사용 가능한 여유 공간이 할당된 스토리지의 10% 이하일 때
- 낮은 스토리지 조건이 5분 이상 지속될 때
- 마지막 스토리지 수정 후 최소 6시간이 지났거나 인스턴스에서 스토리지 최적화가 완료된 시간 둘 중 더 긴 시간이 지났을 때

추가 스토리지는 다음 중 더 큰 값만큼 증가합니다.

- 10GiB
- 현재 할당된 스토리지의 10퍼센트
- 지난 1시간 동안의 `FreeStorageSpace` 지표를 기반으로 향후 7시간 동안 현재 할당된 스토리지 크기를 초과할 것으로 예상되는 스토리지 증가량. 지표에 대한 자세한 내용은 Amazon CloudWatch를 사용하여 모니터링을 참조하세요.

최대 스토리지 임계값은 DB 인스턴스의 Auto Scaling을 위해 설정한 한도입니다. 다음과 같은 제약이 있습니다.

- 최대 스토리지 임계값을 현재 할당된 스토리지보다 10% 이상 크게 설정해야 합니다. 스토리지 크기가 최대 스토리지 임계값에 도달했다는 이벤트 알림을 받지 않으려면 최소 26% 이상으로 설정하는 것이 좋습니다.

  예를 들어, 할당된 스토리지가 1000GiB인 DB 인스턴스가 있는 경우 최대 스토리지 임계값을 1100GiB 이상으로 설정합니다. 그러지 않으면 Invalid max storage size for *engine_name*(engine_name의 최대 스토리지 크기가 잘못됨)과 같은 오류 메시지가 표시됩니다. 하지만 이벤트 알림을 차단하려면 최대 스토리지 임계값을 최소 1260GiB로 설정하는 것이 좋습니다.
- 프로비저닝된 IOPS(io1 또는 io2 Block Express) 스토리지를 사용하는 DB 인스턴스의 경우 IOPS 대 최대 스토리지 임계값(GiB)의 비율이 특정 범위 내에 있어야 합니다. 자세한 내용은 프로비저닝된 IOPS SSD 스토리지 단원을 참조하십시오.

- 오토스케일링 지원 인스턴스의 최대 스토리지 임계값을 데이터베이스 엔진 및 DB 인스턴스 클래스에 할당된 최대 스토리지보다 큰 값으로 설정할 수 없습니다.

  예를 들어, db.m5.xlarge 기반의 SQL Server Standard Edition에 기본 할당된 인스턴스 스토리지는 20GiB(최소), 최대 할당 스토리지는 16,384GiB입니다. autoscaling의 기본 최대 스토리지 임계값은 1,000GiB입니다. 이 기본값을 사용하면 인스턴스가 1,000GiB 넘게 자동 확장되지 않습니다. 인스턴스에 할당된 최대 스토리지가 16,384GiB인 경우에도 마찬가지입니다.

> **Note**
>
> 사용 패턴 및 고객 요구에 따라 최대 스토리지 임계값을 신중하게 선택하는 것이 좋습니다. 사용 패턴에 수차가 있는 경우 자동 크기 조정이 매우 높은 임계값을 예측할 때 최대 스토리지 임계값은 예기치 않게 높은 값으로 스토리지 확장을 방지할 수 있습니다. DB 인스턴스가 자동 확장된 후에는 할당된 스토리지를 줄일 수 없습니다.

주제

- 스토리지 자동 크기 조정의 제한 사항
- 새로운 DB 인스턴스의 스토리지 Autoscaling 활성화
- DB 인스턴스의 스토리지 Autoscaling 설정 변경
- 수동으로 DB 인스턴스 스케일 다운 또는 스케일 인
- DB 인스턴스의 스토리지 Autoscaling 비활성화

스토리지 자동 크기 조정의 제한 사항

스토리지 Autoscaling에는 다음과 같은 제한 사항이 적용됩니다.

- 스토리지 증분으로 최대 스토리지 임계값이 초과할 경우 Autoscaling이 발생하지 않습니다.
- 자동 크기 조정 시 RDS는 후속 자동 크기 조정 작업을 위해 스토리지 크기를 예측합니다. 후속 작업이 최대 스토리지 임계값을 초과할 것으로 예상되는 경우 RDS는 최대 스토리지 임계값보다 작아지도록 자동으로 크기를 조정합니다.
- 자동 크기 조정으로 대용량 데이터 로드에 대한 스토리지가 가득 찬 상황을 완전히 막을 수는 없습니다. 6시간 동안 또는 인스턴스에서 스토리지 최적화 완료 시까지 중 더 긴 시간 동안 스토리지를 추가로 수정할 수 없기 때문입니다.

대용량 데이터 로드를 수행하고 Autoscaling이 충분한 공간을 제공하지 않으면 데이터베이스가 몇 시간 동안 스토리지가 가득 찬 상태로 유지될 수 있습니다. 이는 데이터베이스에 악영향을 미칠 수 있습니다.

- Amazon RDS가 Autoscaling 작업을 시작할 때 동시에 사용자가 스토리지 확장 작업을 시작할 경우 사용자의 스토리지 수정이 우선합니다. Autoscaling 작업이 취소됩니다.
- 자동 크기 조정으로 할당된 스토리지를 줄일 수는 없습니다. 스토리지가 할당된 후에 DB 인스턴스의 스토리지 양을 줄일 수 없습니다.
- Autoscaling은 마그네틱 스토리지와 함께 사용할 수 없습니다.
- Autoscaling은 주문 가능한 스토리지가 6TiB 미만인 이전 세대 인스턴스 클래스(db.m3.large, db.m3.xlarge 및 db.m3.2xlarge)와 함께 사용할 수 없습니다.
- Auto Scaling 작업은 AWS CloudTrail에 의해 기록되지 않습니다. CloudTrail에 대한 자세한 내용은 AWS CloudTrail에서 Amazon RDS API 호출 모니터링 섹션을 참조하세요.

Autoscaling이 Amazon RDS DB 인스턴스의 스토리지를 동적으로 늘리는 데 효과적이기는 하지만 DB 인스턴스의 초기 스토리지는 일반적인 워크로드에 적합한 크기로 구성해야 합니다.

새로운 DB 인스턴스의 스토리지 Autoscaling 활성화

새로운 Amazon RDS DB 인스턴스를 생성할 때 스토리지 Autoscaling의 활성화 여부를 선택할 수 있습니다. 또한 Amazon RDS가 DB 인스턴스에 할당할 수 있는 스토리지의 상한선도 설정할 수 있습니다.

> **Note**
>
> 스토리지 Autoscaling이 활성화된 Amazon RDS DB 인스턴스를 복제할 경우 해당 설정은 복제 인스턴스에서 자동으로 상속되지 않습니다. 새롭게 복제된 DB 인스턴스는 할당되는 스토리지 크기가 원본 인스턴스와 동일합니다. 하지만 복제된 인스턴스에서 스토리지 요건이 계속해서 증가할 경우 새로운 인스턴스에서도 스토리지 Autoscaling을 다시 활성화할 수 있습니다.

콘솔

새로운 DB 인스턴스에서 스토리지 Autoscaling을 활성화하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 콘솔의 오른쪽 상단에서 DB 인스턴스를 생성하려는 AWS 리전을 선택합니다.

3. 탐색 창에서 데이터베이스를 선택합니다.
4. 데이터베이스 생성을 선택합니다. 엔진 선택 페이지에서 데이터베이스 엔진을 선택한 후 Amazon RDS 시작하기.에서 설명하는 대로 DB 인스턴스 정보를 지정합니다.
5. Storage autoscaling(스토리지 Autoscaling) 섹션에서 DB 인스턴스의 Maximum storage threshold(최대 스토리지 임계값) 값을 설정합니다.
6. 나머지 DB 인스턴스 정보를 Amazon RDS 시작하기에서 설명하는 대로 지정합니다.

AWS CLI

새로운 DB 인스턴스에서 스토리지 Autoscaling을 활성화하려면 AWS CLI 명령인 `create-db-instance`를 사용합니다. 다음 파라미터를 설정합니다.

- `--max-allocated-storage` - 스토리지 자동 크기 조정을 활성화하고 스토리지 크기 상한선(기비바이트)을 설정합니다.

Amazon RDS 스토리지 자동 크기 조정을 DB 인스턴스에서 사용할 수 있는지 확인하려면 AWS CLI `describe-valid-db-instance-modifications` 명령을 사용합니다. 인스턴스를 생성하기 전에 인스턴스 클래스를 기준으로 확인하려면 `describe-orderable-db-instance-options` 명령을 사용합니다. 반환 값에서 다음 필드를 확인하십시오.

- `SupportsStorageAutoscaling` - DB 인스턴스 또는 인스턴스 클래스의 스토리지 자동 조정 크기 지원 여부를 나타냅니다.

스토리지에 대한 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 단원을 참조하십시오.

RDS API

새로운 DB 인스턴스에서 스토리지 Autoscaling을 활성화하려면 Amazon RDS API 작업 `CreateDBInstance`를 사용합니다. 다음 파라미터를 설정합니다.

- `MaxAllocatedStorage` - Amazon RDS 스토리지 자동 크기 조정을 활성화하고 스토리지 크기 상한선(기비바이트)을 설정합니다.

DB 인스턴스에 Amazon RDS 스토리지 Autoscaling을 사용할 수 있는지 확인하려면 기존 인스턴스일 경우 Amazon RDS API `DescribeValidDbInstanceModifications` 작업을 사용하고, 인스턴스 생성 전에는 `DescribeOrderableDBInstanceOptions` 작업을 사용합니다. 반환 값에서 다음 필드를 확인하십시오.

- `SupportsStorageAutoscaling` - DB 인스턴스의 스토리지 자동 크기 조정 지원 여부를 나타냅니다.

스토리지에 대한 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 단원을 참조하십시오.

DB 인스턴스의 스토리지 Autoscaling 설정 변경

기존 Amazon RDS DB 인스턴스에서 스토리지 Autoscaling을 활성화할 수 있습니다. 또한 Amazon RDS가 DB 인스턴스에 할당할 수 있는 스토리지의 상한선도 변경할 수 있습니다.

콘솔

DB 인스턴스의 스토리지 Autoscaling 설정을 변경하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 수정하려는 DB 인스턴스를 선택한 후 수정을 선택합니다. [Modify DB instance] 페이지가 나타납니다.
4. Autoscaling 섹션에서 스토리지 제한을 변경합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.
5. 원하는 대로 모두 변경되었으면 계속을 선택하고 수정 사항을 확인합니다.
6. 확인 페이지에서 변경 내용을 검토합니다. 변경 사항이 정확할 경우 DB 인스턴스 수정을 선택하여 변경 내용을 저장합니다. 그렇지 않으면 뒤로를 선택하여 변경 사항을 편집하거나 취소를 선택하여 변경 사항을 취소합니다.

   스토리지 autoscaling 제한 변경은 즉시 적용됩니다. 이 설정은 [Apply immediately] 설정을 무시합니다.

AWS CLI

DB 인스턴스의 스토리지 Autoscaling 설정을 변경하려면 AWS CLI 명령인 `modify-db-instance`를 사용합니다. 다음 파라미터를 설정합니다.

- `--max-allocated-storage` - 스토리지 크기의 상한선(기비바이트)을 설정합니다. 값이 `--allocated-storage` 파라미터보다 크면 스토리지 Autoscaling이 활성화됩니다. 값이 `--allocated-storage` 파라미터와 동일하면 스토리지 Autoscaling이 비활성화됩니다.

Amazon RDS 스토리지 자동 크기 조정을 DB 인스턴스에서 사용할 수 있는지 확인하려면 AWS CLI `describe-valid-db-instance-modifications` 명령을 사용합니다. 인스턴스를 생성하기 전에 인스턴스 클래스를 기준으로 확인하려면 `describe-orderable-db-instance-options` 명령을 사용합니다. 반환 값에서 다음 필드를 확인하십시오.

- `SupportsStorageAutoscaling` - DB 인스턴스의 스토리지 자동 크기 조정 지원 여부를 나타냅니다.

스토리지에 대한 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 단원을 참조하십시오.

RDS API

DB 인스턴스의 스토리지 Autoscaling 설정을 변경하려면 Amazon RDS API 작업 `ModifyDBInstance`를 사용합니다. 다음 파라미터를 설정합니다.

- `MaxAllocatedStorage` - 스토리지 크기의 상한선(기비바이트)을 설정합니다.

DB 인스턴스에 Amazon RDS 스토리지 Autoscaling을 사용할 수 있는지 확인하려면 기존 인스턴스일 경우 Amazon RDS API `DescribeValidDbInstanceModifications` 작업을 사용하고, 인스턴스 생성 전에는 `DescribeOrderableDBInstanceOptions` 작업을 사용합니다. 반환 값에서 다음 필드를 확인하십시오.

- `SupportsStorageAutoscaling` - DB 인스턴스의 스토리지 자동 크기 조정 지원 여부를 나타냅니다.

스토리지에 대한 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 단원을 참조하십시오.

수동으로 DB 인스턴스 스케일 다운 또는 스케일 인

Amazon RDS는 증가하는 수요에 맞게 스토리지 오토 스케일링을 제공합니다. 그러나 스케일 다운 및 스케일 인과 관련된 제한 사항이 있습니다.

- RDS 스토리지 - RDS는 수요 증가에 따라 스토리지의 자동 스케일 업을 지원하지만 스토리지를 자동으로 스케일 다운하지 않습니다.
- 읽기 전용 복제본 - RDS는 읽기 전용 복제본의 자동 스케일 아웃(추가) 또는 스케일 인(삭제)을 지원하지 않습니다. 로드 요구 사항에 따라 읽기 전용 복제본을 수동으로 추가하거나 제거해야 합니다.

RDS 리소스를 스케일 다운하려면 다음 수동 작업을 수행합니다.

- 스토리지의 경우 `modify-db-instance` 명령을 사용하여 DB 인스턴스의 할당된 스토리지를 수동으로 줄일 수 없습니다. 대신 다음 기법 중 하나를 선택합니다.
  - DB 엔진이 지원하는 경우 블루/그린 배포를 사용합니다. 스토리지 크기가 더 작은 그린 데이터베이스를 만든 다음 그린 데이터베이스를 블루 데이터베이스로 승격합니다. 자세한 내용은 스토리지 및 성능 설정 수정 단원을 참조하십시오.
  - 할당된 스토리지가 더 낮은 새 DB 인스턴스를 만들고, 현재 데이터베이스에서 새로 만든 데이터베이스 인스턴스로 데이터를 수동으로 마이그레이션하고, 데이터베이스 엔드포인트를 전환합니다.
- 읽기 전용 복제본의 경우 RDS 콘솔 또는 AWS CLI를 통해 미사용 복제본을 수동으로 삭제합니다.

DB 인스턴스의 스토리지 Autoscaling 비활성화

Amazon RDS에서 더 이상 Amazon RDS DB 인스턴스의 스토리지를 자동으로 늘릴 필요가 없다면 스토리지 Autoscaling을 비활성화할 수 있습니다. 비활성화한 후에도 DB 인스턴스 스토리지의 크기를 수동으로 늘릴 수 있습니다.

콘솔

DB 인스턴스의 스토리지 Autoscaling을 비활성화하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 수정하려는 DB 인스턴스를 선택한 후 수정을 선택합니다. [Modify DB instance] 페이지가 나타납니다.
4. Storage autoscaling(스토리지 Autoscaling) 섹션에서 Enable storage autoscaling(스토리지 Autoscaling 활성화) 확인란을 해제합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.
5. 원하는 대로 모두 변경했으면 계속을 선택하고 수정 사항을 확인합니다.
6. 확인 페이지에서 변경 내용을 검토합니다. 변경 사항이 정확할 경우 DB 인스턴스 수정을 선택하여 변경 내용을 저장합니다. 그렇지 않으면 뒤로를 선택하여 변경 사항을 편집하거나 취소를 선택하여 변경 사항을 취소합니다.

스토리지 autoscaling 제한 변경은 즉시 적용됩니다. 이 설정은 [Apply immediately] 설정을 무시합니다.

AWS CLI

DB 인스턴스에서 스토리지 Autoscaling을 비활성화하려면 AWS CLI 명령인 `modify-db-instance`와 다음 파라미터를 사용합니다.

- `--max-allocated-storage` - `--allocated-storage` 설정과 동일하게 값을 지정하여 지정된 DB 인스턴스에서 Amazon RDS 스토리지 자동 크기 조정을 더 이상 사용하지 않습니다.

스토리지에 대한 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 단원을 참조하십시오.

RDS API

DB 인스턴스의 스토리지 Autoscaling을 비활성화하려면 Amazon RDS API 작업 `ModifyDBInstance`를 사용합니다. 다음 파라미터를 설정합니다.

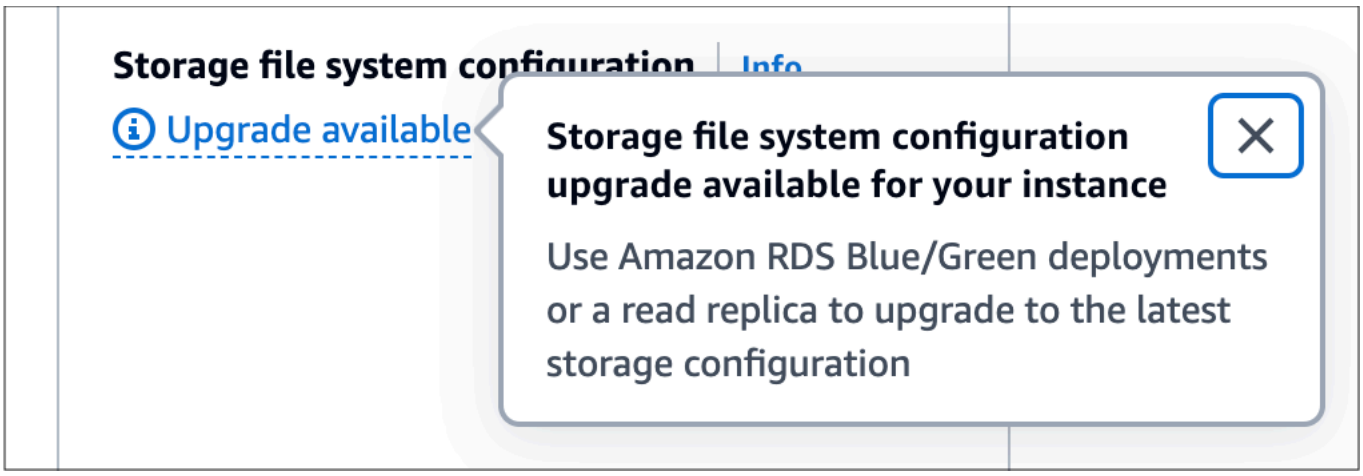
- `MaxAllocatedStorage` - `AllocatedStorage` 설정과 동일하게 값을 지정하여 지정된 DB 인스턴스에서 Amazon RDS 스토리지 자동 크기 조정을 더 이상 사용하지 않습니다.

스토리지에 대한 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 단원을 참조하십시오.

DB 인스턴스의 스토리지 파일 시스템 업그레이드

대부분의 RDS DB 인스턴스는 RDS for MariaDB, MySQL, PostgreSQL 데이터베이스에 대해 최대 64TiB의 스토리지 크기를 제공합니다. 그러나 일부 구형 32비트 파일 시스템은 스토리지 용량이 더 낮습니다. DB 인스턴스의 스토리지 용량을 확인하려면 describe-valid-db-instance-modifications AWS CLI 명령을 사용합니다.

RDS는 스토리지 시스템의 스토리지 크기가 16TiB인지, 파일 크기 제한이 2TiB인지 또는 최적화되지 않은 쓰기인지 확인합니다. DB 인스턴스가 이러한 조건을 충족하는 경우 RDS는 파일 시스템 구성이 업그레이드 대상임을 알립니다. DB 인스턴스 세부 정보 페이지의 스토리지 패널에서 DB 인스턴스의 업그레이드 자격을 확인할 수 있습니다.



DB 인스턴스가 파일 시스템 업그레이드에 적합한 경우 다음 방법 중 하나를 사용합니다.

- 블루/그린 배포를 생성하고 스토리지 파일 시스템 구성 업그레이드를 지정합니다.

  이 옵션은 그린 환경의 파일 시스템을 원하는 구성으로 업그레이드합니다. 그런 다음 블루/그린 배포를 전환하여 그린 환경을 새로운 프로덕션 환경으로 승격합니다. 자세한 지침은 the section called "블루/그린 배포 생성" 섹션을 참조하세요.
- DB 인스턴스 읽기 전용 복제본을 만들고 스토리지 파일 시스템 구성 업그레이드를 지정합니다.

  이 옵션은 읽기 전용 복제본의 파일 시스템을 원하는 구성으로 업그레이드합니다. 그런 다음 읽기 전용 복제본을 독립 실행형 인스턴스로 승격할 수 있습니다. 자세한 지침은 the section called "읽기 전용 복제본 생성" 섹션을 참조하세요.

스토리지 업그레이드 중에는 데이터베이스 엔진을 사용할 수 없습니다. 스토리지 구성 업그레이드는 I/O 집약적인 작업이므로 읽기 전용 복제본과 블루/그린 배포의 생성 시간이 길어집니다. 다음 두 조건이 모두 충족되면 스토리지 업그레이드 프로세스가 더 빨라집니다.

- 소스 DB 인스턴스는 프로비저닝된 IOPS SSD(io1 또는 io2 Block Express) 스토리지를 사용합니다.
- 인스턴스 크기가 4xlarge 이상인 그린 환경 또는 읽기 전용 복제본을 프로비저닝했습니다.

범용 SSD(gp2) 스토리지를 사용하는 스토리지 업그레이드는 I/O 크레딧 밸런스를 고갈할 수 있어 업그레이드 시간이 더 오래 걸릴 수 있습니다. 자세한 내용은 the section called "DB 인스턴스 스토리지" 섹션을 참조하세요.

스토리지 업그레이드 중에 다음 두 조건이 모두 충족되면 RDS는 그린 인스턴스 또는 읽기 전용 복제본에 할당된 스토리지 크기를 10% 늘립니다.

- 소스 DB 인스턴스의 스토리지 사용량이 할당된 스토리지 크기의 90% 이상입니다.
- 스토리지 자동 크기 조정이 활성화되었습니다.

새 스토리지 크기가 인스턴스에 대해 설정된 최대 할당 스토리지 이상이면 RDS는 자동 크기 조정을 끕니다. 스토리지 업그레이드가 시작되기 전에 스토리지 자동 크기 조정이 비활성화되면 업그레이드하는 동안 스토리지 크기가 증가하지 않습니다.

프로비저닝된 IOPS SSD 스토리지 설정 수정

프로비저닝된 IOPS SSD 스토리지를 사용하는 DB 인스턴스의 설정을 Amazon RDS 콘솔, AWS CLI 또는 Amazon RDS API를 사용해 수정할 수 있습니다. 스토리지 유형과 할당된 스토리지, 그리고 프로비저닝된 IOPS 크기를 지정합니다. 범위는 사용하는 데이터베이스 엔진과 인스턴스 유형에 따라 달라집니다.

인스턴스에 프로비저닝된 IOPS의 양을 줄일 수는 있어도 스토리지 크기를 줄일 수는 없습니다.

대부분의 경우, 스토리지를 확장할 때 어떠한 중단도 필요하지 않고 서버 성능을 저하하지 않습니다. DB 인스턴스의 스토리지 IOPS를 수정하면 DB 인스턴스의 상태가 스토리지 최적화로 변경됩니다.

> **Note**
>
> 스토리지 최적화는 몇 시간이 걸릴 수 있습니다. 6시간 또는 인스턴스에서 스토리지 최적화가 완료된 시간 둘 중 더 긴 시간 동안은 스토리지를 추가로 수정할 수 없습니다.

각 데이터베이스 엔진에 사용할 수 있는 할당된 스토리지 및 프로비저닝된 IOPS 범위에 대한 자세한 내용은 프로비저닝된 IOPS SSD 스토리지 섹션을 참조하세요.

콘솔

DB 인스턴스에 대한 프로비저닝된 IOPS 설정을 변경하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.

   DB 인스턴스의 목록을 필터링하려면 Filter databases(데이터베이스 필터링)에 Amazon RDS가 결과를 필터하는 데 사용할 텍스트 문자열을 입력합니다. 이름이 해당 문자열을 포함하는 DB 인스턴스만 표시됩니다.

3. 수정하려는 프로비저닝된 IOPS가 있는 DB 인스턴스를 선택합니다.
4. 수정을 선택합니다.
5. DB 인스턴스 수정 페이지에서 스토리지 유형으로 프로비저닝된 IOPS SSD(io1) 또는 프로비저닝된 IOPS SSD(io2)를 선택합니다.
6. 프로비저닝된 IOPS에 값을 입력합니다.

   할당된 스토리지 또는 프로비저닝된 IOPS에서 지정하는 값이 다른 파라미터에서 지원되는 제한을 벗어나면 경고 메시지가 표시됩니다. 이 메시지는 다른 파라미터에 대하여 요구되는 값의 범위를 보여줍니다.
7. [Continue]를 선택합니다.
8. Scheduling of modifications(수정 사항 예약) 섹션에서 Apply immediately(즉시 적용)를 선택하여 변경 사항을 DB 인스턴스에 즉시 적용합니다. 또는 Apply during the next scheduled maintenance window(예약된 다음 유지 관리 기간에 적용)를 선택하여 다음 유지 관리 기간에 변경 사항을 적용합니다.
9. 변경될 파라미터를 검토하고 Modify DB instance(DB 인스턴스 수정)를 선택하여 수정을 완료합니다.

   할당된 스토리지나 프로비저닝된 IOPS에 대한 새 값은 [Status] 열에 나타납니다.

AWS CLI

DB 인스턴스에서 프로비저닝된 IOPS 설정을 변경하려면 AWS CLI 명령인 `modify-db-instance`를 사용합니다. 다음 파라미터를 설정합니다.

- `--storage-type` - 프로비저닝된 IOPS의 경우 `io1` 또는 `io2`로 설정합니다.
- `--allocated-storage` - DB 인스턴스에 할당할 스토리지 크기(GiB)입니다.
- `--iops` - DB 인스턴스에 대해 새로 설정하는 프로비저닝된 IOPS의 크기이며, 초당 I/O 작업 수로 표현됩니다.
- `--apply-immediately` - `--apply-immediately`를 사용하면 변경 사항이 즉시 적용됩니다. 다음 유지 관리 기간에 변경 사항을 적용하려면 `--no-apply-immediately`(기본값)을 사용합니다.

RDS API

DB 인스턴스에서 프로비저닝된 IOPS 설정을 변경하려면 Amazon RDS API 작업인 `ModifyDBInstance`를 사용합니다. 다음 파라미터를 설정합니다.

- `StorageType` - 프로비저닝된 IOPS의 경우 `io1` 또는 `io2`로 설정합니다.
- `AllocatedStorage` - DB 인스턴스에 할당할 스토리지 크기(GiB)입니다.
- `Iops` - DB 인스턴스에 대해 새로 설정하는 IOPS 속도이며, 초당 I/O 작업 수로 표현됩니다.
- `ApplyImmediately` - 이 옵션을 `True`로 설정하면 변경 사항이 즉시 적용됩니다. 다음 유지 관리 기간에 변경 사항을 적용하려면 이 옵션을 `False`(기본값)로 설정합니다.

I/O 집약적 스토리지 수정

Amazon RDS DB 인스턴스는 데이터베이스 및 로그 스토리지에 Amazon Elastic Block Store(EBS) 볼륨을 사용합니다. RDS(RDS for SQL Server 제외)는 필요한 스토리지 용량에 따라 자동으로 데이터를 여러 Amazon EBS 볼륨에 스트라이핑하여 성능을 강화합니다. SSD 스토리지 유형이 있는 RDS DB 인스턴스는 RAID 0 구성에서 1개 또는 4개의 스트라이프 Amazon EBS 볼륨으로 지원됩니다. 설계상 RDS DB 인스턴스의 스토리지 수정 작업은 진행 중인 데이터베이스 작업에 미치는 영향을 최소화합니다.

대부분의 경우 스토리지 확장 수정은 Amazon EBS 계층으로 완전히 오프로드되며 데이터베이스에 영향을 미치지 않습니다. 이 단계는 보통 몇 분 내에 완료됩니다. 하지만 일부 구형 RDS 스토리지 볼륨은 크기, 프로비저닝된 IOPS 또는 스토리지 유형을 수정하기 위해 다른 프로세스를 필요로 합니다. 여기에는 I/O가 많이 필요할 수 있는 작업을 사용하여 데이터의 전체 복제본을 만드는 작업이 포함됩니다.

스토리지 수정은 다음 요인 중 하나라도 적용되는 경우 입출력이 많은 작업을 사용합니다.

- 소스 스토리지 유형은 마그네틱 스토리지입니다. 마그네틱 스토리지는 엘라스틱 볼륨 수정을 지원하지 않습니다.
- RDS DB 인스턴스는 볼륨 1개 또는 4개 Amazon EBS 레이아웃에 있지 않습니다. 향상된 모니터링 지표를 사용하여 RDS DB 인스턴스에서 사용 중인 Amazon EBS 볼륨의 수를 확인할 수 있습니다. 자세한 내용은 RDS 콘솔에서 OS 지표 보기을 참조하세요.
- 수정 요청의 대상 크기에 따라 할당된 스토리지가 MariaDB, MySQL 및 PostgreSQL 인스턴스의 경우 RDS의 경우 400GiB 이상으로, 오라클의 경우 RDS의 경우 200GiB 이상으로 늘어납니다. 스토리지 자동 확장 작업은 DB 인스턴스에 할당된 스토리지 크기를 이러한 임계값 이상으로 늘리는 경우에도 동일한 효과가 있습니다.

스토리지 수정에 I/O가 많은 작업이 포함되는 경우 I/O 리소스가 소모되고 DB 인스턴스의 부하가 증가합니다. 범용 SSD(gp2) 스토리지를 사용하는 I/O 집약적 작업을 서로 전환할 때는 I/O 크레딧 밸런스가 고갈될 수도 있어 전환 시간이 더 오래 걸릴 수 있습니다.

스토리지 수정 작업을 완료하는 데 필요한 시간을 줄이는 데 도움이 되도록 피크 시간 외에 이러한 스토리지 수정 요청을 예약하는 것이 가장 좋습니다. 또는 DB 인스턴스의 읽기 전용 복제본을 생성한 후 읽기 전용 복제본에서 스토리지를 변경할 수 있습니다. 그러면 복제본이 기본 DB 인스턴스로 승격됩니다. 자세한 내용은 DB 인스턴스 읽기 전용 복제본 작업을 참조하세요.

자세한 내용을 알아보려면 다음 섹션을 참조하세요.할당된 스토리지를 늘리려고 할 때 Amazon RDS DB 인스턴스가 수정 중 상태로 멈추는 이유는 무엇입니까?

범용 SSD(gp3) 스토리지 설정 수정

범용 SSD(gp3) 스토리지를 사용하는 DB 인스턴스의 설정을 Amazon RDS 콘솔, AWS CLI 또는 Amazon RDS API를 사용해 수정할 수 있습니다. 필요한 스토리지 유형, 할당된 스토리지, 프로비저닝된 IOPS 양, 스토리지 처리량(throughput)을 지정합니다.

DB 인스턴스의 프로비저닝된 IOPS 양과 스토리지 처리량을 줄일 수는 있어도 스토리지 크기를 줄일 수는 없습니다.

대부분의 경우 스토리지 크기를 조정할 때 어떠한 중단도 필요하지 않습니다. DB 인스턴스의 스토리지 IOPS를 수정하면 DB 인스턴스의 상태가 스토리지 최적화로 변경됩니다. 스토리지 최적화 중에는 지연 시간이 길어질 수 있지만 여전히 한 자릿수 밀리초 범위 내에 있을 수 있습니다. 스토리지 수정 후 DB 인스턴스가 완전히 작동합니다.

> **Note**
>
> 인스턴스에서 스토리지 최적화가 완료된 후에는 6시간이 지날 때까지 스토리지를 추가로 수정할 수 없습니다.

각 데이터베이스 엔진에 사용할 수 있는 할당된 스토리지, 프로비저닝된 IOPS, 스토리지 처리량(throughput) 범위에 대한 자세한 내용은 gp3 스토리지(권장) 섹션을 참조하세요.

콘솔

DB 인스턴스의 스토리지 성능 설정을 변경하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.

DB 인스턴스의 목록을 필터링하려면 Filter databases(데이터베이스 필터링)에 Amazon RDS가 결과를 필터하는 데 사용할 텍스트 문자열을 입력합니다. 이름이 해당 문자열을 포함하는 DB 인스턴스만 표시됩니다.

3. 수정하려는 gp3 스토리지가 있는 DB 인스턴스를 선택합니다.
4. Modify(수정)를 선택합니다.
5. DB 인스턴스 수정 페이지에서 스토리지 유형으로 범용 SSD(gp3)를 선택한 후 다음을 수행합니다.

   a. 프로비저닝된 IOPS에서 값을 선택합니다.

      할당된 스토리지 또는 프로비저닝된 IOPS에서 지정하는 값이 다른 파라미터에서 지원되는 제한을 벗어나면 경고 메시지가 표시됩니다. 이 메시지는 다른 파라미터에 대하여 요구되는 값의 범위를 보여줍니다.

   b. 스토리지 처리량(throughput)에서 값을 선택합니다.

      프로비저닝된 IOPS 또는 스토리지 처리량(throughput)에서 지정하는 값이 다른 파라미터에서 지원되는 제한을 벗어나면 경고 메시지가 표시됩니다. 이 메시지는 다른 파라미터에 대하여 요구되는 값의 범위를 보여줍니다.

6. [Continue]를 선택합니다.
7. Scheduling of modifications(수정 사항 예약) 섹션에서 Apply immediately(즉시 적용)를 선택하여 변경 사항을 DB 인스턴스에 즉시 적용합니다. 또는 Apply during the next scheduled maintenance window(예약된 다음 유지 관리 기간에 적용)를 선택하여 다음 유지 관리 기간에 변경 사항을 적용합니다.
8. 변경될 파라미터를 검토하고 Modify DB instance(DB 인스턴스 수정)를 선택하여 수정을 완료합니다.

   프로비저닝된 IOPS의 새 값은 Status(상태) 열에 나타납니다.

AWS CLI

DB 인스턴스의 스토리지 성능 설정을 변경하려면 AWS CLI 명령 `modify-db-instance`를 사용합니다. 다음 파라미터를 설정합니다.

- `--storage-type` - 범용 SSD(gp3)의 경우 gp3로 설정합니다.
- `--allocated-storage` - DB 인스턴스에 할당할 스토리지 크기(GiB)입니다.

- `--iops` - DB 인스턴스에 대해 새로 설정하는 프로비저닝된 IOPS의 크기이며, 초당 I/O 작업 수로 표현됩니다.
- `--storage-throughput`- DB 인스턴스의 새 스토리지 처리량(throughput)으로, MiBps로 표현됩니다.
- `--apply-immediately` - `--apply-immediately`를 사용하면 변경 사항이 즉시 적용됩니다. 다음 유지 관리 기간에 변경 사항을 적용하려면 `--no-apply-immediately`(기본값)을 사용합니다.

RDS API

DB 인스턴스의 스토리지 성능 설정을 변경하려면 Amazon RDS API 작업 `ModifyDBInstance`를 사용합니다. 다음 파라미터를 설정합니다.

- `StorageType` - 범용 SSD(gp3)의 경우 `gp3`로 설정합니다.
- `AllocatedStorage` - DB 인스턴스에 할당할 스토리지 크기(GiB)입니다.
- `Iops` - DB 인스턴스에 대해 새로 설정하는 IOPS 속도이며, 초당 I/O 작업 수로 표현됩니다.
- `StorageThroughput`- DB 인스턴스의 새 스토리지 처리량(throughput)으로, MiBps로 표현됩니다.
- `ApplyImmediately` - 이 옵션을 `True`로 설정하면 변경 사항이 즉시 적용됩니다. 다음 유지 관리 기간에 변경 사항을 적용하려면 이 옵션을 `False`(기본값)로 설정합니다.

전용 로그 볼륨(DLV) 사용

프로비저닝된 IOPS(PIOPS) 스토리지를 사용하는 DB 인스턴스 전용 로그 볼륨(DLV)을 사용할 수 있습니다. DLV는 PostgreSQL 데이터베이스 트랜잭션 로그 및 MySQL/MariaDB의 재실행 로그와 바이너리 로그를 데이터베이스 표가 들어 있는 볼륨과 분리된 스토리지 볼륨으로 옮깁니다. DLV는 트랜잭션 쓰기 로깅을 보다 효율적이고 일관되게 만듭니다. DLV는 할당된 스토리지가 크고 초당 I/O(IOPS) 요구 사항이 높거나 지연 시간에 민감한 워크로드가 있는 데이터베이스에 적합합니다.

DLV는 PIOPS 스토리지(io1 및 io2 Block Express)에 지원되며 1,024GiB의 고정 크기와 프로비저닝된 IOPS 3,000으로 만들어집니다.

Amazon RDS는 다음 버전의 경우 모든 AWS 리전에서 DLV를 지원합니다.

- MariaDB 10.6.7 이상의 10 버전
- MySQL 8.0.28 이상 8.0 버전, MySQL 8.4.3 이상 8.4 버전
- PostgreSQL 13.10 이상의 13버전, 14.7 이상의 14 버전, 15.2 이상의 15 버전

RDS는 다중 AZ 배포와 함께 DLV를 지원합니다. 다중 AZ 인스턴스를 수정하거나 생성하면 기본 인스턴스와 보조 인스턴스 모두에 대해 DLV가 생성됩니다.

RDS는 읽기 전용 복제본이 있는 DLV를 지원합니다. 기본 DB 인스턴스에 DLV가 활성화되어 있는 경우 DLV를 활성화한 후 생성되는 모든 읽기 전용 복제본에도 DLV가 포함됩니다. DLV로 전환하기 전에 생성된 읽기 전용 복제본에는 명시적으로 수정하지 않는 한 DLV가 활성화되지 않습니다. DLV가 활성화되기 전에 기본 인스턴스에 연결된 모든 읽기 전용 복제본도 DLV를 사용하도록 수동으로 수정하는 것이 좋습니다.

> **Note**
>
> 5TiB 이상의 데이터베이스 구성에는 DLV를 사용하는 것이 좋습니다.

DLV의 이점에 대한 자세한 내용은 다음 블로그 게시물을 참조하세요.

- Amazon RDS 전용 로그 볼륨으로 데이터베이스 성능 향상
- Benchmark Amazon RDS for PostgreSQL with dedicated log volumes
- Percona 설명서의 Maximizing performance of AWS RDS for MySQL with dedicated log volumes

각 데이터베이스 엔진에 사용할 수 있는 할당된 스토리지, 프로비저닝된 IOPS, 스토리지 처리량(throughput) 범위에 대한 자세한 내용은 프로비저닝된 IOPS SSD 스토리지 섹션을 참조하세요.

주제

- DLV 활성화 및 비활성화 시 고려 사항
- DB 인스턴스를 생성할 때 DLV 활성화
- 기존 DB 인스턴스에서 DLV 활성화
- DLV 스토리지 모니터링

DLV 활성화 및 비활성화 시 고려 사항

DLV 활성화 및 비활성화 시 시간이 많이 걸리고 가동 중지가 발생할 수 있습니다. 이 프로세스에는 활성화한 경우 모든 트랜잭션 로그 또는 다시 실행 및 이진수 로그(데이터베이스 엔진에 따라 다름)를 새 볼륨에 복사하고, 비활성화할 때는 원래 스토리지로 다시 복사하는 작업이 포함됩니다. 이 작업 기간은 다음과 같은 여러 요인의 영향을 받습니다.

- 트랜잭션 로그 수:

  - 데이터베이스가 크고 트랜잭션이 많을수록 로그가 더 많이 생성되므로, 복사에 더 많은 시간이 걸립니다.
  - 복제 슬롯이 비활성 상태이거나 복제가 지연되는 경우 기본 DB 인스턴스에 트랜잭션 로그가 누적되어 복사에 필요한 시간이 늘어날 수 있습니다. 복제가 최신인지 확인하고 불필요한 슬롯은 모두 제거하세요.
- 스토리지 구성:
  - DB 인스턴스 EBS 대역폭 - 대역폭이 높을수록 데이터 전송 속도가 빨라집니다.
  - 프로비저닝된 IOPS 수 - 초당 입출력 작업 처리량(IOPS)이 증가하면 복사 프로세스의 속도를 높일 수 있습니다.
- 데이터베이스 활동 - 구성 중 데이터베이스 활동 수준이 활발하면 프로세스 속도가 느려질 수 있습니다.

가동 중지를 최소화하려면 작업량이 적은 기간이나 유지 관리 기간에 계획을 세우고 일정을 잡는 것이 좋습니다.

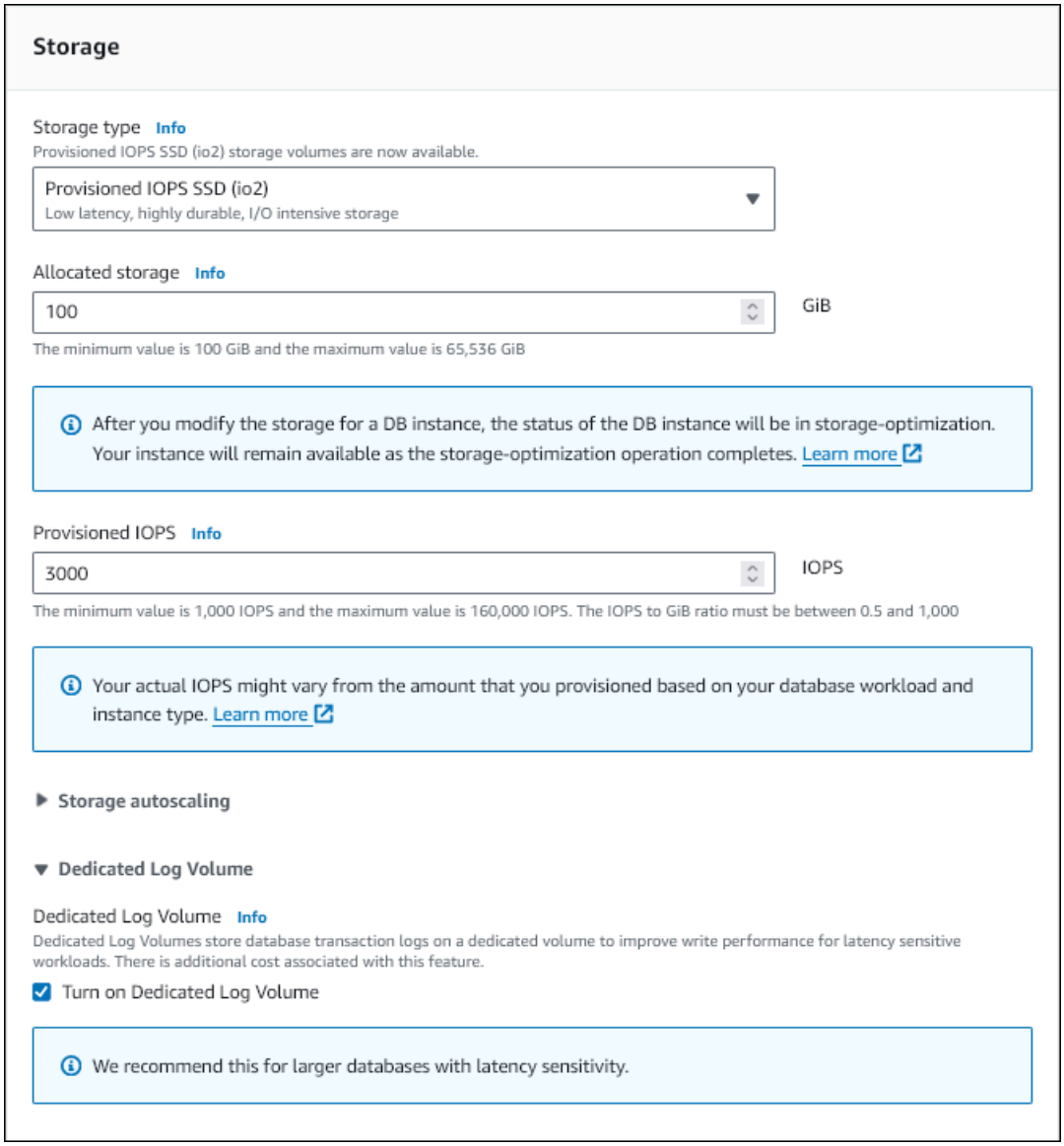
DB 인스턴스를 생성할 때 DLV 활성화

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DLV가 활성화된 DB 인스턴스를 만들 수 있습니다.

콘솔

새로운 DB 인스턴스에서 DLV를 활성화하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 데이터베이스 생성을 선택합니다.
3. DB 인스턴스 생성 페이지에서 DLV를 지원하는 DB 엔진을 선택합니다.
4. 스토리지에서 다음을 수행합니다.
   a. 프로비저닝된 IOPS SSD(io1) 또는 프로비저닝된 IOPS SSD(io2)를 선택합니다.
   b. 할당된 스토리지와 원하는 프로비저닝된 IOPS를 입력합니다.
   c. 전용 로그 볼륨을 확장하고 전용 로그 볼륨 켜기를 선택합니다.



5. 복제본에 필요에 따라 다른 설정을 선택합니다.
6. 데이터베이스 생성를 선택합니다.

데이터베이스가 생성되면 데이터베이스 세부 정보 페이지의 구성 탭에 전용 로그 볼륨 값이 나타납니다.

CLI

프로비저닝된 IOPS 스토리지를 사용하는 DB 인스턴스를 생성할 때 DLV를 활성화하려면 create-db-instance AWS CLI 명령을 사용합니다. 다음 파라미터를 설정합니다.

- `--dedicated-log-volume` - 전용 로그 볼륨을 활성화합니다.
- `--storage-type` - 프로비저닝된 IOPS의 경우 `io1` 또는 `io2`로 설정합니다.
- `--allocated-storage` - DB 인스턴스에 할당할 스토리지 크기(GiB)입니다.
- `--iops` - DB 인스턴스에 대해 설정하는 프로비저닝된 IOPS의 크기이며, 초당 I/O 작업 수로 표현됩니다.

RDS API

프로비저닝된 IOPS 스토리지를 사용하는 DB 인스턴스를 생성할 때 DLV를 활성화하려면 CreateDBInstance Amazon RDS API 작업을 사용합니다. 다음 파라미터를 설정합니다.

- `DedicatedLogVolume` - 전용 로그 볼륨을 활성화하려면 `true`로 설정합니다.
- `StorageType` - 프로비저닝된 IOPS의 경우 `io1` 또는 `io2`로 설정합니다.
- `AllocatedStorage` - DB 인스턴스에 할당할 스토리지 크기(GiB)입니다.
- `Iops` - DB 인스턴스에 대해 설정하는 IOPS 속도이며, 초당 I/O 작업 수로 표현됩니다.

기존 DB 인스턴스에서 DLV 활성화

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DLV를 활성화하도록 DB 인스턴스를 수정할 수 있습니다.

DB 인스턴스의 DLV 설정을 수정한 후에는 DB 인스턴스를 재부팅해야 합니다.

콘솔

기존 DB 인스턴스에서 DLV를 활성화하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.

   DB 인스턴스의 목록을 필터링하려면 Filter databases(데이터베이스 필터링)에 Amazon RDS가 결과를 필터하는 데 사용할 텍스트 문자열을 입력합니다. 이름이 해당 문자열을 포함하는 DB 인스턴스만 표시됩니다.
3. 수정하려는 프로비저닝된 IOPS 스토리지가 있는 DB 인스턴스를 선택합니다.
4. 수정을 선택합니다.

5. DB 인스턴스 수정 페이지에서 다음을 수행합니다.
   - 스토리지에서 전용 로그 볼륨을 확장하고 전용 로그 볼륨 켜기를 선택합니다.
6. 계속을 선택합니다.
7. DB 인스턴스에 대한 변경 사항을 즉시 적용하려면 즉시 적용을 선택합니다. 또는 Apply during the next scheduled maintenance window(예약된 다음 유지 관리 기간에 적용)를 선택하여 다음 유지 관리 기간에 변경 사항을 적용합니다.
8. 변경될 파라미터를 검토하고 Modify DB instance(DB 인스턴스 수정)를 선택하여 수정을 완료합니다.

데이터베이스 세부 정보 페이지의 구성 탭에 새로운 전용 로그 볼륨 값이 나타납니다.

CLI

프로비저닝된 IOPS 스토리지를 사용하는 기존 DB 인스턴스에서 DLV를 활성화 또는 비활성화하려면 AWS CLI 명령인 `modify-db-instance`를 사용합니다. 다음 파라미터를 설정합니다.

- `--dedicated-log-volume` - 전용 로그 볼륨을 활성화합니다.

  전용 로그 볼륨을 비활성화하려면 `--no-dedicated-log-volume`(기본값)을 사용합니다.
- `--apply-immediately` - `--apply-immediately`를 사용하면 변경 사항이 즉시 적용됩니다.

  다음 유지 관리 기간에 변경 사항을 적용하려면 `--no-apply-immediately`(기본값)을 사용합니다.

RDS API

프로비저닝된 IOPS 스토리지를 사용하는 기존 DB 인스턴스에서 DLV를 활성화 또는 비활성화하려면 Amazon RDS API 작업인 `ModifyDBInstance`를 사용합니다. 다음 파라미터를 설정합니다.

- `DedicatedLogVolume` - 전용 로그 볼륨을 활성화하려면 이 옵션을 `true`로 설정합니다.

  전용 로그 볼륨을 비활성화하려면 이 옵션을 `false`로 설정합니다. 이것이 기본값입니다.
- `ApplyImmediately` - 이 옵션을 `True`로 설정하면 변경 사항이 즉시 적용됩니다.

  다음 유지 관리 기간에 변경 사항을 적용하려면 이 옵션을 `False`(기본값)로 설정합니다.

DLV 스토리지 모니터링

CloudWatch의 `FreeStorageSpaceLogVolume` 지표를 사용하여 DLV 스토리지 사용량을 모니터링할 수 있습니다.

RDS for PostgreSQL에 대한 다음 쿼리를 사용하여 트랜잭션 로그가 차지하는 크기를 찾아볼 수 있습니다.

```
SELECT pg_size_pretty(COALESCE(sum(size), 0)) AS total_wal_generated_size
FROM pg_catalog.pg_ls_waldir();
```

DLV의 스토리지가 부족하면 DB 인스턴스가 `storage-full` 상태로 전환되어 가동 중지가 발생합니다.

DB 인스턴스 삭제

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB 인스턴스를 삭제할 수 있습니다. Aurora DB 클러스터의 DB 인스턴스를 삭제하려면 Aurora DB 클러스터 및 DB 인스턴스 삭제를 참조하세요.

주제

- DB 인스턴스를 삭제하기 위한 사전 조건
- DB 인스턴스 삭제 시 고려 사항
- DB 인스턴스 삭제

DB 인스턴스를 삭제하기 위한 사전 조건

DB 인스턴스를 삭제하려면 우선 삭제 방지가 꺼져 있는지 확인합니다. 기본적으로 콘솔에서 생성된 DB 인스턴스에는 삭제 방지가 켜져 있습니다.

DB 인스턴스에 삭제 방지가 켜져 있는 경우 인스턴스 설정을 수정하여 이 기능을 끌 수 있습니다. 데이터베이스 세부 정보 페이지에서 수정을 선택하거나 modify-db-instance 명령을 호출합니다. 이 작업은 중단을 야기하지 않습니다. 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하십시오.

DB 인스턴스 삭제 시 고려 사항

DB 인스턴스를 삭제하면 인스턴스 복구 가능성, 백업 가용성, 읽기 전용 복제본 상태에 영향을 미칩니다. 다음 문제를 고려하세요.

- 최종 DB 스냅샷을 생성할지 여부를 선택할 수 있습니다. 다음과 같은 옵션이 있습니다:
  - 최종 스냅샷을 생성하면 삭제된 DB 인스턴스를 복구하는 데 사용할 수 있습니다. RDS는 최종 스냅샷과 이전에 촬영한 수동 스냅샷 두 가지를 모두 보존합니다. DB 인스턴스가 `Available` 상태일 때는 DB 인스턴스의 최종 DB 스냅샷을 생성할 수 없습니다. 자세한 내용은 Amazon RDS DB 인스턴스 상태 보기 단원을 참조하십시오.
  - 최종 스냅샷을 생성하지 않으면 인스턴스 삭제 속도가 더 빨라집니다. 단점은 나중에 복원할 수 있는 최종 스냅샷이 없다는 것입니다. 삭제된 DB 인스턴스를 복원하려는 경우, 자동 백업을 유지하거나 이전의 수동 스냅샷을 사용하여 DB 인스턴스를 이전 스냅샷 시점으로 복원하세요.
- 자동 백업을 보존할지 여부를 선택할 수 있습니다. 다음과 같은 옵션이 있습니다:
  - 자동 백업을 보관할 경우, RDS는 삭제 시점에 DB 인스턴스에 적용되는 보존 기간 동안 자동 백업을 보존합니다. 자동 백업을 사용하여 DB 인스턴스를 보존 기간 이후가 아닌 보존 기간 중간의 특

정 시점으로 복원할 수 있습니다. 이 보존 기간은 최종 DB 스냅샷을 생성하는지 여부와 관계없이 적용됩니다. 보관된 자동 백업을 삭제하려면 보관된 자동 백업 삭제 섹션을 참조하세요.

- 보존된 자동 백업 및 수동 스냅샷은 삭제될 때까지 청구 요금이 부과됩니다. 자세한 내용은 보존 비용 단원을 참조하십시오.
- 자동 백업을 보관하도록 선택하지 않으면 RDS는 DB 인스턴스와 동일한 AWS 리전에 있는 자동 백업을 삭제합니다. 이러한 백업은 복구할 수 없습니다. 자동 백업을 다른 AWS 리전에 복제할 경우, 자동 백업을 보존하도록 선택하지 않으면 RDS가 백업을 보존합니다. 자세한 내용은 다른 AWS 리전에 자동 백업 복제 단원을 참조하십시오.

> **Note**
>
> 최종 DB 스냅샷을 생성할 경우 일반적으로 자동 백업은 보관할 필요가 없습니다.

- DB 인스턴스를 삭제하면 RDS는 수동 DB 스냅샷을 삭제하지 않습니다. 자세한 내용은 Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성 단원을 참조하십시오.
- 모든 RDS 리소스를 삭제하려는 경우 다음과 같은 리소스에 요금이 청구된다는 점을 유의하세요.
  - DB 인스턴스
  - DB 스냅샷
  - DB 클러스터

  예약 인스턴스를 구입한 경우 인스턴스를 구입했던 당시에 동의한 계약에 따라 요금이 청구됩니다. 자세한 내용은 Amazon RDS용 예약 DB 인스턴스 단원을 참조하십시오. AWS Cost Explorer를 사용하여 모든 AWS 리소스에 대한 요금 청구 정보를 얻을 수 있습니다. 자세한 내용은 AWS Cost Explorer로 비용 분석을 참조하세요.
- 동일한 AWS 리전에서 읽기 전용 복제본이 있는 DB 인스턴스를 삭제할 경우, 각 읽기 전용 복제본은 독립 실행형 DB 인스턴스로 자동 승격됩니다. 자세한 내용은 읽기 전용 복제본을 독립 DB 인스턴스로 승격 단원을 참조하십시오. DB 인스턴스에 다른 AWS 리전의 읽기 전용 복제본이 있는 경우, 교차 리전 읽기 전용 복제본의 소스 DB 인스턴스 삭제와 관련된 내용은 리전 간 복제 시 고려 사항 섹션을 참조하세요.
- DB 인스턴스의 상태가 `deleting`인 경우 CA 인증서 값이 RDS 콘솔이나 AWS CLI 명령 또는 RDS API 작업의 출력에 나타나지 않습니다. CA 인증서에 대한 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 단원을 참조하십시오.
- 해당 DB 인스턴스를 삭제하는 데 필요한 시간은 백업 보존 기간(즉, 삭제할 백업 수), 삭제되는 데이터의 양과 최종 스냅샷 생성 여부에 따라 달라집니다.

DB 인스턴스 삭제

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB 인스턴스를 삭제할 수 있습니다. 필요한 작업은 다음과 같습니다.

- DB 인스턴스 이름 입력
- 인스턴스의 최종 DB 스냅샷을 생성하는 옵션을 활성화하거나 비활성화합니다.
- 자동 백업을 보관하는 옵션을 활성화하거나 비활성화합니다.

> **Note**
>
> 삭제 방지가 켜진 DB 인스턴스는 삭제할 수 없습니다. 자세한 내용은 DB 인스턴스를 삭제하기 위한 사전 조건 단원을 참조하십시오.

콘솔

DB 인스턴스를 삭제하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 후 삭제하려는 DB 인스턴스를 선택합니다.
3. [Actions]에 대해 [Delete]를 선택합니다.
4. DB 인스턴스의 최종 DB 스냅샷을 생성하려면 최종 스냅샷 생성 여부를 선택합니다.
5. 최종 스냅샷을 생성하도록 선택한 경우 최종 스냅샷 이름을 입력합니다.
6. 자동 백업을 보관하려면 자동 백업 보관을 선택합니다.
7. 상자에 **`delete me`**를 입력합니다.
8. 삭제를 선택합니다.

AWS CLI

계정에서 DB 인스턴스의 인스턴스 ID를 찾으려면 describe-db-instances 명령을 호출합니다.

```
aws rds describe-db-instances --query 'DBInstances[*].[DBInstanceIdentifier]' --output
 text
```

AWS CLI를 사용하여 DB 인스턴스를 삭제하려면 다음 옵션과 함께 delete-db-instance 명령을 호출하십시오.

- `--db-instance-identifier`
- `--final-db-snapshot-identifier` 또는 `--skip-final-snapshot`

Example 최종 스냅샷 사용, 자동 백업 보관 안 함

대상 LinuxmacOS, 또는Unix:

```
aws rds delete-db-instance \
    --db-instance-identifier mydbinstance \
    --final-db-snapshot-identifier mydbinstancefinalsnapshot \
    --delete-automated-backups
```

Windows의 경우:

```
aws rds delete-db-instance ^
    --db-instance-identifier mydbinstance ^
    --final-db-snapshot-identifier mydbinstancefinalsnapshot ^
    --delete-automated-backups
```

Example 자동 백업 보관, 최종 스냅샷 없음

대상 LinuxmacOS, 또는Unix:

```
aws rds delete-db-instance \
    --db-instance-identifier mydbinstance \
    --skip-final-snapshot \
    --no-delete-automated-backups
```

Windows의 경우:

```
aws rds delete-db-instance ^
    --db-instance-identifier mydbinstance ^
    --skip-final-snapshot ^
    --no-delete-automated-backups
```

RDS API

Amazon RDS API를 사용하여 DB 인스턴스를 삭제하려면 다음 파라미터와 함께 `DeleteDBInstance` 작업을 호출하십시오.

- `DBInstanceIdentifier`
- `FinalDBSnapshotIdentifier` 또는 `SkipFinalSnapshot`

자습서: 개발에서 프로덕션까지 MySQL DB 인스턴스 환경 관리

소개

Amazon RDS DB 인스턴스를 관리할 때 흔히 발생하는 시나리오는 초기 개발부터 프로덕션 배포까지의 수명 주기를 감독하는 것입니다. 이 자습서에서는 데이터베이스가 최적의 성능을 발휘하고 변화하는 운영 요구 사항에 맞게 조정되도록 주요 태스크를 처리하는 방법을 안내합니다. 또한, 일관성과 신뢰성을 보장하기 위해 개발 환경과 프로덕션 환경 간의 변경 사항을 동기화하는 옵션도 간략하게 설명합니다.

이 단계를 완료하면 다음을 학습하게 됩니다.

- MySQL DB 인스턴스를 활용하여 Amazon RDS 태그 추가 및 업데이트, 스토리지 확장, 읽기 전용 복제본 생성, 리소스 삭제 등 특정 태스크를 수행하는 방법.
- 포괄적인 테스트 및 검증을 위해 프로덕션 환경의 업데이트를 개발 환경으로 동기화하는 방법.

이 자습서를 완료하려면 다음 작업을 수행하세요.

1. MySQL DB 인스턴스를 생성합니다.
2. Amazon RDS 태그를 추가하여 DB 인스턴스를 개발 환경으로 분류합니다.
3. 증가한 워크로드를 수용할 수 있도록 DB 인스턴스의 스토리지 용량을 늘립니다.
4. 읽기 전용 복제본을 생성하여 DB 인스턴스의 복원력과 가용성을 강화합니다.
5. Amazon RDS 태그를 업데이트하여 DB 인스턴스를 프로덕션 환경으로 분류합니다.
6. 더 이상 필요하지 않은 DB 인스턴스는 삭제하여 추가 비용이 발생하지 않도록 합니다.
7. 다음 단계: 환경 간 일관성을 위해 개발 인스턴스를 프로덕션과 동기화

사전 조건

시작하기 전에 다음 섹션에서 다음 단계를 완료하세요.

- AWS 계정에 등록
- 관리자 액세스 권한이 있는 사용자 생성

태그를 추가하여 DB 인스턴스를 개발 환경으로 분류

DB 인스턴스를 개발 환경으로 분류하려면 생성한 인스턴스에 Amazon RDS 태그를 추가합니다. Amazon RDS 태그는 사용자가 정의하고 RDS 인스턴스와 연결하는 키-값 쌍입니다. AWS 리소스에 태그를 지정하면 개발 리소스와 프로덕션 AWS 리소스를 구분하는 데 도움이 됩니다. Amazon RDS 태그에 관한 자세한 내용은 Amazon RDS 리소스에 태그 지정 섹션을 참조하세요.

1. Amazon RDS 콘솔에서 데이터베이스를 선택합니다.
2. 태그할 DB 인스턴스를 선택합니다.
3. 세부 정보 섹션에서 태그 섹션으로 스크롤합니다.
4. 태그 관리를 선택하고 새 태그 추가를 선택합니다.
5. 태그 키와 값에 값을 입력합니다. 예를 들어, dev 값이 있는 태그 키 환경을 사용하여 데이터베이스 인스턴스가 개발 환경의 일부임을 지정할 수 있습니다.
6. 새 태그 추가 및 변경 사항 저장을 선택합니다.

이제 DB 인스턴스는 개발 환경으로 태그가 지정되었습니다. 이렇게 하면 DB 인스턴스를 더 쉽게 식별하고 이 리소스와 관련된 비용을 관리할 수 있습니다.

증가하는 데이터 요구 사항을 수용할 수 있도록 DB 인스턴스의 스토리지 용량 증대

다음으로 MySQL DB 인스턴스의 스토리지 용량을 수정하여 추가 데이터를 수용합니다. 처음에는 DB 인스턴스의 스토리지 용량이 애플리케이션의 즉각적인 요구 사항을 충족하도록 설정됩니다. 하지만 데이터 볼륨이 증가하면 데이터베이스의 성능과 안정성을 지속적으로 유지할 수 있도록 스토리지 설정을 조정해야 할 수도 있습니다. 이 프로세스에는 DB 인스턴스에 할당된 스토리지를 늘리는 작업이 포함됩니다. DB 인스턴스의 스토리지 용량을 수정하는 방법에 대한 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 작업 섹션을 참조하세요.

1. Amazon RDS 콘솔에서 데이터베이스를 선택합니다.
2. 수정하려는 DB 인스턴스를 선택합니다.
3. 수정을 선택합니다.
4. 스토리지에서 할당된 스토리지를 늘립니다. 수정된 스토리지 값은 현재 값보다 커야 합니다.
5. 계속을 선택합니다.

6. 수정 사항 예약에서 즉시 적용을 선택하여 DB 인스턴스에 스토리지 변경 사항을 즉시 적용하거나 예약된 다음 유지 관리 기간에 적용을 선택하여 다음 유지 관리 기간 동안 변경 사항을 적용할 수 있습니다.
7. 원하는 대로 설정이 되었으면 [Modify DB instance]를 선택합니다.

이제 DB 인스턴스의 스토리지 용량이 늘어났습니다. 이를 통해 더 큰 데이터 볼륨을 효과적으로 처리하고 애플리케이션의 데이터 요구가 증가함에 따라 지속적인 성능과 안정성을 보장할 수 있습니다.

읽기 전용 복제본을 생성하여 DB 인스턴스의 복원력과 가용성 강화

MySQL DB 인스턴스의 읽기 전용 복제본을 생성합니다. 읽기 전용 복제본은 DB 인스턴스의 복원력과 가용성을 개선합니다. 기본 DB 인스턴스의 읽기 트래픽을 줄이려면 DB 인스턴스의 읽기 전용 복제본을 만드세요. 이렇게 하면 쿼리가 읽기 전용 복제본으로 라우팅되므로, 부하를 분산하고 전반적인 데이터베이스 성능을 개선할 수 있습니다. DB 인스턴스 읽기 전용 복제본에 대한 자세한 내용은 DB 인스턴스 읽기 전용 복제본 작업 섹션을 참조하세요.

MySQL DB 인스턴스가 복제 소스 역할을 하려면 소스 DB 인스턴스에서 자동 백업을 활성화해야 합니다. 이 작업은 백업 보존 기간을 0이 아닌 다른 값으로 설정하여 수행할 수 있습니다. MySQL 읽기 전용 복제본에 대한 자세한 내용은 MySQL 읽기 전용 복제본 작업 섹션을 참조하세요.

1. Amazon RDS 콘솔에서 데이터베이스를 선택합니다.
2. 읽기 전용 복제본의 소스로 사용할 DB 인스턴스를 선택합니다.
3. 작업에서 읽기 전용 복제본 생성을 선택합니다.
4. DB 인스턴스 식별자에 읽기 전용 복제본의 이름을 모두 소문자로 입력합니다.
5. 인스턴스 구성을 선택합니다. 읽기 전용 복제본의 소스 DB 인스턴스보다 크거나 같은 DB 인스턴스 클래스와 스토리지 유형을 사용하는 것이 좋습니다.
6. AWS 리전에서 읽기 전용 복제본의 대상 리전을 지정합니다.
7. 기본 설정을 그대로 두거나 필요에 따라 수정합니다.
8. 읽기 전용 복제본 생성을 선택합니다.

읽기 전용 복제본은 RDS 콘솔의 데이터베이스 페이지에서 소스 DB 인스턴스 아래에 나타납니다. 역할 열에 복제본이 표시됩니다.

태그를 업데이트하여 DB 인스턴스를 프로덕션 환경으로 분류

DB 인스턴스를 개발 단계에서 프로덕션 단계로 전환할 준비가 되면 전환 단계를 반영하여 태그를 업데이트하는 것이 중요합니다. DB 인스턴스를 운영 및 모니터링 전략에 맞게 조정하려면 DB 인스턴스가 이제 프로덕션 환경의 일부임을 나타내도록 초기 태그를 업데이트하세요. 이렇게 하면 데이터베이스를 더 잘 파악하고 관리할 수 있습니다.

1. Amazon RDS 콘솔에서 데이터베이스를 선택합니다.
2. 업데이트할 DB 인스턴스 선택
3. 세부 정보 섹션에서 태그 섹션으로 스크롤합니다.
4. 태그 관리를 선택합니다.
5. 개발 환경을 나타내는 초기 태그를 제거합니다.
6. 새 태그 추가를 선택합니다.
7. 태그 키와 값에 새 값을 입력합니다. 예를 들어, prod 값과 함께 태그 키 환경을 사용하여 DB 인스턴스가 프로덕션 환경의 일부임을 지정할 수 있습니다.
8. 새 태그 추가 및 변경 사항 저장을 선택합니다.

DB 인스턴스의 태그가 업데이트되어 데이터베이스가 프로덕션 환경으로 전환되었음을 알립니다.

추가 비용이 발생하지 않도록 더 이상 필요하지 않은 DB 인스턴스 삭제

이 자습서를 마치기 전에 리소스 관리 문제를 해결하는 것이 중요합니다. 더 이상 필요하지 않은 리소스가 있는 경우 추가 비용 발생을 방지하고 클라우드 환경을 최적화하려면 해당 리소스를 삭제해야 합니다.

1. Amazon RDS 콘솔에서 데이터베이스를 선택합니다.
2. 삭제할 DB 인스턴스 선택
3. 작업에서 삭제를 선택합니다. DB 인스턴스를 삭제하면 인스턴스가 모든 내용 및 관련 리소스와 함께 영구적으로 삭제됩니다.
4. DB 인스턴스의 삭제를 확인하고 삭제를 선택합니다.

또는 나중에 사용할 수 있도록 DB 인스턴스를 유지 관리하기로 선택한 경우 프로덕션 환경의 일부로 계속 관리할 수 있습니다. 여기에는 포괄적인 테스트 및 검증을 용이하게 하기 위해 동기화된 개발 환경을 유지하는 작업이 포함됩니다. 자세한 내용은 다음 단계: 환경 간 일관성을 위해 개발 인스턴스를 프로덕션과 동기화 단원을 참조하십시오.

다음 단계: 환경 간 일관성을 위해 개발 인스턴스를 프로덕션과 동기화

개발 환경 생성

프로덕션 환경을 관리하려면 포괄적인 테스트 및 검증을 위해 동기화된 개발 환경을 유지하는 것이 중요합니다. 새로운 개발 환경을 만들려면 먼저 현재 프로덕션 DB 인스턴스의 DB 스냅샷을 만드세요. DB 스냅샷은 스토리지 볼륨 스냅샷을 생성하여 전체 DB 인스턴스를 캡처합니다. Amazon RDS 콘솔에서 DB 스냅샷을 생성하는 방법에 대한 지침은 Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성 섹션을 참조하세요.

프로덕션 환경의 DB 스냅샷을 만든 후 DB 스냅샷을 복원하여 개발 환경을 위한 새 DB 인스턴스를 생성합니다. 복원된 DB 인스턴스는 기본 DB 파라미터 및 옵션 그룹과 자동으로 연결됩니다. 하지만 복원 중에 사용자 지정 파라미터 그룹 및 옵션 그룹을 지정하여 적용할 수 있습니다. DB 스냅샷 복원에 대한 지침은 자습서: DB 스냅샷에서 Amazon RDS DB 인스턴스 복원 섹션을 참조하세요.

마지막으로 Amazon RDS 태그를 업데이트하여 새 DB 인스턴스를 새 개발 환경으로 지정합니다. 이 변경 사항을 반영하도록 Amazon RDS 태그를 업데이트하는 방법에 대한 지침은 이전 태그를 업데이트하여 DB 인스턴스를 프로덕션 환경으로 분류 섹션을 참조하세요.

이제 프로덕션 환경의 데이터베이스 구성을 반영하는 새로운 개발 환경이 생겼습니다.

개발 환경을 프로덕션 환경과 동기화

새 개발 환경을 설정한 후에는 프로덕션 환경에서 발생하는 모든 변경 사항과 동기화된 상태를 유지해야 합니다. 이렇게 하면 개발 환경이 현재 프로덕션 상태를 정확하게 반영할 수 있으며, 이는 효과적인 테스트, 검증 및 문제 해결에 필수적입니다. Amazon RDS는 개발 환경을 프로덕션 환경에 맞게 최신 상태로 유지할 수 있는 다양한 방법을 제공합니다. 이러한 옵션에 대한 자세한 내용은 Orchestrating database refreshes for Amazon RDS and Amazon Aurora를 참조하세요.

개발 및 프로덕션 환경을 동기화하는 기본적인 방법 중 하나는 DB 스냅샷을 생성하고 복원하는 것입니다. DB 스냅샷을 사용하면 스냅샷이 생성된 시점의 프로덕션 환경 데이터베이스 구성을 반영하는 개발 환경을 만들 수 있습니다. DB 스냅샷에 대한 자세한 내용은 수동 백업 관리 섹션을 참조하세요. DB 인스턴스 복원에 대한 자세한 내용은 DB 인스턴스 복원 섹션을 참조하세요.

DB 스냅샷은 다음과 같은 사용 사례에 특히 유용합니다.

- 개발 환경의 초기 설정: DB 스냅샷은 스냅샷 생성 당시 프로덕션 환경의 정확한 상태를 반영하는 일관된 기준을 제공하므로, 테스트를 위한 초기 개발 환경을 만드는 데 유용합니다.
- 트래픽이 많은 애플리케이션: 지속적인 운영이 중요한 프로덕션 환경에서 스냅샷에 다중 AZ 배포를 사용하면 기본 데이터베이스의 I/O 중단을 방지하여 끊김 없는 성능과 고가용성을 보장할 수 있습니다.

- 여러 RDS 계정 간 데이터 공유: DB 스냅샷을 서로 다른 AWS 계정 간에 공유할 수 있어 계정 또는 리전 간에 데이터를 쉽게 전송할 수 있습니다. 이는 다양한 목적으로 데이터를 공유해야 하는 공동 프로젝트 또는 시나리오에 유용합니다. 자세한 내용은 Amazon RDS의 DB 스냅샷 공유 단원을 참조하십시오.

이 자습서에서는 DB 인스턴스의 수명 주기 전반에 걸쳐 DB 인스턴스를 관리하기 위한 필수 태스크를 살펴보았습니다. DB 인스턴스 생성, Amazon RDS 태그 추가 및 업데이트, 스토리지 확장, 읽기 전용 복제본 생성 방법을 배웠습니다. 또한, 이러한 기본 작업을 기반으로 구축하고 프로덕션 환경을 효과적으로 관리하는 방법도 배웠습니다. 여기에는 테스트를 위한 개발 환경을 구축하고 일관성을 위해 프로덕션 환경과 동기화하는 작업이 포함되었습니다. 이러한 태스크는 탄력적이고 확장 가능한 데이터베이스 인프라를 유지 관리하여 Amazon RDS 환경을 효율적으로 운영하는 데 도움이 됩니다.

Amazon RDS에 대한 다중 AZ 배포 구성 및 관리

다중 AZ 배포에는 대기 DB 인스턴스가 하나 또는 두 개 있을 수 있습니다. 배포에 하나의 대기 DB 인스턴스가 있는 경우를 다중 AZ DB 인스턴스 배포라고 합니다. 다중 AZ DB 인스턴스 배포에는 장애 조치 지원을 제공하지만, 읽기 트래픽은 처리하지 않는 대기 DB 인스턴스가 하나 있습니다. 배포에 두 개의 대기 DB 인스턴스가 있는 경우를 다중 AZ DB 클러스터 배포라고 합니다. 다중 AZ DB 클러스터 배포에는 장애 조치 지원을 제공하고 읽기 트래픽도 처리할 수 있는 대기 DB 인스턴스가 있습니다.

AWS Management Console을 사용하여 다중 AZ 배포가 다중 AZ DB 인스턴스 배포인지 다중 AZ DB 클러스터 배포인지 확인할 수 있습니다. 탐색 창에서 데이터베이스(Databases)를 선택한 후 DB 식별자(DB identifier)를 선택합니다.

- 다중 AZ DB 인스턴스 배포의 특징은 다음과 같습니다.
  - DB 인스턴스에는 행이 하나만 있습니다.
  - 역할(Role) 값은 인스턴스(Instance) 또는 기본(Primary)입니다.
  - 다중 AZ(Multi-AZ) 값은 예(Yes)입니다.
- 다중 AZ DB 클러스터 배포의 특징은 다음과 같습니다.
  - 클러스터 수준 행에는 세 개의 DB 인스턴스 행이 있습니다.
  - 클러스터 수준 행의 경우 역할(Role) 값은 다중 AZ DB 클러스터(Multi-AZ DB cluster)입니다.
  - 각 인스턴스 수준 행에서 역할(Role) 값은 라이터 인스턴스(Writer instance) 또는 리더 인스턴스(Reader instance)입니다.
  - 각 인스턴스 수준 행에서 다중 AZ(Multi-AZ) 값은 3 영역(3 Zones)입니다.

주제

- Amazon RDS에 대한 다중 AZ 인스턴스 배포
- Amazon RDS용 다중 AZ DB 클러스터 배포

또한 다음 주제는 DB 인스턴스와 다중 AZ DB 클러스터에 모두 적용됩니다.

- the section called “RDS 리소스에 태그 지정”
- the section called “Amazon RDS의 ARN”
- the section called “스토리지 작업”
- the section called “DB 인스턴스 유지 관리”

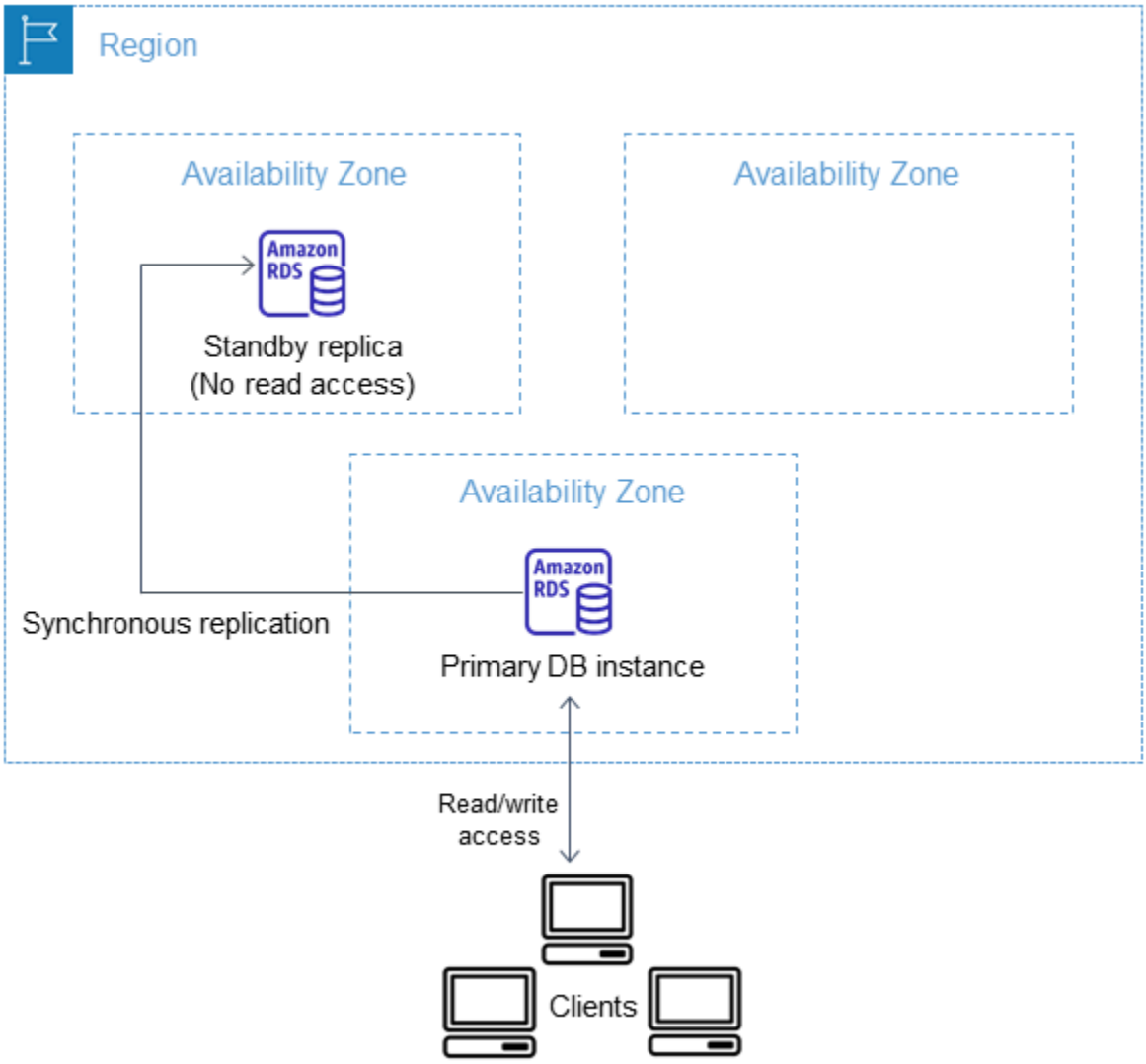
- the section called “엔진 버전 업그레이드”

Amazon RDS에 대한 다중 AZ 인스턴스 배포

Amazon RDS는 단일 대기 DB 인스턴스와 함께 다중 AZ 배포를 사용하여 DB 인스턴스에 대한 고가용성 및 장애 조치 지원을 제공합니다. 이러한 유형의 배포를 다중 AZ DB 인스턴스 배포라고 합니다. Amazon RDS는 다양한 기술을 통해 장애 조치 지원을 제공합니다. MariaDB, MySQL, Oracle, RDS Custom for SQL Server DB인스턴스용 다중 AZ 배포는 Amazon의 장애 조치 기술을 사용합니다. Microsoft SQL Server DB 인스턴스는 SQL Server 데이터베이스 미러링(DBM) 또는 상시 가동 가용성 그룹(AG)을 사용합니다. 다중 AZ에 대한 SQL Server 버전 지원에 대한 자세한 내용은 Amazon RDS for Microsoft SQL Server의 다중 AZ 배포 섹션을 참조하세요. RDS Custom for SQL Server로 다중 AZ 작업을 수행하는 방법에 대한 내용은 RDS Custom for SQL Server에 대한 다중 AZ 배포 구성 및 관리 섹션을 참조하세요.

다중 AZ DB 인스턴스 배포에서 Amazon RDS는 자동으로 서로 다른 가용 영역에 동기식 대기 복제본을 프로비저닝하고 유지합니다. 프라이머리 DB 인스턴스는 전체 가용 영역에서 대기 복제본으로 동기식으로 복제되어 시스템 백업 중에 데이터 이중화를 제공하고 대기 시간 급증을 최소화합니다. DB 인스턴스를 고가용성으로 실행하면 계획된 시스템 유지 관리 중 가용성을 높일 수 있습니다. 또한, DB 인스턴스 오류 및 가용 영역 중단이 일어나지 않도록 방지할 수 있습니다. 가용 영역에 대한 자세한 내용은 리전, 가용 영역 및 로컬 영역 단원을 참조하세요.

> **Note**
>
> 고가용성 옵션은 읽기 전용 시나리오에서는 확장 솔루션이 아닙니다. 대기 복제본을 사용하여 읽기 트래픽을 처리할 수 없습니다. 읽기 전용 트래픽을 처리하려면 대신 다중 AZ DB 클러스터 또는 읽기 전용 복제본을 사용해야 합니다. 다중 AZ DB 클러스터에 대한 자세한 내용은 Amazon RDS용 다중 AZ DB 클러스터 배포 섹션을 참조하세요. 읽기 전용 복제본에 대한 자세한 내용은 DB 인스턴스 읽기 전용 복제본 작업을 참조하세요.



RDS 콘솔을 통해 DB 인스턴스 생성 시 다중 AZ를 지정하여 간편하게 다중 AZ DB 인스턴스 배포를 생성할 수 있습니다. 콘솔을 통해 DB 인스턴스를 수정하고 다중 AZ 옵션을 지정하여 기존 DB 인스턴스를 다중 AZ DB 인스턴스 배포로 변환할 수 있습니다. 또한, AWS CLI 또는 Amazon RDS API를 사용하여 다중 AZ DB 인스턴스 배포를 지정할 수도 있습니다. create-db-instance 또는 modify-db-instance 명령을 사용하거나 CreateDBInstance 또는 ModifyDBInstance API 작업을 사용합니다.

RDS 콘솔에 예비 복제본(보조 AZ라고 함)의 가용 영역이 표시됩니다. 또한 describe-db-instances CLI 명령 또는 DescribeDBInstances API 작업을 사용하여 보조 AZ를 찾을 수도 있습니다.

다중 AZ DB 인스턴스 배포를 사용하는 DB 인스턴스는 단일 AZ 배포에 비해 쓰기 및 커밋 대기 시간이 길어질 수 있습니다. 이러한 현상은 동기식 데이터 복제가 발생하기 때문에 일어날 수 있습니다. AWS는 가용 영역 간 지연 시간이 짧은 네트워크 연결을 제공하도록 설계되었지만 배포가 예비 복제본으로 장애 조치될 경우 지연 시간이 변경될 수 있습니다. 프로덕션 워크로드의 경우 빠르고 일관된 성능을

제공할 수 있도록 프로비저닝된 IOPS(초당 입/출력 작업)를 사용하는 것이 좋습니다. DB 인스턴스 클래스에 대한 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요.

DB 인스턴스를 Amazon RDS용 다중 AZ 배포로 변환

DB 인스턴스를 다중 AZ 배포로 수정하면 다른 가용 영역에 대기 인스턴스를 추가하여 가용성이 향상됩니다. 이 프로세스에는 가동 중지 시간이 최소화되며 스토리지 및 성능 영향에 대한 신중한 계획이 필요합니다. 이 변경 사항은 내결함성을 향상시키고 장애가 발생할 경우 복구 시간을 단축하므로 고가용성 환경에 이상적입니다.

단일 AZ 배포에 DB 인스턴스가 있고 이를 다중 AZ DB 인스턴스 배포로 수정하는 경우 Amazon RDS는 다음과 같은 작업을 수행합니다.

1. 기본 DB 인스턴스의 Amazon Elastic Block Store(EBS) 볼륨의 스냅샷을 만듭니다.
2. 스냅샷에서 스탠바이 복제본용 새 볼륨을 생성합니다. 이러한 볼륨은 백그라운드에서 초기화되며 데이터가 완전히 초기화된 후에 최대 볼륨 성능이 달성됩니다.
3. 기본 복제본과 대기 복제본의 볼륨 간의 동기 블록 수준 복제를 켭니다.

> **Important**
>
> 단일 AZ에서 다중 AZ로 변환하는 동안 스냅샷에서 대기 DB 인스턴스를 생성하면 가동 중지가 방지되지만 특히 쓰기에 민감한 워크로드의 경우 성능에 영향을 미칠 수 있습니다. 동기식 복제는 I/O 지연 시간을 늘려 데이터베이스 성능에 영향을 미칠 수 있습니다. 프로덕션 DB 인스턴스를 다중 AZ DB 인스턴스로 변환하지 않는 것이 가장 좋습니다.
> 대신 읽기 전용 복제본을 생성하고, 백업을 활성화하고, 다중 AZ로 변환하고, 데이터를 볼륨으로 로드한 다음 프라이머리 DB 인스턴스로 승격합니다. 자세한 내용은 DB 인스턴스 읽기 전용 복제본 작업 단원을 참조하십시오.

다중 AZ DB 인스턴스 배포가 되도록 DB 인스턴스 수정하는 방법은 2가지가 있습니다.

주제

- RDS 콘솔을 사용하여 다중 AZ DB 인스턴스 배포로 변환
- 다중 AZ DB 인스턴스 배포가 되도록 DB 인스턴스 수정

RDS 콘솔을 사용하여 다중 AZ DB 인스턴스 배포로 변환

RDS 콘솔을 사용하여 DB 인스턴스를 다중 AZ DB 인스턴스 배포로 변환할 수 있습니다.

콘솔만 사용하여 변환을 완료할 수 있습니다. AWS CLI 또는 RDS API를 사용하려면 다중 AZ DB 인스턴스 배포가 되도록 DB 인스턴스 수정의 지침을 따르십시오.

RDS 콘솔을 사용하여 다중 AZ DB 인스턴스 배포로 변환하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 다음 변경하려는 DB 인스턴스를 선택합니다.
3. Actions(작업)에서 Convert to Multi-AZ deployment(다중 AZ 배포로 변환)를 선택합니다.
4. 확인 페이지에서 Apply immediately(즉시 적용)을 선택하여 변경 사항을 즉시 적용합니다. 이 옵션을 선택하면 다운타임이 발생하지 않지만 성능이 영향을 받을 수 있습니다. 다음 유지 관리 기간에 업데이트를 적용하도록 선택할 수도 있습니다. 자세한 내용은 수정 예약 설정 사용 단원을 참조하십시오.
5. Convert to Multi-AZ(다중 AZ로 변환)를 선택합니다.

다중 AZ DB 인스턴스 배포가 되도록 DB 인스턴스 수정

다음과 같은 방법으로 다중 AZ DB 인스턴스 배포가 되도록 DB 인스턴스를 수정할 수 있습니다.

- RDS 콘솔을 사용하여 DB 인스턴스를 수정하고 Multi-AZ deployment(다중 AZ 배포)를 Yes(예)로 설정합니다.
- AWS CLI를 사용하여 modify-db-instance 명령을 호출하고 `--multi-az` 옵션을 설정합니다.
- RDS API를 사용하여 ModifyDBInstance 작업을 호출하고 `MultiAZ` 파라미터를 `true`로 설정합니다.

DB 인스턴스 수정에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하세요. 수정이 완료되면 Amazon RDS는 과정 완료를 표시하는 이벤트(RDS-EVENT-0025)를 트리거합니다. Amazon RDS 이벤트를 모니터링할 수 있습니다. 이벤트에 대한 자세한 내용은 Amazon RDS 이벤트 알림 작업 단원을 참조하세요.

Amazon RDS에 대한 다중 AZ DB 인스턴스 장애 조치

계획되거나 계획되지 않은 다중 AZ DB 인스턴스의 운영 중단으로 인해 인프라 장애가 발생한 경우, Amazon RDS는 자동으로 다른 가용 영역의 대기 복제본으로 전환됩니다.

장애 조치가 완료되는 데 소요되는 시간은 프라이머리 DB 인스턴스를 사용할 수 없게 된 시점의 데이터베이스 활동 및 기타 조건에 따라 달라집니다. 장애 조치에 소요되는 시간은 일반적으로 60–120초입니다. 그러나 트랜잭션의 규모가 크거나 복구 프로세스가 복잡한 경우 장애 조치에 소요되는 시간이 증가할 수 있습니다. 장애 조치가 완료되면 RDS 콘솔에 새 가용 영역이 반영되는 데 시간이 더 걸릴 수 있습니다.

> **Note**
>
> 다중 AZ DB 인스턴스를 재부팅할 때 장애 조치를 수동으로 강제할 수 있습니다. 자세한 내용은 DB 인스턴스 재부팅 단원을 참조하십시오.

Amazon RDS는 자동으로 장애 조치를 취하여 관리자의 개입 없이 데이터베이스 작업을 신속하게 재개할 수 있도록 합니다. 다음 표에 설명된 조건 중 하나가 발생하면 기본 DB 인스턴스는 자동으로 예비 복제본으로 전환됩니다. 이 장애 조치 이유는 이벤트 로그에서 확인할 수 있습니다.

장애 조치 이유	설명
RDS 데이터베이스 인스턴스 기반의 운영 체제가 오프라인 작업에서 패치되고 있습니다.	OS 패치 또는 보안 업데이트를 위한 유지 관리 기간 동안 장애 조치가 트리거되었습니다. 자세한 내용은 DB 인스턴스 유지 관리 섹션을 참조하세요.
RDS 다중 AZ 인스턴스의 기본 호스트가 비정상입니다.	다중 AZ DB 인스턴스 배포에서 손상된 프라이머리 DB 인스턴스를 감지하여 장애 조치를 수행했습니다.
네트워크 연결 손실로 인해 RDS 다중 AZ 인스턴스의 기본 호스트에 연결할 수 없습니다.	RDS 모니터링이 기본 DB 인스턴스에 대한 네트워크 연결 실패를 감지하여 장애 조치를 트리거했습니다.
RDS 인스턴스를 고객이 수정했습니다.	RDS DB 인스턴스 수정 때문에 장애 조치가 트리거되었습니다.

장애 조치 이유	설명
	자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.
RDS 다중 AZ 기본 인스턴스가 사용 중이며 응답하지 않습니다.	기본 DB 인스턴스가 응답하지 않습니다. 다음을 수행하는 것이 좋습니다. • 이벤트 및 CloudWatch 로그에서 과도한 CPU, 메모리 또는 스왑 공간 사용량을 검사합니다. 자세한 내용은 Amazon RDS 이벤트 알림 작업 및 Amazon RDS 이벤트에서 트리거되는 규칙 생성 단원을 참조하십시오. • 워크로드를 평가하여 적절한 DB 인스턴스 클래스를 사용 중인지 확인합니다. 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요. • 실시간 운영 체제 지표에 대해 향상된 모니터링을 사용합니다. 자세한 내용은 Enhanced Monitoring을 사용하여 OS 지표 모니터링 섹션을 참조하세요. • 성능 개선 도우미를 사용하면 DB 인스턴스의 성능에 영향을 미치는 문제를 분석할 수 있습니다. 자세한 내용은 성능 개선 도우미를 통한 Amazon RDS 모니터링 섹션을 참조하세요. 이러한 권장 사항에 대한 자세한 내용은 Amazon RDS의 모니터링 도구 및 Amazon RDS의 모범 사례 단원을 참조하세요.
RDS 다중 AZ 인스턴스의 기본 호스트 기반의 스토리지 볼륨에 오류가 발생했습니다.	다중 AZ DB 인스턴스 배포가 프라이머리 DB 인스턴스에서 스토리지 문제를 감지하여 장애 조치를 수행했습니다.

장애 조치 이유	설명
사용자가 DB 인스턴스의 장애 조치를 요청했습니다.	사용자가 DB 인스턴스를 재부팅하고 장애 조치를 사용하여 재부팅을 선택했습니다. 자세한 내용은 DB 인스턴스 재부팅을 참조하세요.

다음 단계에 따라 다중 AZ DB 인스턴스가 장애 조치를 수행했는지 확인할 수 있습니다.

- 장애 조치가 시작되었음을 이메일 또는 SMS로 사용자에게 알리도록 DB 이벤트 구독을 설정합니다. 이벤트에 대한 자세한 내용은 Amazon RDS 이벤트 알림 작업 단원을 참조하세요.
- RDS 콘솔 또는 API 작업을 사용하여 DB 이벤트를 확인합니다.
- RDS 콘솔 또는 API 작업을 사용하여 다중 AZ DB 인스턴스 배포의 현재 상태를 확인합니다.

장애 조치, 복구 시간 절감 및 기타 Amazon RDS 모범 사례에 대한 자세한 내용은 Amazon RDS의 모범 사례 단원을 참조하세요.

DNS 이름 조회를 위한 JVM TTL 설정

장애 조치 메커니즘은 DB 인스턴스의 Domain Name System(DNS) 레코드가 예비 DB 인스턴스를 가리키도록 자동으로 변경합니다. 그 결과 DB 인스턴스의 기존 연결을 모두 재설정해야 합니다. Java 가상 머신(JVM) 환경에서 Java DNS 캐싱 메커니즘이 작동하는 방식 때문에 JVM 설정을 다시 구성해야 할 수 있습니다.

JVM은 DNS 이름 조회를 캐시합니다. JVM은 호스트 이름을 IP 주소로 확인하는 경우 유지 시간(TTL)이라고 하는 지정된 기간 동안 IP 주소를 캐싱합니다.

AWS 리소스는 간헐적으로 변경되는 DNS 이름 항목을 사용하므로 TTL 값을 60초 이하로 하여 JVM을 구성하는 것이 좋습니다. 이렇게 하면 리소스의 IP 주소가 변경될 때 애플리케이션이 DNS를 다시 쿼리하여 리소스의 새 IP 주소를 수신하고 사용할 수 있습니다.

일부 Java 구성에서는 JVM이 재시작될 때까지 DNS 항목을 새로 고치지 않도록 JVM 기본 TTL이 설정됩니다. 따라서 애플리케이션이 여전히 실행되는 동안 AWS 리소스의 IP 주소가 변경되는 경우 JVM을 수동으로 재시작하여 캐시된 IP 정보가 새로 고쳐질 때까지는 해당 리소스를 사용할 수 없습니다. 이 경우 캐시된 IP 정보를 정기적으로 새로 고치도록 JVM의 TTL을 설정하는 것이 중요합니다.

`networkaddress.cache.ttl` 속성 값을 검색하여 JVM 기본 TTL을 가져올 수 있습니다.

```
String ttl = java.security.Security.getProperty("networkaddress.cache.ttl");
```

> **Note**
>
> 기본 TTL은 JVM 버전과 보안 관리자 설치 여부에 따라 다를 수 있습니다. 대부분의 JVM은 60초 미만의 기본 TTL을 제공합니다. 이러한 JVM을 사용 중이며 보안 관리자는 사용하지 않는 경우 이 주제의 나머지 내용을 무시해도 좋습니다. Oracle의 보안 관리자에 대한 자세한 내용은 Oracle 설명서의 The Security Manager를 참조하십시오.

JVM의 TTL을 수정하려면 `networkaddress.cache.ttl` 속성 값을 설정합니다. 필요에 따라 다음 방법 중 하나를 사용합니다.

- JVM을 사용하는 모든 애플리케이션에 대해 전역적으로 속성 값을 설정하려면 `networkaddress.cache.ttl` 파일에서 `$JAVA_HOME/jre/lib/security/java.security`을 설정합니다.

  ```
  networkaddress.cache.ttl=60
  ```

- 애플리케이션에만 로컬로 속성을 설정하려면 네트워크 연결이 설정되기 전에 애플리케이션의 초기화 코드에서 `networkaddress.cache.ttl`을 설정합니다.

  ```
  java.security.Security.setProperty("networkaddress.cache.ttl" , "60");
  ```

Amazon RDS용 다중 AZ DB 클러스터 배포

다중 AZ DB 클러스터 배포는 읽을 수 있는 복제 DB 인스턴스가 2개 있는 Amazon RDS의 반동기식 고가용성 배포 모드입니다. 다중 AZ DB 클러스터에는 동일한 AWS 리전에 있는 세 개의 개별 가용 영역에 라이터 DB 인스턴스와 두 개의 리더 DB 인스턴스가 있습니다. 다중 AZ DB 클러스터는 다중 AZ DB 인스턴스 배포에 비해 고가용성, 높은 읽기 워크로드 용량, 낮은 쓰기 대기 시간을 지원합니다.

가동 중지 시간을 줄이면서 Amazon RDS for MySQL 데이터베이스로 데이터 가져오기의 지침에 따라 온프레미스 데이터베이스의 데이터를 다중 AZ DB 클러스터로 가져올 수 있습니다.

다중 AZ DB 클러스터에 대한 예약 DB 인스턴스를 구매할 수 있습니다. 자세한 내용은 다중 AZ DB 클러스터에 대한 예약 DB 인스턴스 단원을 참조하십시오.

기능 가용성 및 해당 지원은 각 데이터베이스 엔진의 특정 버전 및 AWS 리전 리전에 따라 다릅니다. 다중 AZ DB 클러스터가 포함된 Amazon RDS의 버전 및 리전 가용성에 대한 자세한 내용은 Amazon RDS에서 다중 AZ DB 클러스터를 지원하는 리전 및 DB 엔진 단원을 참조하세요.

주제

- 다중 AZ DB 클러스터에서 사용 가능한 인스턴스 클래스
- 다중 AZ DB 클러스터 아키텍처
- 다중 AZ DB 클러스터용 파라미터 그룹
- RDS 프록시를 다중 AZ DB 클러스터와 함께 사용
- 복제본 지연 시간 및 다중 AZ DB 클러스터
- 다중 AZ DB 클러스터 스냅샷
- Amazon RDS용 다중 AZ DB 클러스터 생성
- Amazon RDS에서 다중 AZ DB 클러스터에 연결
- AWS 컴퓨팅 리소스와 다중 AZ DB 클러스터 자동 연결
- Amazon RDS용 다중 AZ DB 클러스터 수정
- Amazon RDS에서 다중 AZ DB 클러스터의 엔진 버전 업그레이드
- Amazon RDS용 다중 AZ DB 클러스터 이름 변경
- Amazon RDS에서 다중 AZ DB 클러스터 및 리더 DB 인스턴스 재부팅
- Amazon RDS용 다중 AZ DB 클러스터 장애 조치
- Amazon RDS의 다중 AZ DB 클러스터에 PostgreSQL 논리적 복제 설정
- Amazon RDS의 다중 AZ DB 클러스터 읽기 전용 복제본 사용
- Amazon RDS의 다중 AZ DB 클러스터에서 외부 복제 설정

- Amazon RDS용 다중 AZ DB 클러스터 삭제
- Amazon RDS 다중 AZ DB 클러스터의 제한

> **Important**
>
> 다중 AZ DB 클러스터는 Aurora DB 클러스터와 동일하지 않습니다. Aurora DB 클러스터에 대한 자세한 내용은 Amazon Aurora 사용 설명서를 참조하세요.

다중 AZ DB 클러스터에서 사용 가능한 인스턴스 클래스

다중 AZ DB 클러스터 배포는 `db.m5d`, `db.m6gd`, `db.m6id`, `db.m6idn`, `db.r5d`, `db.r6gd`, `db.x2iedn`, `db.r6id`, `db.r6idn` 및 `db.c6gd` DB 인스턴스 클래스에 지원됩니다.

> **Note**
>
> c6gd 인스턴스 클래스는 `medium` 인스턴스 크기를 지원하는 유일한 인스턴스 클래스입니다.

DB 인스턴스 클래스에 대한 자세한 내용은 the section called “DB 인스턴스 클래스” 섹션을 참조하세요.

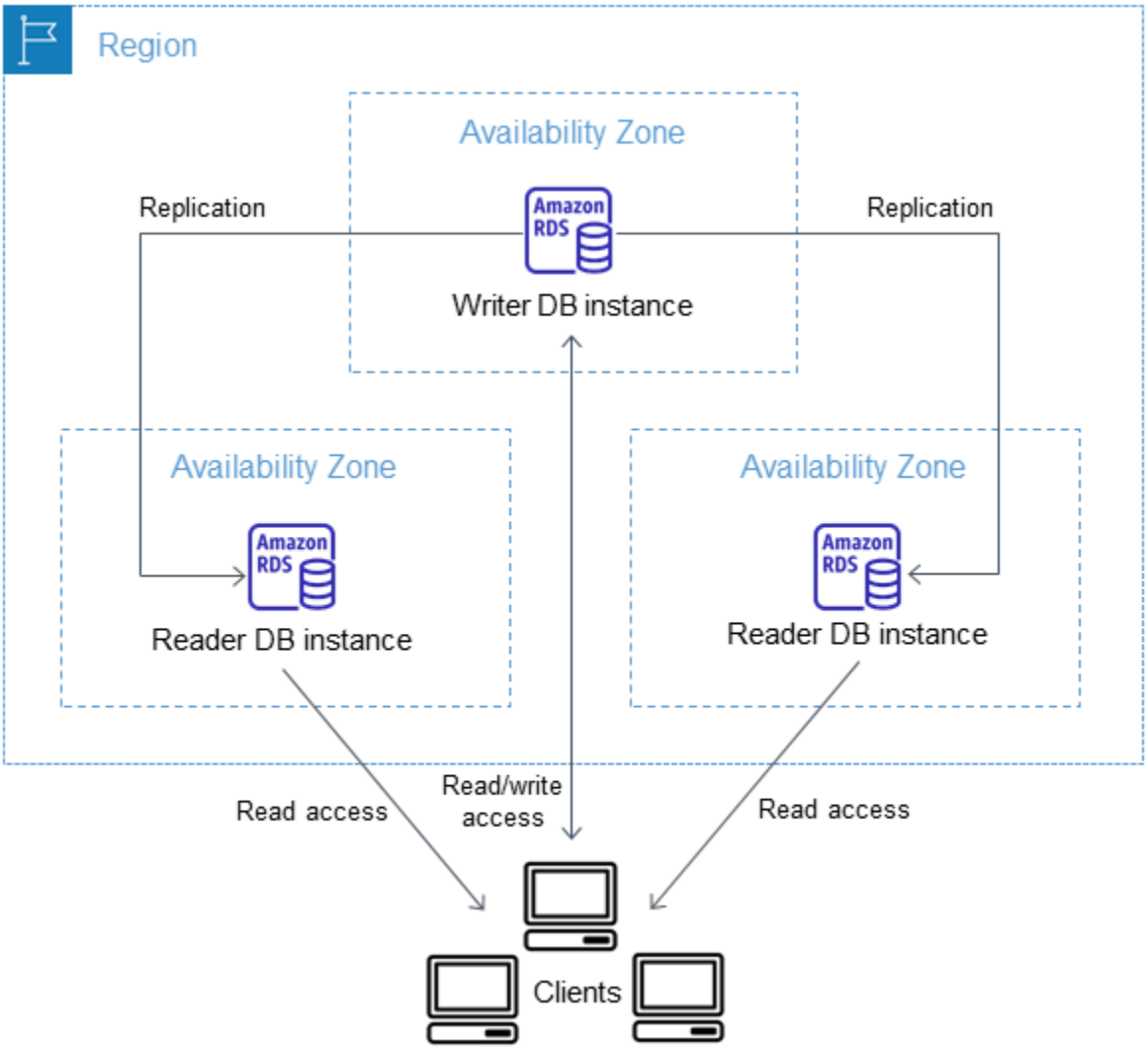
다중 AZ DB 클러스터 아키텍처

다중 AZ DB 클러스터에서는 Amazon RDS가 DB 엔진의 네이티브 복제 기능을 사용하여 라이터 DB 인스턴스의 데이터를 두 리더 DB 인스턴스로 복제합니다. 라이터 DB 인스턴스가 변경되면 각 리더 DB 인스턴스로 전송됩니다.

다중 AZ DB 클러스터 배포에서는 변경 사항을 커밋하려면 하나 이상의 리더 DB 인스턴스의 승인을 받아야 하는 반동기식 복제를 사용합니다. 모든 복제본에서 이벤트가 완전히 실행되고 커밋되었음을 승인하는 작업은 필요하지 않습니다.

리더 DB 인스턴스는 자동 장애 조치 대상 역할을 하며 읽기 트래픽을 제공하여 애플리케이션 읽기 처리량을 높입니다. 라이터 DB 인스턴스에서 중단이 발생하는 경우 RDS는 어느 리더 DB 인스턴스가 장애 조치 대상이 되는지를 관리합니다. RDS는 어느 리더 DB 인스턴스의 가장 최근 변경 기록을 기준으로 이를 수행합니다.

다음 다이어그램은 다중 AZ DB 클러스터를 보여줍니다.



일반적으로 다중 AZ DB 클러스터는 다중 AZ DB 인스턴스 배포에 비해 쓰기 대기 시간이 짧습니다. 또한 읽기 전용 워크로드가 리더 DB 인스턴스에서 실행되도록 허용합니다. RDS 콘솔에는 라이터 DB 인스턴스의 가용 영역과 리더 DB 인스턴스의 가용 영역이 표시됩니다. describe-db-clusters CLI 명령이나 DescribeDBClusters API 작업을 사용하여 이 정보를 찾아볼 수도 있습니다.

> **Important**
>
> RDS for MySQL 다중 AZ DB 클러스터의 복제 오류를 방지하려면 모든 테이블에 프라이머리 키가 있는 것이 좋습니다.

다중 AZ DB 클러스터용 파라미터 그룹

다중 AZ DB 클러스터에서 DB 클러스터 파라미터 그룹은 다중 AZ DB 클러스터의 모든 DB 인스턴스에 적용되는 엔진 구성 값에 대한 컨테이너 역할을 합니다.

다중 AZ DB 클러스터에서 DB 파라미터 그룹은 DB 엔진 및 DB 엔진 버전에 대한 기본 DB 파라미터 그룹으로 설정됩니다. DB 클러스터 파라미터 그룹의 설정은 클러스터의 모든 DB 인스턴스에 사용됩니다.

파라미터 그룹에 대한 자세한 내용은 the section called “DB 클러스터 파라미터 그룹” 단원을 참조하세요.

RDS 프록시를 다중 AZ DB 클러스터와 함께 사용

Amazon RDS 프록시를 사용하여 다중 AZ DB 클러스터의 프록시를 만들 수 있습니다. RDS 프록시를 사용하면 애플리케이션이 데이터베이스 연결을 풀링하고 공유할 수 있어 확장 기능을 향상할 수 있습니다. 각 프록시는 연결 재사용이라고도 하는 연결 멀티플렉싱을 수행합니다. 멀티플렉싱을 사용하면 RDS 프록시가 하나의 기본 데이터베이스 연결을 사용하여 한 트랜잭션에 대한 모든 작업을 수행합니다. 또한, RDS 프록시는 다중 AZ DB 클러스터의 마이너 버전 업그레이드로 인한 가동 중지 시간을 1초 이하로 줄일 수 있습니다. RDS 프록시의 이점에 대한 자세한 내용은 Amazon RDS Proxy 섹션을 참조하세요.

다중 AZ DB 클러스터에 프록시를 설정하려면 클러스터를 생성할 때 RDS 프록시 생성을 선택합니다. RDS 프록시 엔드포인트를 생성하고 관리하는 방법에 대한 지침은 the section called “RDS 프록시 엔드포인트 작업” 섹션을 참조하세요.

복제본 지연 시간 및 다중 AZ DB 클러스터

복제본 지연 시간은 리더 DB 인스턴스에서 가장 최근에 적용된 트랜잭션과 라이터 DB 인스턴스에서 가장 최근 트랜잭션 사이의 시간 차이입니다. Amazon CloudWatch 지표 `ReplicaLag`는 이 시차를 나타냅니다. CloudWatch 지표에 대한 자세한 내용은 Amazon CloudWatch로 Amazon RDS 지표 모니터링 섹션을 참조하세요.

다중 AZ DB 클러스터는 높은 쓰기 성능을 허용하지만 엔진 기반 복제의 특성으로 인해 복제본 지연이 계속 발생할 수 있습니다. 모든 장애 조치는 새 라이터 DB 인스턴스를 승격하기 전에 먼저 복제본 지연을 해결해야 하므로 이 복제본 지연을 모니터링하고 관리하는 것이 고려 사항입니다.

RDS for MySQL 다중 AZ DB 클러스터용 장애 조치 시간은 나머지 리더 DB 인스턴스의 복제본 지연에 따라 달라집니다. 두 리더 DB 인스턴스는 모두 적용되지 않은 트랜잭션을 적용해야 그 중 하나가 새 라이터 DB 인스턴스로 승격됩니다.

RDS for PostgreSQL 다중 AZ DB 클러스터용 장애 조치 시간은 가장 낮은 나머지 리더 DB 인스턴스의 복제본 지연에 따라 달라집니다. 가장 낮은 복제본 지연을 가진 리더 DB 인스턴스는 모두 적용되지 않은 트랜잭션을 적용해야 새 라이터 DB 인스턴스로 승격됩니다.

복제본 지연이 설정된 시간을 초과할 때 CloudWatch 경보를 생성하는 방법을 보여주는 자습서는 자습서: Amazon RDS에 대한 다중 AZ DB 클러스터 복제본 지연에 대한 Amazon CloudWatch 경보 생성 섹션을 참조하세요.

복제본 지연의 일반적인 원인

일반적으로 복제본 지연은 쓰기 워크로드가 너무 높아 리더 DB 인스턴스가 트랜잭션을 효율적으로 적용할 수 없을 때 발생합니다. 다양한 워크로드로 인해 일시적 또는 지속적인 복제본 지연이 발생할 수 있습니다. 다음은 몇 가지 일반적인 원인의 예입니다.

- 라이터 DB 인스턴스의 높은 쓰기 동시성 또는 대량 일괄 업데이트로 인해 리더 DB 인스턴스의 적용 프로세스가 뒤쳐집니다.
- 하나 이상의 리더 DB 인스턴스의 리소스를 사용하는 대량 읽기 워크로드입니다. 느리거나 큰 쿼리를 실행하면 적용 프로세스에 영향이 있고 복제본 지연이 발생할 수 있습니다.
- 데이터베이스가 커밋 순서를 유지해야 하기 때문에 대량의 데이터 또는 DDL 문을 수정하는 트랜잭션으로 인해 복제본 지연 시간이 일시적으로 증가하는 경우가 있습니다.

복제본 지연 완화

RDS for MySQL 및 RDS for PostgreSQL용 다중 AZ DB 클러스터의 경우 라이터 DB 인스턴스의 로드를 줄여 복제본 지연을 완화할 수 있습니다. 흐름 제어를 사용하여 복제본 지연을 줄일 수도 있습니다. 흐름 제어는 라이터 DB 인스턴스에서 쓰기를 제한하여 작동하므로 복제본 지연 시간이 계속해서 무제한으로 증가하지 않도록 합니다. 쓰기 제한은 트랜잭션 끝에 지연을 추가하여 수행되며, 이로 인해 라이터 DB 인스턴스의 쓰기 처리량이 감소합니다. 흐름 제어가 지연 제거를 보장하지는 않지만 많은 워크로드에서 전반적인 지연을 줄이는 데 도움이 될 수 있습니다. 다음 섹션에서는 RDS for MySQL 및 RDS PostgreSQL로 흐름 제어를 사용하는 방법을 제공합니다.

RDS for MySQL에 대한 흐름 제어를 통해 복제본 지연 완화

RDS for MySQL 다중 AZ DB 클러스터를 사용하는 경우 동적 파라미터 `rpl_semi_sync_master_target_apply_lag`을 사용하면 흐름 제어가 기본적으로 활성화됩니다. 이 파라미터는 복제본 지연에 대해 원하는 상한을 지정합니다. 복제본 지연이 구성된 제한에 가까워지면 흐름 제어는 라이터 DB 인스턴스의 쓰기 트랜잭션을 제한하여 지정된 값보다 낮은 복제본 지연을 포함하려고 시도합니다. 경우에 따라 복제본 지연이 지정된 제한을 초과할 수 있습니다. 기본적으로 이

파라미터는 120초로 설정됩니다. 흐름 제어를 비활성화하려면 이 파라미터를 최대 값 86,400초(하루)로 설정합니다.

흐름 제어에 의해 주입된 현재 지연을 보려면 다음 쿼리를 실행하여 `Rpl_semi_sync_master_flow_control_current_delay` 파라미터를 표시하세요.

```
SHOW GLOBAL STATUS like '%flow_control%';
```

출력은 다음과 비슷한 형태가 됩니다.

```
+-------------------------------------------------+-------+
| Variable_name                                   | Value |
+-------------------------------------------------+-------+
| Rpl_semi_sync_master_flow_control_current_delay | 2010  |
+-------------------------------------------------+-------+
1 row in set (0.00 sec)
```

> **Note**
>
> 지연은 마이크로초 단위로 표시됩니다.

RDS for MySQL Multi-AZ DB 클러스터에서 성능 개선 도우미를 활성화하면 흐름 제어에 의해 쿼리가 지연되었음을 나타내는 해당 SQL 문 대기 이벤트를 모니터링할 수 있습니다. 흐름 제어에 의해 지연이 발생했을 때 성능 개선 도우미 대시보드의 해당 SQL 문에서 `/wait/synch/cond/semisync/semi_sync_flow_control_delay_cond` 대기 이벤트를 볼 수 있습니다. 이러한 지표를 보려면 성능 스키마가 켜져 있는지 확인합니다. 성능 개선 도우미에 대한 자세한 내용은 성능 개선 도우미를 통한 Amazon RDS 모니터링 섹션을 참조하세요.

RDS for PostgreSQL에 대한 흐름 제어를 통해 복제본 지연 완화

RDS for PostgreSQL용 다중 AZ DB 클러스터를 사용하면 흐름 제어가 확장으로 배포됩니다. 이는 DB 클러스터의 모든 DB 인스턴스에 대해 백그라운드 작업자를 켭니다. 기본적으로 리더 DB 인스턴스의 백그라운드 작업자는 현재 복제본 지연을 라이터 DB 인스턴스의 백그라운드 작업자에게 알립니다. 리더 DB 인스턴스에서 지연이 2분을 초과하면 라이터 DB 인스턴스의 백그라운드 작업자가 트랜잭션 종료 시 지연을 추가합니다. 지연 임계값을 제어하려면 `flow_control.target_standby_apply_lag` 파라미터를 사용합니다.

흐름 제어가 PostgreSQL 프로세스를 제한하면 `pg_stat_activity` 및 성능 개선 도우미의 `Extension` 대기 이벤트가 이를 나타냅니다. 함수 `get_flow_control_stats`는 현재 추가되고 있는 지연 시간에 대한 세부 정보를 표시합니다.

흐름 제어는 짧지만 동시 트랜잭션이 많은 대부분의 온라인 트랜잭션 처리(OLTP) 워크로드에 이상적입니다. 배치 작업과 같은 장기 실행 트랜잭션으로 인해 지연이 발생하는 경우 흐름 제어는 큰 이점을 제공하지 않습니다.

`shared_preload_libraries`에서 확장을 제거하고 DB 인스턴스를 재부팅하여 흐름 제어를 해제할 수 있습니다.

다중 AZ DB 클러스터 스냅샷

Amazon RDS는 구성된 백업 기간 동안 다중 AZ DB 클러스터의 자동 백업을 생성하고 저장합니다. RDS는 개별 인스턴스가 아닌 전체 클러스터를 백업하여 DB 클러스터의 스토리지 볼륨 스냅샷을 생성합니다.

다중 AZ DB 클러스터의 수동 백업을 수행할 수도 있습니다. 매우 장기간 백업하려면 스냅샷 데이터를 Amazon S3로 내보내는 것을 고려합니다. 자세한 내용은 the section called “다중 AZ DB 클러스터의 스냅샷 생성” 단원을 참조하십시오.

다중 AZ DB 클러스터를 특정 시점으로 복원하여 새 다중 AZ DB 클러스터를 생성할 수 있습니다. 지침은 the section called “다중 AZ DB 클러스터를 특정 시점으로 복원” 단원을 참조하십시오.

또는 다중 AZ DB 클러스터 스냅샷을 단일 AZ 배포 또는 다중 AZ DB 인스턴스 배포에 복원할 수 있습니다. 지침은 the section called “다중 AZ DB 클러스터 스냅샷에서 DB 인스턴스로 복원” 단원을 참조하세요.

Amazon RDS용 다중 AZ DB 클러스터 생성

다중 AZ DB 클러스터에는 라이터 DB 인스턴스와 두 개의 리더 DB 인스턴스가 세 개의 개별 가용 영역에 있습니다. 다중 AZ DB 클러스터는 다중 AZ 배포에 비해 고가용성, 높은 읽기 워크로드 용량 및 짧은 대기 시간을 제공합니다. 다중 AZ DB 클러스터에 대한 자세한 내용은 Amazon RDS용 다중 AZ DB 클러스터 배포 섹션을 참조하세요.

> **Note**
>
> 다중 AZ DB 클러스터는 MySQL 및 PostgreSQL DB 엔진에서만 지원됩니다.

DB 클러스터 사전 조건

> **Important**
>
> 다중 AZ DB 클러스터를 생성할 수 있으려면 먼저 Amazon RDS 환경 설정 단원의 작업들을 완료해야 합니다.

다음은 다중 AZ DB 클러스터를 생성하기 전에 완료해야 하는 전제 조건입니다.

주제

- DB 클러스터의 네트워크 구성
- 추가 사전 조건

DB 클러스터의 네트워크 구성

Amazon VPC 서비스 기반의 Virtual Private Cloud(VPC)에서만 다중 AZ DB 클러스터를 생성할 수 있습니다. 가용 영역이 3개 이상 있는 AWS 리전에 있어야 합니다. DB 클러스터에 대해 선택한 DB 서브넷 그룹은 세 개 이상의 가용 영역을 포함해야 합니다. 이 구성은 DB 클러스터의 각 DB 인스턴스가 서로 다른 가용 영역에 있도록 합니다.

새 DB 클러스터와 동일한 VPC의 Amazon EC2 인스턴스 간에 연결을 설정하려는 경우 DB 클러스터 생성 시 설정합니다. 동일한 VPC의 EC2 인스턴스 이외의 리소스에서 DB 클러스터에 연결하려는 경우 네트워크 연결을 수동으로 구성합니다.

주제

- EC2 인스턴스와의 자동 네트워크 연결 구성
- 네트워크 수동 구성

EC2 인스턴스와의 자동 네트워크 연결 구성

다중 AZ DB 클러스터를 생성할 때 AWS Management Console을 사용하여 EC2 인스턴스와 새 DB 클러스터 간의 연결을 설정할 수 있습니다. 이렇게 하면 RDS가 VPC 및 네트워크 설정을 자동으로 구성합니다. DB 클러스터는 EC2 인스턴스가 DB 클러스터에 액세스할 수 있도록 EC2 인스턴스와 동일한 VPC에 생성됩니다.

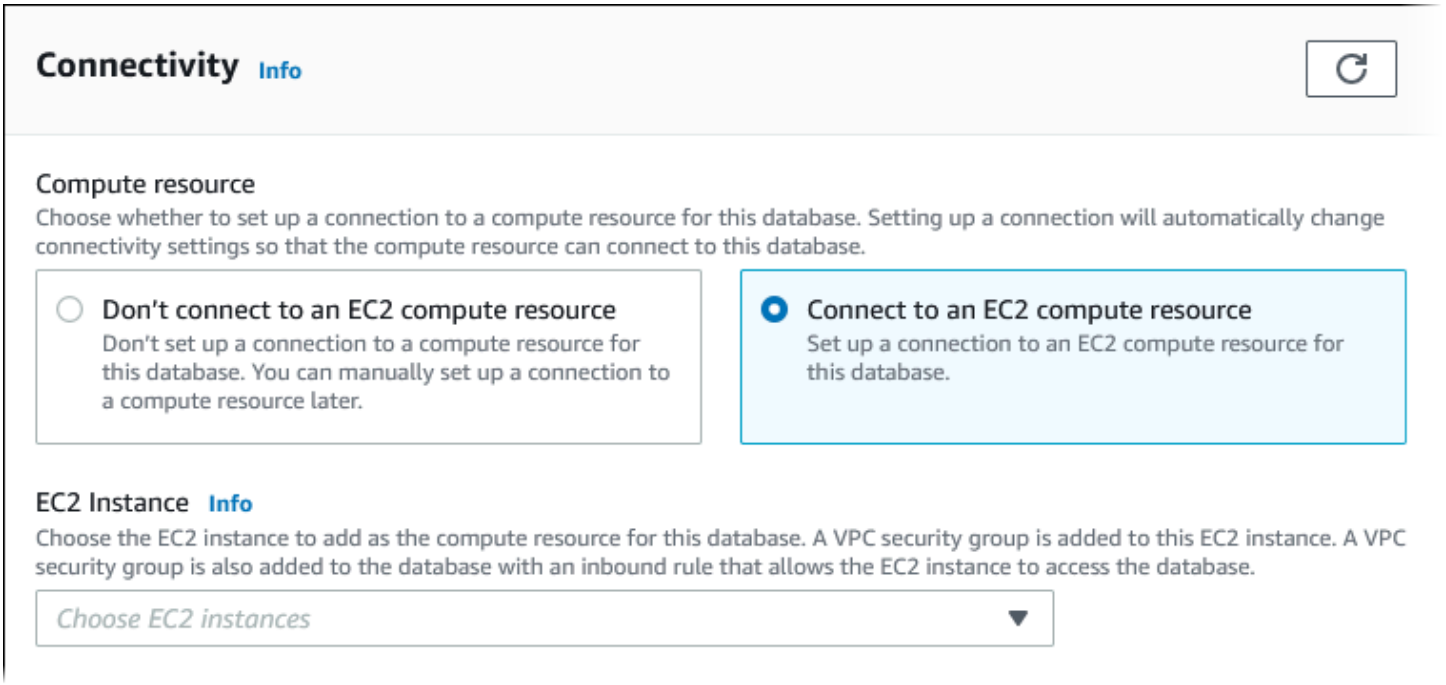
다음은 EC2 인스턴스를 DB 클러스터와 연결하기 위한 요구 사항입니다.

- DB 클러스터를 생성하려면 먼저 EC2 인스턴스가 AWS 리전에 있어야 합니다.

  AWS 리전에 EC2 인스턴스가 없는 경우 콘솔은 인스턴스를 생성할 수 있는 링크를 제공합니다.
- DB 인스턴스를 만드는 사용자는 다음 작업을 수행할 수 있는 권한이 있어야 합니다.
  - `ec2:AssociateRouteTable`
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateRouteTable`
  - `ec2:CreateSubnet`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeRouteTables`
  - `ec2:DescribeSecurityGroups`
  - `ec2:DescribeSubnets`
  - `ec2:ModifyNetworkInterfaceAttribute`
  - `ec2:RevokeSecurityGroupEgress`

이 옵션을 사용하면 프라이빗 DB 클러스터가 생성됩니다. DB 클러스터는 프라이빗 서브넷만 있는 DB 서브넷 그룹을 사용하여 VPC 내 리소스에 대한 액세스를 제한합니다.

EC2 인스턴스를 DB 클러스터에 연결하려면 Create database(데이터베이스 생성) 페이지의 Connectivity(연결) 섹션에서 Connect to an EC2 compute resource(EC2 컴퓨팅 리소스에 연결)을 선택합니다.



Connect to an EC2 compute resource(EC2 컴퓨팅 리소스에 연결)를 선택하면 RDS가 다음 옵션을 자동으로 설정합니다. Don't connect to an EC2 compute resource(EC2 컴퓨팅 리소스에 연결하지 않음)를 선택하여 EC2 인스턴스와의 연결을 설정하지 않도록 선택하지 않는 한 이러한 설정을 변경할 수 없습니다.

콘솔 옵션	자동 설정
Virtual Private Cloud(VPC)	RDS는 VPC를 EC2 인스턴스와 연결된 VPC로 설정합니다.
DB 서브넷 그룹	RDS에는 EC2 인스턴스와 동일한 가용 영역에 프라이빗 서브넷이 있는 DB 서브넷 그룹이 필요합니다. 이 요구 사항을 충족하는 DB 서브넷 그룹이 있다면 RDS는 기존 DB 서브넷 그룹을 사용합니다. 기본적으로 이 옵션은 Automatic setup(자동 설정)으로 설정되어 있습니다. Automatic setup(자동 설정)을 선택하고 이 요구 사항을 충족하는 DB 서브넷 그룹이 없으면 다음 작업이 수행됩니다. RDS는 가용 영역 중 하나가 EC2 인스턴스와 동일한 3개의 가용 영역에서 3개

콘솔 옵션	자동 설정
	의 사용 가능한 프라이빗 서브넷을 사용합니다. 가용 영역에서 프라이빗 서브넷을 사용할 수 없는 경우 RDS는 가용 영역에 프라이빗 서브넷을 생성합니다. 그런 다음 RDS는 DB 서브넷 그룹을 생성합니다. 프라이빗 서브넷을 사용할 수 있는 경우 RDS는 서브넷과 연결된 라우팅 테이블을 사용하고 생성된 모든 서브넷을 이 라우팅 테이블에 추가합니다. 프라이빗 서브넷을 사용할 수 없는 경우 RDS는 인터넷 게이트웨이 액세스가 없는 라우팅 테이블을 생성하고 생성한 서브넷을 라우팅 테이블에 추가합니다. RDS를 사용하면 기존 DB 서브넷 그룹도 사용할 수 있습니다. 선택한 기존 DB 서브넷 그룹을 사용하려면 Choose existing(기존 항목 선택)을 선택합니다.
공개 액세스(Public access)	RDS는 DB 클러스터에 공개적으로 액세스할 수 없도록 아니요를 선택합니다. 보안을 위해 데이터베이스를 비공개로 유지하고 인터넷에서 액세스할 수 없도록 하는 것이 가장 좋습니다.

콘솔 옵션	자동 설정
VPC 보안 그룹(방화벽)	RDS는 DB 클러스터와 연결된 새 보안 그룹을 생성합니다. 보안 그룹의 이름은 `rds-ec2-`*n*이며, 여기서 *n*은 숫자입니다. 이 보안 그룹에는 EC2 VPC 보안 그룹(방화벽)이 소스인 인바운드 규칙이 포함됩니다. DB 클러스터와 연결된 이 보안 그룹을 통해 EC2 인스턴스가 DB 클러스터에 액세스할 수 있습니다. RDS는 DB 인스턴스와 연결된 새 보안 그룹도 생성합니다. 보안 그룹의 이름은 `ec2-rds-`*n*이며, 여기서 *n*은 숫자입니다. 이 보안 그룹에는 DB 클러스터의 VPC 보안 그룹을 소스로 하는 아웃바운드 규칙이 포함됩니다. 이 보안 그룹을 사용하면 EC2 인스턴스가 DB 클러스터에 트래픽을 보내도록 허용할 수 있습니다. Create new(새로 생성)를 선택하고 새 보안 그룹의 이름을 입력하여 다른 새 보안 그룹을 추가할 수 있습니다. Choose existing(기존 항목 선택)을 선택하고 추가할 보안 그룹을 선택하여 기존 보안 그룹을 추가할 수 있습니다.
가용 영역	RDS는 다중 AZ DB 클러스터 배포에서 하나의 DB 인스턴스에 대해 EC2 인스턴스의 가용 영역을 선택합니다. RDS는 다른 두 DB 인스턴스에 대해 서로 다른 가용 영역을 임의로 선택합니다. 라이터 DB 인스턴스는 EC2 인스턴스와 동일한 가용 영역에 생성됩니다. 장애 조치가 발생하고 라이터 DB 인스턴스가 다른 가용 영역에 있는 경우 교차 가용 영역 비용이 발생할 가능성이 있습니다.

이러한 설정에 대한 자세한 내용은 다중 AZ DB 클러스터를 생성하기 위한 설정 단원을 참조하세요.

DB 클러스터를 생성한 후에 이러한 설정을 변경할 경우 변경 사항이 EC2 인스턴스와 DB 클러스터 간의 연결에 영향을 미칠 수 있습니다.

네트워크 수동 구성

동일한 VPC의 EC2 인스턴스 이외의 리소스에서 DB 클러스터에 연결하려는 경우 네트워크 연결을 수동으로 구성합니다. AWS Management Console을 사용하여 다중 AZ DB 클러스터를 생성하는 경우 Amazon RDS에서 자동으로 VPC를 생성하도록 할 수 있습니다. 또는 다중 AZ DB 클러스터에서 기존

VPC를 사용하거나 새 VPC를 생성할 수도 있습니다. 다중 AZ DB 클러스터에서 VPC를 사용하려면 세 개 이상의 가용 영역마다 VPC에 서브넷이 한 개 이상 있어야 합니다. VPC에 대한 자세한 내용은 다음(Amazon VPC 및 Amazon RDS)을 참조하세요.

기본 VPC를 사용하지 않거나 VPC를 생성하지 않았고 콘솔을 사용할 계획이 없다면 다음을 수행하면 됩니다.

- DB 클러스터를 배포하려는 AWS 리전에서 세 개 이상의 가용 영역에 각각 한 개 이상의 서브넷을 갖는 VPC를 생성합니다. 자세한 내용은 VPC에서 DB 인스턴스를 사용한 작업을 참조하세요.
- DB 클러스터에 대한 연결 권한을 부여할 수 있도록 VPC 보안 그룹을 지정합니다. 자세한 내용은 보안 그룹을 생성하여 VPC 내부의 DB 인스턴스에 대한 액세스를 제공 및 보안 그룹을 통한 액세스 제어 단원을 참조하세요.
- VPC에서 다중 AZ DB 클러스터에서 사용할 수 있는 서브넷을 세 개 이상 정의하는 RDS DB 서브넷 그룹을 지정합니다. 자세한 내용은 DB 서브넷 그룹을 사용한 작업을 참조하세요.

다중 AZ DB 클러스터에 적용되는 제한 사항에 대한 자세한 내용은 Amazon RDS 다중 AZ DB 클러스터의 제한 페이지를 참조하세요.

다중 AZ DB 클러스터와 동일한 VPC에 있지 않은 리소스에 연결하려는 경우 VPC에서 DB 인스턴스에 액세스하는 시나리오에서 적절한 시나리오를 참조하십시오.

추가 사전 조건

다중 AZ DB 클러스터를 만들려면 먼저 다음과 같은 추가 사전 조건을 고려하세요.

- DB 클러스터에 대한 구성 파라미터를 사용자 지정하려면 필요한 파라미터 설정으로 DB 클러스터 파라미터 그룹을 지정해야 합니다. DB 클러스터 파라미터 그룹 생성 또는 수정에 대한 자세한 내용은 다중 AZ DB 클러스터용 파라미터 그룹 섹션을 참조하세요.
- DB 클러스터에 지정할 TCP/IP 포트 번호를 정합니다. 일부 기업에서는 방화벽이 이러한 기본 포트 연결을 차단하는 경우도 있습니다. 이처럼 기업 방화벽이 기본 포트를 차단할 경우 DB 클러스터에 다른 포트를 선택해야 합니다. DB 클러스터의 DB 인스턴스는 모두 동일한 포트를 사용합니다.
- 데이터베이스의 메이저 엔진 버전이 RDS 표준 지원 종료일에 다다른 경우 추가 지원 CLI 옵션 또는 RDS API 파라미터를 사용해야 합니다. 자세한 내용은 다중 AZ DB 클러스터를 생성하기 위한 설정의 RDS 추가 지원을 참조하세요.

DB 클러스터 생성

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 다중 AZ DB 클러스터를 생성할 수 있습니다.

콘솔

가용성 및 지속성(Availability and durability) 섹션에서 다중 AZ DB 클러스터(Multi-AZ DB cluster)를 선택하여 다중 AZ DB 클러스터를 생성할 수 있습니다.

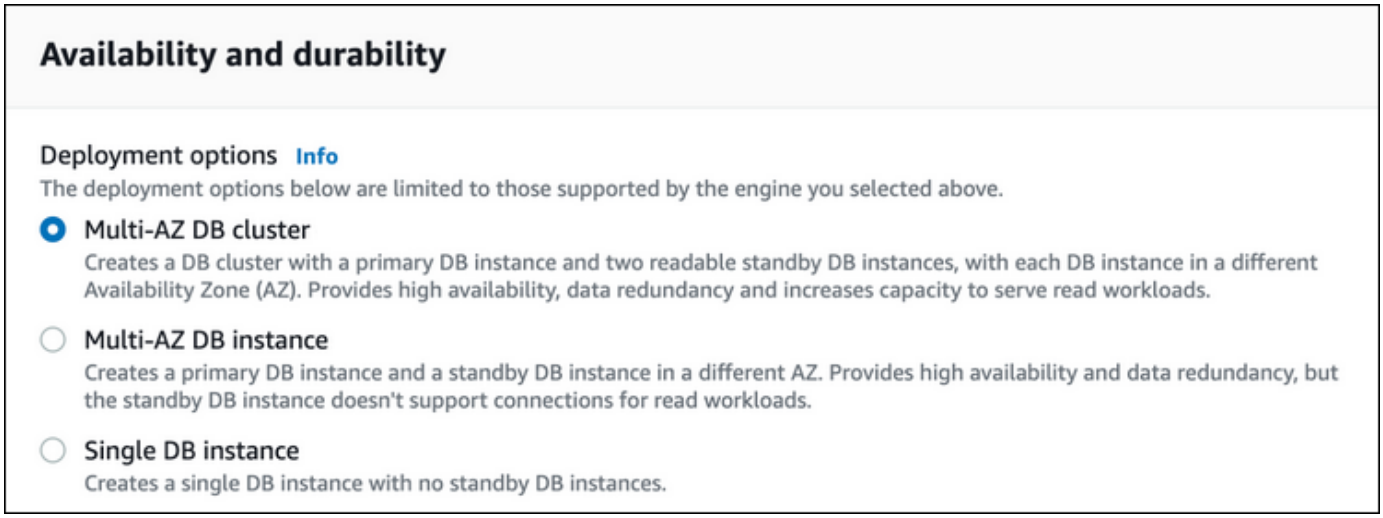
콘솔을 사용하여 다중 AZ DB 클러스터를 생성하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. AWS Management Console의 오른쪽 상단 모서리에서 DB 인스턴스를 생성하려는 AWS 리전을 선택합니다.

   다중 AZ DB 클러스터를 지원하는 AWS 리전에 대한 자세한 내용은 Amazon RDS 다중 AZ DB 클러스터의 제한 페이지를 참조하세요.
3. 탐색 창에서 데이터베이스를 선택합니다.
4. 데이터베이스 생성을 선택합니다.

   다중 AZ DB 클러스터를 생성하려면 표준 생성(Standard Create)이 선택되어 있고 간편 생성(Easy Create)이 선택되어 있지 않은지 확인하세요.
5. 엔진 유형(Engine type)에서 MySQL 또는 PostgreSQL을 선택합니다.
6. 버전(Version)으로 DB 엔진 버전을 선택합니다.

   다중 AZ DB 클러스터를 지원하는 에 대한 자세한 내용은 Amazon RDS 다중 AZ DB 클러스터의 제한 페이지를 참조하세요.
7. 템플릿(Templates)에서 배포에 사용할 적합한 템플릿을 선택합니다.
8. 가용성 및 지속성(Availability and durability)에서 다중 AZ DB 클러스터(Multi-AZ DB cluster)를 선택합니다.



9. DB 클러스터 식별자(DB cluster identifier)에 DB 클러스터의 식별자를 입력합니다.
10. 마스터 사용자 이름(Master username)에 마스터 사용자 이름을 입력하거나 기본 설정을 그대로 사용합니다.
11. 다음과 같이 기본 암호를 입력합니다.

    a. Settings(설정) 섹션에서 Credential Settings(보안 인증 정보 설정)를 엽니다.

    b. 암호를 지정하려는 경우 암호 자동 생성(Auto generate a password) 확인란이 선택되어 있으면 선택을 해제합니다.

    c. (선택 사항) 마스터 사용자 이름(Master username) 값을 변경합니다.

    d. 마스터 암호 및 암호 확인에 동일한 암호를 입력합니다.
12. DB 인스턴스 클래스에서 DB 인스턴스 클래스를 선택합니다. 지원되는 DB 인스턴스 클래스 목록은 the section called “다중 AZ DB 클러스터에서 사용 가능한 인스턴스 클래스” 섹션을 참조하세요.
13. (선택 사항) 이 DB 클러스터의 컴퓨팅 리소스에 대한 연결을 설정합니다.

    DB 클러스터를 생성하는 동안 Amazon EC2 인스턴스와 새 DB 클러스터 간의 연결을 구성할 수 있습니다. 자세한 내용은 EC2 인스턴스와의 자동 네트워크 연결 구성 섹션을 참조하세요.
14. VPC 보안 그룹(방화벽) 아래의 연결 섹션에서 새로 만들기를 선택하면 로컬 컴퓨터의 IP 주소가 데이터베이스에 액세스할 수 있도록 허용하는 인바운드 규칙과 함께 VPC 보안 그룹이 생성됩니다.
15. 나머지 섹션에서 DB 클러스터 설정을 지정합니다. 각 설정에 대한 자세한 내용은 다중 AZ DB 클러스터를 생성하기 위한 설정 단원을 참조하십시오.
16. 데이터베이스 생성을 선택합니다.

자동 생성된 암호를 사용하기로 한 경우에는 Databases(데이터베이스) 페이지에 View credential details(자격 증명 세부 정보 보기) 버튼이 나타납니다.

DB 클러스터의 마스터 사용자 이름 및 암호를 보려면 자격 증명 세부 정보 보기를 선택합니다.

DB 클러스터를 마스터 사용자로 연결하려면 화면에 나타난 사용자 이름과 암호를 사용합니다.

> **Important**
>
> 마스터 사용자 암호를 다시 볼 수는 없습니다.

17. 데이터베이스(Databases)의 경우 새 DB 클러스터의 이름을 선택합니다.

RDS 콘솔에 새 DB 클러스터의 세부 정보가 표시됩니다. DB 클러스터를 생성하고 사용할 준비가 될 때까지 DB 클러스터의 상태는 생성 중(Creating)입니다. 상태가 사용 가능(Available)으로 변경되면 DB 클러스터에 연결할 수 있습니다. 할당된 DB 클러스터 클래스 및 스토리지에 따라 새 DB 클러스터를 사용할 수 있기까지 몇 분 정도 걸릴 수 있습니다.

AWS CLI

AWS CLI를 사용하여 다중 AZ DB 클러스터를 생성하기 전에 필수 사전 조건을 이행해야 합니다. 여기에는 VPC 및 RDS DB 서브넷 그룹 생성이 포함됩니다. 자세한 내용은 DB 클러스터 사전 조건 섹션을 참조하세요.

AWS CLI를 사용하여 다중 AZ DB 클러스터를 생성하려면 create-db-cluster 명령을 호출하면 됩니다. `--db-cluster-identifier`를 지정합니다. `--engine` 옵션으로 `mysql` 또는 `postgres`를 지정합니다.

각 옵션에 대한 자세한 내용은 다중 AZ DB 클러스터를 생성하기 위한 설정 단원을 참조하세요.

다중 AZ DB 클러스터를 지원하는 AWS 리전, DB 엔진 및 DB 엔진 버전에 대한 자세한 내용은 Amazon RDS 다중 AZ DB 클러스터의 제한 페이지를 참조하세요.

`create-db-cluster` 명령은 DB 클러스터를 위한 라이터 DB 인스턴스와 두 개의 리더 DB 인스턴스를 생성합니다. 각 DB 인스턴스는 서로 다른 가용 영역에 있습니다.

예를 들어, 다음 명령을 사용하면 이름이 `mysql-multi-az-db-cluster`인 MySQL 8.0 다중 AZ DB 클러스터가 생성됩니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-cluster \
   --db-cluster-identifier mysql-multi-az-db-cluster \
   --engine mysql \
   --engine-version 8.0.32  \
   --master-username admin \
   --manage-master-user-password  \
   --port 3306 \
   --backup-retention-period 1  \
   --db-subnet-group-name default \
   --allocated-storage 4000 \
   --storage-type io1 \
   --iops 10000 \
   --db-cluster-instance-class db.m5d.xlarge
```

Windows의 경우:

```
aws rds create-db-cluster ^
   --db-cluster-identifier mysql-multi-az-db-cluster ^
   --engine mysql ^
   --engine-version 8.0.32 ^
   --manage-master-user-password ^
   --master-username admin ^
   --port 3306 ^
   --backup-retention-period 1 ^
   --db-subnet-group-name default ^
   --allocated-storage 4000 ^
   --storage-type io1 ^
   --iops 10000 ^
   --db-cluster-instance-class db.m5d.xlarge
```

다음 명령을 사용하면 이름이 `postgresql-multi-az-db-cluster`인 PostgreSQL 13.4 다중 AZ DB 클러스터가 생성됩니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-cluster \
```

```
    --db-cluster-identifier postgresql-multi-az-db-cluster \
    --engine postgres \
    --engine-version 13.4 \
    --manage-master-user-password \
    --master-username postgres \
    --port 5432 \
    --backup-retention-period 1  \
    --db-subnet-group-name default \
    --allocated-storage 4000 \
    --storage-type io1 \
    --iops 10000 \
    --db-cluster-instance-class db.m5d.xlarge
```

Windows의 경우:

```
aws rds create-db-cluster ^
    --db-cluster-identifier postgresql-multi-az-db-cluster ^
    --engine postgres ^
    --engine-version 13.4 ^
    --manage-master-user-password ^
    --master-username postgres ^
    --port 5432 ^
    --backup-retention-period 1 ^
    --db-subnet-group-name default ^
    --allocated-storage 4000 ^
    --storage-type io1 ^
    --iops 10000 ^
    --db-cluster-instance-class db.m5d.xlarge
```

RDS API

RDS API를 사용하여 다중 AZ DB 클러스터를 생성하려면 먼저 VPC 및 RDS DB 서브넷 그룹 생성과 같은 필수 사전 조건을 이행해야 합니다. 자세한 내용은 DB 클러스터 사전 조건을 참조하세요.

RDS API를 사용하여 다중 AZ DB 클러스터를 생성하려면 CreateDBCluster 작업을 호출합니다. `DBClusterIdentifier`를 지정합니다. `Engine` 파라미터에 대해 `mysql` 또는 `postgresql`을 지정합니다.

각 옵션에 대한 자세한 내용은 다중 AZ DB 클러스터를 생성하기 위한 설정 단원을 참조하세요.

`CreateDBCluster` 작업을 통해 DB 클러스터에 대한 라이터 DB 인스턴스와 리더 DB 인스턴스가 생성됩니다. 각 DB 인스턴스는 서로 다른 가용 영역에 있습니다.

다중 AZ DB 클러스터를 생성하기 위한 설정

다중 AZ DB 클러스터를 생성할 때 선택하는 설정에 대한 자세한 내용은 다음 표를 참조하세요. AWS CLI 옵션에 대한 자세한 정보는 create-db-cluster를 참조하세요. RDS API 파라미터에 대한 자세한 내용은 CreateDBCluster를 참조하세요.

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터

할당된 스토리지	DB 클러스터의 각 DB 인스턴스에 할당할 스토리지 양(기비바이트)입니다. 자세한 내용은 Amazon RDS DB 인스턴스 스토리지을 참조하세요.	CLI 옵션:
		`--allocated-storage`
		API 파라미터:
		`AllocatedStorage`

자동 마이너 버전 업그레이드	자동 마이너 버전 업그레이드(Enable auto minor version upgrade)를 선택하면 DB 클러스터가 지원되는 경우 기본 마이너 DB 엔진 버전 업그레이드를 자동으로 수신하도록 할 수 있습니다. Amazon RDS는 유지 관리 기간에 자동 마이너 버전 업그레이드를 수행합니다.	CLI 옵션:
		`--auto-minor-version-upgrade`
		`--no-auto-minor-version-upgrade`
		API 파라미터:
		`AutoMinorVersionUpgrade`

백업 보관 기간	DB 클러스터의 자동 백업을 보존할 기간(단위: 일)입니다. 다중 AZ DB 클러스터의 경우 이 값을 **1** 이상으로 설정해야 합니다.
	자세한 내용은 백업 소개을 참조하세요.
		CLI 옵션:
		`--backup-retention-period`
		API 파라미터:
		`BackupRetentionPeriod`

백업 기간	Amazon RDS가 DB 클러스터의 백업을 자동으로 수행하는 기간입니다. 데이터베이스를 백업할 특정 기간을 지정하지 않으려면 기본값으로	CLI 옵션: `--preferred-backup-window` API 파라미터:

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터
	기본 설정 없음(No preference)을 사용합니다. 자세한 내용은 백업 소개 섹션을 참조하세요.	`PreferredBackupWindow`

인증 기관	DB 클러스터에서 사용하는 서버 인증서의 CA(인증 기관)입니다.
	
	자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 섹션을 참조하세요.
		CLI 옵션:
		
		`--ca-certificate-identifier`
		
		RDS API 파라미터:
		
		`CACertificateIdentifier`

스냅샷으로 태그 복사	이 옵션은 스냅샷을 생성할 때 DB 클러스터 태그를 DB 스냅샷에 복사합니다.
	
	자세한 내용은 Amazon RDS 리소스에 태그 지정을 참조하세요.
		CLI 옵션:
		
		`-copy-tags-to-snapshot`
		
		`-no-copy-tags-to-snapshot`
		
		RDS API 파라미터:
		
		`CopyTagsToSnapshot`

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터

데이터베이스 인증	사용하려는 데이터베이스 인증 옵션입니다.
	
	데이터베이스 암호로만 데이터베이스 사용자를 인증할 수 있도록 Password authentication(암호 인증)을 선택합니다.
	
	사용자 및 역할을 통해 데이터베이스 암호 및 사용자 자격 증명으로 데이터베이스 사용자를 인증할 수 있도록 Password and IAM DB authentication(암호 및 IAM DB 인증)을 선택합니다. 자세한 내용은 MariaDB, MySQL 및 PostgreSQL IAM 데이터베이스 인증 섹션을 참조하세요.
		CLI 옵션:
		
		`--enable-iam-database-authentication`
		
		`--no-enable-iam-database-authentication`
		
		RDS API 파라미터:
		
		`EnableIAMDatabaseAuthentication`

데이터베이스 포트	DB 클러스터에 액세스할 포트입니다. 기본 포트가 표시됩니다.
	
	DB 클러스터를 생성한 후에는 포트를 변경할 수 없습니다.
	
	일부 기업에서는 방화벽이 이러한 기본 포트 연결을 차단하는 경우도 있습니다. 이처럼 기업 방화벽이 기본 포트를 차단할 경우 DB 클러스터에 다른 포트를 입력해야 합니다.
		CLI 옵션:
		
		`--port`
		
		RDS API 파라미터:
		
		`Port`

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터

DB 클러스터 식별자	DB 클러스터의 이름입니다. 온프레미스 서버와 동일한 방식으로 DB 클러스터의 이름을 지정합니다. DB 클러스터 식별자는 최대 63자의 영숫자를 포함할 수 있으며 선택한 AWS 리전의 계정에 대해 고유해야 합니다.	CLI 옵션:
		
		`--db-cluster-identifier`
		
		RDS API 파라미터:
		
		`DBClusterIdentifier`

DB 인스턴스 클래스	다중 AZ DB 클러스터에 있는 각 DB 인스턴스의 컴퓨팅 및 메모리 용량입니다(예: `db.m5d.xlarge`).
	
	가능하면 일반 쿼리 작업 세트가 메모리에 상주할 수 있을 정도로 큰 DB 인스턴스 클래스를 선택합니다. 작업 세트가 메모리에 상주할 경우 시스템의 디스크 쓰기가 불필요하여 성능이 향상됩니다.
	
	지원되는 DB 인스턴스 클래스 목록은 the section called “다중 AZ DB 클러스터에서 사용 가능한 인스턴스 클래스” 섹션을 참조하세요.
		CLI 옵션:
		
		`--db-cluster-instance-class`
		
		RDS API 파라미터:
		
		`DBClusterInstanceClass`

DB 클러스터 파라미터 그룹	DB 클러스터와 연결할 DB 클러스터 파라미터 그룹.
	
	자세한 내용은 다중 AZ DB 클러스터용 파라미터 그룹을 참조하세요.
		CLI 옵션:
		
		`--db-cluster-parameter-group-name`
		
		RDS API 파라미터:
		
		`DBClusterParameterGroupName`

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터

DB 엔진 버전	사용할 데이터베이스 엔진의 버전입니다.	CLI 옵션:
		
		`--engine-version`
		
		RDS API 파라미터:
		
		`EngineVersion`

DB 클러스터 파라미터 그룹	DB 클러스터와 연결할 DB 인스턴스 파라미터 그룹입니다. 자세한 내용은 다중 AZ DB 클러스터용 파라미터 그룹 섹션을 참조하세요.	CLI 옵션: `--db-cluster-parameter-group-name` RDS API 파라미터: **`DBClusterParameterGroupName`**

DB 서브넷 그룹	DB 클러스터에 사용할 DB 서브넷 그룹입니다.
	기존 DB 서브넷 그룹을 사용하려면 Choose existing(기존 항목 선택)을 선택합니다. 그런 다음 Existing DB subnet groups(기존 DB 서브넷 그룹) 드롭다운 목록에서 필요한 서브넷 그룹을 선택합니다.
	
	RDS가 호환되는 DB 서브넷 그룹을 선택하도록 하려면 Automatic setup(자동 설정)을 선택합니다. 해당 그룹이 없으면 RDS는 클러스터에 대한 새 서브넷 그룹을 생성합니다.
	
	자세한 내용은 DB 서브넷 그룹을 사용한 작업 섹션을 참조하세요.
		CLI 옵션:
		
		`--db-subnet-group-name`
		
		RDS API 파라미터:
		
		`DBSubnetGroupName`

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터

삭제 방지	DB 클러스터가 삭제되지 않도록 방지하기 위한 삭제 방지 활성화 콘솔에서 프로덕션 DB 클러스터를 생성하면 삭제 방지가 기본적으로 설정됩니다.
	
	자세한 내용은 DB 인스턴스 삭제을 참조하세요.
		CLI 옵션:
		
		`--deletion-protection`
		
		`--no-deletion-protection`
		
		RDS API 파라미터:
		
		`DeletionProtection`

암호화(Encryption)	암호화 활성화(Enable Encryption)를 선택하면 DB 클러스터에 대해 비활성화되어 있는 암호화를 설정할 수 있습니다.
	
	다중 AZ DB 클러스터의 경우 암호화가 기본적으로 켜져 있습니다.
	
	자세한 내용은 Amazon RDS 리소스 암호화을 참조하세요.
		CLI 옵션:
		
		`--kms-key-id`
		
		`--storage-encrypted`
		
		`--no-storage-encrypted`
		
		RDS API 파라미터:
		
		`KmsKeyId`
		
		`StorageEncrypted`

확장 모니터링	고급 모니터링 활성화(Enable enhanced monitoring)를 선택하면 DB 클러스터가 실행되는 운영 체제에 대해 지표를 실시간으로 수집할 수 있습니다.
	
	자세한 내용은 Enhanced Monitoring을 사용하여 OS 지표 모니터링을 참조하세요.
		CLI 옵션:
		
		`--monitoring-interval`
		
		`--monitoring-role-arn`
		
		RDS API 파라미터:
		
		`MonitoringInterval`
		
		`MonitoringRoleArn`

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터

초기 데이터베이스 이름	DB 클러스터에 있는 데이터베이스의 이름입니다. 이름을 제공하지 않으면 Amazon RDS는 MySQL용 DB 클러스터에 데이터베이스를 생성하지 않습니다. 하지만 PostgreSQL용 DB 클러스터에는 데이터베이스를 생성합니다. 이름은 데이터베이스 엔진에서 예약한 단어는 사용할 수 없습니다. DB 엔진에 따라 다른 제약이 있습니다.
	
	MySQL:
	
	• 1–64자의 영숫자로 구성되어야 합니다.
	
	PostgreSQL:
	
	• 1–63자의 영숫자로 구성되어야 합니다.
	• 글자나 밑줄로 시작해야 합니다. 이후 문자는 글자, 밑줄 또는 숫자(0~9)가 될 수 있습니다.
	• 초기 데이터베이스 이름은 `postgres`입니다.
		CLI 옵션:
		
		`--database-name`
		
		RDS API 파라미터:
		
		`DatabaseName`

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터

로그 내보내기	Amazon CloudWatch Logs에 게시할 데이터베이스 로그 파일의 유형입니다.
	
	자세한 내용은 Amazon CloudWatch Logs에 데이터베이스 로그 게시 섹션을 참조하세요.
		CLI 옵션:
		
		`-enable-cloudwatch-logs-exports`
		
		RDS API 파라미터:
		
		`EnableCloudwatchLogsExports`

유지보수 윈도우	대기 중인 DB 클러스터 설정 변경이 적용되기 위해 경과해야 하는 기간(30분)입니다. 기간이 중요하지 않은 경우 기본 설정 없음을 선택합니다.
	
	자세한 내용은 Amazon RDS 유지 관리 기간 섹션을 참조하세요.
		CLI 옵션:
		
		`--preferred-maintenance-window`
		
		RDS API 파라미터:
		
		`PreferredMaintenanceWindow`

에서 마스터 보안 인증 정보 관리AWS Secrets Manager	Secrets Manager에서 보안 암호의 마스터 사용자 암호를 관리하려면 AWS Secrets Manager에서 마스터 자격 증명 관리를 선택합니다.
	
	원하는 경우 보안 암호를 보호하는 데 사용할 KMS 키를 선택합니다. 자신의 계정에서 KMS 키를 선택하거나 다른 계정의 키를 입력할 수 있습니다.
	
	자세한 내용은 Amazon RDS 및 AWS Secrets Manager를 통한 암호 관리 섹션을 참조하세요.
		CLI 옵션:
		
		`--manage-master-user-password | --no-manage-master-user-password`
		
		`--master-user-secret-kms-key-id`
		
		RDS API 파라미터:
		
		`ManageMasterUserPassword`
		
		`MasterUserSecretKmsKeyId`

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터

마스터 암호	마스터 사용자 계정의 암호입니다.	CLI 옵션:
		
		`--master-user-password`
		
		RDS API 파라미터:
		
		`MasterUserPassword`

마스터 사용자 이름	모든 데이터베이스 권한을 사용하여 DB 클러스터에 로그인하기 위해 마스터 사용자 이름으로 사용할 이름입니다.
	
	• 1–16자의 영숫자와 밑줄을 포함할 수 있습니다.
	• 첫 번째 문자는 글자이어야 합니다.
	• 데이터베이스 엔진에서 예약한 단어는 사용할 수 없습니다.
	
	다중 AZ DB 클러스터를 생성한 후에는 마스터 사용자 이름을 변경할 수 없습니다.
	
	마스터 사용자에게 부여된 권한에 대한 자세한 내용은 마스터 사용자 계정 권한 섹션을 참조하세요.
		CLI 옵션:
		
		`--master-username`
		
		RDS API 파라미터:
		
		`MasterUsername`

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터

성능 개선 도우미	성능 개선 도우미 활성화(Enable Performance Insights)를 선택하면 DB 클러스터 로드를 모니터링하여 데이터베이스 성능을 분석하고 문제를 해결할 수 있습니다.
	
	보존 기간을 선택하여 성능 개선 도우미 데이터 기록을 얼마나 유지할지 결정합니다. 프리 티어의 보존 설정은 기본값(7일)입니다. 성능 데이터를 더 오래 보존하려면 1~24개월을 지정하십시오. 보존 기간에 대한 자세한 내용은 성능 개선 도우미의 요금 및 데이터 보존 단원을 참조하십시오.
	
	이 데이터베이스 볼륨을 암호화하는 데 사용되는 키를 보호하는 데 사용할 KMS 키를 선택합니다. 자신의 계정에서 KMS 키를 선택하거나 다른 계정의 키를 입력할 수 있습니다.
	
	자세한 내용은 성능 개선 도우미를 통한 Amazon RDS 모니터링 섹션을 참조하세요.
		CLI 옵션:
		
		`--enable-performance-insights`
		
		`--no-enable-performance-insights`
		
		`--performance-insights-retention-period`
		
		`--performance-insights-kms-key-id`
		
		RDS API 파라미터:
		
		`EnablePerformanceInsights`
		
		`PerformanceInsightsRetentionPeriod`
		
		`PerformanceInsightsKMSKeyId`

프로비저닝된 IOPS	DB 클러스터에 처음 할당될 프로비저닝된 IOPS의 양(초당 입력/출력 작업 수)입니다.	CLI 옵션:
		
		`--iops`
		
		RDS API 파라미터:
		
		`Iops`

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터

공개 액세스(Public access)	공개적으로 액세스할 수 있음(Publicly accessible)을 선택하면 DB 클러스터에 퍼블릭 IP 주소를 제공하여 VPC 외부에서 액세스할 수 있습니다. 공개적으로 액세스가 가능하려면 DB 클러스터도 VPC의 퍼블릭 서브넷에 있어야 합니다.
	
	공개적으로 액세스할 수 없음(Not publicly accessible)을 선택하면 VPC 내에서만 DB 클러스터에 액세스할 수 있습니다.
	
	자세한 내용은 VPC에 있는 DB 인스턴스를 인터넷에서 숨기기을 참조하세요.
	
	VPC 외부에서 DB 클러스터에 연결하려면 DB 클러스터에 공개적으로 액세스할 수 있어야 합니다. 또한, DB 클러스터 보안 그룹의 인바운드 규칙을 사용하여 액세스 권한을 부여하고 다른 요구 사항을 충족해야 합니다. 자세한 내용은 Amazon RDS DB 인스턴스에 연결할 수 없음을 참조하세요.
	
	DB 클러스터에 공개적으로 액세스할 수 없는 경우 AWS Site-to-Site VPN 연결 또는 AWS Direct Connect 연결을 사용하여 프라이빗 네트워크에서 액세스할 수 있습니다. 자세한 내용은 인터네트워크 트래픽 개인 정보을 참조하세요.
		CLI 옵션:
		
		`--publicly-accessible`
		
		`--no-publicly-accessible`
		
		RDS API 파라미터:
		
		`PubliclyAccessible`

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터

RDS 추가 지원	지원되는 메이저 엔진 버전이 RDS 표준 지원 종료 날짜를 지나서도 계속 실행되도록 하려면 RDS 추가 지원 활성화를 선택합니다.
	
	DB 클러스터를 생성할 때 Amazon RDS는 기본적으로 RDS 확장 지원을 사용합니다. RDS 표준 지원 종료일 후에 새 DB 클러스터가 생성되지 않도록 하고 RDS 확장 지원에 대한 요금이 부과되지 않도록 하려면 이 설정을 비활성화하세요. 기존 DB 클러스터에는 RDS 확장 지원 요금 시작일이 되기 전까지는 요금이 부과되지 않습니다.
	
	자세한 내용은 Amazon RDS를 사용한 Amazon RDS 추가 지원 섹션을 참조하세요.
		CLI 옵션:
		
		`--engine-lifecycle-support`
		
		RDS API 파라미터:
		
		`EngineLifecycleSupport`

스토리지 처리량(throughput)	DB 클러스터의 스토리지 처리량 값입니다. 이 설정은 스토리지 유형으로 범용 SSD(gp3)를 선택한 경우에만 표시됩니다. 이 설정은 구성할 수 없으며, 지정한 IOPS에 따라 자동으로 설정됩니다. 자세한 내용은 gp3 스토리지(권장) 섹션을 참조하세요.	이 값은 자동으로 계산되며 CLI 옵션이 없습니다.

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터
RDS 프록시	Create an RDS Proxy(RDS 프록시 생성)를 선택하여 DB 클러스터용 프록시를 생성합니다. Amazon RDS는 프록시에 대한 IAM 역할과 Secrets Manager 암호를 자동으로 생성합니다.	DB 클러스터를 생성할 때는 사용할 수 없습니다.

스토리지 유형	DB 클러스터의 스토리지 유형입니다.
	
	범용 SSD(gp3), 프로비저닝된 IOPS(io1) 및 프로비저닝된 IOPS SSD(io2) 스토리지만 지원됩니다.
	
	자세한 내용은 Amazon RDS 스토리지 유형 섹션을 참조하세요.
		CLI 옵션:
		
		`--storage-type`
		
		RDS API 파라미터:
		
		`StorageType`

Virtual Private Cloud(VPC)	이 DB 클러스터와 연결할 Amazon VPC 서비스 기반 VPC입니다. 자세한 내용은 Amazon VPC 및 Amazon RDS 섹션을 참조하세요.	CLI 및 API의 경우 VPC 보안 그룹 ID를 지정합니다.

VPC 보안 그룹(방화벽)	DB 클러스터와 연결할 보안 그룹입니다.
	
	자세한 내용은 VPC 보안 그룹 개요을 참조하세요.
		CLI 옵션:
		
		`--vpc-security-group-ids`
		
		RDS API 파라미터:
		
		`VpcSecurityGroupIds`

다중 AZ DB 클러스터를 생성할 때 적용되지 않는 설정

AWS CLI 명령 `create-db-cluster` 및 RDS API 작업 `CreateDBCluster`의 다음 설정은 다중 AZ DB 클러스터에 적용되지 않습니다.

또한, 콘솔에서 다중 AZ DB 클러스터에 대해 이러한 설정을 지정할 수 없습니다.

AWS CLI 설정	RDS API 설정
--availability-zones	AvailabilityZones
--backtrack-window	BacktrackWindow
--character-set-name	CharacterSetName
--domain	Domain
--domain-iam-role-name	DomainIAMRoleName
--enable-global-write-forwarding | --no-enable-global-write-fo rwarding	EnableGlobalWriteForwarding
--enable-http-endpoint | --no-enable-http-endpoint	EnableHttpEndpoint
--global-cluster-identifier	GlobalClusterIdentifier
--option-group-name	OptionGroupName
--pre-signed-url	PreSignedUrl
--replication-source-identifier	ReplicationSourceIdentifier
--scaling-configuration	ScalingConfiguration

Amazon RDS에서 다중 AZ DB 클러스터에 연결

다중 AZ DB 클러스터에는 단일 DB 인스턴스 대신 세 개의 DB 인스턴스가 있습니다. 각 연결은 특정 DB 인스턴스에서 처리합니다. 다중 AZ DB 클러스터에 연결하면 지정한 호스트 이름과 포트가 엔드포인트라는 정규화된 도메인 이름으로 연결됩니다. 다중 AZ DB 클러스터는 엔드포인트 메커니즘을 사용하여 이러한 연결을 추상화합니다. 따라서 DB 클러스터 내에서 연결할 DB 인스턴스를 정확히 지정할 필요가 없습니다. 따라서 일부 DB 인스턴스를 사용할 수 없을 때 모든 호스트 이름을 하드코딩하거나, 연결을 다시 라우팅하기 위해 자체 로직을 작성할 필요가 없습니다.

라이터 엔드포인트는 읽기 및 쓰기 작업을 모두 지원하는 DB 클러스터의 라이터 DB 인스턴스에 연결됩니다. 리더 엔드포인트는 읽기 작업만 지원하는 두 리더 DB 인스턴스 중 하나에 연결됩니다.

엔드포인트를 사용하면 사용 사례에 따라 각 연결을 적절한 DB 인스턴스 또는 DB 인스턴스 그룹에 매핑할 수 있습니다. 예를 들어, DDL 및 DML 문을 수행하기 위해 라이터 DB 인스턴스인 어떤 DB 인스턴스에나 연결할 수 있습니다. 쿼리를 수행하려는 경우 리더 엔드포인트에 연결하면 다중 AZ DB 클러스터가 자동으로 리더 DB 인스턴스 간에 연결을 관리합니다. 진단 또는 튜닝의 경우 특정 DB 인스턴스 엔드포인트에 연결하여 특정 DB 인스턴스에 대한 세부 정보를 검토할 수 있습니다.

DB 인스턴스 연결에 대한 정보는 Amazon RDS DB 인스턴스에 연결 섹션을 참조하세요.

다중 AZ DB 클러스터 연결에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- 클러스터 엔드포인트
- 리더 엔드포인트
- 인스턴스 엔드포인트
- 고가용성 연결
- Amazon RDS용 AWS 드라이버를 사용하여 다중 AZ DB 클러스터에 연결

다중 AZ DB 클러스터 엔드포인트

엔드포인트는 호스트 주소를 포함하는 고유 식별자로 표시됩니다. 다중 AZ DB 클러스터에서 제공하는 엔드포인트 유형은 다음과 같습니다.

클러스터 엔드포인트

다중 AZ DB 클러스터의 클러스터 엔드포인트(또는 라이터 엔드포인트)는 해당 DB 클러스터의 현재 라이터 DB 인스턴스에 연결됩니다. 이 엔드포인트는 DDL 및 DML 문과 같은 쓰기 작업을 수행할 수 있는 유일한 엔드포인트로, 읽기 작업도 수행할 수 있습니다.

각 다중 AZ DB 클러스터에는 클러스터 엔드포인트와 라이터 DB 인스턴스가 하나씩 있습니다.

삽입, 업데이트, 삭제 및 DDL 변경을 비롯하여 DB 클러스터의 모든 쓰기 작업에 대해 클러스터 엔드포인트를 사용합니다. 또한 쿼리와 같은 읽기 작업에도 클러스터 엔드포인트를 사용할 수 있습니다.

DB 클러스터의 현재 라이터 DB 인스턴스에 장애가 발생하면 다중 AZ DB 클러스터가 자동으로 새 라이터 DB 인스턴스로 장애 조치를 수행합니다. 장애 조치가 이루어지는 동안에도 DB 클러스터가 새로운 라이터 DB 인스턴스의 클러스터 엔드포인트 연결 요청을 처리하여 서비스 중단 시간을 최소화합니다.

다음은 다중 AZ DB 클러스터의 클러스터 엔드포인트를 보여주는 예제입니다.

```
mydbcluster.cluster-123456789012.us-east-1.rds.amazonaws.com
```

클러스터 엔드포인트에 연결하는 방법에 대한 자세한 내용은 클러스터 엔드포인트 섹션을 참조하세요.

리더 엔드포인트

다중 AZ DB 클러스터의 리더 엔드포인트는 DB 클러스터에 대한 읽기 전용 연결을 위한 지원을 제공합니다. SELECT 쿼리와 같은 읽기 작업에 리더 엔드포인트를 사용합니다. 리더 DB 인스턴스에서 이러한 문을 처리하여 이 엔드포인트에서 라이터 DB 인스턴스의 오버헤드를 줄입니다. 또한, 클러스터가 동시 SELECT 쿼리를 처리할 수 있는 용량을 확장할 수 있습니다. 각 다중 AZ DB 클러스터에는 리더 엔드포인트가 하나씩 있습니다.

리더 엔드포인트는 리더 DB 인스턴스 중 하나로 각 연결 요청을 전송합니다. 리더 엔드포인트를 세션에 사용하는 경우 해당 세션에서 SELECT와 같은 읽기 전용 문만 수행할 수 있습니다.

다음은 다중 AZ DB 클러스터의 리더 엔드포인트를 보여주는 예제입니다. 리더 엔드포인트의 읽기 전용 인텐트는 클러스터 엔드포인트 이름 내에서 `-ro`로 표시됩니다.

```
mydbcluster.cluster-ro-123456789012.us-east-1.rds.amazonaws.com
```

리더 엔드포인트에 연결하는 방법에 대한 자세한 내용은 리더 엔드포인트 섹션을 참조하세요.

인스턴스 엔드포인트

인스턴스 엔드포인트는 다중 AZ DB 클러스터에 있는 특정 DB 인스턴스에 연결됩니다. DB 클러스터의 DB 인스턴스에는 각각 고유한 인스턴스 엔드포인트가 있습니다. 따라서 DB 클러스터의 현재 라이터 DB 인스턴스에 대해 인스턴스 엔드포인트가 하나씩 있고 DB 클러스터의 각 리더 DB 인스턴스에 대해 인스턴스 엔드포인트가 하나씩 있습니다.

인스턴스 엔드포인트에서는 DB 클러스터에 대한 연결을 직접 제어할 수 있습니다. 이 제어 기능은 클러스터 엔드포인트 또는 리더 엔드포인트를 사용하는 것이 적절하지 않을 수 있는 상황에서 문제를 해결하는 데 도움이 될 수 있습니다. 예를 들어 클라이언트 애플리케이션에서 워크로드 유형에 따라 더욱 세분화된 로드 밸런싱이 필요할 수 있습니다. 이 경우, 여러 클라이언트가 DB 클러스터의 다른 리더 DB 인스턴스에 연결하여 읽기 워크로드를 분산하도록 구성할 수 있습니다.

다음은 다중 AZ DB 클러스터의 DB 인스턴스에 대한 인스턴스 엔드포인트를 보여주는 예제입니다.

```
mydbinstance.123456789012.us-east-1.rds.amazonaws.com
```

인스턴스 엔드포인트에 연결하는 방법에 대한 자세한 내용은 인스턴스 엔드포인트 섹션을 참조하세요.

엔드포인트 보기

콘솔, AWS CLI 또는 Amazon RDS API를 사용하여 클러스터, 리더 및 인스턴스 엔드포인트를 봅니다.

Console

AWS Management Console에서 각 다중 AZ DB 클러스터의 세부 정보 페이지에 클러스터 엔드포인트와 리더 엔드포인트가 표시됩니다. 각 DB 인스턴스의 세부 정보 페이지에서 인스턴스 엔드포인트를 확인할 수 있습니다.

AWS CLI

AWS CLI을 사용하여 describe-db-clusters 명령의 출력에서 라이터 및 리더 엔드포인트를 확인할 수 있습니다. 예를 들어, 다음 명령은 현재 AWS 리전의 모든 클러스터에 대한 엔드포인트 속성을 보여줍니다.

```
aws rds describe-db-cluster-endpoints
```

Amazon RDS API

Amazon RDS API로 DescribeDBClusterEndpoints 작업을 호출하여 엔드포인트를 검색합니다. 출력에는 Amazon Aurora DB 클러스터 엔드포인트(있는 경우)도 표시됩니다.

클러스터 엔드포인트

각 다중 AZ 클러스터에는 기본 제공되는 단일 클러스터 엔드포인트가 있으며, Amazon RDS에서 이 엔드포인트의 이름과 기타 속성을 관리합니다. 이 종류의 엔드포인트는 생성, 삭제 또는 수정할 수 없습니다.

DB 클러스터를 관리하거나, 추출, 변환, 로드(ETL) 작업을 수행하거나, 애플리케이션을 개발 및 테스트할 때 클러스터 엔드포인트를 사용하게 됩니다. 클러스터 엔드포인트는 클러스터의 라이터 DB 인스턴스에 연결됩니다. 라이터 DB 인스턴스는 테이블 및 인덱스를 생성하고, `INSERT` 문을 실행하며, 기타 DDL 및 DML 작업을 수행할 수 있는 유일한 DB 인스턴스입니다.

장애 조치 메커니즘에서 새 DB 인스턴스가 클러스터의 라이터 DB 인스턴스가 되도록 승격하면 클러스터 엔드포인트와 연결되는 물리적 IP 주소가 변경됩니다. 어떤 형식의 연결 풀링이나 기타 멀티플렉싱을 사용하는 경우, 캐싱된 DNS 정보의 TTL(Time to Live)을 플러시하거나 줄이도록 준비합니다. 이렇게 하면 사용할 수 없게 되었거나 장애 조치 후 이제 읽기 전용인 DB 인스턴스에 읽기/쓰기 연결 설정을 시도하지 않아도 됩니다.

리더 엔드포인트

다중 AZ DB 클러스터의 읽기 전용 연결에 리더 엔드포인트를 사용합니다. 이 엔드포인트는 DB 클러스터가 쿼리 집약적인 워크로드를 처리할 수 있도록 도와줍니다. 리더 엔드포인트는 클러스터에서 보고 또는 기타 읽기 전용 작업을 수행하는 애플리케이션에 제공하는 엔드포인트입니다. 리더 엔드포인트는 다중 AZ DB 클러스터에서 사용 가능한 리더 DB 인스턴스에 대한 연결을 전송합니다.

각 다중 AZ 클러스터에는 기본 제공되는 단일 리더 엔드포인트가 있으며, Amazon RDS에서 이 엔드포인트의 이름과 기타 속성을 관리합니다. 이 종류의 엔드포인트는 생성, 삭제 또는 수정할 수 없습니다.

인스턴스 엔드포인트

다중 AZ DB 클러스터의 각 DB 인스턴스에는 Amazon RDS에서 관리하는 고유한 기본 제공 인스턴스 엔드포인트가 있습니다. 이 종류의 엔드포인트는 생성, 삭제 또는 수정할 수 없습니다. 다중 AZ DB 클러스터에서는 일반적으로 인스턴스 엔드포인트보다 라이터 및 리더 엔드포인트를 더 자주 사용합니다.

일상적인 작업에서 인스턴스 엔드포인트를 사용하는 주요 방법은 다중 AZ DB 클러스터의 특정 DB 인스턴스 하나에 영향을 주는 용량 또는 성능 문제를 진단하는 것입니다. 특정 DB 인스턴스에 연결되어 있는 동안 해당 인스턴스의 상태 변수, 지표 등을 검토할 수 있습니다. 이렇게 하면 클러스터의 다른 인스턴스에 일어난 일과 별개로 해당 DB 인스턴스에 일어난 일을 확인하는 데 도움이 됩니다.

고가용성 연결

고가용성이 중요한 다중 AZ DB 클러스터의 경우 라이터 엔드포인트를 읽기/쓰기 또는 범용 연결에 사용하고 리더 엔드포인트를 읽기 전용 연결에 사용합니다. 라이터 및 리더 엔드포인트는 인스턴스 엔드포인트보다 DB 인스턴스 장애 조치를 더 잘 관리합니다. 인스턴스 엔드포인트와 달리, 라이터 및 리더 엔드포인트는 클러스터의 DB 인스턴스를 사용할 수 없게 되면 연결된 DB 인스턴스를 자동으로 변경합니다.

DB 클러스터의 기본 DB 인스턴스에 장애가 발생하면 Amazon RDS가 자동으로 새로운 라이터 DB 인스턴스로 장애 조치하는데, 리더 DB 인스턴스를 새로운 라이터 DB 인스턴스로 승격하여 이를 수행합니다. 장애 조치가 발생하면 라이터 엔드포인트를 사용하여 새롭게 승격된 라이터 DB 인스턴스에 다시 연결할 수 있습니다. 또는 리더 엔드포인트를 사용하여 DB 클러스터에서 리더 DB 인스턴스 중 하나에 다시 연결할 수 있습니다. 장애 조치 중에 리더 DB 인스턴스가 새로운 라이터 DB 인스턴스로 승격된 후 리더 엔드포인트가 잠시 동안 DB 클러스터의 새 라이터 DB 인스턴스에 직접 연결할 수 있습니다. 인스턴스 엔드포인트 연결을 관리하는 애플리케이션 로직을 직접 설계하면 DB 클러스터에서 사용 가능한 DB 인스턴스 집합을 수동으로 또는 프로그래밍 방식으로 검색할 수 있습니다.

Amazon RDS용 AWS 드라이버를 사용하여 다중 AZ DB 클러스터에 연결

더 빠른 전환 및 장애 조치 시간, AWS Secrets Manager, AWS Identity and Access Management(IAM) 및 페더레이션 ID를 사용한 인증을 지원하도록 설계된 AWS 드라이버 제품군입니다. AWS 드라이버는 DB 클러스터 상태 모니터링과 클러스터 토폴로지 파악을 통해 새 라이터를 결정합니다. 이 접근 방식은 전환 및 장애 조치 시간을 오픈 소스 드라이버의 경우 수십 초였던 것에 비해 10초 미만으로 단축합니다.

새로운 서비스 기능이 도입됨에 따라 AWS 드라이버 제품군의 목표는 이러한 서비스 기능에 대한 지원을 기본 제공하는 것입니다.

Amazon Web Service(AWS) JDBC 드라이버를 사용하여 다중 AZ DB 클러스터에 연결

Amazon Web Services(AWS) JDBC 드라이버는 애플리케이션이 클러스터링된 데이터베이스의 기능을 활용할 수 있도록 지원하는 고급 JDBC 래퍼로 설계되었습니다. 이 래퍼는 기존 JDBC 드라이버의 기능을 보완하고 확장합니다. 이 드라이버는 다음 커뮤니티 드라이버와 드롭인 호환됩니다.

- MySQL Connector/J
- MariaDB 커넥터/J
- pgJDBC

AWS JDBC 드라이버를 설치하려면 CLASSPATH 애플리케이션에 있는 AWS JDBC 드라이버 .jar 파일을 추가하고 해당 커뮤니티 드라이버에 대한 참조를 보관해 두세요. 다음과 같이 해당 연결 URL 접두사를 업데이트하세요.

- `jdbc:mysql://~jdbc:aws-wrapper:mysql://`
- `jdbc:mariadb://~jdbc:aws-wrapper:mariadb://`
- `jdbc:postgresql://~jdbc:aws-wrapper:postgresql://`

AWS JDBC 드라이버에 대한 자세한 내용 및 사용 방법에 대한 전체 지침은 Amazon Web Services (AWS) JDBC Driver GitHub repository를 참조하세요.

Amazon Web Service(AWS) Python 드라이버를 사용하여 다중 AZ DB 클러스터에 연결

Amazon Web Services(AWS) Python 드라이버는 고급 Python 래퍼로 설계되었습니다. 이 래퍼는 오픈 소스 Psycopg 드라이버의 기능을 보완하고 확장합니다. AWS Python 드라이버는 Python 버전 3.8 이상을 지원합니다. `pip` 명령을 사용하여 `psycopg` 오픈 소스 패키지와 함께 `aws-advanced-python-wrapper` 패키지를 설치할 수 있습니다.

AWS Python 드라이버에 대한 자세한 내용 및 사용 방법에 대한 전체 지침은 Amazon Web Services (AWS) JDBC Python GitHub repository를 참조하세요.

AWS 컴퓨팅 리소스와 다중 AZ DB 클러스터 자동 연결

다중 AZ DB 클러스터와 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스와 같은 AWS 컴퓨팅 리소스 및 AWS Lambda 함수를 자동으로 연결할 수 있습니다.

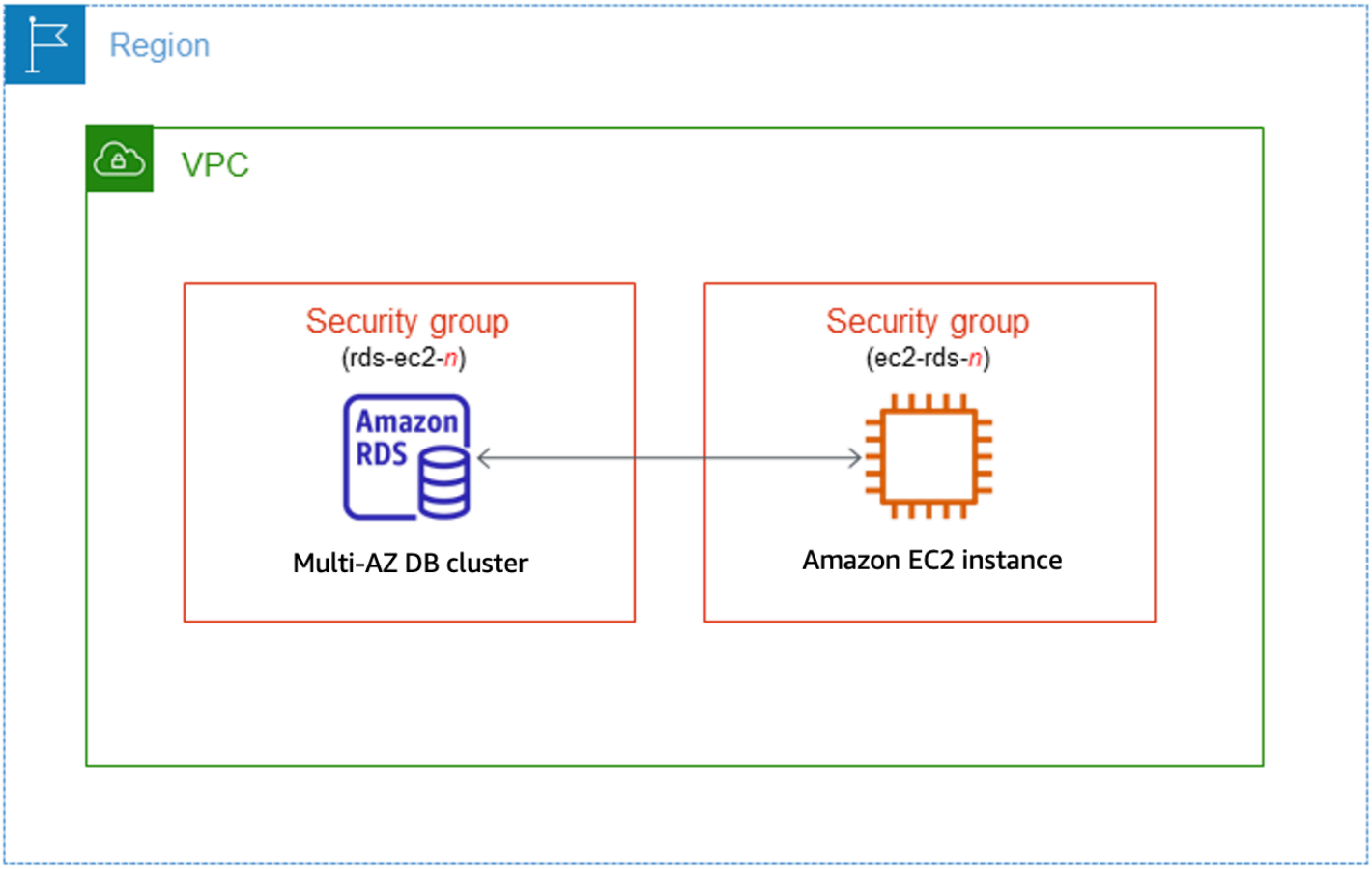
다음 주제에서는 다중 AZ DB 클러스터 배포 내에서 Amazon RDS DB 인스턴스에 안정적으로 연결하기 위해 네트워크 설정, 보안 그룹 및 연결 파라미터를 구성하는 방법에 대한 자세한 지침을 제공합니다. 다중 AZ DB 클러스터와 상호 작용하는 애플리케이션의 네트워크 연결 및 성능을 최적화하여 안전하고 효율적인 데이터 운영을 보장하는 데 중점을 둡니다.

주제

- EC2 인스턴스와 다중 AZ DB 클러스터 자동 연결
- Lambda 함수와 다중 AZ DB 클러스터 자동 연결

EC2 인스턴스와 다중 AZ DB 클러스터 자동 연결

Amazon RDS 콘솔을 사용하여 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스와 다중 AZ DB 클러스터 간의 연결 설정을 간소화할 수 있습니다. 다중 AZ DB 클러스터는 프라이빗 서브넷에 있고, EC2 인스턴스는 VPC 내의 퍼블릭 서브넷에 있는 경우가 많습니다. EC2 인스턴스의 SQL 클라이언트를 사용하여 다중 AZ DB 클러스터에 연결할 수 있습니다. EC2 인스턴스는 프라이빗 다중 AZ DB 클러스터에 액세스하는 웹 서버 또는 애플리케이션을 실행할 수도 있습니다.



다중 AZ DB 클러스터와 동일한 VPC에 있지 않은 EC2 인스턴스에 연결하려는 경우 the section called "VPC에서 DB 인스턴스에 액세스하는 시나리오"의 시나리오를 참조하세요.

주제

- EC2 인스턴스와의 자동 연결 개요
- EC2 인스턴스와 다중 AZ DB 클러스터 자동 연결
- 연결된 컴퓨팅 리소스 보기

EC2 인스턴스와의 자동 연결 개요

EC2 인스턴스와 다중 AZ DB 클러스터 간의 연결을 자동으로 설정할 때 Amazon RDS는 EC2 인스턴스 및 DB 클러스터에 대한 VPC 보안 그룹을 구성합니다.

다음은 EC2 인스턴스를 다중 AZ DB 클러스터와 연결하기 위한 요구 사항입니다.

- EC2 인스턴스는 다중 AZ DB 클러스터와 동일한 VPC에 있어야 합니다.

EC2 인스턴스가 동일한 VPC에 없는 경우 콘솔은 EC2 인스턴스를 생성하기 위한 링크를 제공합니다.

- 연결을 설정하는 사용자는 다음 EC2 작업을 수행할 수 있는 권한이 있어야 합니다.
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeSecurityGroups`
  - `ec2:ModifyNetworkInterfaceAttribute`
  - `ec2:RevokeSecurityGroupEgress`

EC2 인스턴스에 대한 연결을 설정하면 Amazon RDS는 다음 테이블에 설명된 대로 다중 AZ DB 클러스터 및 EC2 인스턴스와 연결된 보안 그룹의 현재 구성을 기반으로 조치를 취합니다.

현재 RDS 보안 그룹 구성	현재 EC2 보안 그룹 구성	RDS 작업
`rds-ec2-`*n* 패턴(*n*은 숫자를 나타냄)과 일치하는 이름의 다중 AZ DB 클러스터와 연결된 보안 그룹이 하나 이상 있습니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 EC2 인스턴스의 VPC 보안 그룹을 소스로 하는 인바운드 규칙이 하나만 포함됩니다.	`rds-ec2-`*n*(*n*은 숫자를 나타냄) 패턴과 일치하는 이름을 가진 EC2 인스턴스와 연결된 보안 그룹이 하나 이상 있습니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 다중 AZ DB 클러스터의 VPC 보안 그룹을 소스로 하는 아웃바운드 규칙이 하나만 포함됩니다.	Amazon RDS는 아무런 조치도 취하지 않습니다. EC2 인스턴스와 다중 AZ DB 클러스터 간의 연결이 이미 자동으로 구성되었습니다. EC2 인스턴스와 RDS 데이터베이스 사이에 이미 연결이 존재하기 때문에 보안 그룹은 수정되지 않습니다.
다음 중 하나의 조건이 적용됩니다. •	다음 중 하나의 조건이 적용됩니다. •	RDS action: create new security groups

현재 RDS 보안 그룹 구성	현재 EC2 보안 그룹 구성	RDS 작업
`rds-ec2-`*n* 패턴과 일치하는 이름의 다중 AZ DB 클러스터와 연결된 보안 그룹이 없습니다. • `rds-ec2-`*n* 패턴과 일치하는 이름의 다중 AZ DB 클러스터와 연결된 보안 그룹이 하나 이상 있습니다. 하지만 이러한 보안 그룹 중 어느 것도 EC2 인스턴스와의 연결에 사용할 수 없습니다. 보안 그룹에 EC2 인스턴스의 VPC 보안 그룹을 소스로 하는 인바운드 규칙 하나가 없는 경우 보안 그룹을 사용할 수 없습니다. 보안 그룹이 수정된 경우에도 사용할 수 없습니다. 수정 사항의 예로는 규칙 추가 또는 기존 규칙의 포트 변경이 있습니다.	`ec2-rds-`*n* 패턴과 일치하는 이름을 가진 EC2 인스턴스와 연결된 보안 그룹이 없습니다. • `ec2-rds-`*n* 패턴과 일치하는 이름을 가진 EC2 인스턴스와 연결된 보안 그룹이 하나 이상 있습니다. 하지만 이러한 보안 그룹 중 어느 것도 다중 AZ DB 클러스터와의 연결에 사용할 수 없습니다. 보안 그룹에 다중 AZ DB 클러스터의 VPC 보안 그룹을 소스로 하는 아웃바운드 규칙 하나가 없는 경우 보안 그룹을 사용할 수 없습니다. 보안 그룹이 수정된 경우에도 사용할 수 없습니다.	

현재 RDS 보안 그룹 구성	현재 EC2 보안 그룹 구성	RDS 작업
`rds-ec2-`*n* 패턴과 일치하는 이름의 다중 AZ DB 클러스터와 연결된 보안 그룹이 하나 이상 있습니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 EC2 인스턴스의 VPC 보안 그룹을 소스로 하는 인바운드 규칙이 하나만 포함됩니다.	`ec2-rds-`*n* 패턴과 일치하는 이름을 가진 EC2 인스턴스와 연결된 보안 그룹이 하나 이상 있습니다. 하지만 이러한 보안 그룹 중 어느 것도 다중 AZ DB 클러스터와의 연결에 사용할 수 없습니다. 보안 그룹에 다중 AZ DB 클러스터의 VPC 보안 그룹을 소스로 하는 아웃바운드 규칙 하나가 없는 경우 보안 그룹을 사용할 수 없습니다. 보안 그룹이 수정된 경우에도 사용할 수 없습니다.	RDS action: create new security groups
`rds-ec2-`*n* 패턴과 일치하는 이름의 다중 AZ DB 클러스터와 연결된 보안 그룹이 하나 이상 있습니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 EC2 인스턴스의 VPC 보안 그룹을 소스로 하는 인바운드 규칙이 하나만 포함됩니다.	연결에 유효한 EC2 보안 그룹이 있지만 EC2 인스턴스와 연결되어 있지 않습니다. 이 보안 그룹의 이름이 `rds-ec2-`*n* 패턴과 일치합니다. 수정되지 않았습니다. 여기에는 다중 AZ DB 클러스터의 VPC 보안 그룹을 소스로 하는 아웃바운드 규칙이 하나만 포함됩니다.	RDS action: associate EC2 security group

현재 RDS 보안 그룹 구성	현재 EC2 보안 그룹 구성	RDS 작업
다음 중 하나의 조건이 적용됩니다. • `rds-ec2-n` 패턴과 일치하는 이름의 다중 AZ DB 클러스터와 연결된 보안 그룹이 없습니다. • `rds-ec2-n` 패턴과 일치하는 이름의 다중 AZ DB 클러스터와 연결된 보안 그룹이 하나 이상 있습니다. 하지만 이러한 보안 그룹 중 어느 것도 EC2 인스턴스와의 연결에 사용할 수 없습니다. 보안 그룹에 EC2 인스턴스의 VPC 보안 그룹을 소스로 하는 인바운드 규칙 하나가 없는 경우 보안 그룹을 사용할 수 없습니다. 보안 그룹이 수정된 경우에도 사용할 수 없습니다.	`rds-ec2-n` 패턴과 일치하는 이름을 가진 EC2 인스턴스와 연결된 보안 그룹이 하나 이상 있습니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 다중 AZ DB 클러스터의 VPC 보안 그룹을 소스로 하는 아웃바운드 규칙이 하나만 포함됩니다.	RDS action: create new security groups

RDS 작업: 새 보안 그룹 생성

Amazon RDS에서 다음 작업을 수행합니다.

- `rds-ec2-n` 패턴과 일치하는 새 보안 그룹을 생성합니다. 이 보안 그룹에는 EC2 인스턴스의 VPC 보안 그룹을 소스로 하는 인바운드 규칙이 하나 포함됩니다. 이 보안 그룹은 다중 AZ DB 클러스터와 연결되어 있으며 EC2 인스턴스가 다중 AZ DB 클러스터에 액세스하도록 허용합니다.

- `ec2-rds-n` 패턴과 일치하는 새 보안 그룹을 생성합니다. 이 보안 그룹에는 다중 AZ DB 클러스터의 VPC 보안 그룹을 소스로 하는 아웃바운드 규칙이 하나 포함됩니다. 이 보안 그룹은 EC2 인스턴스와 연결되어 있으며 EC2 인스턴스가 다중 AZ DB 클러스터에 트래픽을 보내도록 허용합니다.

RDS 작업: EC2 보안 그룹 연결

Amazon RDS는 유효한 기존 EC2 보안 그룹을 EC2 인스턴스와 연결합니다. 이 보안 그룹은 EC2 인스턴스가 다중 AZ DB 클러스터에 트래픽을 보내도록 허용합니다.

EC2 인스턴스와 다중 AZ DB 클러스터 자동 연결

EC2 인스턴스와 RDS 데이터베이스 간의 연결을 설정하기 전에 EC2 인스턴스와의 자동 연결 개요에 설명된 요구 사항을 충족하는지 확인하세요.

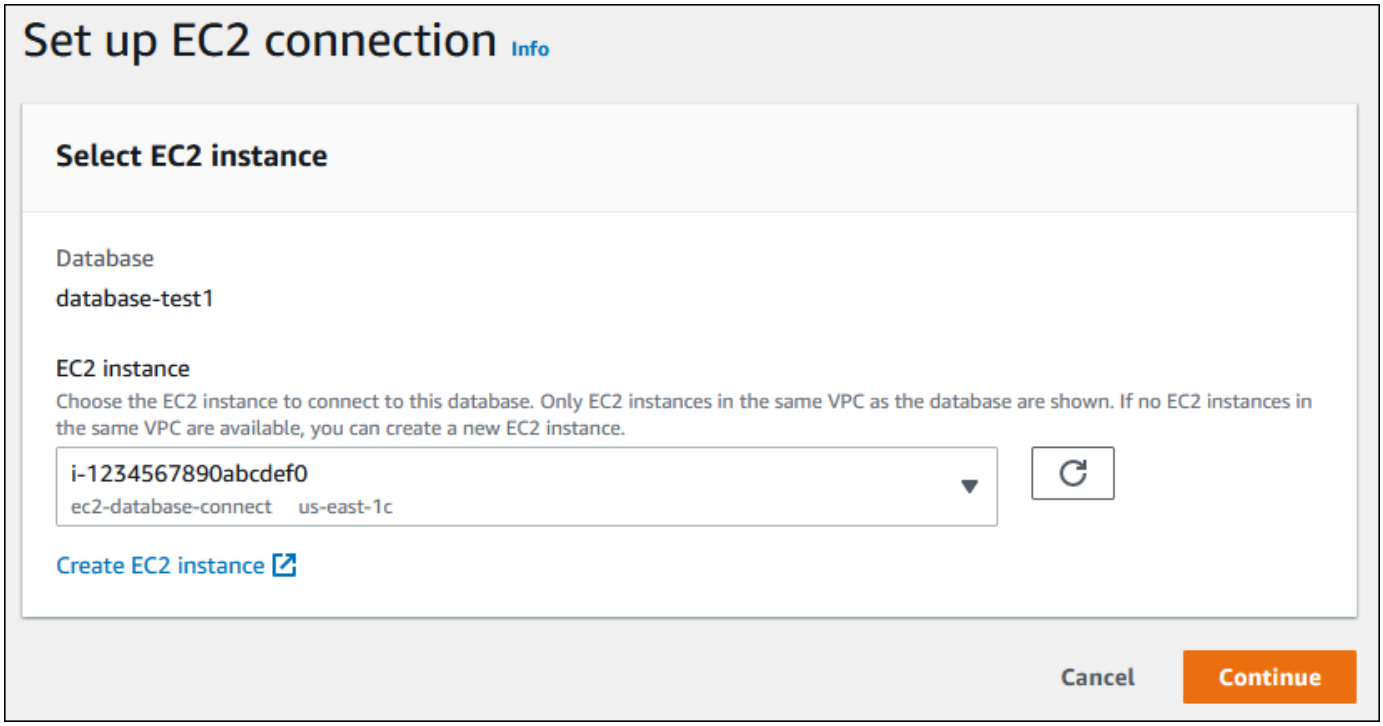
연결을 구성한 후에 보안 그룹을 변경할 경우 변경 사항이 EC2 인스턴스와 RDS 데이터베이스 간의 연결에 영향을 미칠 수 있습니다.

> **Note**
>
> AWS Management Console을 사용해야만 EC2 인스턴스와 RDS 데이터베이스 간의 연결을 자동으로 설정할 수 있습니다. AWS CLI 또는 RDS API로는 연결을 자동으로 설정할 수 없습니다.

EC2 인스턴스와 RDS 데이터베이스를 자동으로 연결하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택하고 RDS 데이터베이스를 선택합니다.
3. 작업에서 EC2 연결 설정을 선택합니다.

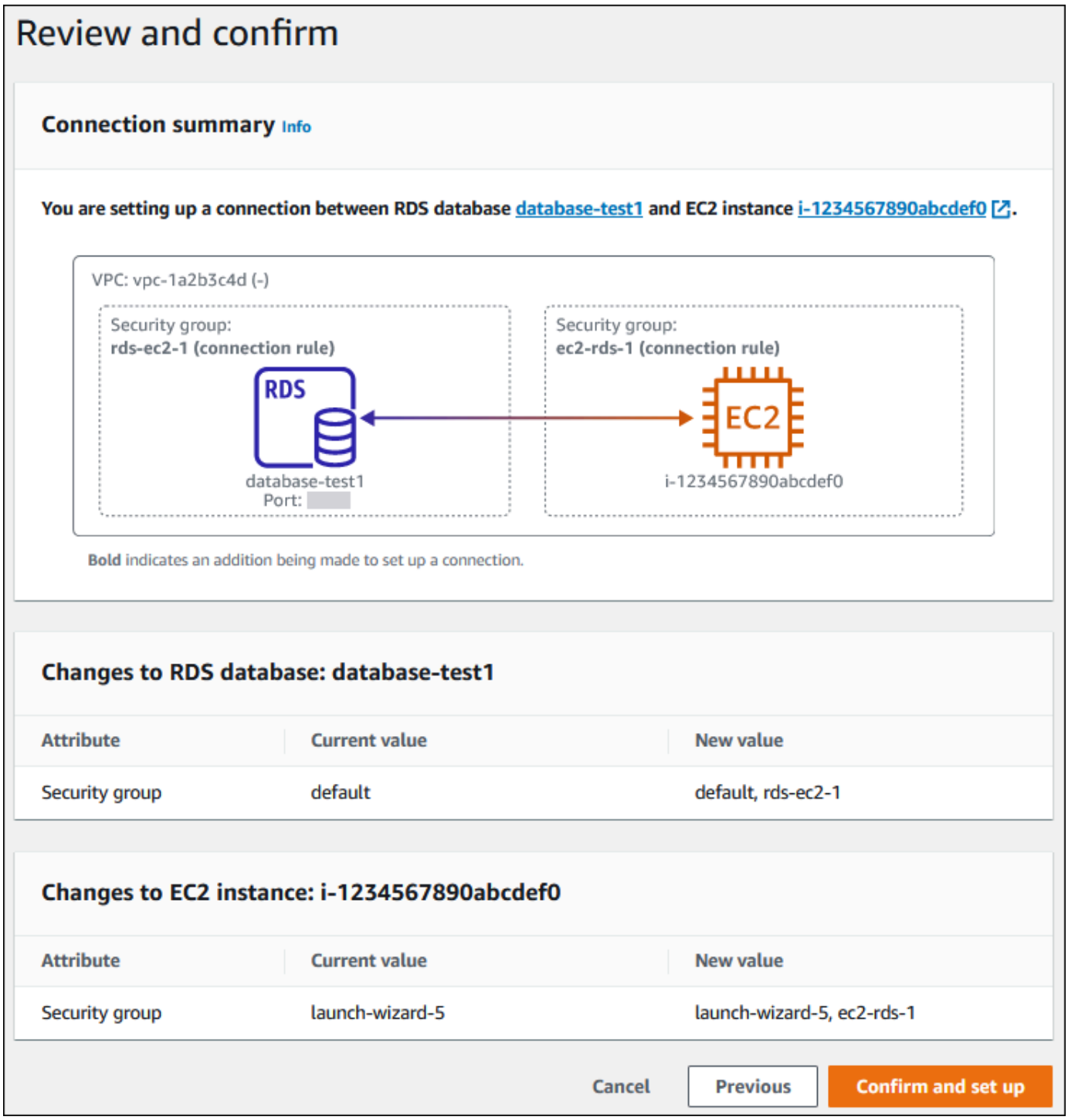
   EC2 연결 설정 페이지가 나타납니다.
4. EC2 연결 설정 페이지에서 EC2 인스턴스를 선택합니다.



동일한 VPC에 EC2 인스턴스가 없는 경우 EC2 인스턴스 생성을 선택하여 인스턴스를 생성합니다. 이 경우 새 EC2 인스턴스가 RDS 데이터베이스와 동일한 VPC 있어야 합니다.

5. 계속을 선택합니다.

   검토 및 확인 페이지가 나타납니다.



6. 검토 및 확인 페이지에서 RDS가 EC2 인스턴스와의 연결을 설정하기 위해 적용할 변경 사항을 검토합니다.

   변경 내용이 올바르면 확인 및 설정을 선택합니다.

   변경 내용이 올바르지 않으면 이전 또는 취소를 선택합니다.

연결된 컴퓨팅 리소스 보기

AWS Management Console을 사용하여 RDS 데이터베이스에 연결된 컴퓨팅 리소스를 볼 수 있습니다. 표시되는 리소스에는 자동으로 설정된 컴퓨팅 리소스 연결이 포함됩니다. 다음과 같은 방법으로 컴퓨팅 리소스와의 연결을 자동으로 설정할 수 있습니다.

- 데이터베이스를 생성할 때 컴퓨팅 리소스를 선택할 수 있습니다.

  자세한 내용은 Amazon RDS DB 인스턴스 생성 및 Amazon RDS용 다중 AZ DB 클러스터 생성 섹션을 참조하세요.
- 기존 데이터베이스와 컴퓨팅 리소스 간의 연결을 설정할 수 있습니다.

  자세한 내용은 EC2 인스턴스와 RDS 데이터베이스 자동 연결 단원을 참조하십시오.

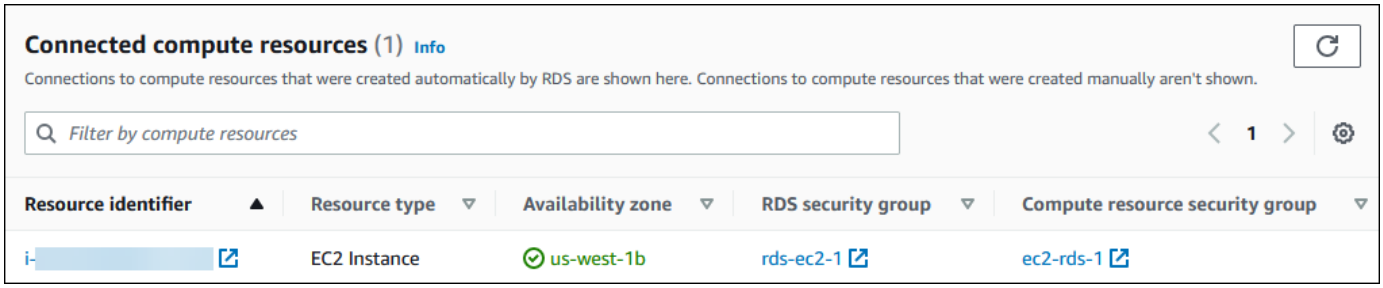
나열된 컴퓨팅 리소스에는 데이터베이스에 수동으로 연결된 리소스는 포함되지 않습니다. 예를 들어 데이터베이스와 연결된 VPC 보안 그룹에 규칙을 추가하여 컴퓨팅 리소스가 데이터베이스에 수동으로 액세스하도록 허용할 수 있습니다.

컴퓨팅 리소스가 나열되려면 다음 조건이 적용되어야 합니다.

- 컴퓨팅 리소스와 연결된 보안 그룹의 이름은 패턴 `ec2-rds-`*n*(여기서*n*은 숫자)과 일치합니다.
- 컴퓨팅 리소스와 연결된 보안 그룹에는 포트 범위가 RDS 데이터베이스에서 사용하는 포트로 설정된 아웃바운드 규칙이 있습니다.
- 컴퓨팅 리소스와 연결된 보안 그룹에는 소스가 RDS 데이터베이스에 연결된 보안 그룹으로 설정된 아웃바운드 규칙이 있습니다.
- RDS 데이터베이스와 연결된 보안 그룹의 이름은 패턴 `rds-ec2-`*n*(여기서 *n*은 숫자)과 일치합니다.
- RDS 데이터베이스와 연결된 보안 그룹에는 포트 범위가 RDS 데이터베이스에서 사용하는 포트로 설정된 인바운드 규칙이 있습니다.
- RDS 데이터베이스와 연결된 보안 그룹에는 소스가 컴퓨팅 리소스에 연결된 보안 그룹으로 설정된 인바운드 규칙이 있습니다.

RDS 데이터베이스에 연결된 컴퓨팅 리소스를 보는 방법

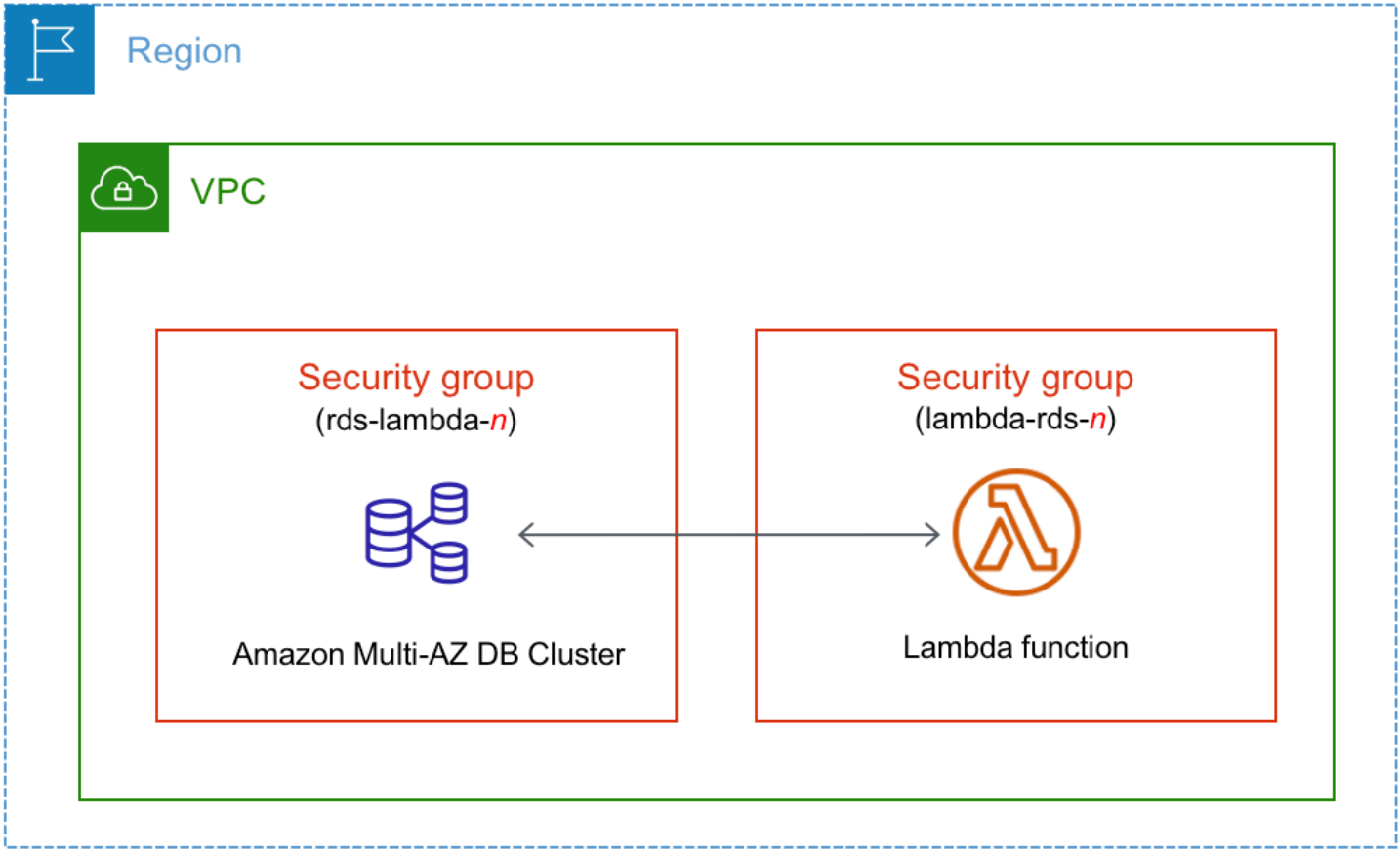
1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 다음 RDS 데이터베이스의 이름을 선택합니다.
3. 연결 및 보안 탭의 연결된 컴퓨팅 리소스에서 컴퓨팅 리소스를 확인합니다.



Lambda 함수와 다중 AZ DB 클러스터 자동 연결

RDS 콘솔을 사용하여 Lambda 함수와 다중 AZ DB 클러스터 간의 연결 설정을 간소화할 수 있습니다. RDS 콘솔을 사용하여 Lambda 함수와 다중 AZ DB 클러스터 간의 연결 설정을 간소화할 수 있습니다. 다중 AZ DB 클러스터가 VPC 내 프라이빗 서브넷에 있는 경우가 많습니다. 애플리케이션에서 프라이빗 다중 AZ DB 클러스터에 액세스하는 데 Lambda 함수를 사용할 수 있습니다.

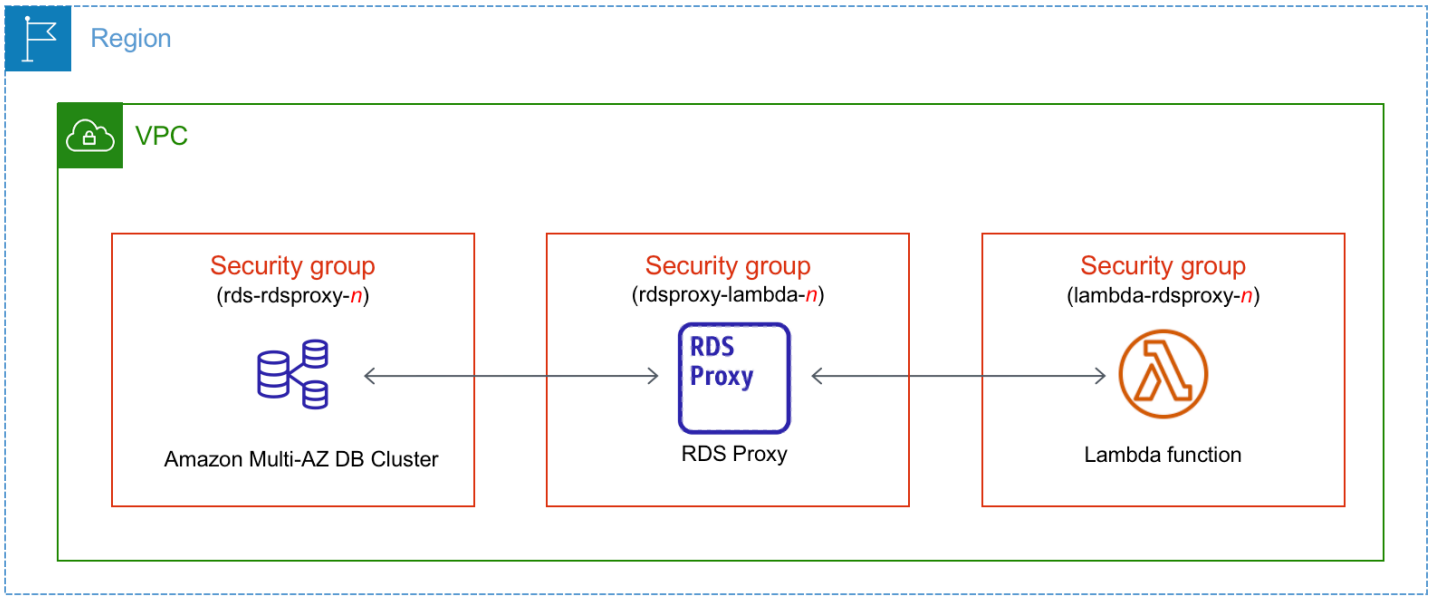
다음 이미지는 다중 AZ DB 클러스터와 Lambda 함수 간의 직접 연결을 보여줍니다.



RDS 프록시를 통해 Lambda 함수와 다중 AZ DB 클러스터 간의 연결을 설정하여 데이터베이스 성능과 복원력을 개선할 수 있습니다. Lambda 함수는 RDS 프록시가 제공하는 연결 풀링의 이점을 누릴 수 있

는 단기 데이터베이스 연결을 자주 만드는 경우가 많습니다. Lambda 애플리케이션 코드에서 데이터베이스 보안 인증 정보를 관리하는 대신 Lambda 함수에 대해 이미 가지고 있는 IAM 인증을 활용할 수 있습니다. 자세한 내용은 Amazon RDS Proxy 단원을 참조하십시오.

콘솔을 사용하여 연결용 프록시를 자동으로 만들 수 있습니다. 기존 프록시를 선택할 수도 있습니다. 콘솔은 데이터베이스 및 Lambda 함수와의 연결을 허용하도록 프록시 보안 그룹을 업데이트합니다. 데이터베이스 보안 인증 정보를 입력하거나 데이터베이스에 액세스하는 데 필요한 Secrets Manager 보안 암호를 선택할 수 있습니다.



주제

- Lambda 함수와의 자동 연결 개요
- Lambda 함수와 다중 AZ DB 클러스터 자동 연결
- 연결된 컴퓨팅 리소스 보기

Lambda 함수와의 자동 연결 개요

Lambda 함수와 다중 AZ DB 클러스터 간의 연결을 자동으로 설정할 때 Amazon RDS는 Lambda 함수 및 DB 클러스터에 대한 VPC 보안 그룹을 구성합니다.

다음은 Lambda 함수를 다중 AZ DB 클러스터와 연결하기 위한 요구 사항입니다.

- Lambda 함수가 다중 AZ DB 클러스터와 같은 VPC에 있어야 합니다.

  Lambda 함수가 동일한 VPC에 없는 경우 콘솔은 생성을 위한 링크를 제공합니다.

- 연결을 설정하는 사용자는 다음과 같은 Amazon RDS, Amazon EC2, Lambda, Secrets Manager 및 IAM 작업을 수행할 권한이 있어야 합니다.
  - Amazon RDS
    - `rds:CreateDBProxies`
    - `rds:DescribeDBInstances`
    - `rds:DescribeDBProxies`
    - `rds:ModifyDBInstance`
    - `rds:ModifyDBProxy`
    - `rds:RegisterProxyTargets`
  - Amazon EC2
    - `ec2:AuthorizeSecurityGroupEgress`
    - `ec2:AuthorizeSecurityGroupIngress`
    - `ec2:CreateSecurityGroup`
    - `ec2:DeleteSecurityGroup`
    - `ec2:DescribeSecurityGroups`
    - `ec2:RevokeSecurityGroupEgress`
    - `ec2:RevokeSecurityGroupIngress`
  - Lambda
    - `lambda:CreateFunctions`
    - `lambda:ListFunctions`
    - `lambda:UpdateFunctionConfiguration`
  - Secrets Manager
    - `sercetsmanager:CreateSecret`
    - `secretsmanager:DescribeSecret`
  - IAM
    - `iam:AttachPolicy`
    - `iam:CreateRole`
    - `iam:CreatePolicy`
  - AWS KMS
    - `kms:describeKey`

Lambda 함수와 다중 AZ DB 클러스터 간의 연결을 설정할 때 Amazon RDS는 함수 및 다중 AZ DB 클러스터에 대한 VPC 보안 그룹을 구성합니다. RDS 프록시를 사용하는 경우 Amazon RDS는 프록시에 대한 VPC 보안 그룹도 구성합니다. Amazon RDS는 다음 테이블에 설명된 것과 같이 다중 AZ DB 클러스터, Lambda 함수 및 프록시와 관련된 보안 그룹의 현재 구성에 따라 작동합니다.

현재 RDS 보안 그룹 구성	현재 Lambda 보안 그룹 구성	현재 프록시 보안 그룹 구성	RDS 작업
모든 리소스의 보안 그룹이 올바른 명명 패턴을 따르고 올바른 인바운드 및 아웃바운드 규칙을 적용하기 때문에 Amazon RDS가 아무런 작업을 수행하지 않습니다.	이름이 `rds-lambda-` *n* 패턴(*n*은 숫자를 나타냄)과 일치하며 다중 AZ DB 클러스터와 연결된 보안 그룹이 하나 이상 있거나 연결된 프록시의 `TargetHealth` 가 `AVAILABLE` 입니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 Lambda 함수 또는 프록시의 VPC 보안 그룹을 소스로 하는 인바운드 규칙이 하나만 있습니다.	이름이 `lambda-rds-` *n* 또는 `lambda-rdsproxy-` *n*(*n*은 숫자를 나타냄) 패턴과 일치하며 Lambda 함수와 연결된 보안 그룹이 하나 이상 있습니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 다중 AZ DB 클러스터 또는 프록시의 VPC 보안 그룹을 대상으로 하는 아웃바운드 규칙이 하나만 있습니다.	이름이 `rdsproxy-lambda-` *n*(*n*은 숫자를 나타냄) 패턴과 일치하며 프록시와 연결된 보안 그룹이 하나 이상 있습니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 Lambda 함수 및 다중 AZ DB 클러스터의 VPC 보안 그룹이 포함된 인바운드 및 아웃바운드 규칙이 있습니다.
다음 중 하나의 조건이 적용됩니다. • 이름이 `rds-lambda-` *n* 패턴과 일치하며 다중 AZ DB 클러스터와 연결된 보안 그룹이 없거나 연결된 프록	다음 중 하나의 조건이 적용됩니다. • 이름이 `lambda-rds-` *n* 또는 `lambda-rdsproxy-` *n* 패턴과 일치하며 Lambda	다음 중 하나의 조건이 적용됩니다. • 이름이 `rdsproxy-lambda-` *n* 패턴과 일치하며 프록시와 연결된 보안 그룹이 없습니다. •	RDS action: create new security groups

현재 RDS 보안 그룹 구성	현재 Lambda 보안 그룹 구성	현재 프록시 보안 그룹 구성	RDS 작업
시의 TargetHealth 가 AVAILABLE 입니다. • 이름이 rds-lambda- *n* 패턴과 일치하며 다중 AZ DB 클러스터와 연결된 보안 그룹이 하나 이상 있거나 연결된 프록시의 TargetHealth 가 AVAILABLE 입니다. 그러나 Amazon RDS는 Lambda 함수와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon RDS는 Lambda 함수 또는 프록시의 VPC 보안 그룹을 소스로 하는 인바운드 규칙 하나가 포함되지 않은 보안 그룹을 사용할 수 없습니다. 또한 수정된 보안 그룹을 사용할 수 없습니다. 수정 사항의 예로는 규칙 추가 또는 기존 규칙의 포트 변경이 있습니다.	함수와 연결된 보안 그룹이 없습니다. • 이름이 lambda-rds- *n* 또는 lambda-rdsproxy- *n* 패턴과 일치하며 Lambda 함수와 연결된 보안 그룹이 하나 이상 있습니다. 그러나 Amazon RDS는 다중 AZ DB 클러스터와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon RDS는 다중 AZ DB 클러스터 또는 프록시의 VPC 보안 그룹을 소스로 하는 아웃바운드 규칙 하나가 포함되지 않은 보안 그룹을 사용할 수 없습니다. 또한 수정된 보안 그룹을 사용할 수 없습니다.	이름이 rdsproxy-lambda- *n* 패턴과 일치하며 프록시와 연결된 보안 그룹이 하나 이상 있습니다. 그러나 Amazon RDS는 다중 AZ DB 클러스터 또는 Lambda 함수와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon RDS는 다중 AZ DB 클러스터 및 Lambda 함수의 VPC 보안 그룹을 대상으로 하는 인바운드 및 아웃바운드 규칙이 포함되지 않은 보안 그룹을 사용할 수 없습니다. 또한 수정된 보안 그룹을 사용할 수 없습니다.	

현재 RDS 보안 그룹 구성	현재 Lambda 보안 그룹 구성	현재 프록시 보안 그룹 구성	RDS 작업
이름이 `rds-lambda-` *n* 패턴과 일치하며 다중 AZ DB 클러스터와 연결된 보안 그룹이 하나 이상 있거나 연결된 프록시의 `TargetHealth` 가 `AVAILABLE` 입니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 Lambda 함수 또는 프록시의 VPC 보안 그룹을 소스로 하는 인바운드 규칙이 하나만 있습니다.	이름이 `lambda-rds-` *n* 또는 `lambda-rdsproxy-` *n* 패턴과 일치하며 Lambda 함수와 연결된 보안 그룹이 하나 이상 있습니다. 그러나 Amazon RDS는 다중 AZ DB 클러스터와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon RDS는 다중 AZ DB 클러스터 또는 프록시의 VPC 보안 그룹을 대상으로 하는 아웃바운드 규칙 하나가 포함되지 않은 보안 그룹을 사용할 수 없습니다. 또한 수정된 보안 그룹을 사용할 수 없습니다.	이름이 `rdsproxy-lambda-` *n* 패턴과 일치하며 프록시와 연결된 보안 그룹이 하나 이상 있습니다. 그러나 Amazon RDS는 다중 AZ DB 클러스터 또는 Lambda 함수와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon RDS는 다중 AZ DB 클러스터 및 Lambda 함수의 VPC 보안 그룹을 대상으로 하는 인바운드 및 아웃바운드 규칙이 포함되지 않은 보안 그룹을 사용할 수 없습니다. 또한 수정된 보안 그룹을 사용할 수 없습니다.	RDS action: create new security groups

현재 RDS 보안 그룹 구성	현재 Lambda 보안 그룹 구성	현재 프록시 보안 그룹 구성	RDS 작업
이름이 `rds-lambda-` *n* 패턴과 일치하며 다중 AZ DB 클러스터와 연결된 보안 그룹이 하나 이상 있거나 연결된 프록시의 `TargetHealth` 가 `AVAILABLE` 입니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 Lambda 함수 또는 프록시의 VPC 보안 그룹을 소스로 하는 인바운드 규칙이 하나만 있습니다.	연결에 유효한 Lambda 보안 그룹이 존재하지만 Lambda 함수와 연결되어 있지 않습니다. 이 보안 그룹의 이름이 `lambda-rds-` *n* 또는 `lambda-rdsproxy-` *n* 패턴과 일치합니다. 수정되지 않았습니다. 다중 AZ DB 클러스터 또는 프록시의 VPC 보안 그룹을 대상으로 하는 아웃바운드 규칙이 하나만 있습니다.	연결에 유효한 프록시 보안 그룹이 있지만 프록시와 연결되어 있지 않습니다. 이 보안 그룹의 이름이 `rdsproxy-lambda-` *n* 패턴과 일치합니다. 수정되지 않았습니다. 다중 AZ DB 클러스터 및 Lambda 함수의 VPC 보안 그룹이 포함된 인바운드 및 아웃바운드 규칙이 있습니다.	RDS action: associate Lambda security group

현재 RDS 보안 그룹 구성	현재 Lambda 보안 그룹 구성	현재 프록시 보안 그룹 구성	RDS 작업
다음 중 하나의 조건이 적용됩니다. • 이름이 `rds-lambda-` *n* 패턴과 일치하며 다중 AZ DB 클러스터와 연결된 보안 그룹이 없거나 연결된 프록시의 `TargetHealth` 가 `AVAILABLE` 입니다. • 이름이 `rds-lambda-` *n* 패턴과 일치하며 다중 AZ DB 클러스터와 연결된 보안 그룹이 하나 이상 있거나 연결된 프록시의 `TargetHealth` 가 `AVAILABLE` 입니다. 그러나 Amazon RDS는 Lambda 함수 또는 프록시와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon RDS는 Lambda 함수 또는 프록시의 VPC 보안 그룹을 소스로 하는 인바운드 규칙 하나	이름이 `lambda-rds-` *n* 또는 `lambda-rdsproxy-` *n* 패턴과 일치하며 Lambda 함수와 연결된 보안 그룹이 하나 이상 있습니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 다중 AZ DB 클러스터 또는 프록시의 VPC 보안 그룹을 대상으로 하는 아웃바운드 규칙이 하나만 있습니다.	이름이 `rdsproxy-lambda-` *n* 패턴과 일치하며 프록시와 연결된 보안 그룹이 하나 이상 있습니다. 해당 패턴과 일치하는 보안 그룹이 수정되지 않았습니다. 이 보안 그룹에는 다중 AZ DB 클러스터 및 Lambda 함수의 VPC 보안 그룹이 포함된 인바운드 및 아웃바운드 규칙이 있습니다.	RDS action: create new security groups

현재 RDS 보안 그룹 구성	현재 Lambda 보안 그룹 구성	현재 프록시 보안 그룹 구성	RDS 작업
가 포함되지 않은 보안 그룹을 사용할 수 없습니다. 또한 수정된 보안 그룹을 사용할 수 없습니다.			

현재 RDS 보안 그룹 구성	현재 Lambda 보안 그룹 구성	현재 프록시 보안 그룹 구성	RDS 작업
`rds-rdsproxy-` *n* 패턴(*n*은 숫자를 나타냄)과 일치하는 이름의 다중 AZ DB 클러스터와 연결된 보안 그룹이 하나 이상 있습니다.	다음 중 하나의 조건이 적용됩니다. • 이름이 `lambda-rds-` *n* 또는 `lambda-rdsproxy-` *n* 패턴과 일치하며 Lambda 함수와 연결된 보안 그룹이 없습니다. • 이름이 `lambda-rds-` *n* 또는 `lambda-rdsproxy-` *n* 패턴과 일치하며 Lambda 함수와 연결된 보안 그룹이 하나 이상 있습니다. 그러나 Amazon RDS는 다중 AZ DB 클러스터와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon RDS는 다중 AZ DB 클러스터 또는 프록시의 VPC 보안 그룹을 대상으로 하는 아웃바운드 규칙 하나가 포함되지 않은 보안 그	다음 중 하나의 조건이 적용됩니다. • 이름이 `rdsproxy-lambda-` *n* 패턴과 일치하며 프록시와 연결된 보안 그룹이 없습니다. • 이름이 `rdsproxy-lambda-` *n* 패턴과 일치하며 프록시와 연결된 보안 그룹이 하나 이상 있습니다. 그러나 Amazon RDS는 다중 AZ DB 클러스터 또는 Lambda 함수와의 연결에 이러한 보안 그룹을 사용할 수 없습니다. Amazon RDS는 다중 AZ DB 클러스터 및 Lambda 함수의 VPC 보안 그룹을 대상으로 하는 인바운드 및 아웃바운드 규칙이 포함되지 않은 보안 그룹을 사용할 수 없습니다. 또한 수정된 보안 그룹을 사용할 수 없습니다.	RDS action: create new security groups

현재 RDS 보안 그룹 구성	현재 Lambda 보안 그룹 구성	현재 프록시 보안 그룹 구성	RDS 작업
	룹을 사용할 수 없습니다. 또한 수정된 보안 그룹을 사용할 수 없습니다.		

RDS 작업: 새 보안 그룹 생성

Amazon RDS에서 다음 작업을 수행합니다.

- `rds-lambda-`*n* 패턴과 일치하는 새 보안 그룹을 생성합니다. 이 보안 그룹에는 Lambda 함수 또는 프록시의 VPC 보안 그룹을 소스로 하는 인바운드 규칙이 하나 있습니다. 이 보안 그룹은 다중 AZ DB 클러스터와 연결되어 있으며 함수 또는 프록시가 다중 AZ DB 클러스터에 액세스하도록 허용합니다.
- `lambda-rds-`*n* 패턴과 일치하는 새 보안 그룹을 생성합니다. 이 보안 그룹에는 다중 AZ DB 클러스터 또는 프록시의 VPC 보안 그룹을 대상으로 하는 아웃바운드 규칙이 하나 있습니다. 이 보안 그룹은 Lambda 함수와 연결되어 있으며, Lambda 함수가 트래픽을 다중 AZ DB 클러스터로 보내거나 프록시를 통해 보내도록 허용합니다.
- `rdsproxy-lambda-`*n* 패턴과 일치하는 새 보안 그룹을 생성합니다. 이 보안 그룹에는 다중 AZ DB 클러스터 및 Lambda 함수의 VPC 보안 그룹이 포함된 인바운드 및 아웃바운드 규칙이 있습니다.

RDS 작업: Lambda 보안 그룹 연결

Amazon RDS는 유효한 기존 Lambda 보안 그룹을 Lambda 함수와 연결합니다. 이 보안 그룹은 함수가 트래픽을 다중 AZ DB 클러스터로 보내거나 프록시를 통해 보내도록 허용합니다.

Lambda 함수와 다중 AZ DB 클러스터 자동 연결

Amazon RDS 콘솔을 사용하여 Lambda 함수를 다중 AZ DB 클러스터에 자동으로 연결할 수 있습니다. 이렇게 하면 이러한 리소스 간의 연결을 설정하는 프로세스가 간소화됩니다.

RDS 프록시를 사용하여 연결에 프록시를 포함할 수도 있습니다. Lambda 함수는 RDS 프록시가 제공하는 연결 풀링의 이점을 누릴 수 있는 단기 데이터베이스 연결을 자주 만듭니다. Lambda 애플리케이션 코드에서 데이터베이스 보안 인증 정보를 관리하는 대신 Lambda 함수에 대해 이미 설정한 IAM 인증을 사용할 수도 있습니다.

Lambda 연결 설정 페이지를 사용하여 기존 다중 AZ DB 클러스터를 신규 및 기존 Lambda 함수에 연결할 수 있습니다. 이 설정 프로세스는 필요한 보안 그룹을 자동으로 설정합니다.

Lambda 함수와 다중 AZ DB 클러스터 간의 연결을 설정하기 전에 다음 요구 사항을 충족해야 합니다.

- Lambda 함수와 다중 AZ DB 클러스터가 동일한 VPC에 있습니다.
- 사용자 계정에 대한 올바른 권한이 있습니다. 요구 사항에 대한 자세한 내용은 Lambda 함수와의 자동 연결 개요 섹션을 참조하세요.

연결을 구성한 후에 보안 그룹을 변경할 경우 변경 사항이 Lambda 함수와 다중 AZ DB 클러스터 간의 연결에 영향을 미칠 수 있습니다.

> **Note**
>
> AWS Management Console에서만 다중 AZ DB 클러스터와 Lambda 함수의 연결을 자동으로 설정할 수 있습니다. Lambda 함수를 연결하려면 다중 AZ DB 클러스터의 모든 인스턴스가 사용 가능 상태여야 합니다.

Lambda 함수와 다중 AZ DB 클러스터를 자동으로 연결하는 방법

<result>

설정을 확인하면 Amazon RDS가 Lambda 함수, RDS 프록시(프록시를 사용한 경우) 및 다중 AZ DB 클러스터 연결 프로세스를 시작합니다. 콘솔에 연결 세부 정보 대화 상자가 표시되며, 여기에 리소스 간 연결을 허용하는 보안 그룹 변경 사항이 나열됩니다.
</result>

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 다음, Lambda 함수에 연결할 다중 AZ DB 클러스터를 선택합니다.
3. 작업에서 Lambda 연결 설정을 선택합니다.
4. Lambda 연결 설정 페이지의 Lambda 함수 선택에서 다음 중 하나를 수행합니다.
    - 다중 AZ DB 클러스터와 동일한 VPC에 기존 Lambda 함수가 있는 경우 기존 함수 선택을 선택하고 해당 함수를 선택합니다.

- 동일한 VPC에 Lambda 함수가 없는 경우 새 함수 생성을 선택한 다음 함수 이름을 입력합니다. 기본 런타임은 Nodejs.18로 설정되어 있습니다. 연결 설정을 완료한 후 Lambda 콘솔에서 새 Lambda 함수의 설정을 수정할 수 있습니다.

5. (선택 사항) RDS 프록시에서 RDS 프록시를 사용하여 연결을 선택하고 다음 중 하나를 수행합니다.
   - 사용하려는 기존 프록시가 있는 경우 기존 프록시 선택을 선택하고 해당 프록시를 선택합니다.
   - 프록시가 없고 Amazon RDS에서 자동으로 프록시를 생성하도록 하려면 새 프록시 생성을 선택합니다. 그런 다음, 데이터베이스 보안 인증 정보에서 다음 중 하나를 수행합니다.

     a. 데이터베이스 사용자 이름 및 암호를 선택한 다음, 다중 AZ DB 클러스터의 사용자 이름과 암호를 입력합니다.

     b. Secrets Manager 보안 암호를 선택합니다. 그런 다음, 보안 암호 선택에서 AWS Secrets Manager 보안 암호를 선택합니다. Secrets Manager 보안 암호가 없다면 새 Secrets Manager 보안 암호 생성을 선택하여 새 보안 암호를 생성합니다. 보안 암호를 생성한 후 보안 암호 선택에서 새 보안 암호를 선택합니다.

     새 프록시를 생성한 후 기존 프록시 선택을 선택하고 해당 프록시를 선택합니다. 프록시를 연결에 사용하려면 다소 시간이 걸릴 수 있습니다.

6. (선택 사항) 연결 요약을 확장하고 리소스에서 강조 표시된 업데이트를 확인합니다.
7. 설정을 선택합니다.

연결된 컴퓨팅 리소스 보기

AWS Management Console을 사용하여 다중 AZ DB 클러스터에 연결된 컴퓨팅 리소스를 볼 수 있습니다. 표시되는 리소스에는 Amazon RDS가 자동으로 설정한 컴퓨팅 리소스 연결이 포함됩니다.

나열된 컴퓨팅 리소스에는 다중 AZ DB 클러스터에 수동으로 연결된 컴퓨팅 리소스가 포함되지 않습니다. 예를 들어 클러스터와 연결된 VPC 보안 그룹에 규칙을 추가하여 컴퓨팅 리소스가 다중 AZ DB 클러스터에 수동으로 액세스하도록 허용할 수 있습니다.

콘솔에 Lambda 함수를 나열하려면 다음 조건을 적용해야 합니다.

- 컴퓨팅 리소스와 연결된 보안 그룹의 이름이 `lambda-rds-`*n* 또는 `lambda-rdsproxy-`*n*(*n*은 숫자를 나타냄) 패턴과 일치합니다.

- 컴퓨팅 리소스와 연결된 보안 그룹에 포트 범위가 다중 AZ DB 클러스터 또는 프록시의 포트로 설정된 아웃바운드 규칙이 있습니다. 아웃바운드 규칙의 대상이 다중 AZ DB 클러스터 또는 관련 프록시와 연결된 보안 그룹으로 설정되어 있어야 합니다.
- 데이터베이스와 연결된 프록시에 연결된 보안 그룹의 이름이 `rds-rdsproxy-`*n*(*n*은 숫자를 나타냄) 패턴과 일치합니다.
- 함수와 연결된 보안 그룹에 포트가 다중 AZ DB 클러스터 또는 관련 프록시가 사용하는 포트로 설정된 아웃바운드 규칙이 있습니다. 대상이 다중 AZ DB 클러스터 또는 관련 프록시와 연결된 보안 그룹으로 설정되어 있어야 합니다.

다중 AZ DB 클러스터에 자동으로 연결된 컴퓨팅 리소스를 보는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 후 다중 AZ DB 클러스터를 선택합니다.
3. 연결 및 보안 탭의 연결된 컴퓨팅 리소스에서 컴퓨팅 리소스를 확인합니다.

Amazon RDS용 다중 AZ DB 클러스터 수정

다중 AZ DB 클러스터에는 라이터 DB 인스턴스와 두 개의 리더 DB 인스턴스가 세 개의 개별 가용 영역에 있습니다. 다중 AZ DB 클러스터는 다중 AZ 배포에 비해 고가용성, 높은 읽기 워크로드 용량 및 짧은 대기 시간을 제공합니다. 다중 AZ DB 클러스터에 대한 자세한 내용은 Amazon RDS용 다중 AZ DB 클러스터 배포 섹션을 참조하세요.

다중 AZ DB 클러스터를 수정하여 설정을 변경할 수 있습니다. 다중 AZ DB 클러스터에서 스냅샷을 만드는 등의 작업을 수행할 수도 있습니다.

> **Important**
>
> 다중 AZ DB 클러스터에 있는 DB 인스턴스를 수정할 수 없습니다. 모든 수정은 DB 클러스터 수준에서 이루어져야 합니다. 다중 AZ DB 클러스터에 있는 DB 인스턴스에 대한 수행할 수 있는 유일한 작업은 인스턴스를 재부팅하는 것입니다.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 다중 AZ DB 클러스터를 수정할 수 있습니다.

콘솔

다중 AZ DB 클러스터를 수정하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스(Databases)를 선택한 다음 수정하려는 다중 AZ DB 클러스터를 선택합니다.
3. 수정을 선택합니다. Modify DB cluster(DB 클러스터 수정) 페이지가 나타납니다.
4. 원하는 설정을 모두 변경합니다. 각 설정에 대한 자세한 내용은 다중 AZ DB 클러스터를 수정하기 위한 설정 단원을 참조하십시오.
5. 원하는 대로 모두 변경되었으면 [Continue]를 선택하고 수정 사항 요약을 확인합니다.
6. (선택 사항) 즉시 적용을 선택하여 변경 내용을 즉시 적용합니다. 일부의 경우 이 옵션을 선택하면 가동 중지 시간이 발생할 수 있습니다. 자세한 내용은 변경 사항 즉시 적용 섹션을 참조하세요.
7. 확인 페이지에서 변경 내용을 검토합니다. 변경 내용이 올바른 경우 DB 클러스터 수정(Modify DB cluster)을 선택하여 변경 내용을 저장합니다.

   또는 뒤로를 선택하여 변경 내용을 편집하거나 취소를 선택하여 변경 내용을 취소합니다.

AWS CLI

AWS CLI를 사용하여 다중 AZ DB 클러스터를 수정하려면 modify-db-cluster 명령을 호출하면 됩니다. 수정할 옵션의 DB 클러스터 식별자 및 값을 지정합니다. 각 옵션에 대한 자세한 내용은 다중 AZ DB 클러스터를 수정하기 위한 설정 단원을 참조하십시오.

Example

다음 코드는 백업 보존 기간을 1주(7일)로 설정하여 `my-multi-az-dbcluster`를 수정합니다. 이 코드는 `--deletion-protection`을 사용하여 삭제 방지를 활성화합니다. `--no-deletion-protection`을 사용하여 삭제 방지를 끌 수 있습니다. 변경 사항은 `--no-apply-immediately`를 사용하여 다음 유지 관리 기간에 적용됩니다. 변경 사항을 바로 적용하려면 `--apply-immediately`를 사용합니다. 자세한 내용은 변경 사항 즉시 적용 섹션을 참조하세요.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier my-multi-az-dbcluster \
    --backup-retention-period 7 \
    --deletion-protection \
    --no-apply-immediately
```

Windows의 경우:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier my-multi-az-dbcluster ^
    --backup-retention-period 7 ^
    --deletion-protection ^
    --no-apply-immediately
```

RDS API

Amazon RDS API를 사용하여 다중 AZ DB 클러스터를 수정하려면 ModifyDBCluster 작업을 호출합니다. DB 클러스터 식별자 및 수정하려는 설정의 파라미터를 지정합니다. 각 파라미터에 대한 자세한 내용은 다중 AZ DB 클러스터를 수정하기 위한 설정 단원을 참조하십시오.

변경 사항 즉시 적용

다중 AZ DB 클러스터를 수정할 때 변경 사항을 즉시 적용할 수 있습니다. 변경 사항을 즉시 적용하려면 AWS Management Console에서 즉시 적용 옵션을 선택합니다. 또는 AWS CLI를 호출할 때 --

`apply-immediately` 옵션을 사용하거나 Amazon RDS API를 사용할 때 `ApplyImmediately` 파라미터를 `true`로 설정합니다.

변경 사항을 즉시 적용하지 않기로 선택하면 변경 사항이 보류 중 수정 사항 대기열로 보내집니다. 다음 유지 관리 기간에 대기열에 있는 보류 중 변경 사항이 적용됩니다. 변경 사항을 즉시 적용하기로 선택하면 새로운 변경 사항과 보류 중인 수정 사항 대기열에 있는 모든 변경 사항이 적용됩니다.

> **Important**
>
> 보류 중인 수정 사항 중에 DB 클러스터가 일시적으로 사용 중단(다운타임)되어야 하는 변경 내용이 하나라도 있으면 즉시 적용 옵션을 선택하는 경우 예기치 않은 다운타임이 발생할 수 있습니다.
> 변경 사항을 즉시 적용하도록 선택하면, 보류 중인 수정 사항도 다음 유지 관리 기간이 아니라 즉시 적용됩니다.
> 보류 중인 변경 사항을 다음 유지 관리 기간에 적용하지 않으려면 변경 사항을 되돌리도록 DB 인스턴스를 수정하면 됩니다. AWS CLI를 사용하고 `--apply-immediately` 옵션을 지정하여 이 작업을 수행할 수 있습니다.

변경을 지연하기로 선택하더라도 일부 데이터베이스 설정 변경 사항은 즉시 적용됩니다. 다른 데이터베이스 설정이 즉시 적용 설정과 상호 작용하는 방식을 보려면 다중 AZ DB 클러스터를 수정하기 위한 설정 단원을 참조하십시오.

다중 AZ DB 클러스터를 수정하기 위한 설정

다중 AZ DB 클러스터를 수정하는 데 사용할 수 있는 설정에 대한 자세한 내용은 다음 표를 참조하세요. AWS CLI 옵션에 대한 자세한 정보는 modify-db-cluster를 참조하세요. RDS API 파라미터에 대한 자세한 내용은 ModifyDBCluster를 참조하세요.

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항
할당된 스토리지	DB 클러스터의 각 DB 인스턴스에 할당할 스토리지 양(기비바이트)입니다. 자세한 내용은 Amazon RDS DB	CLI 옵션: `--allocated-storage` RDS API 파라미터:	변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다. 변경 사항을 즉시 적용하도록 선택하지	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항
	인스턴스 스토리지을 참조하세요.	`AllocatedStorage`	않으면 다음 유지 관리 기간 중 적용됩니다.	

자동 마이너 버전 업그레이드	자동 마이너 버전 업그레이드(Enable auto minor version upgrade)를 선택하면 DB 클러스터가 지원되는 경우 기본 마이너 DB 엔진 버전 업그레이드를 자동으로 수신하도록 할 수 있습니다. Amazon RDS는 유지 관리 기간에 자동 마이너 버전 업그레이드를 수행합니다.	CLI 옵션:
		`--auto-minor-version-upgrade`
		`--no-auto-minor-version-upgrade`
		RDS API 파라미터:
		`AutoMinorVersionUpgrade`
			변경 사항이 즉시 적용됩니다. 이 설정은 즉시 적용 설정을 무시합니다.	이 변경 도중에는 가동 중지 시간이 발생합니다.

백업 보관 기간	DB 클러스터의 자동 백업을 유지할 기간(단위: 일)입니다. 이 값은 0보다 커야 합니다.
	자세한 내용은 백업 소개 섹션을 참조하세요.
		CLI 옵션:
		`--backup-retention-period`
		RDS API 파라미터:
		`BackupRetentionPeriod`
			변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
			변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.
				이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항

백업 기간	Amazon RDS가 DB 클러스터의 백업을 자동으로 수행하는 기간입니다. 데이터베이스를 백업할 특정 기간을 지정하지 않으려면 기본값으로 기본 설정 없음(No preference)을 사용합니다.
	
	자세한 내용은 백업 소개 섹션을 참조하세요.
		CLI 옵션:
		
		`--preferred-backup-window`
		
		RDS API 파라미터:
		
		`PreferredBackupWindow`
			비동기 방식이지만 최대한 빠른 시간 내에 변경 사항이 적용됩니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.

인증 기관	DB 클러스터에서 사용하는 서버 인증서의 CA(인증 기관)입니다.
	
	자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 섹션을 참조하세요.
		CLI 옵션:
		
		`--ca-certificate-identifier`
		
		RDS API 파라미터:
		
		`CACertificateIdentifier`
			변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
			
			변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.
				가동 중단 시간은 DB 엔진이 재시작 없는 교체를 지원하지 않는 경우에만 발생합니다. describe-db-engine-versions AWS CLI 명령을 사용하여 DB 엔진이 재시작 없는 교체를 지원하는지 여부를 확인할 수 있습니다.

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항

스냅샷으로 태그 복사	이 옵션은 스냅샷을 생성할 때 DB 클러스터 태그를 DB 스냅샷에 복사합니다.
	
	자세한 내용은 Amazon RDS 리소스에 태그 지정을 참조하세요.
		CLI 옵션:
		
		`-copy-tags-to-snapshot`
		
		`-no-copy-tags-to-snapshot`
		
		RDS API 파라미터:
		
		`CopyTagsToSnapshot`
			변경 사항이 즉시 적용됩니다. 이 설정은 즉시 적용 설정을 무시합니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.

데이터베이스 인증	다중 AZ DB 클러스터에만 암호 인증이 지원됩니다.	암호 인증이 기본값이므로 지원되지 않습니다.	변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다. 변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항
DB 클러스터 식별자	DB 클러스터 식별자입니다. 이 값은 소문자 문자열로 저장됩니다. DB 클러스터 식별자를 변경하면 DB 클러스터 엔드포인트가 변경됩니다. DB 클러스터에 있는 DB 인스턴스의 식별자와 엔드포인트도 변경됩니다. 새 DB 클러스터 이름은 고유해야 합니다. 최대 길이는 63자입니다. DB 클러스터의 DB 인스턴스 이름이 DB 클러스터의 새 이름에 맞게 변경됩니다. 새 DB 인스턴스 이름은 기존 DB 인스턴스의 이름과 같으면 안 됩니다. 예를 들어 DB 클러스터 이름을 maz로 변경하면 DB 인스턴스 이름도 maz-instance-1 로 변경될 수 있습	CLI 옵션: --new-db-cluster-identifier RDS API 파라미터: NewDBClusterIdentifier	변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다. 변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항
	니다. 이 경우에는 이름이 `maz-instance-1` 로 지정된 기존 DB 인스턴스가 있을 수 없습니다. 자세한 내용은 Amazon RDS용 다중 AZ DB 클러스터 이름 변경 섹션을 참조하세요.			

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항

DB 클러스터 인스턴스 클래스	다중 AZ DB 클러스터에 있는 각 DB 인스턴스의 컴퓨팅 및 메모리 용량입니다(예: `db.r6gd.xlarge`).
	
	가능하면 일반 쿼리 작업 세트가 메모리에 상주할 수 있을 정도로 큰 DB 인스턴스 클래스를 선택합니다. 작업 세트가 메모리에 상주할 경우 시스템의 디스크 쓰기가 불필요하여 성능이 향상됩니다.
	
	자세한 내용은 the section called “다중 AZ DB 클러스터에서 사용 가능한 인스턴스 클래스” 섹션을 참조하세요.
		CLI 옵션:
		
		`--db-cluster-instance-class`
		
		RDS API 파라미터:
		
		`DBClusterInstanceClass`
			변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
			
			변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.
				이 변경 도중에는 가동 중지 시간이 발생합니다.

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항

DB 클러스터 파라미터 그룹	DB 클러스터와 연결할 DB 클러스터 파라미터 그룹.
	
	자세한 내용은 다중 AZ DB 클러스터용 파라미터 그룹을 참조하세요.
		CLI 옵션:
		
		`--db-cluster-parameter-group-name`
		
		RDS API 파라미터:
		
		`DBClusterParameterGroupName`
			파라미터 그룹 변경 사항은 즉시 적용됩니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다. 파라미터 그룹을 변경하면 일부 파라미터의 변경 내용이 재부팅 없이 다중 AZ DB 클러스터의 DB 인스턴스에 즉시 적용됩니다. 다른 파라미터의 변경 사항은 DB 인스턴스를 재부팅한 후에만 적용됩니다.

DB 엔진 버전	사용할 데이터베이스 엔진의 버전입니다.	CLI 옵션:
		
		`--engine-version`
		
		RDS API 파라미터:
		
		`EngineVersion`
			변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
			
			변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.
				이 변경 도중에는 가동 중지 시간이 발생합니다.

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항

삭제 방지	DB 클러스터가 삭제되지 않도록 방지하기 위한 삭제 방지 활성화
	
	자세한 내용은 DB 인스턴스 삭제 섹션을 참조하세요.
		CLI 옵션:
		
		`--deletion-protection`
		
		`--no-deletion-protection`
		
		RDS API 파라미터:
		
		`DeletionProtection`
			변경 사항이 즉시 적용됩니다. 이 설정은 즉시 적용 설정을 무시합니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.

유지보수 윈도우	대기 중인 DB 클러스터 설정 변경이 적용되기 위해 경과해야 하는 기간(30분)입니다. 기간이 중요하지 않은 경우 기본 설정 없음을 선택합니다.
	
	자세한 내용은 Amazon RDS 유지 관리 기간 섹션을 참조하세요.
		CLI 옵션:
		
		`--preferred-maintenance-window`
		
		RDS API 파라미터:
		
		`PreferredMaintenanceWindow`
			변경 사항이 즉시 적용됩니다. 이 설정은 즉시 적용 설정을 무시합니다.	가동 중지를 유발할 수 있는 작업이 하나 이상 보류 중이고, 유지 관리 기간이 현재 시간을 포함하여 변경된 경우 보류 중인 작업이 즉시 적용되고 가동 중지 시간이 발생합니다.

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항

AWS Secrets Manager에서 마스터 자격 증명 관리	Secrets Manager에서 보안 암호의 마스터 사용자 암호를 관리하려면 AWS Secrets Manager에서 마스터 자격 증명 관리를 선택합니다.
	
	원하는 경우 보안 암호를 보호하는 데 사용할 KMS 키를 선택합니다. 자신의 계정에서 KMS 키를 선택하거나 다른 계정의 키를 입력할 수 있습니다.
	
	RDS에서 이미 DB 클러스터의 마스터 사용자 암호를 관리하고 있는 경우 보안 암호 즉시 교체를 선택하여 마스터 사용자 암호를 교체할 수 있습니다.
	
	자세한 내용은 Amazon RDS 및 AWS Secrets Manager를 통한
		CLI 옵션:
		
		`--manage-master-user-password | --no-manage-master-user-password`
		
		`--master-user-secret-kms-key-id`
		
		`--rotate-master-user-password | --no-rotate-master-user-password`
		
		RDS API 파라미터:
		
		`ManageMasterUserPassword`
		
		`MasterUserSecretKmsKeyId`
		
		`RotateMasterUserPassword`
			자동 마스터 사용자 암호 관리를 켜거나 해제하면 변경 내용이 즉시 적용됩니다. 이 변경은 즉시 적용 설정을 무시합니다.
			
			마스터 사용자 암호를 교체하는 경우 변경 내용이 즉시 적용되도록 지정해야 합니다.
				이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항
	암호 관리 섹션을 참조하세요.			

새 마스터 암호	마스터 사용자 계정의 암호입니다.	CLI 옵션:
		`--master-user-password`
		RDS API 파라미터:
		`MasterUserPassword`
			비동기 방식이지만 최대한 빠른 시간 내에 변경 사항이 적용됩니다. 이 설정은 즉시 적용 설정을 무시합니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.

프로비저닝된 IOPS	DB 클러스터에 처음 할당될 프로비저닝된 IOPS의 양(초당 입력/출력 작업 수)입니다.	CLI 옵션:
		`--iops`
		RDS API 파라미터:
		`Iops`
			변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
			변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.
				이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항
공개 액세스(Public access)	공개적으로 액세스할 수 있음(Publicly accessible)을 선택하면 DB 클러스터에 퍼블릭 IP 주소를 부여하여 Virtual Private Cloud(VPC) 외부에서 액세스할 수 있습니다. 공개적으로 액세스가 가능하려면 DB 클러스터도 VPC의 퍼블릭 서브넷에 있어야 합니다. 공개적으로 액세스할 수 없음(Not publicly accessible)을 선택하면 VPC 내에서만 DB 클러스터에 액세스할 수 있습니다. 자세한 내용은 VPC에 있는 DB 인스턴스를 인터넷에서 숨기기을 참조하세요. VPC 외부에서 DB 클러스터에 연결하려면 DB 클러스	DB 클러스터를 수정할 때는 사용할 수 없습니다.	변경 사항이 즉시 적용됩니다. 이 설정은 즉시 적용 설정을 무시합니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항
	터에 공개적으로 액세스할 수 있어야 합니다. 또한, DB 클러스터 보안 그룹의 인바운드 규칙을 사용하여 액세스 권한을 부여하고 다른 요구 사항을 충족해야 합니다. 자세한 내용은 Amazon RDS DB 인스턴스에 연결할 수 없음을 참조하세요. DB 클러스터에 공개적으로 액세스할 수 없는 경우 AWS Site-to-Site VPN 연결 또는 AWS Direct Connect 연결을 사용하여 프라이빗 네트워크에서 액세스할 수 있습니다. 자세한 내용은 인터네트워크 트래픽 개인 정보 섹션을 참조하세요.			

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터	변경이 발생할 때	가동 중지 참고 사항

스토리지 유형	DB 클러스터의 스토리지 유형입니다.
	
	범용 SSD(gp3), 프로비저닝된 IOPS(io1) 및 프로비저닝된 IOPS SSD(io2) 스토리지만 지원됩니다.
	
	자세한 내용은 Amazon RDS 스토리지 유형 섹션을 참조하세요.
		CLI 옵션:
		
		`--storage-type`
		
		RDS API 파라미터:
		
		`StorageType`
			변경 사항을 즉시 적용하도록 선택하면 즉시 적용됩니다.
			
			변경 사항을 즉시 적용하도록 선택하지 않으면 다음 유지 관리 기간 중 적용됩니다.
				이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.

VPC 보안 그룹	DB 클러스터와 연결할 보안 그룹입니다.
	
	자세한 내용은 VPC 보안 그룹 개요을 참조하세요.
		CLI 옵션:
		
		`--vpc-security-group-ids`
		
		RDS API 파라미터:
		
		`VpcSecurityGroupIds`
			비동기 방식이지만 최대한 빠른 시간 내에 변경 사항이 적용됩니다. 이 설정은 즉시 적용 설정을 무시합니다.	이 변경 도중에는 가동 중지 시간이 발생하지 않습니다.

다중 AZ DB 클러스터를 수정할 때 적용되지 않는 설정

AWS CLI 명령 modify-db-cluster 및 RDS API 작업 ModifyDBCluster의 다음 설정은 다중 AZ DB 클러스터에 적용되지 않습니다.

또한, 콘솔에서 다중 AZ DB 클러스터에 대한 관련 설정을 수정할 수 없습니다.

AWS CLI 설정	RDS API 설정
--backtrack-window	BacktrackWindow
--cloudwatch-logs-export-configuration	CloudwatchLogsExportConfiguration
--copy-tags-to-snapshot | --no-copy-tags-to-snapshot	CopyTagsToSnapshot
--db-instance-parameter-group-name	DBInstanceParameterGroupName
--domain	Domain
--domain-iam-role-name	DomainIAMRoleName
--enable-global-write-forwarding | --no-enable-global-write-forwarding	EnableGlobalWriteForwarding
--enable-http-endpoint | --no-enable-http-endpoint	EnableHttpEndpoint
--option-group-name	OptionGroupName
--port	Port
--scaling-configuration	ScalingConfiguration
--storage-type	StorageType

Amazon RDS에서 다중 AZ DB 클러스터의 엔진 버전 업그레이드

Amazon RDS는 지원되는 각 데이터베이스 엔진의 최신 버전을 제공하여 다중 AZ DB 클러스터를 최신 상태로 유지합니다. 이 주제에서는 다중 AZ DB 클러스터를 최신 버전으로 업그레이드하는 프로세스를 설명합니다.

다중 AZ DB 클러스터를 업그레이드하려면 새 호환 엔진 버전을 선택하고 잠재적 가동 중지 시간을 계획해야 합니다. 이 프로세스는 다중 AZ 아키텍처의 장애 조치 기능을 활용하여 중단을 최소화합니다. 업그레이드 중에 기본 인스턴스가 먼저 업데이트된 다음 가용성을 유지하기 위해 대기 인스턴스에 대한 장애 조치가 수행됩니다. 모범 사례에는 트래픽이 적은 기간 동안 업그레이드 수행, 비프로덕션 환경에서 테스트, 새 버전과의 애플리케이션 호환성 확인이 포함됩니다.

주제

- 마이너 버전 업그레이드
- 메이저 버전 업그레이드
- 다중 AZ DB 클러스터 업그레이드
- 다중 AZ DB 클러스터 읽기 전용 복제본 업그레이드

마이너 버전 업그레이드

마이너 버전 업그레이드에는 기존 애플리케이션과 호환되는 변경 사항만 포함됩니다. 마이너 버전 업그레이드를 시작하면 Amazon RDS는 먼저 리더 DB 인스턴스를 한 번에 하나씩 업그레이드합니다. 그러면 리더 DB 인스턴스 중 하나가 새 라이터 DB 인스턴스로 전환됩니다. 그러면 Amazon RDS가 이전 라이터 인스턴스(현재는 리더 인스턴스)를 업그레이드합니다.

업그레이드 중의 가동 중지 시간이 리더 DB 인스턴스가 새 라이터 DB 인스턴스가 되는 데 걸리는 시간으로 제한됩니다. 이 가동 중지는 자동 장애 조치와 같은 역할을 합니다. 자세한 내용은 the section called “다중 AZ DB 클러스터 장애 조치” 단원을 참조하십시오. 다중 AZ DB 클러스터의 복제 지연이 가동 중지 시간에 영향을 줄 수 있습니다. 자세한 내용은 the section called “복제본 지연 시간 및 다중 AZ DB 클러스터” 단원을 참조하십시오.

RDS for PostgreSQL 다중 AZ DB 클러스터 읽기 전용 복제본의 경우 Amazon RDS는 클러스터 멤버 인스턴스를 한 번에 하나씩 업그레이드합니다. 리더 및 라이터 클러스터 역할은 업그레이드 중에 전환되지 않습니다. 따라서 Amazon RDS가 클러스터 라이터 인스턴스를 업그레이드하는 동안 DB 클러스터에 가동 중지가 발생할 수 있습니다.

> **Note**
>
> 다중 AZ DB 클러스터 마이너 버전 업그레이드의 가동 중지 시간은 일반적으로 35초입니다. RDS 프록시와 함께 사용하면 가동 중지 시간을 1초 이하로 더 줄일 수 있습니다. 자세한 내용은 Amazon RDS Proxy 단원을 참조하십시오. 또는 ProxySQL, PgBouncer 또는 AWS 고급 JDBC 래퍼 드라이버와 같은 오픈 소스 데이터베이스 프록시를 사용할 수 있습니다.

메이저 버전 업그레이드

메이저 버전 업그레이드로 기존 애플리케이션과 호환되지 않는 변경 사항이 도입될 수 있습니다.

RDS for PostgreSQL 다중 AZ DB 클러스터의 메이저 버전 업그레이드를 시작하면 Amazon RDS가 리더 및 라이터 인스턴스를 동시에 업그레이드합니다. 따라서 업그레이드가 완료될 때까지 DB 클러스터를 사용할 수 없습니다.

RDS for MySQL 다중 AZ DB 클러스터의 메이저 버전 업그레이드를 시작하면 Amazon RDS는 클러스터 멤버 인스턴스를 한 번에 하나씩 업그레이드하므로 낮은 엔진 버전에서 높은 엔진 버전으로 복제가 이루어집니다. 엔진 버전은 구문과 기능이 다를 수 있으므로 메이저 버전 업그레이드 중에 워크로드가 소스 및 대상 엔진 버전 둘 다와 호환되는지 확인하는 것이 중요합니다.

> **Note**
>
> 마이너 버전 업그레이드와 마찬가지로 RDS for MySQL 메이저 버전 업그레이드의 가동 중지 시간은 일반적으로 35초입니다. RDS 프록시와 함께 사용하면 다운타임을 1초 이하로 더 줄일 수 있습니다. 자세한 내용은 Amazon RDS Proxy 단원을 참조하십시오.

다중 AZ DB 클러스터 업그레이드

다중 AZ DB 클러스터의 엔진 버전을 업그레이드하는 프로세스는 DB 인스턴스 엔진 버전을 업그레이드하는 프로세스와 동일합니다. 지침은 the section called “엔진 버전 업그레이드” 단원을 참조하십시오. 유일한 차이점은 AWS Command Line Interface(AWS CLI)를 사용할 때 modify-db-cluster 명령을 사용하고 `--allow-major-version-upgrade` 파라미터와 함께 `--db-cluster-identifier` 파라미터를 지정한다는 것입니다.

메이저 및 마이너 버전 업그레이드에 대한 자세한 내용은 다음 DB 엔진 설명서를 참조하세요.

- the section called “PostgreSQL DB 엔진 업그레이드”

- the section called “MySQL DB 엔진 업그레이드”

다중 AZ DB 클러스터 읽기 전용 복제본 업그레이드

Amazon RDS는 다중 AZ DB 클러스터 읽기 전용 복제본을 자동으로 업그레이드하지 않습니다. 마이너 버전 업그레이드 시 먼저 모든 읽기 전용 복제본을 수동으로 업그레이드한 다음 클러스터를 업그레이드해야 합니다. 그렇지 않으면 업그레이드가 차단됩니다. 클러스터의 메이저 버전 업그레이드를 수행할 때 모든 읽기 전용 복제본의 복제 상태가 종료로 변경됩니다. 업그레이드가 완료된 다음, 읽기 전용 복제본을 삭제하고 재생성해야 합니다. 자세한 내용은 the section called “읽기 전용 복제본 모니터링” 단원을 참조하십시오.

Amazon RDS용 다중 AZ DB 클러스터 이름 변경

AWS Management Console, AWS CLI `modify-db-cluster` 명령 또는 Amazon RDS API `ModifyDBCluster` 작업을 사용하여 다중 AZ DB 클러스터의 이름을 바꿀 수 있습니다. 다중 AZ DB 클러스터의 이름 바꾸기는 상당한 영향을 미칠 수 있습니다. 다음은 다중 AZ DB 클러스터의 이름을 바꾸기 전에 고려해야 할 사항의 목록입니다.

- 다중 AZ DB 클러스터의 이름을 바꾸면 다중 AZ DB 클러스터의 클러스터 엔드포인트가 변경됩니다. 이러한 엔드포인트는 다중 AZ DB 클러스터에 할당한 이름을 포함하므로 변경됩니다. 이전 엔드포인트에서 새 엔드포인트로 트래픽을 리디렉션할 수 있습니다. 다중 AZ DB 클러스터 엔드포인트에 대한 자세한 내용은 Amazon RDS에서 다중 AZ DB 클러스터에 연결 섹션을 참조하세요.
- 다중 AZ DB 클러스터 이름을 변경하면 다중 AZ DB 클러스터에서 이전에 사용된 DNS 이름은 삭제되지만 캐시는 몇 분 더 남을 수도 있습니다. 이름이 바뀐 다중 AZ DB 클러스터의 새로운 DNS 이름은 약 2분 내에 적용됩니다. 이름이 바뀐 다중 AZ DB 클러스터를 사용하려면 새로운 이름이 적용될 때까지 기다려야 합니다.
- 클러스터 이름을 바꾸면 기존 다중 AZ DB 클러스터 이름을 사용할 수 없습니다.
- 다중 AZ DB 클러스터 이름을 재사용하면 DB 클러스터 이름에 연결된 지표와 이벤트가 유지됩니다.
- 다중 AZ DB 클러스터 태그는 이름 바꾸기에 관계없이 다중 AZ DB 클러스터에 유지됩니다.
- DB 클러스터 스냅샷은 바뀐 이름의 다중 AZ DB 클러스터에 유지됩니다.

> **Note**
>
> 다중 AZ DB 클러스터는 클라우드에서 실행되는 격리된 데이터베이스 환경입니다. 다중 AZ DB 클러스터는 여러 데이터베이스를 호스팅할 수 있습니다. 데이터베이스 이름 변경에 대한 자세한 내용은 DB 엔진 설명서를 참조하세요.

기존 다중 AZ DB 클러스터 교체를 위한 이름 바꾸기

가장 일반적인 다중 AZ DB 클러스터 이름 바꾸기 시나리오에는 DB 클러스터 스냅샷에서 데이터를 복구하거나 시점 복구(PITR)를 수행하는 것이 포함됩니다. 다중 AZ DB 클러스터의 이름을 바꾸면 다중 AZ DB 클러스터를 참조하는 애플리케이션 코드를 변경하지 않고도 다중 AZ DB 클러스터를 교체할 수 있습니다. 이러한 경우 다음 단계를 완료합니다.

1. 다중 AZ DB 클러스터로 전송되는 트래픽을 모두 차단합니다. 다중 AZ DB 클러스터의 데이터베이스에 액세스하는 트래픽을 리디렉션하거나 트래픽이 DB 클러스터의 데이터베이스에 액세스하지 못하도록 차단하는 또 다른 방법을 선택할 수 있습니다.
2. 기존 다중 AZ DB 클러스터의 이름을 바꿉니다.
3. DB 클러스터 스냅샷에서 복원하거나 특정 시점으로 복구하여 새로운 다중 AZ DB 클러스터를 생성합니다. 그런 다음 새 다중 AZ DB 클러스터에 이전 다중 AZ DB 클러스터의 이름을 지정합니다.

이전 다중 AZ DB 클러스터를 삭제할 때 이전 다중 AZ DB 클러스터의 원하지 않는 DB 클러스터 스냅샷까지 삭제되는 경우 사용자 본인에게 책임이 있습니다.

콘솔

다중 AZ DB 클러스터 이름을 바꾸는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
3. 이름을 바꾸려는 다중 AZ DB 클러스터를 선택합니다.
4. 수정을 선택합니다.
5. Settings(설정)에서 DB 클러스터 식별자에 새 이름을 입력합니다.
6. Continue(계속)를 선택합니다.
7. 변경 사항을 즉시 적용하려면 즉시 적용을 선택합니다. 일부의 경우 이 옵션을 선택하면 중단이 발생할 수 있습니다. 자세한 내용은 변경 사항 즉시 적용 단원을 참조하십시오.
8. 확인 페이지에서 변경 내용을 검토합니다. 변경 내용이 올바른 경우 클러스터 수정을 선택하여 변경 내용을 저장합니다.

   그렇지 않으면 Back(뒤로)을 선택하여 변경 내용을 편집하거나 Cancel(취소)을 선택하여 변경 내용을 취소합니다.

AWS CLI

다중 AZ DB 클러스터의 이름을 바꾸려면 AWS CLI 명령 modify-db-cluster를 사용합니다. 현재 `--db-cluster-identifier` 값 및 `--new-db-cluster-identifier` 파라미터를 다중 AZ DB 클러스터의 새 이름과 함께 제공합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier DBClusterIdentifier \
    --new-db-cluster-identifier NewDBClusterIdentifier
```

Windows의 경우:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier DBClusterIdentifier ^
    --new-db-cluster-identifier NewDBClusterIdentifier
```

RDS API

다중 AZ DB 클러스터 이름을 바꾸려면 Amazon RDS API 작업 ModifyDBCluster를 다음 파라미터와 함께 호출합니다.

- `DBClusterIdentifier` – DB 클러스터의 기존 이름입니다.
- `NewDBClusterIdentifier` – DB 클러스터의 새 이름입니다.

Amazon RDS에서 다중 AZ DB 클러스터 및 리더 DB 인스턴스 재부팅

일반적으로 유지 보수 목적으로 다중 AZ DB 클러스터를 재부팅해야 하는 경우가 있습니다. 예를 들어, 특정 내용을 수정하거나 DB 클러스터와 연결된 DB 클러스터 파라미터 그룹을 변경하는 경우 DB 클러스터를 재부팅해야 합니다. 이렇게 하면 변경 사항이 적용됩니다.

DB 클러스터에서 연결된 DB 클러스터 파라미터 그룹의 최신 변경 사항을 사용하지 않는 경우 AWS Management Console에서는 보류 중-재부팅(pending-reboot) 상태의 DB 클러스터 파라미터 그룹이 표시됩니다. 재시작 보류중 파라미터 그룹 상태로 인해 다음번 유지 관리 기간 중에 자동 재부팅이 되지는 않습니다. 최신 파라미터 변경 사항을 해당 DB 클러스터에 적용하려면 DB 클러스터를 수동으로 재부팅하면 됩니다. 파라미터 그룹에 대한 자세한 내용은 다중 AZ DB 클러스터용 파라미터 그룹 단원을 참조하세요.

DB 클러스터를 재부팅하면 데이터베이스 엔진 서비스가 재시작됩니다. DB 클러스터를 재부팅하면 DB 클러스터 상태가 재부팅(rebooting)으로 설정되면서 잠시 중단됩니다.

사용 가능(Available) 상태가 아닌 경우 DB 클러스터를 재부팅할 수 없습니다. 백업이 진행 중이거나 이전에 수정을 요청했거나 유지 관리 기간 작업 등 여러 원인으로 인해 데이터베이스를 사용할 수 없습니다.

DB 클러스터를 재부팅하는 데 걸리는 시간은 충돌 복구 프로세스, 재부팅 시 데이터베이스 활동 및 특정 DB 클러스터의 동작에 따라 달라집니다. 따라서 재부팅 시간을 단축하려면 재부팅 프로세스에서 데이터베이스 작업을 최소화하는 것이 좋습니다. 데이터베이스 작업을 줄이면 중간 트랜잭션의 롤백 작업이 줄어듭니다.

> **Important**
>
> 다중 AZ DB 클러스터는 장애 조치와 함께 재부팅을 지원하지 않습니다. 다중 AZ DB 클러스터의 라이터 인스턴스를 재부팅하면 해당 DB 클러스터의 리더 DB 인스턴스에는 영향을 주지 않으며, 장애 조치가 발생하지 않습니다. 리더 DB 인스턴스를 재부팅하면 장애 조치가 발생하지 않습니다. 다중 AZ DB 클러스터를 장애 조치하려면 콘솔에서 장애 조치(Failover)를 선택하거나, AWS CLI 명령 `failover-db-cluster`를 호출하거나, API 작업 `FailoverDBCluster`를 호출합니다.

콘솔

DB 클러스터를 재부팅하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스(Databases)를 선택한 다음 재부팅하려는 다중 AZ DB 클러스터를 선택합니다.
3. 작업에서 재부팅을 선택합니다.

   DB 클러스터 재부팅(Reboot DB cluster) 페이지가 표시됩니다.
4. DB 클러스터를 재부팅하려면 재부팅(Reboot)을 선택합니다.

   또는 [취소(Cancel)]를 선택합니다.

AWS CLI

AWS CLI를 사용하여 다중 AZ DB 클러스터를 재부팅하려면 reboot-db-cluster 명령을 호출합니다.

```
aws rds reboot-db-cluster --db-cluster-identifier mymultiazdbcluster
```

RDS API

Amazon RDS API를 사용하여 다중 AZ DB 클러스터를 재부팅하려면 RebootDBCluster 작업을 호출하면 됩니다.

Amazon RDS용 다중 AZ DB 클러스터 장애 조치

다중 AZ DB 클러스터의 라이터 DB 인스턴스에 계획되거나 계획되지 않은 중단이 발생하면 Amazon RDS는 자동으로 다른 가용 영역에 있는 리더 DB 인스턴스로 장애 조치합니다. 이를 통해 중단 시간을 최소화하면서 고가용성을 보장할 수 있습니다. 장애 조치는 하드웨어 장애, 네트워크 문제 또는 수동 요청 중에 발생할 수 있습니다. 이 주제에서는 실패 자동 감지, 장애 조치 중 이벤트 시퀀스, 읽기 및 쓰기 작업에 미치는 영향을 간략하게 설명합니다. 또한 장애 조치 시간을 모니터링하고 최소화하기 위한 모범 사례를 제공합니다.

장애 조치가 완료되는 데 걸리는 시간은 라이터 DB 인스턴스를 사용할 수 없게 된 데이터베이스 활동 및 기타 조건에 따라 달라집니다. 장애 조치에 소요되는 시간은 일반적으로 35초 아래입니다. 장애 조치는 두 리더 DB 인스턴스에 모두 장애가 발생한 라이터의 처리되지 않은 트랜잭션이 적용되면 완료됩니다. 장애 조치가 완료되면 RDS 콘솔에 새 가용 영역이 반영되는 데 시간이 더 걸릴 수 있습니다.

주제

- 자동 장애 조치
- 다중 AZ DB 클러스터 수동 장애 조치
- 다중 AZ DB 클러스터가 장애 조치되었는지 여부 확인
- DNS 이름 조회를 위한 JVM TTL 설정

자동 장애 조치

Amazon RDS는 자동으로 장애 조치를 취하여 관리자의 개입 없이 데이터베이스 작업을 신속하게 재개할 수 있도록 합니다. 장애 조치를 수행하려면 라이터 DB 인스턴스가 자동으로 리더 DB 인스턴스로 전환합니다.

다중 AZ DB 클러스터 수동 장애 조치

다중 AZ DB 클러스터를 수동으로 장애 조치하는 경우 RDS는 먼저 기본 DB 인스턴스를 종료합니다. 그러면 내부 모니터링 시스템이 기본 DB 인스턴스가 비정상임을 감지하고 읽기 가능한 복제 DB 인스턴스로 승격합니다. 장애 조치에 소요되는 시간은 일반적으로 35초 아래입니다.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 다중 AZ DB 클러스터를 수동으로 장애 조치할 수 있습니다.

콘솔

다중 AZ DB 클러스터를 수동으로 장애 조치하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
3. 장애 조치하려는 다중 AZ DB 클러스터를 선택합니다.
4. 작업(Actions)으로 장애 조치(Failover)를 선택합니다.

   DB 클러스터 장애 조치 페이지가 표시됩니다.
5. 장애 조치(Failover)를 선택하여 수동 장애 조치를 확인합니다.

AWS CLI

다중 AZ DB 클러스터를 수동으로 장애 조치하려면 AWS CLI 명령 failover-db-cluster를 사용하세요.

Example

```
aws rds failover-db-cluster --db-cluster-identifier mymultiazdbcluster
```

RDS API

다중 AZ DB 클러스터를 수동으로 장애 조치하려면 Amazon RDS API FailoverDBCluster를 호출하고 `DBClusterIdentifier`를 지정하면 됩니다.

다중 AZ DB 클러스터가 장애 조치되었는지 여부 확인

다음 단계에 따라 다중 AZ DB 클러스터가 장애 조치를 수행했는지 확인할 수 있습니다.

- 장애 조치가 시작되었음을 이메일 또는 SMS로 사용자에게 알리도록 DB 이벤트 구독을 설정합니다. 이벤트에 대한 자세한 내용은 Amazon RDS 이벤트 알림 작업 단원을 참조하세요.
- Amazon RDS 콘솔 또는 API 작업을 사용하여 DB 이벤트를 확인합니다.
- Amazon RDS 콘솔, AWS CLI 및 RDS API를 사용하여 다중 AZ DB 클러스터의 현재 상태를 확인합니다.

장애 조치, 복구 시간 절감 및 기타 Amazon RDS 모범 사례에 대한 자세한 내용은 Amazon RDS의 모범 사례 단원을 참조하세요.

DNS 이름 조회를 위한 JVM TTL 설정

장애 조치 메커니즘은 리더 DB 인스턴스로 연결되도록 DB 인스턴스의 도메인 이름 시스템(DNS) 레코드를 자동으로 변경합니다. 그 결과 DB 인스턴스의 기존 연결을 모두 재설정해야 합니다. Java 가상 머신(JVM) 환경에서 Java DNS 캐싱 메커니즘이 작동하는 방식 때문에 JVM 설정을 다시 구성해야 할 수 있습니다.

JVM은 DNS 이름 조회를 캐시합니다. JVM은 호스트 이름을 IP 주소로 확인하는 경우 유지 시간(TTL)이라고 하는 지정된 기간 동안 IP 주소를 캐싱합니다.

AWS 리소스는 간헐적으로 변경되는 DNS 이름 항목을 사용하므로 TTL 값을 60초 이하로 하여 JVM을 구성하는 것이 좋습니다. 이렇게 하면 리소스의 IP 주소가 변경될 때 애플리케이션이 DNS를 다시 쿼리하여 리소스의 새 IP 주소를 수신하고 사용할 수 있습니다.

일부 Java 구성에서는 JVM이 재시작될 때까지 DNS 항목을 새로 고치지 않도록 JVM 기본 TTL이 설정됩니다. 따라서 애플리케이션이 여전히 실행되는 동안 AWS 리소스의 IP 주소가 변경되는 경우 JVM을 수동으로 재시작하여 캐시된 IP 정보가 새로 고쳐질 때까지는 해당 리소스를 사용할 수 없습니다. 이 경우 캐시된 IP 정보를 정기적으로 새로 고치도록 JVM의 TTL을 설정하는 것이 중요합니다.

> **Note**
>
> 기본 TTL은 JVM 버전과 보안 관리자 설치 여부에 따라 다를 수 있습니다. 대부분의 JVM은 60초 미만의 기본 TTL을 제공합니다. 이러한 JVM을 사용 중이며 보안 관리자는 사용하지 않는 경우 이 주제의 나머지 내용을 무시해도 좋습니다. Oracle의 보안 관리자에 대한 자세한 내용은 Oracle 설명서의 The Security Manager를 참조하십시오.

JVM의 TTL을 수정하려면 `networkaddress.cache.ttl` 속성 값을 설정합니다. 필요에 따라 다음 방법 중 하나를 사용합니다.

- JVM을 사용하는 모든 애플리케이션에 대해 전역적으로 속성 값을 설정하려면 `networkaddress.cache.ttl` 파일에서 `$JAVA_HOME/jre/lib/security/java.security`을 설정합니다.

  ```
  networkaddress.cache.ttl=60
  ```

- 애플리케이션에만 로컬로 속성을 설정하려면 네트워크 연결이 설정되기 전에 애플리케이션의 초기화 코드에서 `networkaddress.cache.ttl`을 설정합니다.

```
java.security.Security.setProperty("networkaddress.cache.ttl" , "60");
```

Amazon RDS의 다중 AZ DB 클러스터에 PostgreSQL 논리적 복제 설정

PostgreSQL의 논리적 복제 기능을 다중 AZ DB 클러스터와 함께 사용하면 전체 데이터베이스 인스턴스가 아닌 개별 테이블을 복제하고 동기화할 수 있습니다. 논리적 복제에서는 게시 및 구독 모델을 사용하여 원본에서 한 명 이상의 수신자에게 변경 내용을 복제합니다. 이 작업은 PostgreSQL 미리 쓰기 로그(WAL)의 변경 레코드를 사용하여 작동합니다. 자세한 내용은 the section called “논리적 복제 수행” 단원을 참조하십시오.

다중 AZ DB 클러스터의 라이터 DB 인스턴스에 새 논리적 복제 슬롯을 만들면 해당 슬롯이 클러스터의 각 리더 DB 인스턴스에 비동기적으로 복사됩니다. 리더 DB 인스턴스의 슬롯은 라이터 DB 인스턴스의 슬롯과 지속적으로 동기화됩니다.

논리적 복제는 RDS for PostgreSQL 버전 14.8-R2 이상 및 15.3-R2 이상을 실행하는 다중 AZ DB 클러스터에서 지원됩니다.

> **Note**
>
> 기본 PostgreSQL 논리적 복제 기능 외에 RDS for PostgreSQL을 실행하는 다중 AZ DB 클러스터도 `pglogical` 확장을 지원합니다.

PostgreSQL 논리적 복제에 대한 자세한 내용은 PostgreSQL 설명서의 논리적 복제를 참조하세요.

주제

- 사전 조건
- 논리적 복제 설정
- 제한 및 권장 사항

사전 조건

다중 AZ DB 클러스터를 위한 PostgreSQL 논리적 복제를 구성하려면 다음 사전 조건을 충족해야 합니다.

- 사용자 계정이 `rds_superuser` 그룹의 구성원이어야 하며 `rds_superuser` 권한을 보유해야 합니다. 자세한 내용은 the section called “PostgreSQL 역할 및 권한 이해” 단원을 참조하십시오.
- 다중 AZ DB 클러스터를 사용자 지정 DB 클러스터 파라미터 그룹과 연결해야 다음 절차에 설명된 파라미터 값을 구성할 수 있습니다. 자세한 내용은 the section called “DB 클러스터 파라미터 그룹” 단원을 참조하십시오.

논리적 복제 설정

다중 AZ DB 클러스터의 논리적 복제를 설정하려면 연결된 DB 클러스터 파라미터 그룹 내에서 특정 파라미터를 활성화한 후 논리적 복제 슬롯을 생성해야 합니다.

> **Note**
>
> PostgreSQL 버전 16부터는 다중 AZ DB 클러스터의 리더 DB 인스턴스를 논리적 복제에 사용할 수 있습니다.

RDS for PostgreSQL 다중 AZ DB 클러스터의 논리적 복제를 설정하려면

1. RDS for PostgreSQL 다중 AZ DB 클러스터와 연결된 사용자 지정 DB 클러스터 파라미터 그룹을 엽니다.
2. 파라미터 검색 필드에서 `rds.logical_replication` 정적 파라미터를 찾고 값을 1로 설정합니다. 이 파라미터 변경은 WAL 생성을 증가시킬 수 있으므로, 논리적 슬롯을 사용하는 경우에만 활성화해야 합니다.
3. 이러한 변경의 하나로 다음과 같은 DB 클러스터 파라미터를 구성합니다.

   - `max_wal_senders`
   - `max_replication_slots`
   - `max_connections`

   예상 사용량에 따라 다음 파라미터의 값을 변경해야 할 수도 있습니다. 하지만 대부분의 경우 기본값이면 충분합니다.

   - `max_logical_replication_workers`
   - `max_sync_workers_per_subscription`
4. 파라미터 값을 적용하려면 다중 AZ DB 클러스터를 재부팅합니다. 지침은 the section called “다중 AZ DB 클러스터 재부팅” 단원을 참조하십시오.
5. the section called “논리적 복제 슬롯 작업”에 설명된 대로 다중 AZ DB 클러스터의 라이터 DB 인스턴스에 논리적 복제 슬롯을 생성합니다. 이 프로세스에서는 디코딩 플러그인을 지정해야 합니다. 현재 RDS for PostgreSQL은 PostgreSQL과 함께 전송되는 `test_decoding`, `wal2json`, `pgoutput` 플러그인을 지원합니다.

   슬롯은 클러스터의 각 리더 DB 인스턴스에 비동기적으로 복사됩니다.

6. 다중 AZ DB 클러스터의 모든 리더 DB 인스턴스에 있는 슬롯의 상태를 확인합니다. 이렇게 하려면 모든 리더 DB 인스턴스의 `pg_replication_slots` 보기를 검사하고 애플리케이션이 논리적 변경 사항을 적극적으로 소비하는 동안 `confirmed_flush_lsn` 상태가 진행되고 있는지 확인해야 합니다.

   다음 명령은 리더 DB 인스턴스의 복제 상태를 검사하는 방법을 보여줍니다.

```
% psql -h test-postgres-instance-2.abcdefabcdef.us-west-2.rds.amazonaws.com

postgres=> select slot_name, slot_type, confirmed_flush_lsn from
 pg_replication_slots;
  slot_name   | slot_type | confirmed_flush_lsn
--------------+-----------+---------------------
 logical_slot | logical   | 32/D0001700
(1 row)

postgres=> select slot_name, slot_type, confirmed_flush_lsn from
 pg_replication_slots;
  slot_name   | slot_type | confirmed_flush_lsn
--------------+-----------+---------------------
 logical_slot | logical   | 32/D8003628
(1 row)

% psql -h test-postgres-instance-3.abcdefabcdef.us-west-2.rds.amazonaws.com

postgres=> select slot_name, slot_type, confirmed_flush_lsn from
 pg_replication_slots;
  slot_name   | slot_type | confirmed_flush_lsn
--------------+-----------+---------------------
 logical_slot | logical   | 32/D0001700
(1 row)

postgres=> select slot_name, slot_type, confirmed_flush_lsn from
 pg_replication_slots;
  slot_name   | slot_type | confirmed_flush_lsn
--------------+-----------+---------------------
 logical_slot | logical   | 32/D8003628
(1 row)
```

복제 작업을 완료한 후에는 복제 프로세스를 중지하고 복제 슬롯을 삭제한 다음 논리적 복제를 비활성화합니다. 논리적 복제를 비활성화하려면 DB 클러스터 파라미터 그룹을 수정하

고 rds.logical_replication 값을 다시 0으로 설정합니다. 파라미터 변경 사항을 적용하려면 클러스터를 재부팅하세요.

제한 및 권장 사항

PostgreSQL 버전 16을 실행하는 다중 AZ DB 클러스터에서 논리적 복제를 사용하는 경우 다음과 같은 제한 및 권장 사항이 적용됩니다.

- 라이터 DB 인스턴스만 사용하여 논리적 복제 슬롯을 만들거나 삭제할 수 있습니다. 예를 들어, CREATE SUBSCRIPTION 명령은 호스트 연결 문자열에서 클러스터 라이터 엔드포인트를 사용해야 합니다.
- 테이블 동기화 또는 재동기화 중에는 클러스터 라이터 엔드포인트를 사용해야 합니다. 예를 들어, 다음 명령을 사용하여 새로 추가된 테이블을 다시 동기화할 수 있습니다.

```
Postgres=>ALTER SUBSCRIPTION subscription-name CONNECTION host=writer-endpoint
Postgres=>ALTER SUBSCRIPTION subscription-name REFRESH PUBLICATION
```

- 논리적 복제에 리더 DB 인스턴스를 사용하기 전에 테이블 동기화가 완료될 때까지 기다려야 합니다. pg_subscription_rel 카탈로그 테이블을 사용하여 테이블 동기화를 모니터링할 수 있습니다. srsubstate 열이 준비 상태(r)로 설정되면 테이블 동기화가 완료됩니다.
- 초기 테이블 동기화가 완료되면 논리적 복제 연결에 인스턴스 엔드포인트를 사용하는 것이 좋습니다. 다음 명령은 리더 DB 인스턴스 중 하나로 복제를 오프로드하여 라이터 DB 인스턴스의 부하를 줄여줍니다.

```
Postgres=>ALTER SUBSCRIPTION subscription-name CONNECTION host=reader-instance-
endpoint
```

  한 번에 둘 이상의 DB 인스턴스에서 동일한 슬롯을 사용할 수 없습니다. 둘 이상의 애플리케이션이 클러스터의 서로 다른 DB 인스턴스에서 논리적 변경 사항을 복제하는 경우 클러스터 장애 조치 또는 네트워크 문제로 인해 일부 변경 사항이 손실될 수 있습니다. 이러한 상황에서는 호스트 연결 문자열의 논리적 복제에 인스턴스 엔드포인트를 사용할 수 있습니다. 동일한 구성을 사용하는 다른 애플리케이션은 다음과 같은 오류 메시지를 표시합니다.

```
replication slot slot_name is already active for PID x providing immediate feedback.
```

- pglogical 확장을 사용하는 동안에는 클러스터 라이터 엔드포인트만 사용할 수 있습니다. 확장에는 테이블 동기화 중에 사용되지 않는 논리적 복제 슬롯을 생성할 수 있는 알려진 제한 사항이 있습

니다. 오래된 복제 슬롯은 미리 쓰기 로그(WAL) 파일을 예약하므로, 디스크 공간 문제가 발생할 수 있습니다.

Amazon RDS의 다중 AZ DB 클러스터 읽기 전용 복제본 사용

DB 클러스터 읽기 전용 복제본은 소스 DB 인스턴스에서 생성하는 특수한 유형의 클러스터입니다. 읽기 전용 복제본 생성 후 기본 DB 인스턴스에 적용되는 업데이트는 다중 AZ DB 클러스터 읽기 전용 복제본에 비동기식으로 복사됩니다. 애플리케이션에서 읽기 전용 복제본으로 읽기 쿼리를 라우팅하여 기본 DB 인스턴스의 로드를 줄일 수 있습니다. 읽기 전용 복제본을 사용하면 읽기 중심의 데이터베이스 워크로드에 대한 단일 DB 인스턴스의 용량 제한에서 벗어나 탄력적으로 늘릴 수 있습니다.

다중 AZ DB 클러스터에서 하나 이상의 DB 인스턴스 전용 복제본을 생성할 수도 있습니다. DB 인스턴스 읽기 전용 복제본을 사용하면 초과 읽기 트래픽을 읽기 전용 복제본으로 전달하여 소스 다중 AZ DB 클러스터의 컴퓨팅 또는 I/O 용량을 초과하여 확장할 수 있습니다. 현재는 기존 다중 AZ DB 클러스터에서 읽기 전용 복제본을 생성할 수 없습니다.

읽기 전용 복제본을 사용하여 다중 AZ DB 클러스터로 마이그레이션하거나 다중 AZ DB 클러스터에서 DB 인스턴스 읽기 전용 복제본을 생성할 때 특정 사용 사례 및 성능 요구 사항을 고려하세요.

읽기 전용 복제본을 사용하여 다중 AZ DB 클러스터로 마이그레이션

이 접근 방식은 가동 중지 시간을 최소화하면서 데이터베이스의 가용성과 내구성을 개선해야 하는 경우에 적합합니다. 읽기 전용 복제본을 사용하여 다중 AZ DB 클러스터로 전환하면 지속적인 작업과 데이터 일관성을 보장할 수 있습니다. 이 방법은 가용성을 유지하고 라이브 워크로드에 미치는 영향을 줄이는 것이 중요한 프로덕션 환경에 특히 유용합니다.

다중 AZ DB 클러스터에서 DB 인스턴스 읽기 전용 복제본 생성

이 방법은 읽기 작업을 확장하거나 기본 데이터베이스 인스턴스에서 읽기 트래픽을 오프로드하려는 경우에 적합합니다. 기존 다중 AZ DB 클러스터에서 읽기 전용 복제본을 생성하면 프라이머리 인스턴스의 안정성에 영향을 주지 않고 읽기 작업이 많은 워크로드를 배포하고 성능을 개선할 수 있습니다.

올바른 접근 방식을 선택하는 것은 고가용성 및 내구성을 보장하는 것이 우선순위인지 아니면 읽기 성능을 확장하는 것이 우선순위인지에 따라 달라집니다. 워크로드 특성 및 운영 요구 사항을 평가하여 정보에 입각한 결정을 내립니다.

주제

- 읽기 전용 복제본을 사용하여 다중 AZ DB 클러스터로 마이그레이션
- 다중 AZ DB 클러스터에서 DB 인스턴스 읽기 전용 복제본 생성

읽기 전용 복제본을 사용하여 다중 AZ DB 클러스터로 마이그레이션

단일 AZ 배포 또는 다중 AZ DB 인스턴스 배포를 다중 AZ DB 클러스터 배포로 마이그레이션하여 가동 중지 시간을 줄이려면 다중 AZ DB 클러스터 읽기 전용 복제본을 생성할 수 있습니다. 소스의 경우 단일 AZ 배포의 DB 인스턴스 또는 다중 AZ DB 인스턴스 배포의 기본 DB 인스턴스를 지정합니다. DB 인스턴스는 다중 AZ DB 클러스터로의 마이그레이션 중에 쓰기 트랜잭션을 처리할 수 있습니다.

다중 AZ DB 클러스터 읽기 전용 복제본을 생성하기 전에 다음 사항을 고려하세요.

- 소스 DB 인스턴스는 다중 AZ DB 클러스터를 지원하는 버전에 있어야 합니다. 자세한 내용은 Amazon RDS에서 다중 AZ DB 클러스터를 지원하는 리전 및 DB 엔진 단원을 참조하십시오.
- 다중 AZ DB 클러스터 읽기 전용 복제본은 소스와 동일한 메이저 버전 또는 같거나 더 높은 마이너 버전에 있어야 합니다.
- 백업 보존 기간을 0이 아닌 다른 값으로 설정하여 소스 DB 인스턴스의 자동 백업을 켜야 합니다.
- 소스 DB 인스턴스의 할당된 스토리지는 100GiB 이상이어야 합니다.
- RDS for MySQL의 경우 소스 DB 인스턴스에 대해 `gtid-mode` 및 `enforce_gtid_consistency` 파라미터를 모두 ON으로 설정해야 합니다. 기본 파라미터 그룹이 아니라 사용자 지정 파라미터 그룹을 사용해야 합니다. 자세한 내용은 the section called “DB 파라미터 그룹” 단원을 참조하십시오.
- 활성 상태의 장기 실행 트랜잭션은 읽기 전용 복제본 생성 프로세스를 늦출 수 있습니다. 읽기 전용 복제본을 생성하기 전에 장기 실행 트랜잭션이 완료되기를 기다리는 것이 좋습니다.
- 다중 AZ DB 클러스터 읽기 전용 복제본의 소스 DB 인스턴스를 삭제하는 경우 읽기 전용 복제본이 독립된 다중 AZ DB 클러스터로 승격됩니다.

다중 AZ DB 클러스터 읽기 전용 복제본 생성 및 승격

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 다중 AZ DB 클러스터를 생성하고 승격할 수 있습니다.

> **Note**
>
> 모든 읽기 전용 복제본은 소스 DB 인스턴스와 동일한 Amazon VPC 기반의 Virtual Private Cloud(VPC)에 생성하는 것이 좋습니다.
> 소스 DB 인스턴스와 다른 VPC에 읽기 전용 복제본을 생성하는 경우 Classless Inter-Domain Routing(CIDR) 범위가 복제본과 RDS 시스템 간에 겹칠 수 있습니다. CIDR이 겹치면 복제본이 불안정해져 복제본에 연결하는 애플리케이션에 부정적인 영향을 줄 수 있습니다. 읽기 전용

복제본을 생성할 때 오류가 발생하면 다른 대상 DB 서브넷 그룹을 선택합니다. 자세한 내용은 VPC에서 DB 인스턴스를 사용한 작업 단원을 참조하십시오.

콘솔

읽기 전용 복제본을 사용하여 단일 AZ 배포 또는 다중 AZ DB 인스턴스 배포를 다중 AZ DB 클러스터로 마이그레이션하려면 AWS Management Console을 사용하여 다음 단계를 완료합니다.

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 다중 AZ DB 클러스터 읽기 전용 복제본을 생성합니다.
   a. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
   b. 읽기 전용 복제본의 소스로 사용할 DB 인스턴스를 선택합니다.
   c. 작업에서 읽기 전용 복제본 만들기를 선택합니다.
   d. Availability and durability(가용성 및 내구성)에서 Multi-AZ DB cluster(다중 AZ DB 클러스터)를 선택합니다.
   e. DB 인스턴스 식별자에 읽기 전용 복제본의 이름을 입력합니다.
   f. 나머지 섹션에서 DB 클러스터 설정을 지정합니다. 설정에 대한 자세한 내용은 다중 AZ DB 클러스터를 생성하기 위한 설정 섹션을 참조하세요.
   g. 읽기 전용 복제본 생성을 선택합니다.
3. 준비가 되면 읽기 복제본을 독립형 다중 AZ DB 클러스터로 승격합니다.
   a. 어떤 트랜잭션도 소스 DB 인스턴스에 쓰지 못하도록 한 후 읽기 전용 복제본의 업데이트가 모두 끝날 때까지 기다립니다.

      읽기 전용 복제본에서 수행된 데이터베이스 업데이트는 기본 DB 인스턴스의 업데이트가 끝난 후에 이어집니다. 이 복제 지연은 서로 크게 다를 수 있습니다. `ReplicaLag` 지표를 사용하여 읽기 전용 복제본의 업데이트가 모두 완료되는 시간을 측정합니다. 복제 지연에 대한 자세한 내용은 읽기 전용 복제본 모니터링 섹션을 참조하세요.
   b. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
   c. Amazon RDS 콘솔에서 데이터베이스를 선택합니다.

      데이터베이스 창이 표시됩니다. 각 읽기 전용 복제본은 역할 열에 복제본이라고 표시됩니다.

d. 승격하려는 다중 AZ DB 클러스터 읽기 전용 복제본을 선택합니다.

e. Actions(작업)에서 Promote(승격)를 선택합니다.

f. Promote read replica(읽기 전용 복제본 승격) 페이지에서 새롭게 승격된 다중 AZ DB 클러스터의 백업 보존 기간과 백업 기간을 입력합니다.

g. 원하는 대로 설정되었으면 Promote read replica(읽기 전용 복제본 승격)를 선택합니다.

h. 승격된 다중 AZ DB 클러스터의 상태가 `Available`이 될 때까지 기다립니다.

i. 애플리케이션이 승격된 다중 AZ DB 클러스터를 사용하도록 설정합니다.

더 이상 필요하지 않은 경우 선택적으로 단일 AZ 배포 또는 다중 AZ DB 인스턴스 배포를 삭제하세요. 지침은 DB 인스턴스 삭제 단원을 참조하십시오.

AWS CLI

읽기 전용 복제본을 사용하여 단일 AZ 배포 또는 다중 AZ DB 인스턴스 배포를 다중 AZ DB 클러스터로 마이그레이션하려면 AWS CLI을 사용하여 다음 단계를 완료합니다.

1. 다중 AZ DB 클러스터 읽기 전용 복제본을 생성합니다.

   소스 DB 인스턴스에서 읽기 복제본을 생성하려면 AWS CLI 명령 `create-db-cluster`를 사용합니다. `--replication-source-identifier`의 경우 소스 DB 인스턴스의 Amazon 리소스 이름(ARN)을 지정합니다.

   대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-cluster \
  --db-cluster-identifier mymultiazdbcluster \
  --replication-source-identifier arn:aws:rds:us-
east-2:123456789012:db:mydbinstance
  --engine postgres \
  --db-cluster-instance-class db.m5d.large \
  --storage-type io1 \
  --iops 1000 \
  --db-subnet-group-name defaultvpc \
  --backup-retention-period 1
```

   Windows의 경우:

```
aws rds create-db-cluster ^
```

```
  --db-cluster-identifier mymultiazdbcluster ^
  --replication-source-identifier arn:aws:rds:us-
east-2:123456789012:db:mydbinstance
  --engine postgres ^
  --db-cluster-instance-class db.m5d.large ^
  --storage-type io1 ^
  --iops 1000 ^
  --db-subnet-group-name defaultvpc ^
  --backup-retention-period 1
```

2. 어떤 트랜잭션도 소스 DB 인스턴스에 쓰지 못하도록 한 후 읽기 전용 복제본의 업데이트가 모두 끝날 때까지 기다립니다.

   읽기 전용 복제본에서 수행된 데이터베이스 업데이트는 기본 DB 인스턴스의 업데이트가 끝난 후에 이어집니다. 이 복제 지연은 서로 크게 다를 수 있습니다. `Replica Lag` 지표를 사용하여 읽기 전용 복제본의 업데이트가 모두 완료되는 시간을 측정합니다. 복제 지연에 대한 자세한 내용은 읽기 전용 복제본 모니터링 섹션을 참조하세요.

3. 준비가 되면 읽기 복제본을 독립형 다중 AZ DB 클러스터로 승격합니다.

   다중 AZ DB 클러스터 읽기 전용 복제본을 승격하려면 AWS CLI 명령 `promote-read-replica-db-cluster`를 사용합니다. `--db-cluster-identifier`의 경우 다중 AZ DB 클러스터 읽기 전용 복제의 식별자를 지정합니다.

   ```
   aws rds promote-read-replica-db-cluster --db-cluster-identifier mymultiazdbcluster
   ```

4. 승격된 다중 AZ DB 클러스터의 상태가 `Available`이 될 때까지 기다립니다.
5. 애플리케이션이 승격된 다중 AZ DB 클러스터를 사용하도록 설정합니다.

더 이상 필요하지 않은 경우 선택적으로 단일 AZ 배포 또는 다중 AZ DB 인스턴스 배포를 삭제하세요. 지침은 DB 인스턴스 삭제 단원을 참조하십시오.

RDS API

읽기 전용 복제본을 사용하여 단일 AZ 배포 또는 다중 AZ DB 인스턴스 배포를 다중 AZ DB 클러스터로 마이그레이션하려면 RDS API를 사용하여 다음 단계를 완료합니다.

1. 다중 AZ DB 클러스터 읽기 전용 복제본을 생성합니다.

   다중 AZ DB 클러스터 읽기 전용 복제본을 생성하려면 필요한 파라미터 `DBClusterIdentifier`와 함께 `CreateDBCluster` 작업을 사용합니다.

ReplicationSourceIdentifier의 경우 소스 DB 인스턴스의 Amazon 리소스 이름(ARN)을 지정합니다.

2. 어떤 트랜잭션도 소스 DB 인스턴스에 쓰지 못하도록 한 후 읽기 전용 복제본의 업데이트가 모두 끝날 때까지 기다립니다.

   읽기 전용 복제본에서 수행된 데이터베이스 업데이트는 기본 DB 인스턴스의 업데이트가 끝난 후에 이어집니다. 이 복제 지연은 서로 크게 다를 수 있습니다. Replica Lag 지표를 사용하여 읽기 전용 복제본의 업데이트가 모두 완료되는 시간을 측정합니다. 복제 지연에 대한 자세한 내용은 읽기 전용 복제본 모니터링 섹션을 참조하세요.

3. 준비가 되면 읽기 복제본을 독립형 다중 AZ DB 클러스터로 승격합니다.

   다중 AZ DB 클러스터 읽기 전용 복제본을 승격하려면 필요한 파라미터 DBClusterIdentifier와 함께 PromoteReadReplicaDBCluster 작업을 사용합니다. 다중 AZ DB 클러스터 읽기 전용 복제의 식별자를 지정합니다.

4. 승격된 다중 AZ DB 클러스터의 상태가 Available이 될 때까지 기다립니다.

5. 애플리케이션이 승격된 다중 AZ DB 클러스터를 사용하도록 설정합니다.

더 이상 필요하지 않은 경우 선택적으로 단일 AZ 배포 또는 다중 AZ DB 인스턴스 배포를 삭제하세요. 지침은 DB 인스턴스 삭제 단원을 참조하십시오.

다중 AZ DB 클러스터 읽기 전용 복제본 생성 시 제한

단일 AZ 배포 또는 다중 AZ DB 인스턴스 배포에서 다중 AZ DB 클러스터 읽기 전용 복제본을 생성하는 데는 다음과 같은 제한이 적용됩니다.

- 소스 DB 인스턴스를 소유한 AWS 계정와 다른 AWS 계정에서는 다중 AZ DB 클러스터 읽기 전용 복제본을 생성할 수 없습니다.
- 소스 DB 인스턴스와 다른 AWS 리전에는 다중 AZ DB 클러스터 읽기 전용 복제본을 생성할 수 없습니다.
- 다중 AZ DB 클러스터 읽기 전용 복제본을 특정 시점으로 복구할 수 없습니다.
- 스토리지 암호화는 소스 DB 인스턴스와 다중 AZ DB 클러스터에서 동일한 설정을 가져야 합니다.
- 소스 DB 인스턴스가 암호화된 경우 다중 AZ DB 클러스터 읽기 전용 복제본은 동일한 KMS 키를 사용하여 암호화되어야 합니다.
- 소스 DB 인스턴스가 범용 SSD(gp3) 스토리지를 사용하고 할당된 스토리지가 400GiB 미만인 경우 다중 AZ DB 클러스터 읽기 전용 복제본의 프로비저닝된 IOPS를 수정할 수 없습니다.

- 소스 DB 인스턴스에서 마이너 버전 업그레이드를 수행하려면 먼저 다중 AZ DB 클러스터 읽기 전용 복제본에서 마이너 버전 업그레이드를 수행해야 합니다.
- PostgreSQL 다중 AZ DB 클러스터 읽기 전용 복제본용 RDS에서 마이너 버전 업그레이드를 수행할 때, 업그레이드 후 리더 DB 인스턴스가 라이터 DB 인스턴스로 전환되지 않습니다. 따라서 Amazon RDS가 라이터 인스턴스를 업그레이드하는 동안 DB 클러스터에 가동 중지가 발생할 수 있습니다.
- 다중 AZ DB 클러스터 읽기 전용 복제본에서 메이저 버전 업그레이드를 수행할 수 없습니다.
- 다중 AZ DB 클러스터 읽기 전용 복제본의 소스 DB 인스턴스에서 메이저 버전 업그레이드를 수행할 수 있지만, 읽기 전용 복제본으로의 복제가 중지되고 다시 시작할 수 없습니다.
- 다중 AZ DB 클러스터 읽기 전용 복제본은 읽기 전용 복제본 계단식 읽기를 지원하지 않습니다.
- RDS for PostgreSQL의 경우 다중 AZ DB 클러스터 읽기 전용 복제본은 장애 조치할 수 없습니다.

다중 AZ DB 클러스터에서 DB 인스턴스 읽기 전용 복제본 생성

읽기 중심의 데이터베이스 워크로드를 위해 클러스터의 컴퓨팅 또는 I/O 용량을 초과하여 확장하려는 경우, 다중 AZ DB 클러스터에서 DB 인스턴스 읽기 전용 복제본을 생성할 수 있습니다. 이 과도한 읽기 트래픽을 하나 이상의 DB 인스턴스 읽기 전용 복제본으로 이동할 수 있습니다. 읽기 전용 복제본을 사용하여 다중 AZ DB 클러스터에서 DB 인스턴스로 마이그레션할 수도 있습니다.

읽기 전용 복제본을 만들려면 다중 AZ DB 클러스터를 복제 소스로 지정합니다. 다중 AZ DB 클러스터의 리더 인스턴스 중 하나는 항상 복제 소스이며 라이터 인스턴스가 아닙니다. 이 조건은 장애 조치가 발생한 경우에도 복제본이 항상 소스 클러스터와 동기화되도록 보장합니다.

주제

- 리더 DB 인스턴스와 DB 인스턴스 읽기 전용 복제본 비교
- 고려 사항
- DB 인스턴스 전용 복제본 생성
- DB 인스턴스 읽기 전용 복제본 승격
- 다중 AZ DB 클러스터에서 DB 인스턴스 읽기 전용 복제본 생성 시 제한

리더 DB 인스턴스와 DB 인스턴스 읽기 전용 복제본 비교

다중 AZ DB 클러스터의 DB 인스턴스 읽기 전용 복제본과 다중 AZ DB 클러스터의 리더 DB 인스턴스의 차이점은 다음과 같습니다.

- 리더 DB 인스턴스는 자동 장애 조치 대상으로 작동하지만, DB 인스턴스 읽기 전용 복제본은 그렇지 않습니다.

- 리더 DB 인스턴스는 변경 사항을 커밋하기 전에 라이터 DB 인스턴스에서 변경 사항을 승인해야 합니다. 그러나 DB 인스턴스 전용 복제본의 업데이트는 승인 없이도 읽기 전용 복제본에 비동기식으로 복사됩니다.
- 리더 DB 인스턴스는 다중 AZ DB 클러스터의 라이터 DB 인스턴스와 동일한 인스턴스 클래스, 스토리지 유형, 엔진 버전을 공유합니다. 하지만 DB 인스턴스 읽기 전용 복제본이 반드시 소스 클러스터와 동일한 구성을 공유할 필요는 없습니다.
- DB 인스턴스 읽기 전용 복제본을 독립 실행형 DB 인스턴스로 승격시킬 수 있습니다. 다중 AZ DB 클러스터의 리더 DB 인스턴스는 독립 실행형 인스턴스로 승격시킬 수 없습니다.
- 리더 엔드포인트는 다중 AZ DB 클러스터의 리더 DB 인스턴스에 대한 요청만 라우팅합니다. 이는 요청을 DB 인스턴스 읽기 전용 복제본으로 라우팅하지 않습니다.

리더 및 라이터 DB 인스턴스에 대한 자세한 내용은 the section called “다중 AZ DB 클러스터 아키텍처” 섹션을 참조하세요.

고려 사항

다중 AZ DB 클러스터에서 DB 인스턴스 읽기 전용 복제본을 생성하기 전에 다음 사항을 고려하세요.

- DB 인스턴스를 생성할 경우, 이는 소스 클러스터와 동일한 메이저 버전 또는 같거나 더 높은 마이너 버전에 있어야 합니다. 읽기 전용 복제본을 생성한 후 선택에 따라, 읽기 전용 복제본을 소스 클러스터보다 상위 마이너 버전으로 업그레이드할 수 있습니다.
- DB 인스턴스 읽기 전용 복제본을 생성할 경우, 할당된 스토리지는 소스 다중 AZ DB 클러스터의 할당된 스토리지와 동일해야 합니다. 읽기 전용 복제본을 생성한 후 할당된 스토리지를 변경할 수 있습니다.
- RDS for MySQL의 경우 소스 다중 AZ DB 클러스터의 `gtid-mode` 파라미터를 `ON`으로 설정해야 합니다. 자세한 내용은 the section called “DB 클러스터 파라미터 그룹” 단원을 참조하십시오.
- 활성 상태의 장기 실행 트랜잭션은 읽기 전용 복제본 생성 프로세스를 늦출 수 있습니다. 읽기 전용 복제본을 생성하기 전에 장기 실행 트랜잭션이 완료되기를 기다리는 것이 좋습니다.
- DB 인스턴스 전용 복제본에 대한 다중 AZ DB 클러스터를 삭제할 경우, 쓰기 작업을 수행 중인 모든 읽기 전용 복제본은 독립 실행형 DB 인스턴스로 승격됩니다.

DB 인스턴스 전용 복제본 생성

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 다중 AZ DB 클러스터에서 DB 인스턴스 읽기 전용 복제본을 생성할 수 있습니다.

> **Note**
>
> 모든 읽기 전용 복제본은 소스 다중 AZ DB 클러스터와 동일한 Amazon VPC 기반의 Virtual Private Cloud(VPC)에 생성하는 것이 좋습니다.
> 소스 AZ DB 클러스터와 다른 VPC에 읽기 전용 복제본을 생성하는 경우 Classless Inter-Domain Routing(CIDR) 범위가 복제본과 RDS 시스템 간에 겹칠 수 있습니다. CIDR이 겹치면 복제본이 불안정해져 복제본에 연결하는 애플리케이션에 부정적인 영향을 줄 수 있습니다. 읽기 전용 복제본을 생성할 때 오류가 발생하면 다른 대상 DB 서브넷 그룹을 선택합니다. 자세한 내용은 the section called “VPC에서 DB 인스턴스를 사용한 작업” 단원을 참조하십시오.

콘솔

다중 AZ DB 클러스터에서 DB 인스턴스 전용 복제본을 생성하려면 AWS Management Console을 사용하여 다음 단계를 완료합니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
3. 읽기 전용 복제본의 소스로 사용할 다중 AZ DB 인스턴스를 선택합니다.
4. 작업에서 읽기 전용 복제본 만들기를 선택합니다.
5. 복제본 소스의 경우 올바른 다중 AZ DB 클러스터를 선택했는지 확인하세요.
6. DB 식별자에 읽기 전용 복제본의 이름을 입력합니다.
7. 나머지 섹션에서 DB 인스턴스 설정을 지정합니다. 설정에 대한 자세한 내용은 the section called “사용 가능한 설정” 섹션을 참조하세요.

   > **Note**
   >
   > DB 인스턴스 읽기 전용 복제본의 할당된 스토리지는 소스 다중 AZ DB 클러스터의 할당된 스토리지와 동일해야 합니다.

8. 읽기 전용 복제본 생성을 선택합니다.

AWS CLI

다중 AZ DB 클러스터에서 DB 인스턴스 전용 복제본을 생성하려면 AWS CLI 명령 `create-db-instance-read-replica`를 사용하세요. `--source-db-cluster-identifier`의 경우 다중 AZ DB 클러스터의 식별자를 지정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance-read-replica \
  --db-instance-identifier myreadreplica \
  --source-db-cluster-identifier mymultiazdbcluster
```

Windows의 경우:

```
aws rds create-db-instance-read-replica ^
  --db-instance-identifier myreadreplica ^
  --source-db-cluster-identifier mymultiazdbcluster
```

RDS API

다중 AZ DB 클러스터에서 DB 인스턴스 전용 복제본을 생성하려면 `CreateDBInstanceReadReplica` 작업을 사용하세요.

DB 인스턴스 읽기 전용 복제본 승격

DB 인스턴스 읽기 전용 복제본이 더 이상 필요하지 않은 경우 이를 독립 실행형 DB 인스턴스로 승격시킬 수 있습니다. 읽기 전용 복제본을 승격하면 DB 인스턴스가 먼저 재부팅된 후에 사용할 수 있습니다. 지침은 the section called “읽기 전용 복제본 승격” 단원을 참조하십시오.

읽기 전용 복제본을 사용하여 다중 AZ DB 클러스터 배포를 단일 AZ 또는 다중 AZ DB 인스턴스 배포로 마이그레이션하려면 소스 DB 클러스터에 쓰기 작업을 수행 중인 모든 트랜잭션을 중단해야 합니다. 그런 다음, 읽기 전용 복제본의 업데이트가 모두 끝날 때까지 기다립니다. 다중 AZ DB 클러스터의 리더 DB 인스턴스 중 하나에서 업데이트가 실행되면 읽기 전용 복제본에서 데이터베이스 업데이트가 실행됩니다. 이 복제 지연은 서로 크게 다를 수 있습니다. `ReplicaLag` 지표를 사용하여 읽기 전용 복제본의 업데이트가 모두 완료되는 시간을 측정합니다. 복제 지연에 대한 자세한 내용은 the section called “읽기 전용 복제본 모니터링” 섹션을 참조하세요.

읽기 전용 복제본을 승격시킨 후, 승격된 DB 인스턴스를 사용하도록 애플리케이션을 리디렉션하려면 우선 승격된 DB 인스턴스의 상태가 `Available`이 될 때까지 기다려야 합니다. 또는 더 이상 필요하지

않은 경우에는 다중 AZ DB 클러스터 배포를 삭제합니다. 지침은 the section called “다중 AZ DB 클러스터 삭제” 단원을 참조하십시오.

다중 AZ DB 클러스터에서 DB 인스턴스 읽기 전용 복제본 생성 시 제한

다중 AZ DB 인스턴스 배포에서 DB 클러스터 읽기 전용 복제본을 생성하는 데는 다음과 같은 제한이 적용됩니다.

- 소스 DB 클러스터를 소유한 AWS 계정와 다른 AWS 계정에서는 DB 인스턴스 읽기 전용 복제본을 생성할 수 없습니다.
- 소스 다중 AZ DB 클러스터와 다른 AWS 리전에는 다중 DB 인스턴스 읽기 전용 복제본을 생성할 수 없습니다.
- DB 인스턴스 읽기 전용 복제본을 특정 시점으로 복구할 수 없습니다.
- 스토리지 암호화는 소스 다중 AZ DB 클러스터와 DB 인스턴스 읽기 전용 복제본에서 동일한 설정을 가져야 합니다.
- 다중 AZ DB 클러스터가 암호화된 경우 DB 인스턴스 읽기 전용 복제본은 동일한 KMS 키를 사용하여 암호화되어야 합니다.
- 소스 다중 AZ DB 클러스터에서 마이너 버전 업그레이드를 수행하려면 먼저 다중 DB 인스턴스 읽기 전용 복제본에서 마이너 버전 업그레이드를 수행해야 합니다.
- DB 인스턴스 읽기 전용 복제본은 읽기 전용 복제본 계단식 읽기를 지원하지 않습니다.
- RDS for PostgreSQL의 경우 DB 인스턴스 읽기 전용 복제본을 만들려면 소스 다중 AZ DB 클러스터에서 PostgreSQL 버전 13.11, 14.8 또는 15.2.R2 이상을 실행해야 합니다.
- DB 인스턴스 읽기 전용 복제본의 소스 다중 AZ DB 클러스터에서 메이저 버전 업그레이드를 수행할 수 있지만, 읽기 전용 복제본으로의 복제가 중지되고 다시 시작할 수 없습니다.

Amazon RDS의 다중 AZ DB 클러스터에서 외부 복제 설정

다중 AZ DB 클러스터와 Amazon RDS 외부에 있는 데이터베이스 간의 복제를 설정할 수 있습니다.

외부 복제를 통해 다중 AZ DB 클러스터는 온프레미스 또는 다른 클라우드 환경에서 RDS DB 인스턴스와 외부 데이터베이스 간에 데이터를 복제할 수 있습니다. 재해 복구, 데이터 마이그레이션 및 서로 다른 위치에 있는 시스템 간의 일관성 유지에 유용합니다. 이 섹션에서는 복제를 설정하기 위한 사전 조건, 프로세스를 구성하는 방법 및 복제 지연 시간, 대역폭, 다양한 데이터베이스 엔진과의 호환성과 같은 주요 고려 사항을 다룹니다.

RDS for MySQL

RDS for MySQL 다중 AZ DB 클러스터의 외부 복제를 설정하려면 Amazon RDS가 binlog 파일을 삭제하기 전에 변경 내용이 복제본에 적용되도록 클러스터 내 DB 인스턴스에 이진 로그 파일을 충분히 오랫동안 유지해야 합니다. 이렇게 하려면 `mysql.rds_set_configuration` 저장 절차를 직접적으로 호출하고 `binlog retention hours` 파라미터를 지정하여 이진 로그 유지를 구성하세요. 자세한 내용은 the section called “binlog retention hours” 단원을 참조하십시오.

`binlog retention hours`의 기본값은 `NULL`이며, 이는 이진 로그가 유지되지 않음을 의미합니다(0시간). 다중 AZ DB 클러스터에 대한 외부 복제를 설정하려면 파라미터를 `NULL` 이외의 값으로 설정해야 합니다.

다중 AZ DB 클러스터의 라이터 DB 인스턴스에서만 이진 로그 유지를 구성할 수 있으며, 설정은 모든 리더 DB 인스턴스에 비동기적으로 전파됩니다.

또한 외부 복제본에서 GTID 기반 복제를 활성화하는 것이 좋습니다. 그러면 DB 인스턴스 중 하나에 장애가 발생하는 경우 클러스터 내의 다른 정상 DB 인스턴스에서 복제를 재개할 수 있습니다. 자세한 내용은 MySQL 설명서의 전역 트랜잭션 ID를 사용한 복제를 참조하세요.

RDS for PostgreSQL

RDS for PostgreSQL 다중 AZ DB 클러스터의 외부 복제를 설정하려면 논리 복제를 활성화해야 합니다. 지침은 the section called “다중 AZ DB 클러스터의 PostgreSQL 논리적 복제” 단원을 참조하세요.

Amazon RDS용 다중 AZ DB 클러스터 삭제

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB 다중 AZ DB 클러스터를 삭제할 수 있습니다.

다중 AZ DB 클러스터를 삭제하는 데 필요한 시간은 다음과 같은 요인에 따라 달라질 수 있습니다.

- 백업 보존 기간(삭제할 백업 수).
- 삭제되는 데이터의 양.
- 최종 스냅샷 생성 여부.

다중 AZ DB 클러스터를 삭제하려면 먼저 삭제 방지를 비활성화해야 합니다. 자세한 내용은 the section called “DB 인스턴스를 삭제하기 위한 사전 조건” 단원을 참조하십시오. 다중 AZ DB 클러스터를 수정하여 삭제 방지를 비활성화할 수 있습니다. 자세한 내용은 the section called “다중 AZ DB 클러스터 수정” 단원을 참조하십시오.

콘솔

다중 AZ DB 클러스터를 삭제하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스(Databases)를 선택한 다음 삭제하려는 다중 AZ DB 클러스터를 선택합니다.
3. 작업에 대해 삭제를 선택합니다.
4. 최종 스냅샷을 생성하시겠습니까?(Create final snapshot?)를 선택하여 다중 AZ DB 클러스터의 최종 DB 스냅샷을 생성합니다.

   최종 스냅샷을 생성한 경우 최종 스냅샷 이름(Final snapshot name)에 이름을 입력합니다.
5. 자동 백업 보관(Retain automated backups)을 선택하여 자동 백업을 보관합니다.
6. 상자에 **`delete me`**를 입력합니다.
7. 삭제를 선택합니다.

AWS CLI

AWS CLI를 사용하여 다중 AZ DB 클러스터를 삭제하려면 다음 옵션과 함께 delete-db-cluster 명령을 호출합니다.

- `--db-cluster-identifier`
- `--final-db-snapshot-identifier` 또는 `--skip-final-snapshot`

Example 최종 스냅샷 포함

대상 LinuxmacOS, 또는Unix:

```
aws rds delete-db-cluster \
    --db-cluster-identifier mymultiazdbcluster \
    --final-db-snapshot-identifier mymultiazdbclusterfinalsnapshot
```

Windows의 경우:

```
aws rds delete-db-cluster ^
    --db-cluster-identifier mymultiazdbcluster ^
    --final-db-snapshot-identifier mymultiazdbclusterfinalsnapshot
```

Example 최종 스냅샷 제외

대상 LinuxmacOS, 또는Unix:

```
aws rds delete-db-cluster \
    --db-cluster-identifier mymultiazdbcluster \
    --skip-final-snapshot
```

Windows의 경우:

```
aws rds delete-db-cluster ^
    --db-cluster-identifier mymultiazdbcluster ^
    --skip-final-snapshot
```

RDS API

Amazon RDS API를 사용하여 다중 AZ DB 클러스터를 삭제하려면 다음 파라미터와 함께 DeleteDBCluster 작업을 호출합니다.

- `DBClusterIdentifier`
- `FinalDBSnapshotIdentifier` 또는 `SkipFinalSnapshot`

Amazon RDS 다중 AZ DB 클러스터의 제한

다중 AZ DB 클러스터에는 라이터 DB 인스턴스와 두 개의 리더 DB 인스턴스가 세 개의 개별 가용 영역에 있습니다. 다중 AZ DB 클러스터는 다중 AZ 배포에 비해 고가용성, 높은 읽기 워크로드 용량 및 짧은 대기 시간을 제공합니다. 다중 AZ DB 클러스터에 대한 자세한 내용은 Amazon RDS용 다중 AZ DB 클러스터 배포 섹션을 참조하세요.

다중 AZ DB 클러스터에는 다음과 같은 제한 사항이 적용됩니다.

- 다중 AZ DB 클러스터는 다음 기능을 지원하지 않습니다.
  - IPv6 연결(듀얼 스택 모드)
  - 교차 리전 자동 백업
  - Kerberos 인증
  - 포트 수정 또는 다중 AZ DB 클러스터를 특정 시점으로 복원하고 다른 포트 지정 가능
  - 옵션 그룹 수
  - 삭제된 클러스터의 시점 복구(PITR)
  - 할당된 최대 스토리지를 설정하여 스토리지 자동 크기 조정 또는 수동으로 스토리지 크기 조정 가능
  - 다중 AZ DB 클러스터 중지 및 시작
  - 다중 AZ DB 클러스터의 스냅샷 복사
  - 암호화되지 않은 다중 AZ DB 클러스터 암호화
- RDS for MySQL 다중 AZ DB 클러스터는 다음 시스템 저장 프로시저만 지원합니다.
  - `mysql.rds_rotate_general_log`
  - `mysql.rds_rotate_slow_log`
  - `mysql.rds_show_configuration`
  - `mysql.rds_set_external_master_with_auto_position`
  - `mysql.rds_set_configuration`
- RDS for PostgreSQL 다중 AZ DB 클러스터는 `aws_s3` 및 `pg_transport`와 같은 확장을 지원하지 않습니다.
- PostgreSQL 다중 AZ DB 클러스터용 RDS는 아웃바운드 네트워크 액세스에 사용자 지정 DNS 서버 사용을 지원하지 않습니다.

Amazon RDS를 사용한 Amazon RDS 추가 지원

RDS 추가 지원을 사용하면 RDS 표준 지원 종료일이 지난 후 메이저 엔진 버전에서 추가 비용을 지불하고 데이터베이스를 계속 실행할 수 있습니다.

DB 인스턴스를 처음 생성하거나 복원할 때 RDS 추가 지원을 활성화해야만 RDS 추가 지원에 데이터베이스를 등록할 수 있습니다. 복원하지 않으면 기존 DB 인스턴스에서 RDS 추가 지원 등록 상태를 업데이트할 수 없습니다.

DB 인스턴스를 생성 또는 복원하는 동안 RDS 추가 지원을 활성화했다면 RDS의 표준 지원 종료일 이후, Amazon RDS가 자동으로 해당 DB 인스턴스를 RDS 추가 지원에 등록합니다. RDS 추가 지원에 자동 등록해도 데이터베이스 엔진은 변경되지 않으며, DB 인스턴스의 가동 시간이나 성능에도 영향을 미치지 않습니다.

RDS 추가 지원은 다음과 같은 업데이트 및 기술 지원을 제공합니다.

- 데이터베이스 엔진을 포함하여 DB 인스턴스 또는 DB 클러스터의 중요 및 높은 CVE에 대한 보안 업데이트
- 중요한 문제에 대한 버그 수정 및 패치
- 표준 Amazon RDS 서비스 수준 계약 내에서 지원 사례를 열고 문제 해결 지원을 받을 수 있는 기능

이 유료 기능을 사용하면 지원되는 메이저 엔진 버전으로 업그레이드하는 데 더 많은 시간을 할애할 수 있습니다. 예를 들어 RDS for MySQL의 경우 RDS의 표준 지원 종료일은 2024년 2월 29일입니다. 하지만 종료일 이전에 RDS for MySQL 버전 8.0으로 수동 업그레이드할 준비가 되지 않을 수 있습니다. 이 경우, 2024년 2월 29일부터 Amazon RDS는 RDS 추가 지원에 데이터베이스를 자동으로 등록합니다. 따라서 사용자는 계속해서 RDS for MySQL 버전 5.7을 실행할 수 있습니다. 2024년 3월 1일부터 Amazon RDS에서 RDS 추가 지원 요금을 자동으로 청구합니다.

RDS 추가 지원은 메이저 엔진 버전의 RDS 표준 지원 종료일 이후 최대 3년 동안 사용할 수 있습니다. 3년이 지난 후에도 지원되는 버전으로 메이저 엔진 버전을 업그레이드하지 않으면 Amazon RDS는 메이저 엔진 버전을 자동 업그레이드합니다. 따라서 지원되는 메이저 엔진 버전으로 최대한 빨리 업그레이드하는 것이 좋습니다.

RDS 표준 지원 종료일 및 RDS 추가 지원 종료일에 대한 자세한 내용은 Amazon RDS에서 지원되는 MySQL 메이저 버전 섹션과 Amazon RDS for PostgreSQL 릴리스 일정을 참조하세요.

주제

- Amazon RDS 추가 지원 개요
- Amazon RDS 추가 지원 요금
- Amazon RDS 추가 지원이 포함된 버전
- Amazon RDS 및 Amazon RDS 추가 지원 사용 시 고객 책임
- Amazon RDS 추가 지원이 포함된 DB 인스턴스 또는 다중 AZ DB 클러스터 생성
- Amazon RDS 추가 지원의 DB 인스턴스 또는 다중 AZ DB 클러스터 등록 보기
- Amazon RDS 추가 지원에서 엔진 버전에 대한 지원 날짜 보기
- Amazon RDS 추가 지원이 포함된 DB 인스턴스 또는 다중 AZ DB 클러스터 복원

Amazon RDS 추가 지원 개요

RDS의 표준 지원 종료일 이후, DB 인스턴스를 생성 또는 복원하는 동안 RDS 추가 지원을 비활성화하지 않은 경우 Amazon RDS가 자동으로 해당 인스턴스를 RDS 추가 지원에 등록합니다. Amazon RDS는 RDS 표준 지원 종료일이 되기 전 마지막 마이너 버전 릴리스(아직 이 버전을 실행하지 않는 경우)로 DB 인스턴스를 자동 업그레이드합니다. Amazon RDS는 메이저 엔진 버전에 적용되는 RDS 표준 지원 종료일이 지날 때까지 마이너 버전을 업그레이드하지 않습니다.

또한 RDS 표준 지원 종료일에 도달한 메이저 엔진 버전으로 새 데이터베이스를 생성할 수 있습니다. RDS는 이러한 새 데이터베이스를 RDS 확장 지원에 자동으로 등록하고 이 서비스에 대한 비용을 청구합니다.

RDS의 표준 지원 종료일 이전에 아직 RDS 표준 지원이 적용되는 엔진으로 업그레이드하는 경우 Amazon RDS는 엔진을 RDS 추가 지원에 등록하지 않습니다.

RDS 표준 지원 종료일이 지났지만 RDS 추가 지원에 등록되지 않은 엔진과 호환되는 데이터베이스의 스냅샷을 복원하려고 시도하는 경우 Amazon RDS는 아직 RDS 표준 지원이 적용되는 최신 엔진 버전과 호환되도록 스냅샷을 업그레이드하려고 시도합니다. 복원이 실패할 경우 Amazon RDS는 스냅샷과 호환되는 버전으로 엔진을 RDS 추가 지원에 자동으로 등록합니다.

언제든지 RDS 추가 지원 등록을 종료할 수 있습니다. 등록을 종료하려면 등록된 각 엔진을 아직 RDS 표준 지원이 적용되는 최신 엔진 버전으로 업그레이드하세요. RDS 추가 지원 등록 종료는 아직 RDS 표준 지원이 적용되는 최신 엔진 버전으로 업그레이드를 완료하는 날부터 유효합니다.

RDS 표준 지원 종료일 및 RDS 추가 지원 종료일에 대한 자세한 내용은 Amazon RDS에서 지원되는 MySQL 메이저 버전 섹션과 Amazon RDS for PostgreSQL 릴리스 일정을 참조하세요.

Amazon RDS 추가 지원 요금

RDS 표준 지원 종료일 이후부터 RDS 추가 지원에 등록된 모든 엔진에 대해 요금이 부과됩니다. RDS 표준 지원 종료일에 대해 알아보려면 지원되는 MySQL 메이저 버전 섹션과 Amazon RDS for PostgreSQL에 대한 릴리스 일정을 참조하세요. RDS 확장 지원 요금은 다중 AZ 배포에서 대기 인스턴스에 적용됩니다.

다음 작업 중 하나를 수행하는 경우 RDS 추가 지원에 대한 추가 요금이 자동으로 중지됩니다.

- 표준 지원이 적용되는 엔진 버전으로 업그레이드하세요.
- RDS 표준 지원 종료일이 지난 메이저 버전을 실행 중인 데이터베이스를 삭제합니다.

향후 대상 엔진 버전이 RDS 추가 지원으로 전환되면 요금이 다시 부과됩니다.

예를 들어, RDS for PostgreSQL 11은 2024년 3월 1일에 확장 지원이 시작되지만, 2024년 4월 1일까지는 요금이 청구되지 않습니다. 2024년 4월 30일에 RDS for PostgreSQL 11 데이터베이스를 RDS for PostgreSQL 12로 업그레이드합니다. RDS for PostgreSQL 11에 대한 30일간의 추가 지원 비용만 청구됩니다. RDS 표준 지원 종료일인 2025년 2월 28일 이후에도 이 DB 인스턴스에서 RDS for PostgreSQL 12를 계속 실행할 수 있습니다. 2025년 3월 1일부터 데이터베이스에 RDS 추가 지원 요금이 다시 부과됩니다.

자세한 내용을 알아보려면 Amazon RDS for MySQL 요금 및 Amazon RDS for PostgreSQL 요금을 참조하세요.

Amazon RDS 추가 지원에 대한 비용 방지

RDS 표준 지원 종료일이 지난 후 RDS가 DB 인스턴스 또는 다중 AZ DB 클러스터를 생성 또는 복원하지 못하게 하여 RDS 추가 지원 요금이 부과되지 않도록 할 수 있습니다. 이렇게 하려면 AWS CLI 또는 RDS API를 사용하세요.

AWS CLI에서 `--engine-lifecycle-support` 옵션에 대해 `open-source-rds-extended-support-disabled`를 지정합니다. RDS API에서 `LifeCycleSupport` 파라미터에 `open-source-rds-extended-support-disabled`를 지정합니다. 자세한 내용은 DB 인스턴스 또는 다중 AZ DB 클러스터 생성 또는 DB 인스턴스 또는 다중 AZ DB 클러스터 복원 단원을 참조하세요.

Amazon RDS 추가 지원이 포함된 버전

RDS 확장 지원은 메이저 버전에서만 사용할 수 있습니다. 마이너 버전에서는 사용할 수 없습니다.

RDS 추가 지원은 RDS for MySQL 및 RDS for PostgreSQL에서 사용할 수 있습니다. 자세한 내용은 지원되는 MySQL 메이저 버전 및 Amazon RDS for PostgreSQL 릴리스 정보에서 Release calendar for Amazon RDS for PostgreSQL을 참조하세요.

AWS CLI 또는 RDS API를 사용하여 엔진 버전의 지원 날짜에 대한 정보를 볼 수도 있습니다. 자세한 내용은 Amazon RDS 추가 지원에서 엔진 버전에 대한 지원 날짜 보기 섹션을 참조하세요.

Amazon RDS 추가 지원 버전 명명 규칙

Amazon RDS는 RDS 추가 지원을 통해 엔진에 대한 수정 사항 및 CVE 패치가 포함된 새로운 마이너 버전을 출시할 예정입니다. 자세한 내용은 Amazon RDS for MySQL에 대한 Amazon RDS 추가 지원 버전 및 Amazon RDS for PostgreSQL 릴리스 정보에서 Amazon RDS Extended Support updates for RDS for PostgreSQL을 참조하세요.

이러한 마이너 릴리스의 이름은 major.minor-RDS.YYYYMMDD.patch.YYYYMMDD형식으로, RDS for MySQL의 경우 5.7.44-RDS.20240208.R2.20240210, RDS for PostgreSQL의 경우 11.22-RDS.20240208.R2.20240210입니다.

메이저

MySQL의 경우, 버전 번호의 정수 부분과 첫 번째 소수 부분 모두가 메이저 버전 번호입니다(예: 8.0). 메이저 버전 업그레이드는 버전 번호의 메이저 부분이 증가합니다. 예를 들어, 5.7.44에서 8.0.33으로 업그레이드하는 것은 메이저 버전 업그레이드입니다. 여기서 5.7과 8.0이 메이저 버전 번호입니다.

PostgreSQL의 경우, 메이저 버전 번호는 정수입니다(예: 11).

minor-RDS.YYYYMMDD

MySQL의 경우, 버전 번호의 세 번째 부분이 마이너 버전 번호입니다(예: 5.7.44-RDS.20240208의 44-RDS.20240208).

PostgeSQL의 경우, 버전 번호의 두 번째 부분이 마이너 버전 번호입니다(예: 11.22-RDS.20240208의 22-RDS.20240208).

날짜는 Amazon RDS에서 Amazon RDS 마이너 버전을 만든 날짜입니다.

패치

패치 버전은 Amazon RDS에서 Amazon RDS 마이너 버전을 만든 날짜를 따릅니다(예: 5.7.44-RDS.20240208.R2 또는 11.22-RDS.20240208.R2의 R2).

Amazon RDS 패치 버전에는 릴리스 후 Amazon RDS 마이너 버전에 추가된 중요한 버그 수정이 포함되어 있습니다.

YYYYMMDD(날짜)

날짜는 Amazon RDS에서 패치 버전을 만든 날짜입니다(예: 5.7.44-RDS.20240208.R2.20240210 또는 11.22-RDS.20240208.R2.20240210의 20240210).

Amazon RDS 날짜 버전은 보안 패치로, 릴리스 후 마이너 버전에 추가된 중요한 보안 수정이 포함되어 있습니다. 엔진 동작을 변경할 수 있는 수정 사항은 포함되지 않습니다.

Amazon RDS 및 Amazon RDS 추가 지원 사용 시 고객 책임

다음 콘텐츠에서는 Amazon RDS의 책임과 RDS 추가 지원 사용 시 사용자 책임에 대해 설명합니다.

주제

- Amazon RDS 책임
- 귀하의 책임

Amazon RDS 책임

RDS의 표준 지원 종료일 이후, Amazon RDS는 RDS 추가 지원에 등록된 엔진에 대한 패치, 버그 수정 및 업그레이드를 제공합니다. 이는 최대 3년 동안 또는 엔진 사용을 중지할 때까지(둘 중 먼저 발생하는 날짜) 지원됩니다.

패치를 통해 국가 취약성 데이터베이스(NVD) CVSS 심각도 등급에 정의된 대로 중요 및 상위 CVE에 대해 지원합니다. 자세한 내용을 알아보려면 Vulnerability Metrics 페이지를 참조하세요.

귀하의 책임

RDS 추가 지원에 등록된 DB 인스턴스 또는 다중 AZ DB 클러스터에 제공된 패치, 버그 수정 및 업그레이드를 적용하는 것은 사용자의 책임입니다. Amazon RDS는 언제든지 이러한 패치, 버그 수정 및 업그레이드를 변경, 교체 또는 철회할 권한을 보유합니다. 보안 또는 중요한 안정성 문제를 해결하기 위해 패치가 필요한 경우 Amazon RDS는 패치를 사용하여 DB 인스턴스 또는 다중 AZ DB 클러스터를 업데이트하거나 패치 설치를 요구할 권한을 보유합니다.

또한 RDS의 추가 지원 종료일 이전에 엔진을 최신 엔진 버전으로 업그레이드하는 것은 사용자의 책임입니다. RDS 추가 지원 종료일은 일반적으로 RDS 표준 지원일로부터 3년입니다. 데이터베이스 메

이저 엔진 버전의 RDS 추가 지원 종료일은 지원되는 MySQL 메이저 버전 섹션 및 Amazon RDS for PostgreSQL의 릴리스 일정을 참조하세요.

엔진을 업그레이드하지 않으면 RDS 추가 지원 종료일 이후에 Amazon RDS가 RDS 표준 지원에서 지원되는 최신 엔진 버전으로 엔진 업그레이드를 시도합니다. 업그레이드가 실패할 경우 Amazon RDS는 RDS 표준 지원 종료일이 지난 후 엔진을 실행 중인 DB 인스턴스 또는 다중 AZ DB 클러스터를 삭제할 권한을 보유합니다. 하지만 이렇게 하기 전에 Amazon RDS는 해당 엔진의 데이터를 보존합니다.

Amazon RDS 추가 지원이 포함된 DB 인스턴스 또는 다중 AZ DB 클러스터 생성

DB 인스턴스 또는 다중 AZ DB 클러스터를 생성할 때는 콘솔에서 RDS 추가 지원 활성화를 선택하거나 AWS CLI의 추가 지원 옵션이나 RDS API의 파라미터를 사용하세요. Amazon RDS 추가 지원에서 DB 인스턴스 또는 다중 AZ DB 클러스터를 등록하면 DB 인스턴스 또는 다중 AZ DB 클러스터의 수명 주기 동안 RDS 추가 지원에 영구적으로 등록됩니다.

콘솔을 사용하는 경우 RDS 추가 지원 활성화를 선택해야 합니다. 이 설정은 기본적으로 선택되어 있지 않습니다.

AWS CLI 또는 RDS API를 사용하며 RDS 추가 지원 설정을 지정하지 않는 경우 Amazon RDS는 기본적으로 RDS 추가 지원을 활성화합니다. AWS CloudFormation 또는 기타 서비스를 사용하여 자동화하면 이 기본 동작은 RDS 표준 지원 종료일 이후에도 데이터베이스의 가용성을 유지합니다.

AWS CLI 또는 RDS API를 사용하여 DB 인스턴스 또는 다중 AZ DB 클러스터를 생성하면 RDS 추가 지원에 등록되지 않도록 할 수 있습니다.

주제

- RDS 추가 지원 동작
- RDS 추가 지원 고려 사항
- RDS 추가 지원이 포함된 DB 인스턴스 또는 다중 AZ DB 클러스터 생성

RDS 추가 지원 동작

다음 표에는 메이저 엔진 버전이 RDS 표준 지원 종료 시점에 도달했을 때 발생하는 상황이 요약되어 있습니다.

RDS 추가 지원 상태*	동작
활성화됨	Amazon RDS가 RDS 추가 지원에 대한 요금 청구
비활성	Amazon RDS는 DB 인스턴스 또는 다중 AZ DB 클러스터를 지원되는 엔진 버전으로 업그레이드합니다. 이 업그레이드는 RDS 표준 지원 종료일 당일 또는 그 직후에 진행됩니다.

* RDS 콘솔에서 RDS 추가 지원 상태는 '예' 또는 '아니요'로 표시됩니다. AWS CLI 또는 RDS API에서 RDS 추가 지원 상태는 `open-source-rds-extended-support` 또는 `open-source-rds-extended-support-disabled`로 표시됩니다.

RDS 추가 지원 고려 사항

DB 인스턴스 또는 다중 AZ DB 클러스터를 만들기 전에 다음 항목을 고려하세요.

- RDS 표준 지원 종료일이 지난 후에 새 DB 인스턴스 또는 새 다중 AZ DB 클러스터가 생성되지 않게 하고 RDS 확장 지원 요금을 피할 수 있습니다. 이렇게 하려면 AWS CLI 또는 RDS API를 사용하세요. AWS CLI에서 `--engine-lifecycle-support` 옵션에 대해 `open-source-rds-extended-support-disabled`를 지정합니다. RDS API에서 `LifeCycleSupport` 파라미터에 `open-source-rds-extended-support-disabled`를 지정합니다. `open-source-rds-extended-support-disabled`를 지정하고 RDS 표준 지원 종료일이 지난 경우, DB 인스턴스 또는 다중 AZ DB 클러스터 생성이 항상 실패합니다.
- RDS 추가 지원은 클러스터 수준에서 설정됩니다. 클러스터 멤버는 RDS 콘솔, AWS CLI의 `--engine-lifecycle-support`, RDS API의 `EngineLifecycleSupport`에서 항상 동일한 RDS 추가 지원 설정을 가집니다.

자세한 내용은 MySQL 버전 및 Amazon RDS for PostgreSQL용 릴리스 일정을 참조하세요.

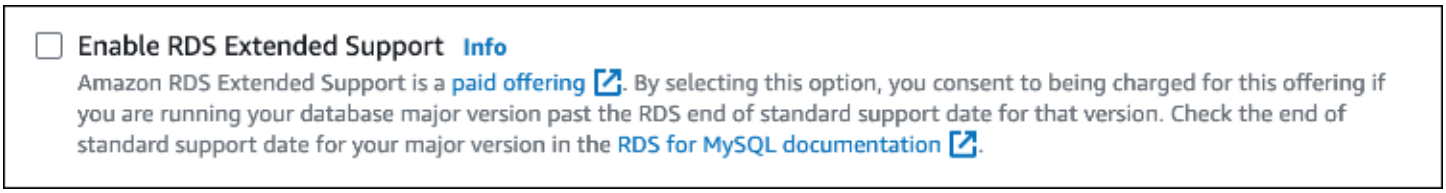
RDS 추가 지원이 포함된 DB 인스턴스 또는 다중 AZ DB 클러스터 생성

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 RDS 추가 지원 버전이 포함된 DB 인스턴스 또는 다중 AZ DB 클러스터를 만들 수 있습니다.

콘솔

DB 인스턴스 또는 다중 AZ DB 클러스터를 생성할 때는 엔진 옵션 섹션에서 RDS 확장 지원 활성화를 선택합니다. 이 설정은 기본적으로 선택되어 있지 않습니다.

다음 이미지는 RDS 추가 지원 활성화 설정을 보여줍니다.



AWS CLI

create-db-instance 또는 create-db-cluster(다중 AZ DB 클러스터) AWS CLI 명령을 실행하는 경우 `--engine-lifecycle-support` 옵션에 대해 `open-source-rds-extended-support`를 지정하여 RDS 추가 지원을 선택합니다. 기본적으로 이 옵션은 `open-source-rds-extended-support`(으)로 설정되어 있습니다.

RDS 표준 지원 종료일 이후에 새 DB 인스턴스 또는 다중 AZ DB 클러스터가 생성되지 않도록 하려면 `--engine-lifecycle-support` 옵션에 `open-source-rds-extended-support-disabled`를 지정하세요. 이렇게 하면 RDS 확장 지원 관련 요금을 피할 수 있습니다.

RDS API

CreateDBInstance 또는 CreateDBCluster(다중 AZ DB 클러스터) Amazon RDS API 작업을 사용할 때는 `EngineLifecycleSupport` 파라미터를 `open-source-rds-extended-support`로 설정하여 RDS 추가 지원을 선택합니다. 이 파라미터는 기본적으로 `open-source-rds-extended-support`로 설정되어 있습니다.

RDS 표준 지원 종료일 이후에 새 DB 인스턴스 또는 다중 AZ DB 클러스터가 생성되지 않도록 하려면 `EngineLifecycleSupport` 파라미터에 `open-source-rds-extended-support-disabled`를 지정하세요. 이렇게 하면 RDS 확장 지원 관련 요금을 피할 수 있습니다.

자세한 정보는 다음의 주제를 참조하세요.

- DB 인스턴스를 생성하려면 Amazon RDS DB 인스턴스 생성의 DB 엔진에 대한 지침을 따르십시오.
- DB 클러스터를 만들려면 Amazon RDS용 다중 AZ DB 클러스터 생성 섹션의 지침을 따르세요.

Amazon RDS 추가 지원의 DB 인스턴스 또는 다중 AZ DB 클러스터 등록 보기

AWS Management Console(AWS CLI) 또는 RDS API를 사용하여 RDS 추가 지원에서 DB 인스턴스 또는 다중 AZ DB 클러스터 등록을 확인할 수 있습니다.

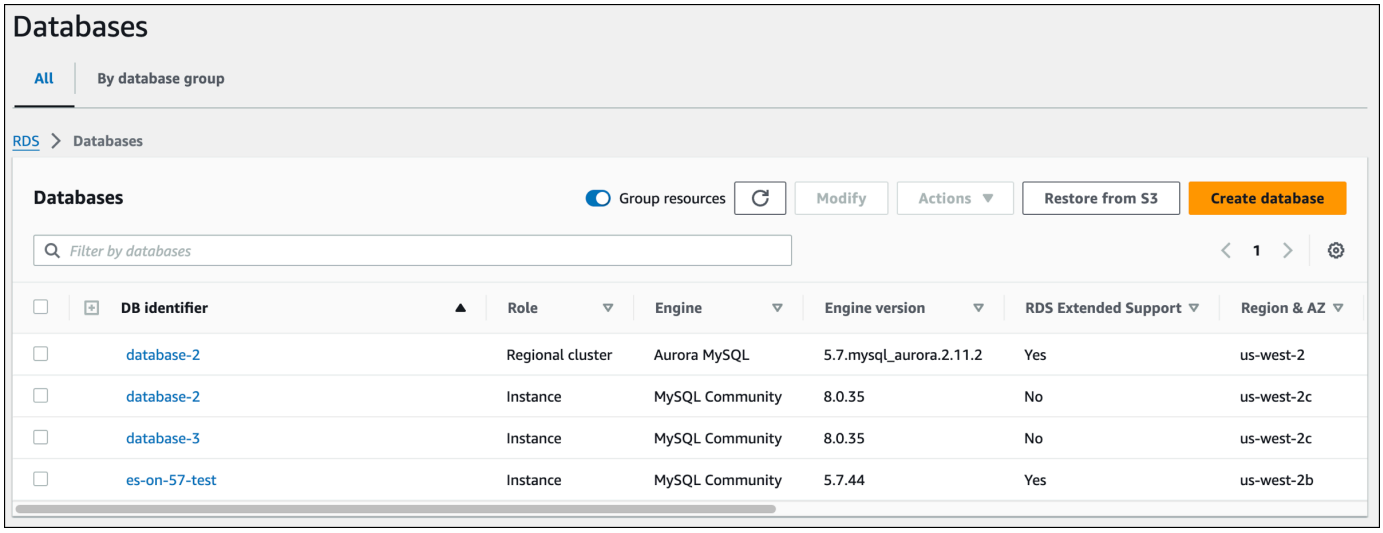
> **Note**
>
> 콘솔의 RDS 추가 지원 열, AWS CLI의 `--engine-lifecycle-support` 옵션 및 RDS API의 `EngineLifecycleSupport` 파라미터는 RDS 추가 지원 등록만 나타냅니다. RDS 추가 지원 요금은 DB 엔진 버전이 RDS 표준 지원 종료 시점에 도달한 경우에만 시작됩니다. 자세한 내용은 지원되는 MySQL 메이저 버전 및 Amazon RDS for PostgreSQL 릴리스 정보에서 Release calendar for Amazon RDS for PostgreSQL을 참조하세요.
> 예를 들어, RDS 추가 지원에 등록되어 있는 RDS for MySQL 5.7 데이터베이스가 있습니다. 2024년 3월 1일부터 Amazon RDS는 이 데이터베이스에 대해 RDS 추가 지원 요금을 청구하기 시작했습니다. 2024년 7월 31일에 이 데이터베이스를 RDS for MySQL 8.0으로 업그레이드했습니다. 이 데이터베이스의 RDS 추가 지원 상태는 활성화된 상태로 유지됩니다. 하지만 RDS for MySQL 8.0이 아직 RDS 표준 지원 종료 시점에 도달하지 않았기 때문에 이 데이터베이스에 대한 RDS 추가 지원 요금 청구는 중단되었습니다. Amazon RDS는 2026년 8월 1일까지 이 데이터베이스에 대한 RDS 추가 지원 요금을 청구하지 않습니다. 이 날짜가 되면 RDS for MySQL 8.0에 대한 RDS 표준 지원이 종료됩니다.

콘솔

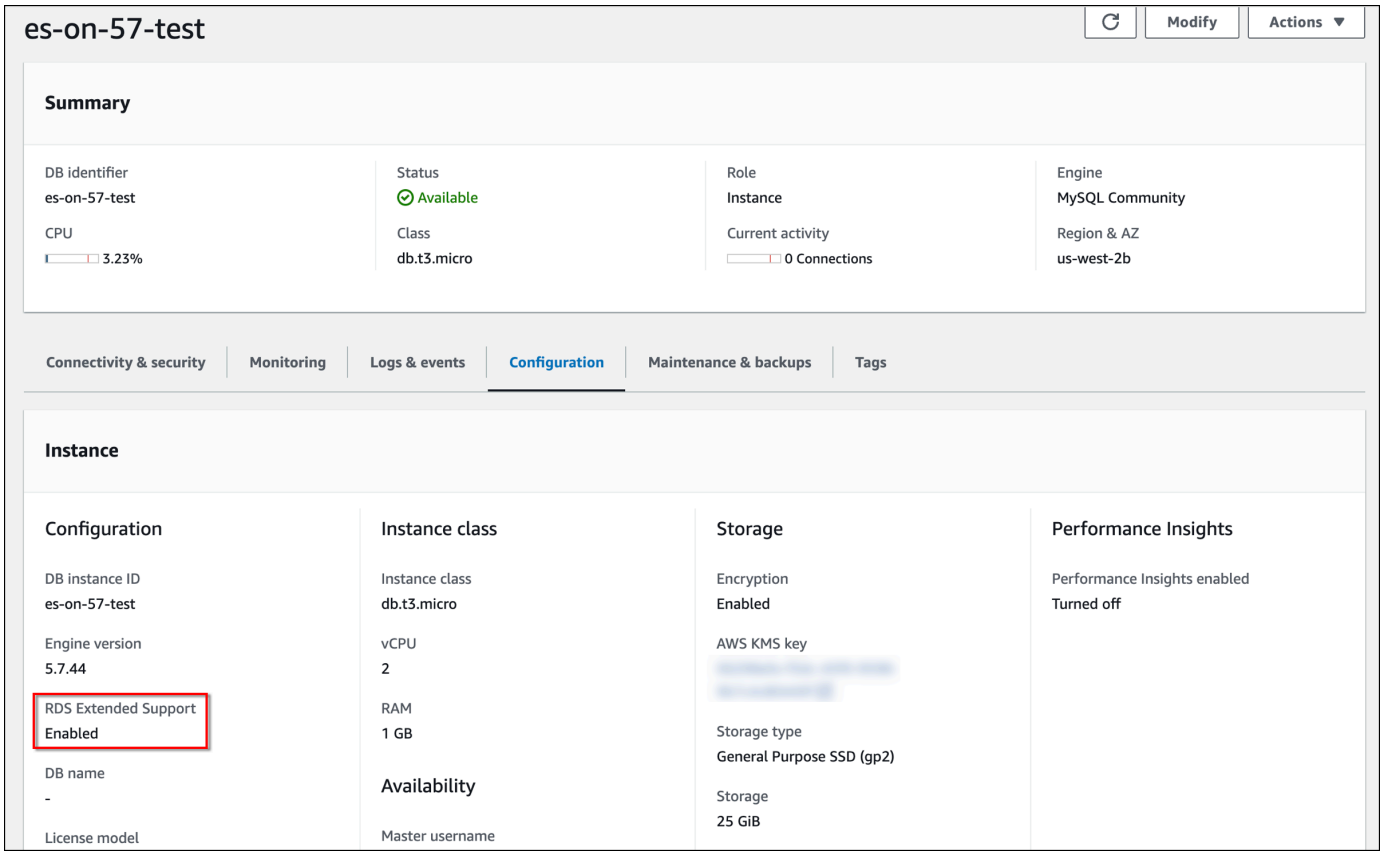
RDS 추가 지원의 DB 인스턴스 또는 다중 AZ DB 클러스터 등록을 보려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택합니다. RDS 추가 지원의 값은 DB 인스턴스 또는 다중 AZ DB 클러스터가 RDS 추가 지원에 등록되었는지 여부를 나타냅니다. 값이 표시되지 않으면 데이터베이스에 대해 RDS 확장 지원을 사용할 수 없는 것입니다.

> **Tip**
>
> RDS 추가 지원 열이 표시되지 않는 경우 기본 설정 아이콘을 선택한 다음 RDS 추가 지원을 활성화합니다.



3. 각 데이터베이스의 구성 탭에서 등록을 볼 수도 있습니다. DB 식별자에서 데이터베이스를 선택합니다. 구성 탭의 추가 지원에서 데이터베이스가 등록되었는지 여부를 확인합니다.



AWS CLI

AWS CLI를 사용하여 RDS 추가 지원에서 데이터베이스의 등록을 보려면 describe-db-instances 또는 describe-db-clusters(Multi-AZ DB 클러스터) 명령을 실행하세요.

데이터베이스에 대해 RDS 추가 지원을 사용할 수 있는 경우, 응답에 `EngineLifecycleSupport` 파라미터가 포함됩니다. `open-source-rds-extended-support` 값은 DB 인스턴스 또는 Multi-AZ DB 클러스터가 RDS 추가 지원에 등록되어 있음을 나타냅니다. `open-source-rds-extended-support-disabled` 값은 RDS 추가 지원에서 DB 인스턴스 또는 Multi-AZ DB 클러스터의 등록이 비활성화되었음을 나타냅니다.

예

다음 명령은 모든 DB 인스턴스에 대한 정보를 반환합니다.

```
aws rds describe-db-instances
```

다음 응답은 DB 인스턴스 `database-1`에서 실행 중인 PostgreSQL 엔진이 RDS 추가 지원에 등록되어 있음을 나타냅니다.

```
{
    "DBInstanceIdentifier": "database-1",
    "DBInstanceClass": "db.t3.large",
    "Engine": "postgres",
    ...
    "EngineLifecycleSupport": "open-source-rds-extended-support"
}
```

RDS API

Amazon RDS API를 사용하여 RDS 추가 지원에서 데이터베이스의 등록을 보려면 DescribeDBInstances 또는 DescribeDBClusters 작업을 사용하세요.

데이터베이스에 대해 RDS 추가 지원을 사용할 수 있는 경우, 응답에 `EngineLifecycleSupport` 파라미터가 포함됩니다. `open-source-rds-extended-support` 값은 DB 인스턴스 또는 Multi-AZ DB 클러스터가 RDS 추가 지원에 등록되어 있음을 나타냅니다. `open-source-rds-extended-support-disabled` 값은 RDS 추가 지원에서 DB 인스턴스 또는 Multi-AZ DB 클러스터의 등록이 비활성화되었음을 나타냅니다.

Amazon RDS 추가 지원에서 엔진 버전에 대한 지원 날짜 보기

AWS CLI 또는 RDS API를 사용하여 Amazon RDS 추가 지원에서 DB 인스턴스 또는 다중 AZ DB 클러스터의 엔진 버전에 대한 지원 날짜 정보를 볼 수 있습니다. 이 정보는 업그레이드를 계획하는 데 도움이 될 수 있습니다.

AWS CLI 명령 및 RDS API 작업은 RDS 표준 지원 및 RDS 추가 지원에 대한 시작 및 종료 날짜를 반환합니다. 이러한 날짜는 메이저 엔진 버전 테이블에서도 확인할 수 있습니다. 자세한 내용은 Amazon RDS에서 지원되는 MySQL 메이저 버전 및 Amazon RDS for PostgreSQL 릴리스 정보의 Release calendar for Amazon RDS for PostgreSQL을 참조하세요.

AWS CLI

주요 엔진 버전에 대한 RDS 표준 지원 및 RDS 추가 지원의 시작 날짜와 종료 날짜를 확인하려면 AWS CLI를 사용하여 describe-db-major-engine-versions 명령을 실행합니다.

이 명령은 오픈 소스 엔진(MariaDB, MySQL 및 PostgreSQL)에 대해 다음과 같은 관련 파라미터를 반환합니다. 상용 엔진(Db2, SQL Server 및 Oracle)에 대해서는 이러한 파라미터를 반환하지 않습니다.

- `SupportedEngineLifecycles` - 이 파라미터는 `LifecycleSupportName`, `LifecycleSupportStartDate` 및 `LifecycleSupportEndDate`를 포함하는 객체의 배열입니다.
- `LifecycleSupportName` - 이 파라미터는 엔진 버전이 속한 지원 유형(RDS 표준 지원(`open-source-rds-standard-support`) 또는 RDS 추가 지원(`open-source-rds-extended-support`))을 나타냅니다. MariaDB의 경우 이 파라미터는 RDS 표준 지원(`open-source-rds-standard-support`)만 반환합니다.
- `LifecycleSupportStartDate` - 이 파라미터는 `LifecycleSupportName`의 값에 따라 메이저 엔진 버전의 RDS 표준 지원 또는 RDS 추가 지원에 대한 시작 날짜를 나열합니다.
- `LifecycleSupportEndDate` - 이 파라미터는 `LifecycleSupportName`의 값에 따라 메이저 엔진 버전의 RDS 표준 지원 또는 RDS 추가 지원에 대한 종료 날짜를 나열합니다.

예제

응답 예제는 다음 정보를 보여 줍니다.

- MySQL 5.7의 지원되는 엔진 수명 주기 `open-source-rds-standard-support` 및 `open-source-rds-extended-support`에 대한 시작 및 종료 날짜입니다. MySQL 5.7에서는 RDS 추가 지원을 사용할 수 있습니다.

- MariaDB 10.6의 지원되는 엔진 수명 주기 `open-source-rds-standard-support`에 대한 시작 및 종료 날짜입니다. MariaDB 10.6에서는 RDS 추가 지원을 사용할 수 없습니다.
- SQL Server는 상용 엔진이므로 SQL Server Enterprise Edition 13에서 지원되는 엔진 수명 주기에 대한 정보는 없습니다.

```
{
    "DBMajorEngineVersions": [
        {
            "Engine": "mysql",
            "MajorEngineVersion": "5.7",
            "SupportedEngineLifecycles": [
                {
                    "LifecycleSupportName": "open-source-rds-standard-support",
                    "LifecycleSupportStartDate": "2016-02-22T00:00:00+00:00",
                    "LifecycleSupportEndDate": "2024-02-29T23:59:59.999000+00:00"
                },
                {
                    "LifecycleSupportName": "open-source-rds-extended-support",
                    "LifecycleSupportStartDate": "2024-03-01T00:00:00+00:00",
                    "LifecycleSupportEndDate": "2027-02-28T23:59:59.999000+00:00"
                }
            ]
        },
        {
            "Engine": "mariadb",
            "MajorEngineVersion": "10.6",
            "SupportedEngineLifecycles": [
                {
                    "LifecycleSupportName": "open-source-rds-standard-support",
                    "LifecycleSupportStartDate": "2022-02-03T00:00:00+00:00",
                    "LifecycleSupportEndDate": "2026-07-31T23:59:59.999000+00:00"
                }
            ]
        },
        {
            "Engine": "sqlserver-ee",
            "MajorEngineVersion": "13.00"
        },
        ...
    ]
}
```

RDS API

RDS API를 사용하여 메이저 엔진 버전의 RDS 표준 지원 및 RDS 추가 지원에 대한 시작 및 종료 날짜를 보려면 DescribeDBMajorEngineVersions 작업을 사용합니다.

엔진 버전에 대해 RDS 추가 지원을 사용할 수 있는 경우 응답에는 두 개의 객체가 있는 배열로 `SupportedEngineLifeCycles` 파라미터가 포함됩니다. 하나의 객체에는 RDS 표준 지원을 위한 시작 및 종료 날짜가 포함됩니다. 두 번째 객체에는 RDS 추가 지원의 시작 및 종료 날짜가 포함됩니다.

오픈 소스 엔진 버전(MariaDB, MySQL 및 PostgreSQL)에서 RDS 추가 지원을 사용할 수 없는 경우, 응답에는 단일 객체가 있는 배열로 `SupportedEngineLifeCycles` 파라미터만 포함됩니다. 이 객체에는 RDS 표준 지원의 시작 및 종료 날짜가 포함됩니다.

엔진 버전이 상용 엔진(Db2, SQL Server 및 Oracle)용인 경우 응답에 `SupportedEngineLifeCycles` 파라미터가 포함되지 않습니다.

Amazon RDS 추가 지원이 포함된 DB 인스턴스 또는 다중 AZ DB 클러스터 복원

DB 인스턴스 또는 다중 AZ DB 클러스터를 복원할 때는 콘솔에서 RDS 추가 지원 활성화를 선택하거나 AWS CLI의 추가 지원 옵션이나 RDS API의 파라미터를 사용하세요. RDS 추가 지원에서 DB 인스턴스 또는 다중 AZ DB 클러스터를 등록하면 DB 인스턴스 또는 다중 AZ DB 클러스터의 수명 주기 동안 RDS 추가 지원에 영구적으로 등록됩니다.

RDS 추가 지원 설정의 기본값은 데이터베이스를 복원할 때 콘솔, AWS CLI, RDS API 중 무엇을 사용하는지에 따라 달라집니다. 콘솔을 사용할 때 RDS 추가 지원 활성화를 선택하지 않고 복원 중인 메이저 엔진 버전이 RDS 표준 지원 종료 시점이 지난 경우 Amazon RDS는 자동으로 DB 인스턴스를 최신 엔진 버전으로 업그레이드합니다. AWS CLI 또는 RDS API를 사용하며 RDS 추가 지원 설정을 지정하지 않는 경우 Amazon RDS는 기본적으로 RDS 추가 지원을 활성화합니다. AWS CloudFormation 또는 기타 서비스를 사용하여 자동화하면 이 기본 동작은 RDS 표준 지원 종료일 이후에도 데이터베이스의 가용성을 유지합니다. AWS CLI 또는 RDS API를 사용하여 RDS 추가 지원을 비활성화할 수 있습니다.

주제

- RDS 추가 지원 동작
- RDS 추가 지원 고려 사항
- RDS 확장 지원이 적용되는 DB 인스턴스 또는 다중 AZ DB 클러스터 복원

RDS 추가 지원 동작

다음 표에는 복원 중인 DB 인스턴스 또는 다중 AZ DB 클러스터의 메이저 엔진 버전이 RDS 표준 지원 종료 시점에 도달했을 때 발생하는 상황이 요약되어 있습니다.

RDS 추가 지원 상태*	동작
활성화됨	Amazon RDS가 RDS 추가 지원에 대한 요금 청구
비활성화됨**	복원이 완료되면 Amazon RDS는 향후 유지 관리 기간에 DB 인스턴스 또는 다중 AZ DB 클러스터를 최신 엔진 버전으로 자동 업그레이드합니다.

* RDS 콘솔에서 RDS 추가 지원 상태는 '예' 또는 '아니요'로 표시됩니다. AWS CLI 또는 RDS API에서 RDS 추가 지원 상태는 `open-source-rds-extended-support` 또는 `open-source-rds-extended-support-disabled`로 표시됩니다.

** 이 옵션은 PostgreSQL 12 이상 또는 MySQL 8 이상을 실행하는 DB 인스턴스 또는 다중 AZ DB 클러스터를 복원할 때만 사용할 수 있습니다.

RDS 추가 지원 고려 사항

DB 인스턴스 또는 다중 AZ DB 클러스터를 복원하기 전에 다음 항목을 고려하세요.

- RDS 표준 지원 종료일이 지난 후 Amazon S3에서 DB 인스턴스 또는 다중 AZ DB 클러스터를 복원하려는 경우 AWS CLI 또는 RDS API를 사용해서만 복원할 수 있습니다. restore-db-cluster-from-s3 AWS CLI 명령의 `--engine-lifecycle-support` 옵션을 사용하거나 RestoreDBClusterFromS3 RDS API 작업의 `EngineLifecycleSupport` 파라미터를 사용하세요.
- RDS가 데이터베이스를 RDS 확장 지원 버전으로 복원하지 못하게 하려면 AWS CLI 또는 RDS API에서 `open-source-rds-extended-support-disabled`를 지정하세요. 이렇게 하면 RDS 확장 지원 관련 요금을 피할 수 있습니다.

  이 설정을 지정하면 Amazon RDS가 복원된 데이터베이스를 지원되는 최신 메이저 버전으로 자동 업그레이드합니다. 업그레이드가 업그레이드 검사에 실패하는 경우 Amazon RDS는 RDS 확장 지원 엔진 버전으로 안전하게 롤백합니다. 이 데이터베이스는 RDS 확장 지원 모드로 유지되며, 데이터베이스를 수동으로 업그레이드하기 전까지 Amazon RDS에서 RDS 확장 지원 요금을 청구합니다.

  예를 들어, RDS 추가 지원을 사용하지 않고 MySQL 5.7 스냅샷을 복원하는 경우 Amazon RDS는 데이터베이스를 MySQL 8.0으로 자동 업그레이드하려고 시도합니다. 해결해야 할 문제로 인해 업그레

이드가 실패할 경우 Amazon RDS는 데이터베이스를 MySQL 5.7로 롤백합니다. Amazon RDS는 문제를 해결할 수 있을 때까지 데이터베이스를 RDS 확장 지원으로 유지합니다. 예를 들어 스토리지 공간이 부족하여 업그레이드가 실패할 수 있습니다. 문제를 해결한 후에 업그레이드를 시작해야 합니다. 데이터베이스 업그레이드를 처음 시도한 후에는 Amazon RDS에서 데이터베이스를 다시 업그레이드하려고 시도하지 않습니다.

- RDS 추가 지원은 클러스터 수준에서 설정됩니다. 클러스터 멤버는 RDS 콘솔, AWS CLI의 `--engine-lifecycle-support`, RDS API의 `EngineLifecycleSupport`에서 항상 동일한 RDS 추가 지원 설정을 가집니다.

자세한 내용은 MySQL 버전 및 Amazon RDS for PostgreSQL용 릴리스 일정을 참조하세요.

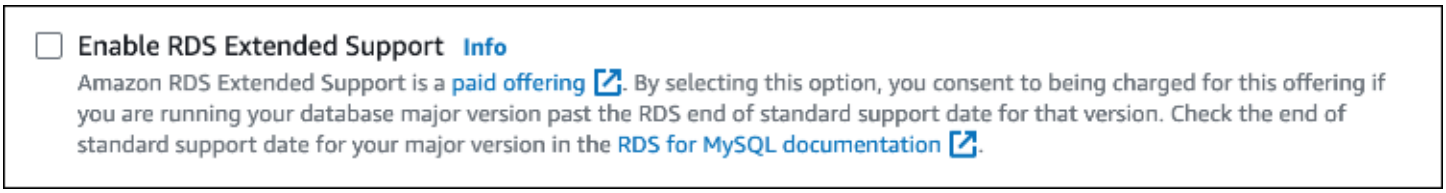
RDS 확장 지원이 적용되는 DB 인스턴스 또는 다중 AZ DB 클러스터 복원

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 RDS 추가 지원 버전이 포함된 DB 인스턴스 또는 다중 AZ DB 클러스터를 복원할 수 있습니다.

콘솔

DB 인스턴스 또는 다중 AZ DB 클러스터를 복원할 때는 엔진 옵션 섹션에서 RDS 추가 지원 활성화를 선택합니다. 이 설정을 선택하지 않았으며 복원 중인 메이저 엔진 버전이 RDS 표준 지원 종료 시점이 지난 경우 Amazon RDS는 DB 인스턴스 또는 다중 AZ DB 클러스터를 RDS 표준 지원 버전으로 자동 업그레이드합니다.

다음 이미지는 RDS 추가 지원 활성화 설정을 보여줍니다.



AWS CLI

restore-db-instance-from-db-snapshot 또는 restore-db-cluster-from-snapshot AWS CLI 명령을 실행하는 경우 `--engine-lifecycle-support` 옵션에 대해 `open-source-rds-extended-support`를 지정하여 RDS 추가 지원을 선택합니다.

RDS 확장 지원과 관련된 비용을 피하려면 `--engine-lifecycle-support` 옵션을 `open-source-rds-extended-support-disabled`로 설정하세요. 기본적으로 이 옵션은 `open-source-rds-extended-support`(으)로 설정되어 있습니다.

다음 AWS CLI 명령을 사용하여 이 값을 지정할 수도 있습니다.

- restore-db-cluster-from-s3
- restore-db-cluster-to-point-in-time
- restore-db-instance-from-s3
- restore-db-instance-to-point-in-time

RDS API

RestoreDBInstanceFromDBSnapshot 또는 RestoreDBClusterFromSnapshot Amazon RDS API 작업을 사용하는 경우 `EngineLifecycleSupport` 파라미터를 `open-source-rds-extended-support`로 설정하여 RDS 추가 지원을 선택합니다.

RDS 확장 지원과 관련된 비용을 피하려면 `EngineLifecycleSupport` 파라미터를 `open-source-rds-extended-support-disabled`로 설정하세요. 이 파라미터는 기본적으로 `open-source-rds-extended-support`로 설정되어 있습니다.

다음 RDS API 작업으로 이 값을 지정할 수도 있습니다.

- RestoreDBClusterFromS3
- RestoreDBClusterToPointInTime
- RestoreDBInstanceFromS3
- RestoreDBInstanceToPointInTime

DB 인스턴스 또는 다중 AZ DB 클러스터 복원에 대한 자세한 내용은 DB 인스턴스 복원 섹션의 DB 엔진에 대한 지침을 따르세요.

데이터베이스 업데이트에 Amazon RDS 블루/그린 배포 사용

블루/그린 배포는 프로덕션 데이터베이스 환경을 별도의 동기화된 스테이징 환경에 복사합니다. Amazon RDS 블루/그린 배포를 사용하면 프로덕션 환경에 영향을 주지 않고 스테이징 환경에서 데이터베이스를 변경할 수 있습니다. 예를 들어 메이저 또는 마이너 DB 엔진 버전을 업그레이드하거나, 스테이징 환경 내 스키마를 변경할 수 있습니다. 준비가 되면 대부분의 경우 1분 미만의 가동 중지 시간으로 스테이징 환경을 새로운 프로덕션 데이터베이스 환경으로 승격할 수 있습니다.

> **Note**
>
> 현재 블루/그린 배포는 RDS for MariaDB, RDS for MySQL, RDS for PostgreSQL에서만 지원됩니다. Amazon Aurora 가용성에 대해서는 Amazon Aurora 사용 설명서의 데이터베이스 업데이트에 Amazon Aurora 블루/그린 배포 사용을 참조하세요.

주제

- 용 Amazon RDS 블루/그린 배포 개요
- Amazon RDS에서 블루/그린 배포 생성
- Amazon RDS에서 블루/그린 배포 보기
- Amazon RDS에서 블루/그린 배포 전환
- Amazon RDS에서 블루/그린 배포 삭제

용 Amazon RDS 블루/그린 배포 개요

Amazon RDS 블루/그린 배포를 사용하면 프로덕션 환경에서 구현하기 전에 데이터베이스를 변경하고 테스트할 수 있습니다. 블루/그린 배포는 프로덕션 환경을 복사하는 스테이징 환경을 만듭니다. 블루/그린 배포에서는 블루 환경이 현재 프로덕션 환경입니다. 그린 환경은 스테이징 환경입니다. 스테이징 환경은 논리적 복제를 사용하여 현재 프로덕션 환경과 계속 동기화됩니다.

프로덕션 워크로드에 영향을 주지 않고 그린 환경에서 RDS DB 인스턴스 를 변경할 수 있습니다. 예를 들어 메이저 또는 마이너 DB 엔진 버전을 업그레이드하거나 기본 파일 시스템 구성을 업그레이드하거나 스테이징 환경에서 데이터베이스 파라미터를 변경할 수 있습니다. 그린 환경의 변경 사항을 철저하게 테스트할 수 있습니다. 준비가 되면 환경을 전환하여 그린 환경을 새로운 프로덕션 환경으로 승격할 수 있습니다. 전환은 일반적으로 1분도 걸리지 않으며 데이터 손실이 발생하지 않고 애플리케이션을 변경할 필요도 없습니다.

그린 환경은 프로덕션 환경 토폴로지의 복사본이므로, 그린 환경에는 DB 인스턴스에서 사용하는 기능이 포함됩니다. 대표적인 기능은 읽기 전용 복제본, 스토리지 구성, DB 스냅샷, 자동 백업, 성능 개선 도우미와 확장 모니터링입니다. 블루 DB 인스턴스가 다중 AZ DB 인스턴스 배포인 경우 그린 DB 인스턴스도 다중 AZ DB 인스턴스 배포입니다.

> **Note**
>
> 현재 블루/그린 배포는 RDS for MariaDB, RDS for MySQL, RDS for PostgreSQL에서만 지원됩니다. Amazon Aurora 가용성에 대해서는 Amazon Aurora 사용 설명서의 데이터베이스 업데이트에 Amazon RDS 블루/그린 배포 사용을 참조하세요.

주제

- 리전 및 버전 사용 가능 여부
- Amazon RDS 블루/그린 배포 사용의 장점
- 블루/그린 배포 워크플로우
- Amazon RDS 블루/그린 배포 작업에 대한 액세스 권한 부여
- Amazon RDS 블루/그린 배포 관련 제한 사항 및 고려 사항
- Amazon RDS 블루/그린 배포 모범 사례

리전 및 버전 사용 가능 여부

기능 가용성 및 해당 지원은 각 데이터베이스 엔진의 특정 버전 및 AWS 리전 리전에 따라 다릅니다. 자세한 내용은 the section called “블루/그린 배포” 단원을 참조하십시오.

Amazon RDS 블루/그린 배포 사용의 장점

Amazon RDS 블루/그린 배포를 사용하면 보안 패치를 최신 상태로 유지하고, 데이터베이스 성능을 개선할 수 있으며 짧고 예측 가능한 다운타임으로 최신 데이터베이스 기능을 채택할 수 있습니다. 블루/그린 배포를 통해 메이저 또는 마이너 엔진 버전 업그레이드와 같이 데이터베이스 업데이트로 인해 발생가능한 리스크 및 다운타임을 줄일 수 있습니다.

블루/그린 배포는 다음과 같은 장점을 제공합니다.

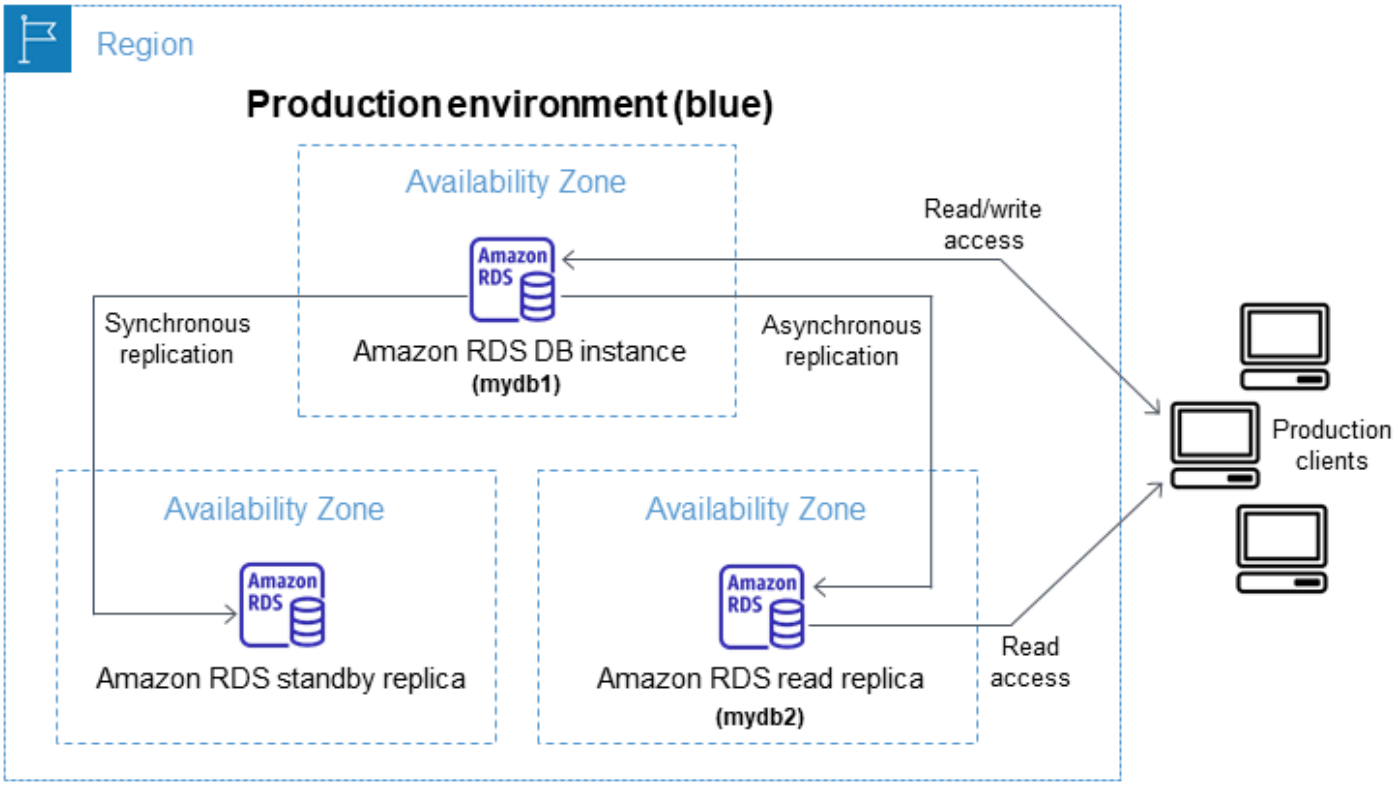
- 프로덕션 준비가 된 스테이징 환경을 쉽게 만들 수 있습니다.
- 데이터베이스 변경 사항을 프로덕션 환경에서 스테이징 환경으로 자동으로 복제합니다.
- 프로덕션 환경에 영향을 주지 않고 안전한 스테이징 환경에서 데이터베이스 변경 사항을 테스트합니다.
- 데이터베이스 패치 및 시스템 업데이트로 최신 상태를 유지하세요.
- 새로운 데이터베이스 기능을 구현하고 테스트합니다.
- 애플리케이션을 변경하지 않고도 스테이징 환경을 새 프로덕션 환경으로 전환합니다.
- 내장된 전환 가드레일을 사용하여 안전하게 전환합니다.
- 전환 중 데이터 손실이 발생하지 않습니다.
- 워크로드에 따라 대부분 1분 이내에 빠르게 전환합니다.

블루/그린 배포 워크플로우

데이터베이스 업데이트에 블루/그린 배포를 사용한다면 다음 주요 단계를 완료해주세요.

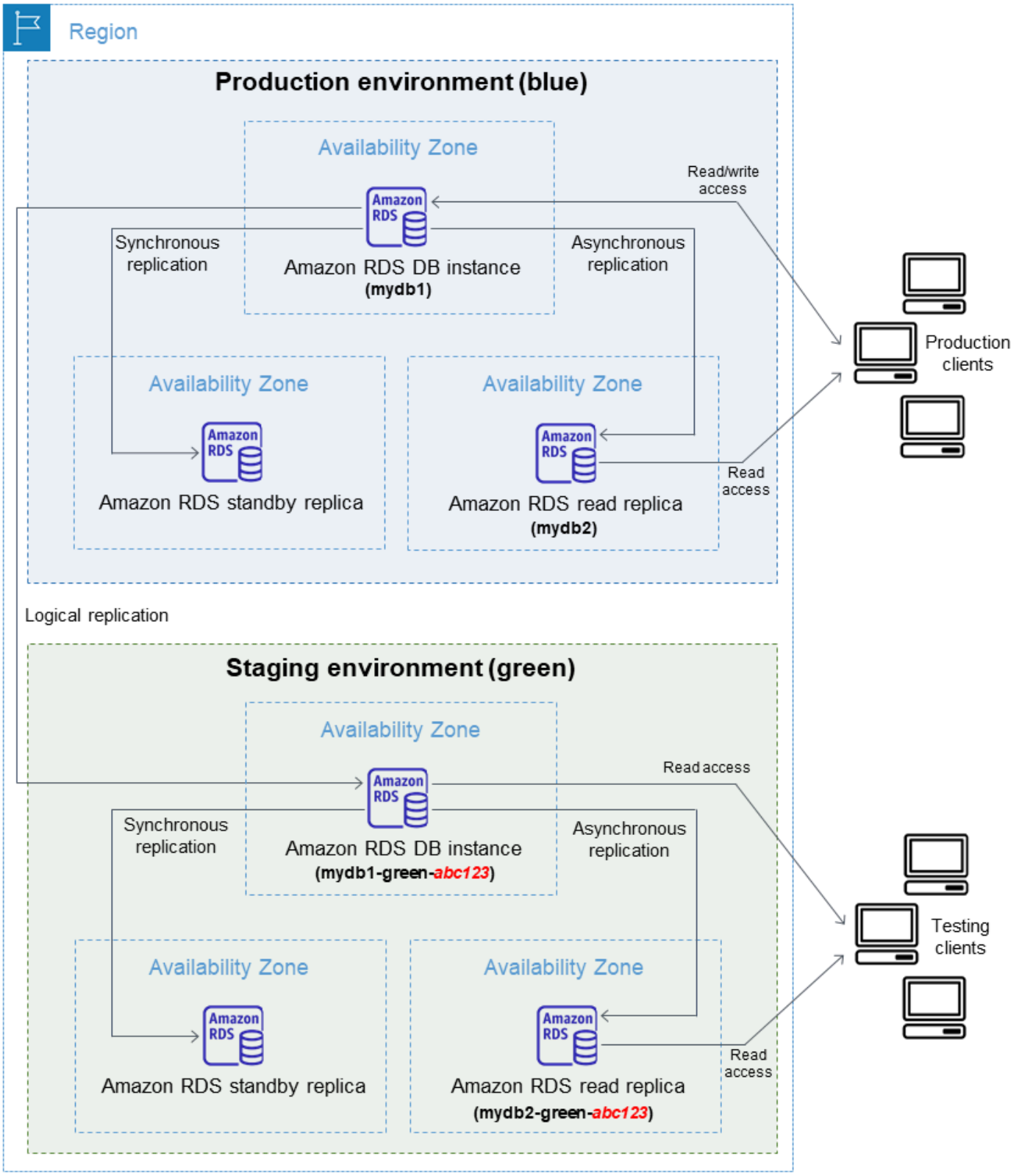
1. 업데이트가 필요한 프로덕션 환경을 식별합니다.

   예를 들어 이 이미지의 프로덕션 환경에는 다중 AZ DB 인스턴스 배포(mydb1) 읽기 전용 복제본(mydb2)이 있습니다.



2. 블루/그린 배포 생성 지침은 Amazon RDS에서 블루/그린 배포 생성 단원을 참조하십시오.

   다음 이미지는 1단계에서 설명하는 프로덕션 환경의 블루/그린 배포 예시입니다. 블루/그린 배포를 생성하는 동안, RDS는 기본 DB 인스턴스의 전체 토폴로지 및 구성을 복사하여 그린 환경을 만듭니다. 복사된 DB 인스턴스 이름 뒤에는 `-green-random-characters`가 추가됩니다. 이미지의 스테이징 환경에는 다중 AZ DB 인스턴스 배포(mydb1-green-*abc123*)와 읽기 전용 복제본(mydb2-green-*abc123*)이 포함됩니다.


Region
Production environment (blue)
Availability Zone
Amazon RDS
Amazon RDS DB instance
(mydb1)
Read/write access
Synchronous replication
Asynchronous replication
Availability Zone
Amazon RDS
Amazon RDS standby replica
Availability Zone
Amazon RDS
Amazon RDS read replica
(mydb2)
Read access
Production clients
Logical replication
Staging environment (green)
Availability Zone
Amazon RDS
Amazon RDS DB instance
(mydb1-green-abc123)
Read access
Synchronous replication
Asynchronous replication
Availability Zone
Amazon RDS
Amazon RDS standby replica
Availability Zone
Amazon RDS
Amazon RDS read replica
(mydb2-green-abc123)
Read access
Testing clients

블루/그린 배포를 생성하는 도중 DB 엔진 버전을 업그레이드하고, 그린 환경에 있는 DB 인스턴스에 다른 DB 파라미터 그룹을 지정할 수 있습니다. 또한 RDS는 블루 환경의 기본 DB 인스턴스에서 그린 환경의 기본 DB 인스턴스로의 논리적 복제를 구성합니다.

블루/그린 배포가 생성되면 그린 환경의 DB 인스턴스는 기본적으로 읽기 전용 상태가 됩니다.

3. 필요한 경우 스테이징 환경에 추가 변경 사항을 적용합니다.

   예를 들어 데이터베이스의 스키마를 변경하거나 그린 환경에 있는 하나 이상의 DB 인스턴스에서 사용하는 DB 인스턴스 클래스를 변경할 수 있습니다.

   DB 인스턴스 수정에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하세요.

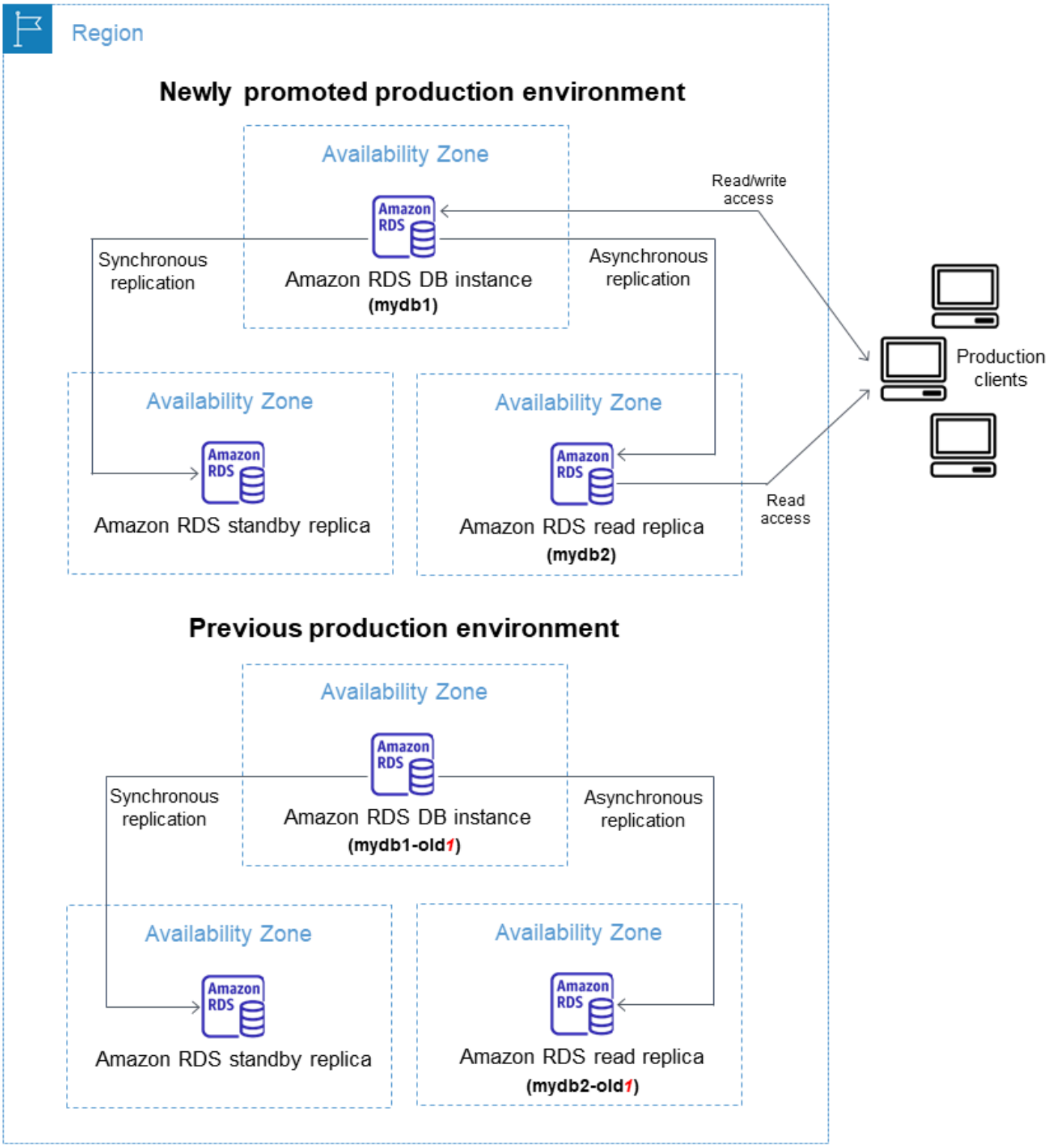
4. 스테이징 환경을 테스트합니다.

   테스트하는 동안에는 그린 환경의 데이터베이스를 읽기 전용으로 유지하는 것이 좋습니다. 복제 충돌이 발생할 수 있으므로 그린 환경에서 쓰기 작업을 사용 설정하세요. 전환 후 프로덕션 데이터베이스에 의도하지 않은 데이터가 저장될 수도 있습니다. RDS for MySQL에서 쓰기 작업을 활성화하려면 `read_only` 파라미터를 `0`으로 설정한 다음, DB 인스턴스를 재부팅합니다. RDS for PostgreSQL의 경우 `default_transaction_read_only` 파라미터를 세션 수준에서 `off`로 설정합니다.

5. 준비가 되면 전환하여 그린 환경을 새로운 프로덕션 환경으로 승격합니다. 지침은 Amazon RDS에서 블루/그린 배포 전환 단원을 참조하십시오.

   전환하면 다운타임이 발생하게 됩니다. 다운타임은 보통 1분 미만이지만 워크로드에 따라 더 길어질 수 있습니다.

   다음 이미지는 전환 이후의 DB 인스턴스를 보여줍니다.



전환이 끝나면 그린 환경에 있던 DB 인스턴스가 새로운 프로덕션 DB 인스턴스가 됩니다. 현재 프로덕션 환경의 이름과 엔드포인트가 새로 승격된 프로덕션 환경에 할당되므로, 애플리케이션을 변경할 필요가 없습니다. 따라서 이제 프로덕션 트래픽이 새 프로덕션 환경으로 이동합니다. 이전 블

루 환경의 DB 인스턴스는 현재 이름에 `-old`*n*이 추가됩니다(여기서 *n*은 숫자입니다). 예를 들어 블루 환경의 DB 인스턴스 이름이 `mydb1`이라고 가정하겠습니다. 전환이 끝나면 DB 인스턴스 이름은 `mydb1-old1`이 됩니다.

이미지의 예시에서는 전환 중에 다음과 같은 변경 사항이 발생합니다.

- 이름이 `mydb1-green-abc123`인 그린 환경 다중 AZ DB 인스턴스 배포가 이름이 `mydb1`인 프로덕션 다중 AZ DB 인스턴스 배포가 됩니다.
- 이름이 `mydb2-green-abc123`인 그린 환경의 읽기 전용 복제본이 이름이 `mydb2`인 프로덕션 읽기 전용 복제본이 됩니다.
- 이름이`mydb1`인 블루 환경 다중 AZ DB 인스턴스 배포가 `mydb1-old1`이 됩니다.
- 이름이 `mydb2`인 블루 환경 읽기 전용 복제본이 `mydb2-old1`이 됩니다.

6. 블루/그린 배포가 더 이상 필요하지 않다면 삭제해도 됩니다. 지침은 Amazon RDS에서 블루/그린 배포 삭제 단원을 참조하십시오.

   전환 후에도 이전 프로덕션 환경이 삭제되지 않으므로, 필요하다면 회귀 테스트에 사용할 수 있습니다.

Amazon RDS 블루/그린 배포 작업에 대한 액세스 권한 부여

사용자는 블루/그린 배포와 관련된 작업을 수행하는 데 필요한 권한이 있어야 합니다. 지정된 리소스에서 특정 API 태스크를 수행할 수 있는 권한을 사용자와 역할에게 부여하는 IAM 정책을 생성할 수 있습니다. 그런 다음 해당 권한이 필요한 IAM 권한 세트 또는 역할에 이러한 정책을 연결할 수 있습니다. 자세한 내용은 Amazon RDS의 자격 증명 및 액세스 관리 단원을 참조하십시오.

블루/그린 배포를 만드는 사용자는 다음 RDS 작업을 수행할 권한이 있어야 합니다.

- `rds:CreateBlueGreenDeployment`
- `rds:AddTagsToResource`
- `rds:CreateDBInstanceReadReplica`

블루/그린 배포로 전환하는 사용자는 다음 RDS 작업을 수행할 권한이 있어야 합니다.

- `rds:SwitchoverBlueGreenDeployment`
- `rds:ModifyDBInstance`
- `rds:PromoteReadReplica`

블루/그린 배포를 삭제하는 사용자는 다음 RDS 작업을 수행할 권한이 있어야 합니다.

- `rds:DeleteBlueGreenDeployment`
- `rds:DeleteDBInstance`

Amazon RDS는 사용자를 대신하여 스테이징 환경에서 리소스를 프로비저닝하고 수정합니다. 이러한 리소스에는 내부적으로 정의된 명명 규칙을 사용하는 DB 인스턴스가 포함됩니다. 따라서 연결된 IAM 정책에는 `my-db-prefix-*`와 같은 부분적인 리소스 이름 패턴이 포함될 수 없습니다. 와일드카드(*)만 지원됩니다. 일반적으로 와일드카드 대신 리소스 태그와 기타 지원되는 속성을 사용하여 이러한 리소스에 대한 액세스를 제어하는 것이 좋습니다. 자세한 내용은 Amazon RDS에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

Amazon RDS 블루/그린 배포 관련 제한 사항 및 고려 사항

Amazon RDS에서 블루/그린 배포를 하려면 복제 슬롯, 리소스 관리, 인스턴스 크기 조정, 데이터베이스 성능에 미치는 잠재적 영향과 같은 요소를 신중하게 고려해야 합니다. 다음 섹션은 데이터베이스 환경의 가동 중지 시간을 최소화하고, 원활한 전환을 보장하고, 효과적으로 관리할 수 있도록 배포 전략을 최적화하는 데 도움이 되는 지침을 제공합니다.

주제

- 블루/그린 배포 관련 제한 사항
- 블루/그린 배포 관련 고려 사항

블루/그린 배포 관련 제한 사항

블루/그린 배포에는 다음과 같은 제한 사항이 적용됩니다.

주제

- 블루/그린 배포 관련 일반 제한 사항
- 블루/그린 배포에 대한 RDS for MySQL 제한 사항
- 물리적 복제를 사용하는 블루/그린 배포에 대한 RDS for PostgreSQL 제한 사항
- 논리적 복제를 사용하는 블루/그린 배포에 대한 RDS for PostgreSQL 제한 사항

블루/그린 배포 관련 일반 제한 사항

블루/그린 배포에는 다음과 같은 일반 제한 사항이 적용됩니다.

- 블루/그린 배포의 경우 AWS Secrets Manager에서 마스터 사용자 암호 관리를 지원하지 않습니다.
- 블루 데이터베이스에서 전용 로그 볼륨(DLV)이 활성화된 경우, 읽기 복제본을 포함한 모든 DB 인스턴스에서 활성화해야 합니다.
- 전환 중에는 블루 및 그린 환경을 Amazon Redshift와 제로 ETL로 통합할 수 없습니다. 먼저 통합을 삭제하고 전환한 다음 통합을 다시 생성해야 합니다.
- 블루/그린 배포를 생성할 때는 그린 환경에서 Event Scheduler(`event_scheduler` 파라미터)를 비활성화해야 합니다. 이렇게 하면 그린 환경에서 이벤트가 생성되어 불일치가 발생하는 것을 방지할 수 있습니다.
- 암호화되지 않은 DB 인스턴스 는 암호화된 DB 인스턴스로 변경할 수 없습니다. 더욱이 암호화된 DB 인스턴스 는 암호화되지 않은 DB 인스턴스로 변경할 수 없습니다.
- 블루 DB 인스턴스 를 해당하는 그린 DB 인스턴스보다 높은 엔진 버전으로 변경할 수 없습니다.
- 블루 환경과 그린 환경의 리소스는 동일한 AWS 계정에 있어야 합니다.
- 블루/그린 배포는 다음 기능에는 지원되지 않습니다.
  - Amazon RDS Proxy
  - 계단식 읽기 전용 복제본
  - 리전 간 읽기 전용 복제본
  - AWS CloudFormation
  - 다중 AZ DB 클러스터 배포

    블루/그린 배포는 다중 AZ DB 인스턴스 배포에서 지원됩니다. 다중 AZ 배포에 대한 자세한 정보는 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 단원을 참조하세요.

블루/그린 배포에 대한 RDS for MySQL 제한 사항

RDS for MySQL 블루/그린 배포에는 다음과 같은 제한 사항이 적용됩니다.

- 블루 DB 인스턴스는 외부 binlog 복제본이 될 수 없습니다.
- 소스 데이터베이스가 사용자 지정 옵션 그룹과 연결된 경우, 블루/그린 배포를 생성할 때 메이저 버전 업그레이드를 지정할 수 없습니다.

  이 경우 메이저 버전 업그레이드를 지정하지 않고 블루/그린 배포를 생성할 수 있습니다. 그런 다음 그린 환경에서 데이터베이스를 업그레이드하면 됩니다. 자세한 내용은 DB 인스턴스 엔진 버전 업그레이드 섹션을 참조하세요.
- 블루/그린 배포는 MySQL용 AWS JDBC 드라이버를 지원하지 않습니다. 자세한 내용은 GitHub의 Known Limitations를 참조하세요.

물리적 복제를 사용하는 블루/그린 배포에 대한 RDS for PostgreSQL 제한 사항

물리적 복제를 사용하는 RDS for PostgreSQL 블루/그린 배포에는 다음과 같은 제한 사항이 적용됩니다. 블루/그린 배포가 논리적 복제 대신 물리적 복제를 사용하는 상황에 대한 설명은 ??? 섹션을 참조하세요.

- 그린 환경이 만들어진 후에는 수동 메이저 버전 업그레이드를 수행할 수 없습니다.
- 물리적 복제를 사용하는 블루/그린 배포는 엄격하게 읽기 전용이므로 그린 환경에서 스키마 변경을 지원하지 않습니다.
- 블루 DB 인스턴스는 논리적 소스(게시자) 또는 복제본(구독자)일 수 없습니다.

논리적 복제를 사용하는 블루/그린 배포에 대한 RDS for PostgreSQL 제한 사항

다음 제한 사항은 논리적 복제를 사용하는 RDS for PostgreSQL 블루/그린 배포에 적용됩니다. 블루/그린 배포가 물리적 복제 대신 논리적 복제를 사용하는 상황에 대한 설명은 ??? 섹션을 참조하세요.

- .
- 블루 DB 인스턴스는 논리적 소스(게시자) 또는 복제본(구독자)일 수 없습니다.
- 블루 DB 인스턴스가 외부 데이터 래퍼(FDW) 확장의 외부 서버로 구성된 경우 IP 주소 대신 인스턴스 엔드포인트 이름을 사용해야 합니다. 이렇게 하면 전환 후에도 구성이 계속 작동합니다.
- 블루/그린 배포에서는 각 데이터베이스에 논리적 복제 슬롯이 필요합니다. 데이터베이스 수가 늘어나면 리소스 오버헤드가 증가하고, 특히 DB 인스턴스 규모가 충분히 조정되지 않은 경우 복제 지연이 발생할 수 있습니다. 영향은 데이터베이스 워크로드 및 연결 수와 같은 요인에 따라 달라집니다. 이를 완화하려면 DB 인스턴스 클래스 규모를 조정하거나 소스 인스턴스의 데이터베이스 수를 줄이는 것이 좋습니다.
- 그린 환경의 논리적 복제 적용 프로세스는 단일 스레드입니다. 블루 환경이 대량의 쓰기 트래픽을 생성하는 경우 그린 환경이 따라가지 못할 수 있습니다. 이로 인해 복제 지연 또는 실패가 발생할 수 있으며, 특히 쓰기 처리량이 지속적으로 높은 워크로드의 경우 더욱 그렇습니다. 워크로드를 철저히 테스트해야 합니다. 메이저 버전 업그레이드와 대용량 쓰기 워크로드 처리가 필요한 시나리오의 경우 AWS Database Migration Service(AWS DMS)를 사용하는 등의 대체 접근 방식을 고려합니다.
- RDS for PostgreSQL에 대한 블루/그린 배포 중에는 파티셔닝된 테이블에 새 파티션을 생성할 수 없습니다. 새 파티션을 생성하려면 블루 환경에서 그린 환경으로 복제되지 않는 `CREATE TABLE` 등의 데이터 정의 언어(DDL) 작업이 필요합니다. 그러나 기존에 파티셔닝된 테이블과 해당 데이터는 그린 환경에 복제됩니다.
- PostgreSQL 확장에는 다음과 같은 제한 사항이 적용됩니다.

- 블루/그린 배포를 만들 때는 블루 환경에서 pg_partman 확장을 비활성화해야 합니다. 확장은 블루 환경에서 그린 환경으로의 논리적 복제를 중단하는 CREATE TABLE과 같은 DDL 작업을 수행합니다.
- 블루/그린 배포를 생성한 후에는 모든 그린 데이터베이스에서 pg_cron 확장을 비활성화한 상태로 유지해야 합니다. 확장에는 슈퍼유저로 실행되고 그린 환경의 읽기 전용 설정을 우회하는 백그라운드 작업자가 있어 복제 충돌이 발생할 수 있습니다.
- 블루/그린 배포를 만들 때는 블루 환경에서 pglogical 및 pgactive 확장을 비활성화해야 합니다. 그린 환경을 새로운 프로덕션 환경으로 전환한 후 확장 기능을 다시 활성화할 수 있습니다. 또한 블루 데이터베이스는 외부 인스턴스의 논리적 구독자가 될 수 없습니다.
- pgAudit 확장을 사용하는 경우, 해당 확장은 블루 및 그린 DB 인스턴스 모두에 대한 사용자 지정 DB 파라미터 그룹의 공유 라이브러리(shared_preload_libraries)에 남아 있어야 합니다. 자세한 내용은 the section called “pgAudit 확장 설정” 섹션을 참조하세요.

블루/그린 배포의 논리적 복제 관련 제한 사항

PostgreSQL에는 논리적 복제와 관련된 몇 가지 제한 사항이 있으며, 이로 인해 RDS for PostgreSQL DB 인스턴스에 대한 블루/그린 배포를 생성할 때 제한이 적용됩니다.

다음 표에는 RDS for PostgreSQL의 블루/그린 배포에 적용되는 논리적 복제 제한 사항이 설명되어 있습니다. 자세한 내용은 PostgreSQL 논리적 복제 설명서의 제한 사항을 참조하세요.

제한 사항	설명
CREATE TABLE 및 CREATE SCHEMA와 같은 데이터 정의 언어(DDL) 문은 블루 환경에서 그린 환경으로 복제되지 않습니다.	Amazon RDS가 블루 환경에서 DDL 변경을 감지하면 그린 데이터베이스는 복제 성능 저하 상태로 전환됩니다. 블루/그린 배포와 모든 그린 데이터베이스를 삭제한 다음 다시 생성해야 합니다.
GRANT 및 REVOKE와 같은 데이터 제어 언어(DCL) 문은 블루 환경에서 그린 환	Amazon RDS PostgreSQL이 블루 환경에서 DCL 문 실행 시도를 감지하면 경고 메시지가 표시됩니다. 블루/그린 배포 프로세스의 제한 사항이므로 이 동작을 변경하는 데 사용할 수 있는 구성 또는 API는 없습니다.

제한 사항	설명
경으로 복제되지 않습니다.	
시퀀스 객체에 대한 NEXTVAL 작업은 블루 환경과 그린 환경 간에 동기화되지 않습니다.	전환 중에 Amazon RDS는 그린 환경의 시퀀스 값을 증가시켜 블루 환경의 시퀀스 값과 일치하도록 합니다. 수천 개의 시퀀스가 있는 경우 전환이 지연될 수 있습니다.
블루 환경의 대형 객체는 그린 환경에 복제되지 않습니다. 여기에는 기존 대형 객체와 블루/그린 배포 프로세스 중에 새로 생성되거나 수정된 대형 객체가 모두 포함됩니다.	Amazon RDS가 블루 환경에서 **pg_largeobject** 시스템 테이블에 저장된 대형 객체의 생성 또는 수정을 감지하면 그린 데이터베이스는 복제 성능 저하 상태로 전환됩니다. 블루/그린 배포와 모든 그린 데이터베이스를 삭제한 다음 다시 생성해야 합니다.
그린 환경에서는 구체화된 뷰가 자동으로 새로 고쳐지지 않습니다.	블루 환경에서 구체화된 뷰를 새로 고쳐도 그린 환경에서는 새로 고쳐지지 않습니다. 전환 후에는 REFRESH MATERIALIZED VIEW 명령을 사용하여 수동으로 새로 고치거나 새로 고침 일정을 예약할 수 있습니다.
프라이머리프라이머리 키가 없는 테이블에서는 UPDATE 및 DELETE 작업이 허용되지 않습니다.	블루/그린 배포를 생성하기 전에 모든 테이블에 프라이머리 키가 있거나 REPLICA IDENTITY FULL을 사용하는지 확인하세요. 그러나 프라이머리 키 또는 고유 키가 없는 경우 REPLICA IDENTITY FULL만 사용합니다. 복제 성능에 영향을 미치기 때문입니다. 자세한 내용은 PostgreSQL 설명서를 참조하세요.

블루/그린 배포 관련 고려 사항

Amazon RDS는 각 리소스의 `DbiResourceId`를 사용하여 블루/그린 배포의 리소스를 추적합니다. 이 리소스 ID는 AWS 리전별로 고유한, 리소스의 변경 불가능한 식별자입니다.

리소스 ID는 DB 인스턴스 ID와 다릅니다. 각 항목은 RDS 콘솔의 데이터베이스 구성에 나열되어 있습니다.

블루/그린 배포로 전환하면 리소스 이름(인스턴스스 ID)이 변경되지만, 각 리소스는 동일한 리소스 ID를 유지합니다. 예를 들어 DB 인스턴스 식별자는 블루 환경에서는 `mydb`입니다. 전환이 끝나면 동일한 DB 인스턴스의 이름이 `mydb-old1`으로 변경됩니다. 하지만 DB 인스턴스의 리소스 ID는 전환 중에 변경되지 않습니다. 따라서 그린 리소스를 새 프로덕션 리소스로 전환하면 관련 리소스 ID는 이전에 프로덕션에 있었던 블루 리소스 ID와 일치하지 않게 됩니다.

블루/그린 배포로 전환한 후에는, 리소스 ID를 프로덕션 리소스와 함께 사용한 통합형 기능 및 서비스를 위해 새로 전환된 프로덕션 리소스의 ID로 업데이트하는 것을 고려해 보세요. 구체적으로는 다음 업데이트를 고려해 보십시오.

- RDS API 및 리소스 ID를 사용하여 필터링을 수행할 경우, 전환 후 필터링에 사용한 리소스 ID를 조정하세요.
- 리소스 감사에 CloudTrail을 사용한다면, 전환 후 새 리소스 ID를 추적하도록 CloudTrail의 소비자를 조정하십시오. 자세한 내용은 AWS CloudTrail에서 Amazon RDS API 호출 모니터링 섹션을 참조하세요.
- Performance Insights API를 사용한다면, 전환 후 API 호출의 리소스 ID를 조정하십시오. 자세한 내용은 성능 개선 도우미를 통한 Amazon RDS 모니터링 섹션을 참조하세요.

  전환 후 동일한 이름의 데이터베이스를 모니터링할 수 있지만, 전환 전의 데이터는 포함되지 않습니다.
- IAM 정책에서 리소스 ID를 사용한다면, 필요한 경우 새로 전환된 리소스의 리소스 ID를 추가해야 합니다. 자세한 내용은 Amazon RDS의 자격 증명 및 액세스 관리 섹션을 참조하세요.
- DB 인스턴스와 연결된 IAM 역할이 있는 경우 전환 후 해당 역할을 다시 연결해야 합니다. 연결된 역할은 그린 환경으로 자동 복사되지 않습니다.
- IAM 데이터베이스 인증을 사용하여 DB 인스턴스를 인증하는 경우, 데이터베이스 액세스에 사용되는 IAM 정책의 `Resource` 요소 아래에 블루 데이터베이스와 그린 데이터베이스가 모두 나열되어 있는지 확인하세요. 이는 전환 후 그린 데이터베이스에 연결하기 위해 필요합니다. 자세한 내용은 the section called “IAM 데이터베이스 액세스를 위한 IAM 정책 생성 및 사용” 섹션을 참조하세요.

- 블루/그린 배포에서 리소스의 자동 백업을 관리하는 데 AWS Backup를 사용한다면, 전환 후에 AWS Backup에서 사용하는 리소스 ID를 조정하십시오. 자세한 내용은 AWS Backup를 사용하여 Amazon RDS에 대한 자동 백업 관리 섹션을 참조하세요.
- 블루/그린 배포의 일부였던 DB 인스턴스의 수동 또는 자동 DB 스냅샷을 복원하려면, 스냅샷을 촬영한 시간을 확인하여 올바른 DB 스냅샷을 복원해야 합니다. 자세한 내용은 DB 인스턴스 복원 섹션을 참조하세요.
- 이전 블루 환경 DB 인스턴스 자동 백업을 설명하거나 특정 시점으로 복원하려면, 작업의 리소스 ID를 사용해야 합니다.

  전환 중에 DB 인스턴스의 이름이 변경되므로 `DescribeDBInstanceAutomatedBackups` 또는`RestoreDBInstanceToPointInTime` 작업에 이전 이름을 사용해선 안 됩니다.

  자세한 내용은 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 섹션을 참조하세요.
- 블루/그린 배포의 그린 환경에 있는 DB 인스턴스에 읽기 전용 복제본을 추가하면, 새 읽기 전용 복제본은 전환할 때 블루 환경의 읽기 전용 복제본을 대체하지 않습니다. 하지만 새 읽기 전용 복제본은 전환 후에도 새 프로덕션 환경에 보존됩니다.
- 전환한 후에는 블루 환경의 체크포인트가 그린 환경에서 유효하지 않기 때문에 AWS Database Migration Service(AWS DMS) 복제 작업을 재개할 수 없습니다. 복제를 계속하려면 새 체크포인트로 DMS 작업을 다시 만들어야 합니다.
- 블루/그린 배포의 그린 환경에 있는 DB 인스턴스를 삭제하면, 블루/그린 배포에서 이를 대체할 새 DB 인스턴스를 만들 수 없습니다.

  삭제된 DB 인스턴스와 동일한 이름 및 Amazon 리소스 이름(ARN)으로 새 DB 인스턴스를 생성하면, `DbiResourceId`가 다르기 때문에 그린 환경에 속하지 않게 됩니다.

  그린 환경에서 DB 인스턴스를 삭제하면 다음과 같은 동작이 발생합니다.
  - 블루 환경에 이름이 같은 DB 인스턴스가 있는 경우, 이 인스턴스는 그린 환경의 DB 인스턴스로 전환되지 않습니다. 이 DB 인스턴스는 DB 인스턴스 이름에 `-old`*n*을 추가하여 이름이 바뀌지 않습니다.
  - 블루 환경의 DB 인스턴스를 가리키는 모든 애플리케이션은 전환 후에도 동일한 DB 인스턴스를 계속 사용합니다.

  DB 인스턴스와 읽기 전용 복제본에도 동일한 동작이 적용됩니다.

Amazon RDS 블루/그린 배포 모범 사례

다음은 블루/그린 배포 모범 사례입니다.

주제

- 블루/그린 배포 일반 모범 사례
- 블루/그린 배포에 대한 RDS for MySQL 모범 사례
- 블루/그린 배포 모범 사례에 대한 RDS for PostgreSQL 모범 사례

블루/그린 배포 일반 모범 사례

블루/그린 배포를 만들 때 다음 일반 모범 사례를 고려하세요.

- 전환하기 전에 DB 인스턴스를 철저하게 테스트합니다.
- 그린 환경에서 데이터베이스를 읽기 전용으로 유지합니다. 복제 충돌이 발생할 수 있으므로 그린 환경에서 쓰기 작업을 사용 설정할 때는 신중해야 합니다. 전환 후 프로덕션 데이터베이스에 의도하지 않은 데이터가 저장될 수도 있습니다.
- 블루/그린 배포를 사용하여 스키마 변경을 구현할 때는 복제와 호환되는 변경만 적용합니다.

  예를 들어 블루 배포에서 그린 배포로의 복제를 중단하지 않고 테이블 끝에 새 열을 추가할 수 있습니다. 그러나 열 이름 변경이나 테이블 이름 변경 같은 스키마 변경은 그린 배포로의 복제 중단을 유발합니다.

  복제 호환 변경 사항에 대한 자세한 내용은 MySQL 설명서의 소스 및 복제본에서 서로 다른 테이블 정의를 사용하는 복제 및 PostgreSQL 논리적 복제 설명서의 제한 사항을 참조하세요.

  > **Note**
  >
  > 이 제한 사항은 물리적 복제를 사용하는 RDS for PostgreSQL 블루/그린 배포에는 적용되지 않습니다. 자세한 내용은 the section called “물리적 복제를 사용하는 블루/그린 배포에 대한 RDS for PostgreSQL 제한 사항” 섹션을 참조하세요.

- 블루/그린 배포를 생성한 후 필요하다면 지연 로딩을 처리합니다. 전환하기 전에 데이터 로드가 완료되었는지 확인합니다. 자세한 내용은 블루/그린 배포를 위한 지연 로딩 및 스토리지 초기화 섹션을 참조하세요.
- 블루/그린 배포로 전환할 때는 전환 모범 사례를 따르세요. 자세한 내용은 the section called “전환 모범 사례” 섹션을 참조하세요.

블루/그린 배포에 대한 RDS for MySQL 모범 사례

RDS for MySQL DB 인스턴스에서 블루/그린 배포를 만들 때 다음 모범 사례를 고려하세요.

- 복제에 최적화되지 않은 MyISAM 같은 비트랜잭션 스토리지 엔진 사용은 자제하시기 바랍니다.
- 이진 로그 복제용 읽기 전용 복제본 및 그린 환경을 최적화합니다. DB 엔진에서 지원하는 경우 GTID, 병렬 및 충돌 방지 복제를 활성화하여 블루/그린 배포를 생성하기 전에 데이터 일관성과 내구성을 보장합니다. 자세한 내용은 GTID 기반 복제 사용 섹션을 참조하세요.
- 그린 환경에 복제본 지연이 발생하는 경우 다음을 고려하세요.
  - 그린 DB 파라미터 그룹에서 `innodb_flush_log_at_trx_commit` 파라미터를 일시적으로 2으로 설정합니다. 복제가 최신 상태로 동기화된 후 전환하기 전에 1의 기본값으로 되돌립니다. 임시 파라미터 값에서 예기치 않은 종료 또는 충돌이 발생하는 경우 탐지되지 않은 데이터 손상을 방지하기 위해 그린 환경을 다시 빌드합니다.
  - 쓰기 지연 시간을 줄이고 복제 처리량을 개선하기 위해 그린 다중 AZ DB 인스턴스를 일시적으로 단일 AZ DB 인스턴스로 변경합니다. 전환 직전에 다중 AZ를 다시 활성화합니다.

블루/그린 배포 모범 사례에 대한 RDS for PostgreSQL 모범 사례

RDS for PostgreSQL DB 인스턴스에서 블루/그린 배포를 만들 때 다음 모범 사례를 고려하세요.

주제

- 블루/그린 배포에 대한 RDS for PostgreSQL 일반 모범 사례
- 물리적 복제를 사용하는 블루/그린 배포에 대한 RDS for PostgreSQL 모범 사례
- 논리적 복제를 사용하는 블루/그린 배포에 대한 RDS for PostgreSQL 모범 사례

블루/그린 배포에 대한 RDS for PostgreSQL 일반 모범 사례

RDS for PostgreSQL DB 인스턴스에서 블루/그린 배포를 만들 때 다음 일반 모범 사례를 고려하세요.

- 블루/그린 배포를 생성하기 전에 모든 PostgreSQL 확장을 최신 버전으로 업데이트합니다. 자세한 내용은 the section called “PostgreSQL 확장 버전 업그레이드” 섹션을 참조하세요.
- 트랜잭션이 오래 실행되면 심각한 복제 지연이 발생할 수 있습니다. 복제 지연을 줄이려면 다음과 같이 해 보세요.
  - 그린 환경이 블루 환경을 따라잡을 때까지 지연될 수 있는 장기 실행 트랜잭션을 줄이세요.
  - 그린 환경이 블루 환경을 따라잡을 때까지 블루 환경의 대량 작업을 줄이세요.
  - 블루/그린 배포를 생성하기 전에 사용량이 많은 테이블에서 수동 vacuum freeze 작업을 시작하세요.

- PostgreSQL 버전 12 이상의 경우, 크거나 사용량이 많은 테이블에서 `index_cleanup` 파라미터를 비활성화하여 블루 데이터베이스의 일반 유지 관리 속도를 높이세요. 자세한 내용은 the section called “테이블에 최대한 신속하게 Vacuum을 실행하는 방법” 섹션을 참조하세요.

  > **Note**
  >
  > Vacuum 중에 인덱스 정리를 정기적으로 건너뛰면 인덱스 팽창이 발생하여 스캔 성능이 저하될 수 있습니다. 블루/그린 배포를 사용하는 동안에만 이 접근 방식을 사용하는 것이 가장 좋습니다. 배포가 완료되면 정기적인 인덱스 유지 관리 및 정리를 재개하는 것이 좋습니다.

- 복제 속도가 느리면 발신자와 수신자가 자주 다시 시작되어 동기화가 지연될 수 있습니다. 이들의 활성 상태를 유지하려면 블루 환경에서는 `wal_sender_timeout` 파라미터를 `0`으로 설정하고 그린 환경에서는 `wal_receiver_timeout` 파라미터를 `0`으로 설정하여 시간 초과를 비활성화하세요.
- 미리 쓰기 로그(WAL) 세그먼트가 블루 환경에서 제거되지 않도록 하려면 PostgreSQL 버전 13 이하에서 `wal_keep_segments` 파라미터를 15625로 설정하세요. 버전 14 이상에서는 사용 가능한 스토리지 공간이 충분하면 `wal_keep_size` 파라미터를 1TiB로 설정하세요.

물리적 복제를 사용하는 블루/그린 배포에 대한 RDS for PostgreSQL 모범 사례

물리적 복제를 사용해 Amazon RDS는 소스 DB 인스턴스의 읽기 전용 복제본을 만듭니다. 관련 파라미터, 모니터링, 조정 및 문제 해결은 the section called “RDS for PostgreSQL의 읽기 전용 복제본 작업” 섹션을 참조하세요.

블루/그린 배포가 논리적 복제 대신 물리적 복제를 사용하는 상황에 대한 설명은 ??? 섹션을 참조하세요.

논리적 복제를 사용하는 블루/그린 배포에 대한 RDS for PostgreSQL 모범 사례

논리적 복제를 사용하는 블루/그린 배포를 만들 때 다음 모범 사례를 고려하세요. 블루/그린 배포가 물리적 복제 대신 논리적 복제를 사용하는 상황에 대한 설명은 ??? 섹션을 참조하세요.

- 데이터베이스에 사용 가능한 메모리가 충분할 경우 블루 환경에서 `logical_decoding_work_mem` DB 파라미터 값을 늘립니다. 이렇게 하면 디스크 디코딩이 줄어들고 대신 메모리가 사용됩니다. 자세한 내용은 PostgreSQL 설명서를 참조하세요.
  - `ReplicationSlotDiskUsage` CloudWatch 지표를 사용하여 디스크에 기록되는 트랜잭션 오버플로를 모니터링할 수 있습니다. 이 지표는 복제 슬롯의 디스크 사용량에 대한 인사이트를 제공하므로 트랜잭션 데이터가 메모리 용량을 초과하고 디스크에 저장되는 시점을 식별하는 데 도움이

됩니다. `FreeableMemory` CloudWatch 지표로 여유 메모리를 모니터링할 수 있습니다. 자세한 내용은 the section called “Amazon RDS에 대한 Amazon CloudWatch 지표” 섹션을 참조하세요.

  - RDS for PostgreSQL 버전 14 이상에서는 `pg_stat_replication_slots` 시스템 뷰를 사용하여 논리적 오버플로 파일의 크기를 모니터링할 수 있습니다.

- `aws_s3` 확장을 사용하는 경우 그린 환경이 만들어진 후 IAM 역할을 통해 그린 DB 인스턴스에 Amazon S3에 대한 액세스 권한을 부여합니다. 이렇게 하면 전환 후에도 가져오기 및 내보내기 명령이 계속 작동합니다. 지침은 the section called “Amazon S3 버킷에 대한 액세스 권한 설정” 섹션을 참조하세요.
- UPDATE 및 DELETE 문 성능을 검토하고 WHERE 절에서 사용되는 열에 인덱스를 생성하여 이러한 쿼리를 최적화할 수 있는지 평가합니다. 이렇게 하면 그린 환경에서 작업을 재생할 때 성능이 향상될 수 있습니다.
- 트리거를 사용하는 경우 이름이 'rds'로 시작하는 `pg_catalog.pg_publication`, `pg_catalog.pg_subscription` 및 `pg_catalog.pg_replication_slots` 객체를 만들고, 업데이트하고, 삭제하는 데 트리거가 방해가 되지 않는지 확인하세요.
- 그린 환경에 더 높은 엔진 버전을 지정하는 경우 모든 데이터베이스에서 `ANALYZE` 작업을 실행하여 `pg_statistic` 테이블을 새로 고칩니다. 옵티마이저 통계는 메이저 버전 업그레이드 중에 전송되지 않으므로 성능 문제를 방지하려면 모든 통계를 다시 생성해야 합니다. 메이저 버전 업그레이드 중 추가 모범 사례에 대해 알아보려면 the section called “메이저 버전 업그레이드를 수행하는 방법” 섹션을 참조하세요.
- 데이터를 조작하기 위해 트리거를 소스에 사용하는 경우 트리거를 `ENABLE REPLICA` 또는 `ENABLE ALWAYS`로 구성하지 마세요. 이렇게 구성하면 복제 시스템이 변경 내용을 전파하고 트리거를 실행하므로 중복이 발생합니다.

Amazon RDS에서 블루/그린 배포 생성

블루/그린 배포를 만들 때는 배포에서 복사할 소스 DB 인스턴스를 지정합니다. 선택한 DB 인스턴스는 프로덕션 DB 인스턴스이며, 블루 환경에서는 이것이 기본 DB 인스턴스가 됩니다. 이 DB 인스턴스는 그린 환경으로 복사되고, RDS는 블루 환경의 DB 인스턴스에서 그린 환경의 DB 인스턴스로의 복제를 구성합니다.

RDS는 블루 환경의 토폴로지와 기능을 스테이징 영역에 복사합니다. 블루 DB 인스턴스에 읽기 전용 복제본이 있는 경우, 그린 DB 인스턴스의 읽기 전용 복제본으로 복사됩니다. 모든 그린 복제본의 할당된 스토리지는 그린 기본 인스턴스와 일치하는 반면, 다른 스토리지 파라미터는 블루 복제본에서 상속됩니다.

블루 DB 인스턴스가 다중 AZ DB 인스턴스 배포인 경우, 블루 DB 인스턴스는 다중 AZ DB 인스턴스 배포로서 생성됩니다.

주제

- 블루/그린 배포 준비
- 블루/그린 배포 생성 시 변경 사항 지정
- 블루/그린 배포를 위한 지연 로딩 및 스토리지 초기화
- 블루/그린 배포 생성
- 블루/그린 배포를 생성하기 위한 설정

블루/그린 배포 준비

DB 인스턴스가 실행 중인 엔진에 따라 블루/그린 배포를 생성하기 전에 수행해야 하는 몇 가지 단계가 있습니다.

주제

- 블루/그린 배포를 위해 RDS for MySQL 또는 RDS for MariaDB DB 인스턴스 준비
- 물리적 복제를 사용하는 블루/그린 배포를 위해 RDS for PostgreSQL DB 인스턴스 준비
- 논리적 복제를 사용하는 블루/그린 배포를 위해 RDS for PostgreSQL DB 인스턴스 준비

블루/그린 배포를 위해 RDS for MySQL 또는 RDS for MariaDB DB 인스턴스 준비

RDS for MySQL 또는 RDS for MariaDB DB 인스턴스에 블루/그린 배포를 만들기 전에 먼저 자동 백업을 활성화해야 합니다. 지침은 the section called “자동 백업 활성화” 섹션을 참조하세요.

물리적 복제를 사용하는 블루/그린 배포를 위해 RDS for PostgreSQL DB 인스턴스 준비

물리적 복제를 사용하는 RDS for PostgreSQL 블루/그린 배포를 만들기 전에 다음 작업을 수행해야 합니다. 물리적 복제와 논리적 복제를 사용하는 버전의 비교 목록은 ??? 섹션을 참조하세요.

- DB 인스턴스에서 자동 백업을 사용 설정합니다. 지침은 the section called “자동 백업 활성화” 섹션을 참조하세요.
- DB 인스턴스가 외부 복제의 소스 또는 대상이 아닌지 확인하세요. 자세한 내용은 the section called “일반 제한 사항” 섹션을 참조하세요.

논리적 복제를 사용하는 블루/그린 배포를 위해 RDS for PostgreSQL DB 인스턴스 준비

논리적 복제를 사용하는 RDS for PostgreSQL 블루/그린 배포를 만들기 전에 다음 작업을 수행해야 합니다. 논리적 복제와 물리적 복제를 사용하는 버전의 비교 목록은 ??? 섹션을 참조하세요.

- 논리적 복제(`rds.logical_replication`)가 켜진 상태에서 인스턴스를 사용자 지정 DB 파라미터 그룹과 연결합니다. 블루 환경에서 그린 환경으로 복제하려면 논리적 복제가 필요합니다. 지침은 the section called “DB 파라미터 그룹의 파라미터 수정” 섹션을 참조하세요.

  블루/그린 배포에는 데이터베이스당 최소 1명의 백그라운드 작업자가 필요하므로, 워크로드에 따라 다음 구성 설정을 조정해야 합니다. 각 설정을 조정하는 방법에 대한 지침은 PostgreSQL 설명서의 구성 설정을 참조하세요.

  - `max_replication_slots`
  - `max_wal_senders`
  - `max_logical_replication_workers`
  - `max_worker_processes`

  논리적 복제를 활성화하고 모든 구성 옵션을 설정한 후에는 DB 인스턴스를 재부팅하여 변경 사항을 적용해야 합니다. 블루/그린 배포의 요구 사항은 DB 인스턴스가 DB 파라미터 그룹과 동기화되는 것이며, 그렇지 않으면 생성에 실패합니다. 자세한 내용은 the section called “DB 인스턴스 재부팅” 섹션을 참조하세요.
- DB 인스턴스가 외부 복제의 소스 또는 대상이 아닌지 확인하세요. 자세한 내용은 the section called “일반 제한 사항” 섹션을 참조하세요.
- DB 인스턴스의 모든 테이블에 프라이머리 키가 있는지 확인하세요. PostgreSQL 논리적 복제는 프라이머리 키가 없는 테이블에 대한 `UPDATE` 또는 `DELETE` 작업을 허용하지 않습니다.
- RDS for PostgreSQL은 PostgreSQL의 기본 논리적 복제를 사용하여 그린 환경에서 재생될 때까지 미리 쓰기 로그(WAL) 세그먼트를 블루 인스턴스에 저장합니다. 블루/그린 배포를 만들기 전에 다음 지표를 확인하여 블루 인스턴스의 용량이 적절한지 확인합니다.

  - `FreeStorageSpace`
  - `TransactionLogsGeneration`
  - `TransactionLogsDiskUsage`
  - `OldestReplicationSlotLag`

  블루 인스턴스에 필요한 추가 스토리지를 추정하려면 피크 워크로드 기간 동안 `TransactionLogsGeneration` CloudWatch 지표를 모니터링합니다. 예를 들어 워크로드가 24시간 동안 100GB의 WAL 데이터를 생성하는 경우 하루 분량의 WAL 세그먼트를 수용할 수 있는 최소

100GB의 추가 스토리지가 있는지 확인합니다. 자세한 내용은 DB 인스턴스에서 지표 모니터링 섹션을 참조하세요.

블루/그린 배포 생성 시 변경 사항 지정

블루/그린 배포를 생성할 때 그린 환경의 DB 인스턴스에 다음 변경 사항을 적용할 수 있습니다.

배포가 끝나면 그린 환경의 DB 인스턴스 추가로 수정할 수 있습니다. 예를 들어 더 높은 엔진 버전 또는 다른 파라미터 그룹을 지정할 수 있습니다.

DB 인스턴스 수정에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하세요.

주제

- 더 높은 엔진 버전 지정
- 다른 DB 파라미터 그룹 지정
- 스토리지 및 성능 설정 수정
- RDS 최적화된 쓰기 활성화
- 스토리지 구성 업그레이드

더 높은 엔진 버전 지정

DB 엔진 업그레이드를 테스트하고 싶다면 더 높은 엔진 버전을 지정할 수 있습니다. 전환 시 데이터베이스는 지정한 메이저 또는 마이너 DB 엔진 버전으로 업그레이드됩니다.

다른 DB 파라미터 그룹 지정

그린 환경에서 파라미터 변경이 DB 인스턴스에 미치는 영향을 테스트하거나, 업그레이드할 때 새 메이저 DB 엔진 버전에 대한 파라미터 그룹을 지정할 수 있습니다.

다른 DB 파라미터 그룹을 지정하면, 지정된 DB 파라미터 그룹이 그린 환경의 모든 DB 인스턴스와 연결됩니다. 다른 파라미터 그룹을 지정하지 않으면 그린 환경의 각 DB 인스턴스는 대응하는 블루 DB 인스턴스의 파라미터 그룹과 연결됩니다.

스토리지 및 성능 설정 수정

그린 환경에서 스토리지 및 성능 설정을 조정하여 리소스 할당을 최적화합니다. 이러한 설정에는 할당된 스토리지, 프로비저닝된 IOPS, 스토리지 유형 및 스토리지 처리량(gp3 스토리지용)이 포함됩니다.

그린 DB 인스턴스의 스토리지 유형을 gp2, gp3, io1 또는 io2로 변경할 수 있습니다. gp3 스토리지의 경우 스토리지 처리량을 조정하여 수요가 많은 워크로드의 데이터 전송 성능을 향상하거나 덜 집약적인 애플리케이션의 비용을 절감할 수도 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 섹션을 참조하세요.

그린 환경에서 할당된 스토리지를 늘리거나 줄이도록 선택할 수도 있습니다. 그러나 스토리지 축소는 할당된 대상 스토리지가 현재 스토리지 사용량보다 20% 이상 많은 경우에만 발생합니다. 할당된 스토리지를 줄이면 Amazon RDS가 스토리지 구성 업그레이드를 시작합니다. 자세한 내용은 the section called “스토리지 구성 업그레이드” 섹션을 참조하세요.

블루 DB 인스턴스가 마그네틱 스토리지를 사용하는 경우 할당된 스토리지를 늘리거나 줄이려면 그린 DB 인스턴스를 범용 또는 프로비저닝된 IOPS 스토리지 유형으로 변경해야 합니다.

RDS 최적화된 쓰기 활성화

RDS 최적화된 쓰기를 지원하는 DB 인스턴스 클래스로 업그레이드하는 데 블루/그린 배포를 사용할 수 있습니다. 지원되는 DB 인스턴스 클래스로 생성된 데이터베이스에서만 RDS 최적화된 쓰기를 활성화할 수 있습니다. 따라서 이 옵션을 사용하면 지원되는 DB 인스턴스 클래스를 사용하는 그린 데이터베이스가 생성되며, 이를 통해 그린 DB 인스턴스에 RDS 최적화된 쓰기를 활성화할 수 있습니다.

RDS 최적화된 쓰기를 지원하지 않는 DB 인스턴스 클래스에서 지원하는 클래스로 업그레이드하는 경우 그린 DB 인스턴스의 스토리지 구성도 업그레이드해야 합니다. 자세한 내용은 the section called “스토리지 구성 업그레이드” 섹션을 참조하세요.

기본 그린 DB 인스턴스의 DB 인스턴스 클래스만 업그레이드할 수 있습니다. 그린 환경의 읽기 전용 복제본은 블루 환경의 DB 인스턴스 설정을 기본적으로 상속합니다. 그린 환경을 성공적으로 만든 후에는 그린 환경에서 읽기 전용 복제본의 DB 인스턴스 클래스를 수동으로 수정해야 합니다.

블루 DB 인스턴스의 엔진 버전과 인스턴스 클래스에 따라 일부 인스턴스 클래스 업그레이드가 지원되지 않습니다. DB 인스턴스 클래스에 대한 자세한 내용은 the section called “DB 인스턴스 클래스” 섹션을 참조하세요.

스토리지 구성 업그레이드

블루 데이터베이스가 최신 스토리지 구성에 없는 경우 RDS는 그린 DB 인스턴스를 이전 스토리지 구성(32비트 파일 시스템)에서 원하는 구성으로 마이그레이션할 수 있습니다. RDS 블루/그린 배포를 사용하면 이전 32비트 파일 시스템의 스토리지 및 파일 크기 조정 제한을 극복할 수 있습니다. 또한 이 설정은 지정된 DB 인스턴스 클래스가 최적화된 쓰기를 지원하는 경우 RDS 최적화된 쓰기와 호환되도록 스토리지 구성을 변경합니다.

> **Note**
>
> 스토리지 구성 업그레이드는 I/O 집약적인 작업이므로 블루/그린 배포의 생성 시간이 길어집니다. 블루 DB 인스턴스가 프로비저닝된 IOPS SSD(io1 또는 io2 Block Express) 스토리지를 사용하고 그린 환경을 인스턴스 크기가 4xlarge 이상이 되도록 프로비저닝한 경우 스토리지 업그레이드 프로세스가 더 빨라집니다. 범용 SSD(gp2) 스토리지를 사용하는 스토리지 업그레이드는 I/O 크레딧 밸런스를 고갈할 수 있어 업그레이드 시간이 더 오래 걸릴 수 있습니다. 자세한 내용은 the section called “DB 인스턴스 스토리지” 섹션을 참조하세요.
> 스토리지 업그레이드 중에는 그린 DB 인스턴스를 일시적으로 사용할 수 없는 반면 블루 DB 인스턴스는 계속 사용할 수 있습니다. 이 시간 동안 복제가 일시 중지됩니다. 블루 인스턴스의 스토리지를 모니터링하고, 업그레이드 후 그린 인스턴스가 자동으로 10% 확장되므로 스토리지가 90%에 도달하면 규모 조정을 고려하세요.

이 옵션은 블루 데이터베이스가 최신 스토리지 구성을 사용하지 않거나 동일한 요청으로 DB 인스턴스 클래스를 변경하는 경우에만 사용할 수 있습니다. 블루/그린 배포를 처음 생성할 때는 스토리지 구성만 업그레이드할 수 있습니다.

블루/그린 배포를 위한 지연 로딩 및 스토리지 초기화

블루/그린 배포를 생성하면 Amazon RDS는 DB 스냅샷에서 복원하여 그린 환경에서 기본 DB 인스턴스를 생성합니다. 만들어진 그린 DB 인스턴스와 해당 읽기 전용 복제본은 지연 로딩이라는 프로세스를 통해 백그라운드에서 데이터를 계속 로드합니다.

지연 로딩은 애플리케이션이 데이터 블록을 요청할 때만 로드합니다. 아직 로드하지 않은 데이터에 액세스하려고 하면 Amazon EBS는 Amazon S3에서 즉시 데이터를 검색하지만 나머지 데이터는 백그라운드에서 계속 로드됩니다. 자세한 내용은 Amazon EBS 스냅샷을 참조하세요.

전체 볼륨 성능을 가속화하기 위해 Amazon RDS는 그린 환경 볼륨의 모든 블록을 읽는 스토리지 초기화를 제공합니다. Amazon EBS는 Amazon S3에서 블록을 사전에 다운로드하여 처음 사용할 때부터 최대 볼륨 성능을 제공합니다. 스토리지 초기화는 전적으로 백그라운드에서 수행되므로 DB 인스턴스 가용성이나 패치 적용 또는 업그레이드와 같은 진행 중인 활동에 영향을 주지 않습니다.

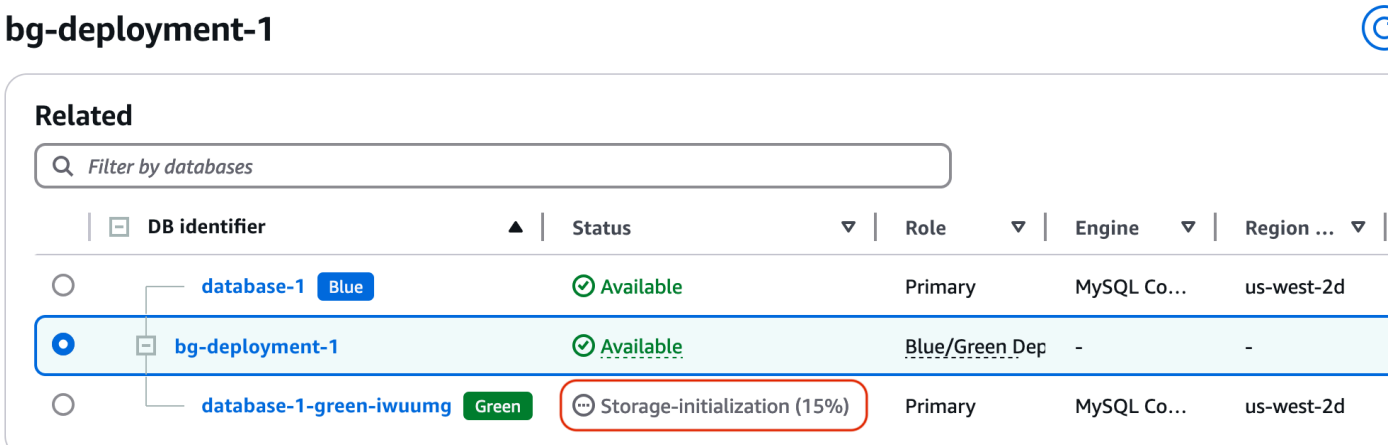
스토리지 초기화는 gp2, gp3, io1 및 io2 볼륨 유형을 사용하는 블루/그린 배포의 인스턴스에만 사용할 수 있습니다. t3 및 t4 패밀리를 제외한 모든 인스턴스 클래스를 지원합니다. 단일 AZ 배포의 그린 DB 인스턴스를 다중 AZ DB 인스턴스 배포로 수정하는 경우 스토리지 초기화에는 다중 AZ 구성의 보조 노드가 포함됩니다.

스토리지를 초기화하는 동안 초기화가 완료될 때까지 스토리지가 전체 성능에 도달하지 못할 수 있지만 인스턴스는 완전히 사용 가능하고 데이터베이스 작업에 사용할 수 있습니다. 스토리지 초기화가 진행되는 동안 전체 인스턴스 상태가 스토리지 초기화로 변경되고 진행률 표시기에 DB 인스턴스의 모든 볼륨에서 최소 초기화 수준이 표시됩니다.

콘솔, AWS CLI 또는 Amazon RDS API를 사용하여 스토리지 초기화를 모니터링합니다.

Console

AWS Management Console에서 DB 인스턴스 상태와 함께 스토리지 초기화 진행 상황을 볼 수 있습니다.



AWS CLI

AWS CLI를 사용하면 describe-db-instances 명령을 사용하여 스토리지 초기화를 모니터링할 수 있습니다. 응답의 `PercentProgress` 필드는 Amazon S3에서 검색된 데이터의 비율을 보여줍니다.

```
aws rds describe-db-instances --db-instance-identifier my-db-instance

{
    "DBInstances": [
        {
            "DBInstanceIdentifier": "my-db-instance",
            "DBInstanceClass": "db.m5.2xlarge",
            "Engine": "postgres",
            "DBInstanceStatus": "storage-initialization",
            ...
            "PercentProgress": "34"
        }
    ]
}
```

Amazon RDS API

Amazon RDS API로 DescribeDBInstances 작업을 직접 호출하여 스토리지 초기화의 상태를 검색합니다.

백그라운드 초기화 작업이 진행됨에 따라 진행률 표시기가 업데이트되므로 전체 스토리지 초기화가 완료되기 전에 스토리지 준비 상태를 추적할 수 있습니다. 스토리지 초기화를 통해 그린 DB 인스턴스가 완전히 작동하면 성능을 최적화할 수 있습니다.

블루/그린 배포 생성

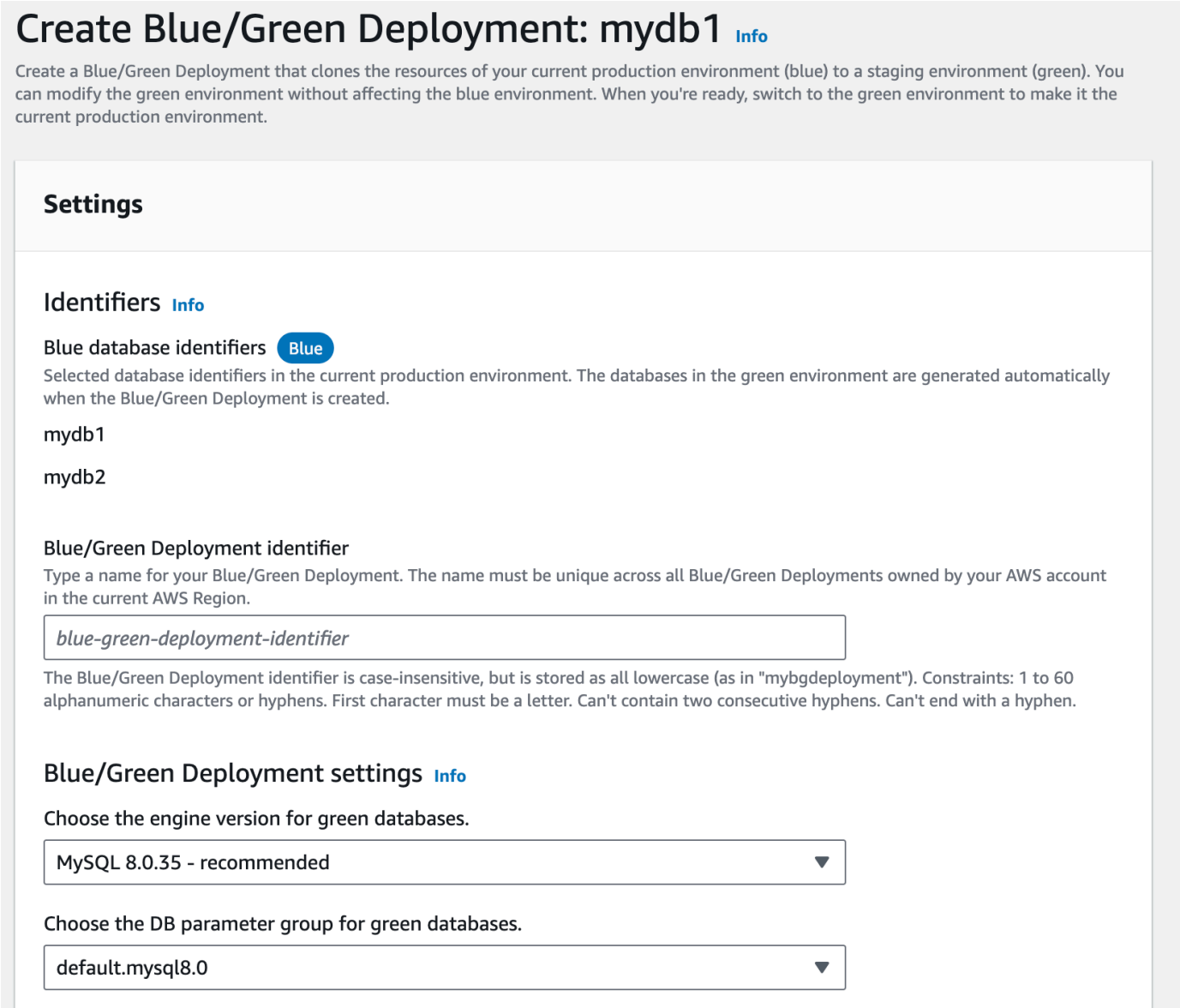
AWS Management Console, AWS CLI 또는 RDS API를 사용하여 블루/그린 배포를 생성할 수 있습니다.

콘솔

블루/그린 배포를 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택한 다음 그린 환경에 복사할 DB 인스턴스를 선택합니다.
3. 작업, 블루/그린 배포 생성을 선택합니다.

   블루/그린 배포 생성 페이지가 표시됩니다.



4. 블루 데이터베이스 식별자를 검토합니다. 해당 식별자가 블루 환경에 있어야 하는 DB 인스턴스와 일치하는지 확인합니다. 일치하지 않는다면 Cancel(취소)를 선택합니다.
5. 블루/그린 배포 이름에 블루/그린 배포의 이름을 입력합니다.
6. 나머지 섹션에서 그린 환경에 대한 설정을 지정합니다. 각 설정에 대한 자세한 내용은 the section called “사용 가능한 설정” 단원을 참조하세요.

   배포가 끝나면 그린 환경의 데이터베이스를 추가로 수정할 수 있습니다.
7. 생성(Create)을 선택합니다.

AWS CLI

AWS CLI를 사용하여 블루/그린 배포를 생성하려면 create-blue-green-deployment 명령을 사용해야 합니다. 사용 가능한 모든 옵션에 대한 자세한 내용은 the section called “사용 가능한 설정” 섹션을 참조하세요.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-blue-green-deployment \
    --blue-green-deployment-name my-blue-green-deployment \
    --source arn:aws:rds:us-east-2:123456789012:db:mydb1 \
    --target-engine-version 8.0.31 \
    --target-db-parameter-group-name mydbparametergroup
```

Windows의 경우:

```
aws rds create-blue-green-deployment ^
    --blue-green-deployment-name my-blue-green-deployment ^
    --source arn:aws:rds:us-east-2:123456789012:db:mydb1 ^
    --target-engine-version 8.0.31 ^
    --target-db-parameter-group-name mydbparametergroup
```

RDS API

Amazon RDS API를 사용하여 블루/그린 배포를 생성하려면 `CreateBlueGreenDeployment` 작업을 사용해야 합니다. 각 옵션에 대한 자세한 내용은 the section called “사용 가능한 설정” 단원을 참조하세요.

블루/그린 배포를 생성하기 위한 설정

블루/그린 배포를 만들 때 선택할 수 있는 설정에 대한 설명은 다음 표에 나와 있습니다. AWS CLI 옵션에 대한 자세한 내용은 create-blue-green-deployment 섹션을 참조하세요. RDS API 파라미터에 대한 자세한 내용은 CreateBlueGreenDeployment를 참조하세요.

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터
할당된 스토리지	그린 DB 인스턴스에 할당할 스토리지의 양(기비바이트 단위)입니다. 할	CLI 옵션: `--target-allocated-storage`

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터
	당된 스토리지를 늘리거나 줄일 수 있습니다. 블루 DB 인스턴스가 마그네틱(standard) 스토리지를 사용하는 경우 그린 환경에서 할당된 스토리지를 수정하려면 그린 DB 인스턴스를 범용 또는 프로비저닝된 IOPS 스토리지 유형으로 변경해야 합니다. 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 섹션을 참조하세요.	API 파라미터: `TargetAllocatedStorage`

블루/그린 배포 식별자	블루/그린 배포의 이름	CLI 옵션:
		
		`--blue-green-deployment-name`
		
		API 파라미터:
		
		`BlueGreenDeploymentName`

블루 데이터베이스 식별자	그린 환경으로 복사할 인스턴스 의 식별자입니다. CLI 또는 API를 사용하는 경우 인스턴스 Amazon 리소스 이름(ARN)을 지정합니다.	CLI 옵션:
		
		`--source`
		
		API 파라미터:
		
		`Source`

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터

그린 데이터베이스를 위한 DB 파라미터 그룹	그린 환경의 데이터베이스와 연결할 파라미터 그룹입니다.	CLI 옵션:
		
		`--target-db-parameter-group-name`
		
		`--target-db-cluster-parameter-group-name`
		
		API 파라미터:
		
		`TargetDBParameterGroupName`
		
		`TargetDBClusterParameterGroupName`

그린 데이터베이스를 위한 최적화된 쓰기 활성화	그린 기본 DB 인스턴스에서 RDS 최적화된 쓰기를 활성화하는 옵션입니다. 자세한 내용은 the section called “RDS 최적화된 쓰기 활성화” 섹션을 참조하세요. 최적화된 쓰기를 지원하지 않는 DB 인스턴스 클래스에서 지원하는 클래스로 변경하는 경우 스토리지 구성도 업그레이드해야 합니다. 자세한 내용은 the section called “스토리지 구성 업그레이드” 섹션을 참조하세요.	CLI 및 API의 경우, RDS 최적화된 쓰기를 지원하는 대상 DB 인스턴스 클래스를 지정하면 그린 기본 DB 인스턴스에서 자동으로 활성화됩니다.

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터

그린 데이터베이스를 위한 엔진 버전	그린 환경의 데이터베이스 를 지정된 DB 엔진 버전으로 업그레이드합니다.
	
	지정하지 않으면 그린 환경의 각 데이터베이스는 블루 환경의 해당 DB 인스턴스와 동일한 엔진 버전으로 생성됩니다.
	
	논리적 복제를 사용하는 RDS for PostgreSQL DB 인스턴스를 선택하는 경우 논리적 복제 제한 사항을 검토하고 확인하세요. 자세한 내용은 the section called “블루/그린 배포의 논리적 복제 관련 제한 사항” 섹션을 참조하세요.
		CLI 옵션:
		
		`--target-engine-version`
		
		RDS API 파라미터:
		
		`TargetEngineVersion`

그린 DB 인스턴스 클래스	그린 환경에 있는 DB 인스턴스의 컴퓨팅 및 메모리 용량입니다(예: `db.m5d.xlarge`).
	
	그린 데이터베이스에 대해 RDS 최적화 쓰기를 사용하도록 설정한 경우에만 이 옵션이 표시됩니다.
		CLI 옵션:
		
		`--target-db-instance-class`
		
		RDS API 파라미터:
		
		`TargetDBInstanceClass`

프로비저닝된 IOPS	그린 데이터베이스에 처음 할당될 초당 입출력 작업량(IOPS)입니다.
	
	이 값은 그린 기본 DB 인스턴스에만 적용되며 그린 복제본에는 적용되지 않습니다.
		CLI 옵션:
		
		`--target-iops`
		
		RDS API 파라미터:
		
		`TargetIops`

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터

스토리지 구성 업그레이드	스토리지 파일 시스템 구성을 업그레이드할지 여부를 선택합니다. 이 설정을 활성화하면 RDS는 기존 스토리지 파일 시스템에서 원하는 구성으로 그린 데이터베이스를 마이그레이션합니다.
	
	이 옵션은 블루 데이터베이스가 최신 스토리지 구성을 사용하지 않거나 동일한 요청으로 RDS 최적화된 쓰기를 활성화하는 경우에만 사용할 수 있습니다. 블루/그린 배포를 처음 생성할 때는 스토리지 구성만 업그레이드할 수 있습니다.
	
	자세한 내용은 the section called “스토리지 파일 시스템 업그레이드” 섹션을 참조하세요.
		CLI 옵션:
		
		`--upgrade-target-storage-config`
		
		RDS API 파라미터:
		
		`UpgradeTargetStorageConfig`

스토리지 처리량 (throughput)	그린 데이터베이스의 스토리지 처리량 값입니다. 이 설정은 스토리지 유형으로 범용 SSD(gp3)를 선택한 경우에만 표시됩니다.
	
	이 값은 그린 기본 DB 인스턴스에만 적용되며 그린 복제본에는 적용되지 않습니다.
	
	자세한 내용은 gp3 스토리지(권장) 섹션을 참조하세요.
		CLI 옵션:
		
		`--target-storage-throughput`
		
		RDS API 파라미터:
		
		`TargetStorageThroughput`

콘솔 설정	설정 설명	CLI 옵션 및 RDS API 파라미터

스토리지 유형	그린 데이터베이스의 스토리지 유형입니다. 다음 스토리지 유형이 지원됩니다.
	• 범용 SSD(gp2)
	• 범용 SSD(gp3)
	• 프로비저닝된 IOPS(io1)
	• 프로비저닝된 IOPS SSD(io2)
	이 값은 그린 기본 DB 인스턴스에만 적용되며 그린 복제본에는 적용되지 않습니다.
	자세한 내용은 Amazon RDS 스토리지 유형 섹션을 참조하세요.
		CLI 옵션:
		`--target-storage-type`
		RDS API 파라미터:
		`TargetStorageType`

Amazon RDS에서 블루/그린 배포 보기

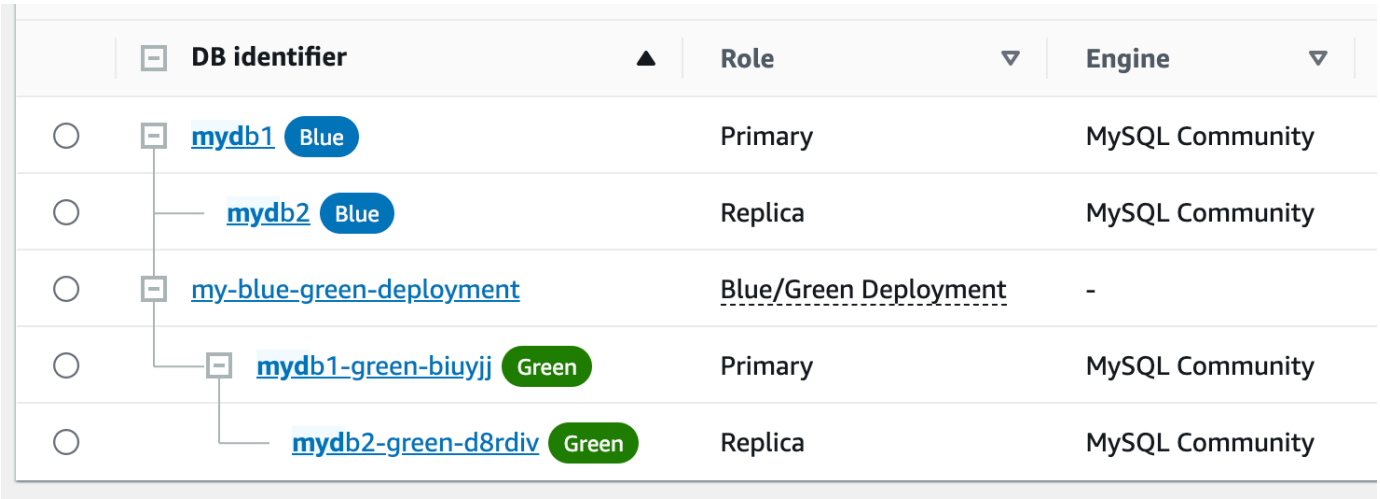
AWS Management Console, AWS CLI 또는 RDS API를 사용하여 블루/그린 배포의 세부 정보를 확인할 수 있습니다.

이벤트를 보고 구독하여 블루/그린 배포 관련 정보를 확인할 수도 있습니다. 자세한 내용은 블루/그린 배포 이벤트 단원을 참조하십시오.

콘솔

블루/그린 배포의 세부 정보를 확인하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택한 다음 목록에서 블루/그린 배포를 찾습니다.

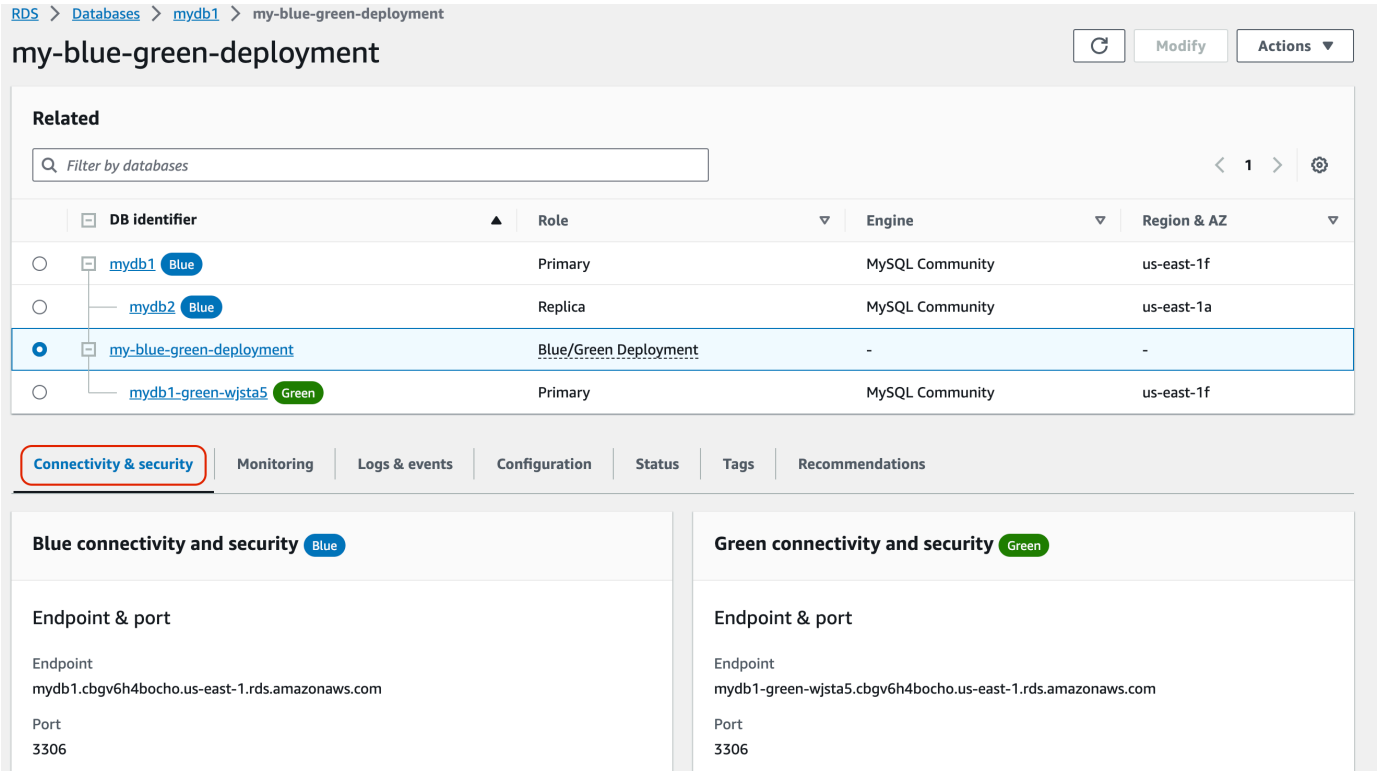


블루/그린 배포의 Role(역할) 값은 Blue/Green Deployment(블루/그린 배포)입니다.

3. 세부 정보를 표시할 블루/그린 배포의 이름을 선택합니다.

각 탭에는 블루 배포 섹션과 그린 배포 섹션이 있습니다. 예를 들어, 그린 환경에서 DB 엔진 버전을 업그레이드한다면 블루 환경과 그린 환경의 구성 탭에 표시되는 DB 엔진 버전이 다를 수 있습니다.

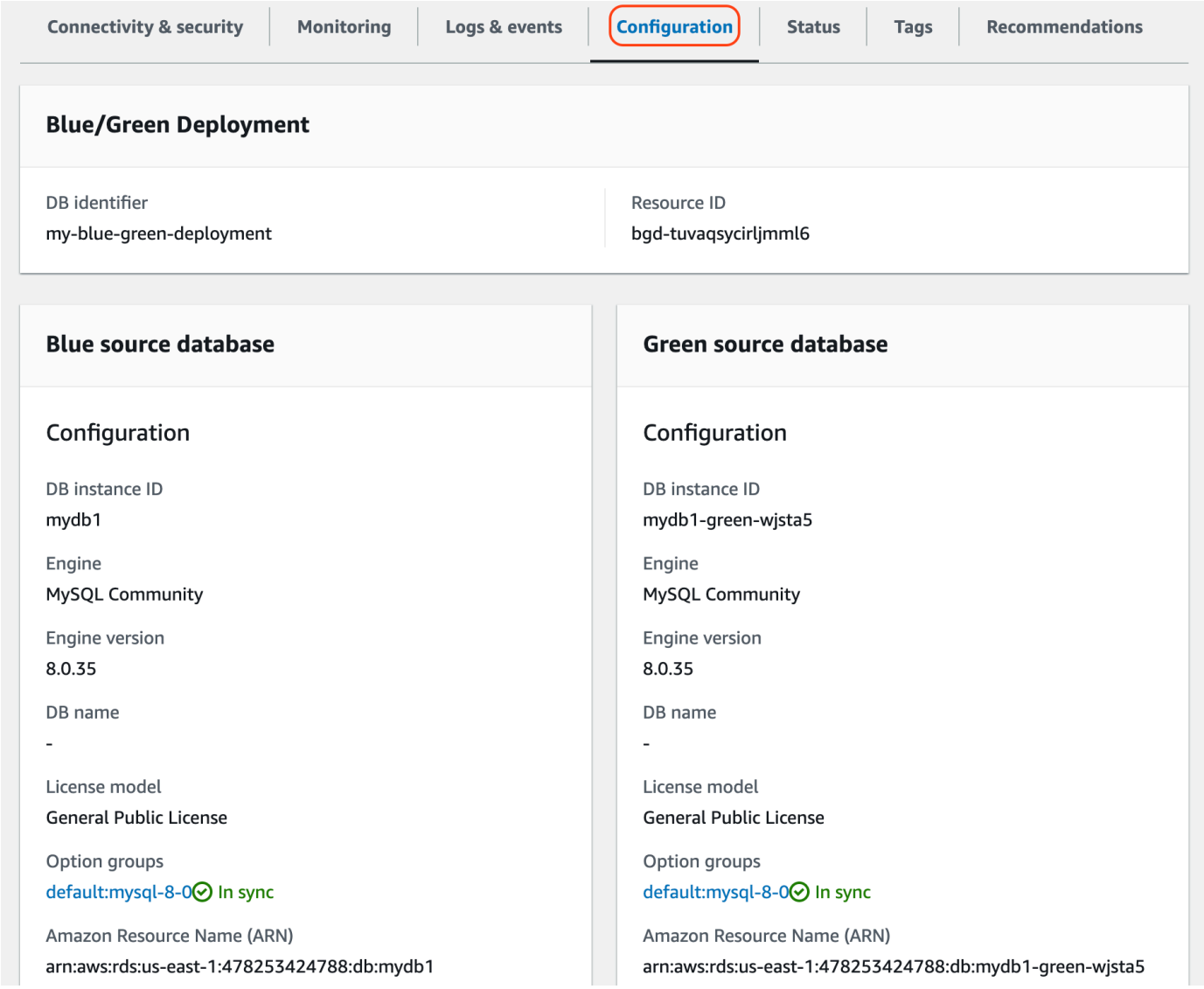
다음 이미지에서는 연결 및 보안 탭 예시를 확인할 수 있습니다.



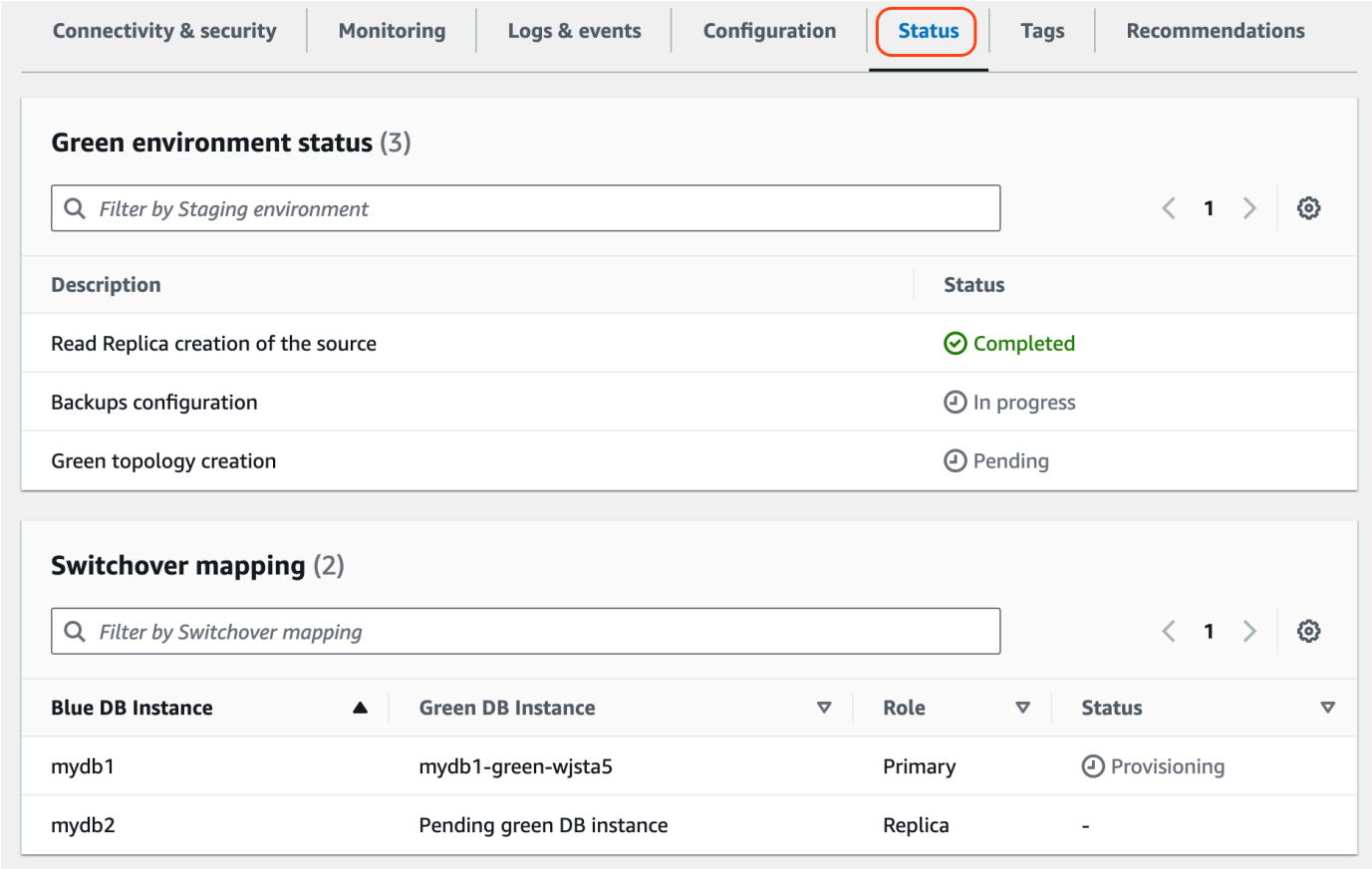
연결성 및 보안 탭에는 블루와 그린 환경 사이의 복제 및 복제 지연의 현재 상태를 보여주는 복제라는 섹션도 있습니다. 복제 상태가 `Replicating`인 경우 블루/그린 배포가 성공적으로 복제되고 있는 것입니다.

논리적 복제를 사용하는 RDS for PostgreSQL 블루/그린 배포의 경우 블루 환경에서 지원되지 않는 DDL이나 대규모 객체를 변경하면 복제 상태가 `Replication degraded`로 변경될 수 있습니다. 자세한 내용은 the section called “블루/그린 배포의 논리적 복제 관련 제한 사항” 단원을 참조하십시오.

다음 이미지에서는 구성 탭 예시를 확인할 수 있습니다.



다음 이미지에서는 상태 탭 예시를 확인할 수 있습니다.



AWS CLI

AWS CLI를 사용하여 블루/그린 배포 세부 정보를 보려면 describe-blue-green-deployments 명령을 사용해야 합니다.

Example 이름을 필터링하여 블루/그린 배포 세부 정보 확인

describe-blue-green-deployments 명령을 사용하는 경우 `--blue-green-deployment-name`으로 필터링할 수 있습니다.

다음 예에서는 *`my-blue-green-deployment`*라는 블루/그린 배포의 세부 정보를 확인할 수 있습니다.

```
aws rds describe-blue-green-deployments \
  --filters Name=blue-green-deployment-name,Values=my-blue-green-deployment
```

Example 식별자를 지정하여 블루/그린 배포 세부 정보 확인

describe-blue-green-deployments 명령을 사용하는 경우 `--blue-green-deployment-identifier` 옵션을 지정할 수 있습니다.

다음 예에서는 *bgd-1234567890abcdef*라는 식별자가 있는 블루/그린 배포의 세부 정보를 확인할 수 있습니다.

```
aws rds describe-blue-green-deployments \
  --blue-green-deployment-identifier bgd-1234567890abcdef
```

RDS API

Amazon RDS API를 사용하여 블루/그린 배포 세부 정보를 보려면 DescribeBlueGreenDeployments 작업을 사용하고 BlueGreenDeploymentIdentifier를 지정해야 합니다.

Amazon RDS에서 블루/그린 배포 전환

전환은 그린 환경을 새로운 프로덕션 환경으로 전환합니다. 그린 DB 인스턴스에 읽기 전용 복제본이 있다면 이 복제본도 전환됩니다. 전환하기 전에는 프로덕션 트래픽이 블루 환경의 DB 인스턴스 및 읽기 전용 복제본으로 라우팅됩니다. 전환한 후에는 프로덕션 트래픽이 그린 환경의 DB 인스턴스 및 읽기 전용 복제본으로 라우팅됩니다.

블루/그린 배포로의 전환은 블루/그린 배포 내에서 그린 DB 인스턴스 를 승격하는 것과 다릅니다. 작업 메뉴에서 승격을 선택하여 그린 DB 인스턴스 를 수동으로 승격하는 경우 블루 환경과 그린 환경 간의 복제가 중단되고 블루/그린 배포가 잘못된 구성 상태로 전환됩니다.

주제

- 전환 제한 시간
- 전환 가드레일
- 전환 작업
- 전환 모범 사례
- 전환 전 CloudWatch 지표 확인
- 전환 전 복제 지연 모니터링
- 블루/그린 배포로의 전환
- 전환 후

전환 제한 시간

30초에서 3,600초(1시간) 사이의 전환 제한 시간을 지정할 수 있습니다. 전환이 지정된 기간보다 오래 걸린다면, 모든 변경 사항이 롤백되고 어느 환경도 변경되지 않습니다. 기본 제한 시간은 300초(5분)입니다.

전환 가드레일

전환을 시작하면 Amazon RDS는 몇 가지 기본 검사를 실행하여 블루 및 그린 환경이 전환 준비가 되었는지 테스트합니다. 이러한 검사를 전환 가드레일이라고 합니다. 이러한 전환 가드레일은 환경이 준비되지 않았다면 전환이 진행되지 않게 합니다. 따라서 예상보다 긴 다운타임을 방지하고, 전환이 시작되면 발생할 수 있는 블루 및 그린 환경 간의 데이터 손실을 방지할 수 있습니다.

Amazon RDS는 그린 환경에서 다음 가드레일 검사를 실행합니다.

- 복제 상태 – 그린 기본 DB 인스턴스 복제 상태가 정상인지 확인합니다. 그린 기본 DB 인스턴스는 블루 기본 DB 인스턴스의 복제본입니다.
- 복제 지연 - 그린 기본 DB 인스턴스의 복제 지연이 전환의 허용 한계 이내인지 확인합니다. 허용 한계는 지정된 제한 시간을 기준으로 합니다. 복제 지연은 그린 기본 DB 인스턴스가 자신의 블루 기본 DB 인스턴스보다 얼마나 뒤쳐져 있는지를 나타냅니다. 자세한 내용은 the section called “전환 전 복제 지연 모니터링” 단원을 참조하십시오.
- 활성 쓰기 - 그린 기본 DB 인스턴스에서 활성 쓰기가 없는지 확인합니다.

Amazon RDS는 블루 환경에서 다음 가드레일 검사를 실행합니다.

- 외부 복제 - RDS for PostgreSQL의 경우 블루 환경이 자체 관리되는 논리적 소스(게시자) 또는 복제본(구독자)이 아니어야 합니다. 자체 관리되는 논리적 소스 또는 복제본인 경우 블루 환경의 모든 데이터베이스에서 자체 관리형 복제 슬롯과 구독을 삭제하고 전환을 진행한 다음 다시 생성하여 복제를 재개하는 것이 좋습니다. RDS for MySQL 및 RDS for MariaDB의 경우 파란색 데이터베이스가 외부 binlog 복제본이 아닌지 확인합니다. 있는 경우 활발하게 복제하고 있지 않은지 확인합니다.
- 장기 실행 활성 쓰기 - 블루 기본 DB 인스턴스에 복제 지연을 늘릴 수 있는 장기 실행 활성 쓰기가 없는지 확인합니다.
- 장기 실행 DDL 문 - 블루 기본 DB 인스턴스에 복제 지연을 늘릴 수 있는 장기 실행 DDL 문이 없는지 확인합니다.
- 지원되지 않는 PostgreSQL 변경 - 논리적 복제를 사용하는 RDS for PostgreSQL블루/그린 배포의 경우 블루 환경에서 DDL 변경이나 대형 객체의 추가 또는 수정이 수행되지 않았는지 확인합니다. 자세한 내용은 the section called “블루/그린 배포의 논리적 복제 관련 제한 사항” 단원을 참조하십시오.

Amazon RDS가 지원되지 않는 PostgreSQL 변경을 감지하면 복제 상태를 `Replication degraded`로 변경하고 블루/그린 배포에서 전환을 사용할 수 없음을 알립니다. 전환을 진행하려면 블루/그린 배포와 모든 그린 데이터베이스를 삭제하고 다시 생성하는 것이 좋습니다. 이렇게 하려면 작업, 그린 데이터베이스를 사용한 삭제를 선택합니다.

전환 작업

블루/그린 배포로 전환하면 RDS는 다음 작업을 수행합니다.

1. 가드레일 검사를 실행하여 블루 및 그린 환경이 전환 준비가 되었는지 확인합니다.
2. 두 환경 모두에서 기본 DB 인스턴스 에 대한 신규 쓰기 작업을 중지합니다.
3. 두 환경 모두에서 DB 인스턴스에 대한 연결을 끊고 새 연결을 허용하지 않습니다.
4. 그린 환경이 블루 환경과 동기화되도록, 그린 환경에서 복제가 충분히 진행될 때까지 기다립니다.
5. 두 환경 모두에서 DB 인스턴스 의 이름을 바꿉니다.

   RDS는 그린 환경의 DB 인스턴스 의 이름을 블루 환경의 대응하는 DB 인스턴스 와 일치하도록 변경합니다. 예를 들어 블루 환경의 DB 인스턴스 이름이 `mydb`라고 가정하겠습니다. 또한 그린 환경의 대응하는 DB 인스턴스의 이름은 `mydb-green-abc123`이라고 가정하겠습니다. 전환 중에는 그린 환경의 DB 인스턴스 이름이 `mydb`로 변경됩니다.

   RDS는 블루 환경의 DB 인스턴스 의 현재 이름에 `-old`*n*을 추가하여 이름을 변경합니다(*n*은 숫자입니다). 예를 들어 블루 환경의 DB 인스턴스 이름이 `mydb`라고 가정하겠습니다. 전환이 끝나면 DB 인스턴스의 이름은 `mydb-old1`이 됩니다.

   또한 RDS는 블루 환경의 대응하는 엔드포인트와 일치하도록 그린 환경의 엔드포인트 이름을 변경하므로, 애플리케이션은 변경하지 않아도 됩니다.
6. 두 환경 모두에서 데이터베이스 연결을 허용합니다.
7. 새 프로덕션 환경에서 기본 DB 인스턴스 에 대한 쓰기 작업을 허용합니다.

   전환이 끝나면 이전 프로덕션 기본 DB 인스턴스는 `read_only` 파라미터(RDS for MySQL의 경우) 또는 `default_transaction_read_only` 파라미터(RDS for PostgreSQL의 경우)를 `0`으로 설정하고 DB 인스턴스를 재부팅할 때까지 읽기 작업만 허용합니다.

Amazon EventBridge를 사용하여 전환 상태를 모니터링할 수 있습니다. 자세한 내용은 the section called “블루/그린 배포 이벤트” 단원을 참조하십시오.

블루 환경에서 태그를 구성했다면 이러한 태그는 전환 중에 새 프로덕션 환경으로 복사됩니다. 태그에 대한 자세한 내용은 Amazon RDS 리소스에 태그 지정 섹션을 참조하세요.

어떤 이유로든 전환이 시작된 후 종료되기 전에 중지되면, 모든 변경 사항이 롤백되고 두 환경 모두 변경되지 않습니다.

전환 모범 사례

전환하기 전에 다음 작업을 완료하여 모범 사례를 준수하는 것이 좋습니다.

- 그린 환경에서 리소스를 철저히 테스트합니다. 정상적이고 효율적으로 작동하는지 확인합니다.
- 관련 Amazon CloudWatch 지표를 모니터링합니다. 자세한 내용은 the section called “전환 전 CloudWatch 지표 확인” 단원을 참조하십시오.
- 전환에 가장 적합한 시간을 확인합니다.

  전환 중에는 두 환경 모두의 데이터베이스에서 쓰기가 차단됩니다. 프로덕션 환경에서 트래픽이 가장 적은 시간을 확인합니다. 활성 DDL 같이 오래 실행되는 트랜잭션은 전환 시간이 길며, 그 결과 프로덕션 워크로드의 다운타임이 길어질 수 있습니다.

  DB 인스턴스에 다수의 연결이 있는 경우 블루/그린 배포로 전환하기 전에 애플리케이션에 필요한 최소한의 개수로 연결을 수동으로 줄이는 것이 좋습니다. 이를 위한 한 가지 방법은 블루/그린 배포의 상태를 모니터링하고 상태가 `SWITCHOVER_IN_PROGRESS`로 변경되었음을 감지하면 연결 정리를 시작하는 스크립트를 만드는 것입니다.
- 두 환경 모두에서 DB 인스턴스가 `Available` 상태인지 확인합니다.
- 그린 환경의 기본 DB 인스턴스 가 정상 상태이고 복제 중인지 확인합니다.
- 네트워크 및 클라이언트 구성으로 인해 DNS 캐시 TTL(Time-to-Live)이 RDS DNS 영역의 기본값인 5초 이상으로 늘어나지 않도록 해야 합니다.
  그렇지 않으면 애플리케이션은 전환 후에도 계속해서 쓰기 트래픽을 블루 환경으로 전송합니다.
- 전환하기 전에 데이터 로드가 완료되었는지 확인합니다. 자세한 내용은 the section called “지연 로딩 및 스토리지 초기화” 단원을 참조하십시오.
- 논리적 복제를 사용하는 RDS for PostgreSQL 블루/그린 배포의 경우 다음을 수행합니다.
  - 논리적 복제 제한을 검토하고 전환하기 전에 필요한 조치를 취합니다. 자세한 내용은 the section called “블루/그린 배포의 논리적 복제 관련 제한 사항” 단원을 참조하십시오.
  - `ANALYZE` 작업을 실행하여 `pg_statistics` 테이블을 새로 고칩니다. 이렇게 하면 전환 후 성능 문제가 발생할 위험이 줄어듭니다.

> **Note**
>
> 전환 중에는 전환에 포함된 DB 인스턴스 를 수정할 수 없습니다.

전환 전 CloudWatch 지표 확인

블루/그린 배포를 전환하기 전에 Amazon CloudWatch 내에서 다음 지표를 확인하는 것이 좋습니다.

- `DatabaseConnections` - 이 지표를 사용하여 블루/그린 배포의 작업 수준을 예측하고 전환하기 전에 값이 배포에 적합한 수준인지 확인합니다. 성능 개선 도우미가 활성화되어 있으면 `DBLoad`가 더 정확한 지표입니다.

자세한 내용은 the section called “RDS에 대한 CloudWatch 지표” 단원을 참조하십시오.

전환 전 복제 지연 모니터링

블루/그린 배포를 전환하기 전에 가동 중지 시간을 줄이기 위해 복제 지연이 0에 가까운지 확인합니다.

RDS for MySQL 및 RDS for MariaDB

MySQL 및 MariaDB 블루/그린 배포의 경우 그린 환경에서 `ReplicaLag` CloudWatch 지표를 확인하여 현재 복제본 지연을 식별합니다. 자세한 내용은 the section called “읽기 전용 복제본 간 지연 문제 진단 및 해결” 단원을 참조하십시오.

RDS for PostgreSQL

물리적 복제를 사용하는 PostgreSQL 블루/그린 배포의 경우 the section called “복제 프로세스 모니터링 및 튜닝”의 지침을 참조하여 현재 복제본 지연을 식별합니다.

논리적 복제를 사용하는 PostgreSQL 블루/그린 배포의 경우 블루 환경에서 `OldestReplicationSlotLag` CloudWatch 지표를 확인하여 현재 복제본 지연을 식별합니다. 자세한 내용은 the section called “Amazon RDS에 대한 Amazon CloudWatch 지표” 단원을 참조하십시오.

또한 블루 환경에서 다음 SQL 쿼리를 실행할 수 있습니다.

```
SELECT slot_name,
       confirmed_flush_lsn as flushed,
```

```
       pg_current_wal_lsn(),
       (pg_current_wal_lsn() - confirmed_flush_lsn) AS lsn_distance
FROM pg_catalog.pg_replication_slots
WHERE slot_type = 'logical';

slot_name        |    flushed    | pg_current_wal_lsn | lsn_distance
-----------------+---------------+--------------------+------------
logical_replica1 | 47D97/CF32980 | 47D97/CF3BAC8      | 37192
```

`confirmed_flush_lsn`은 복제본으로 전송된 마지막 로그 시퀀스 번호(LSN)을 나타냅니다. `pg_current_wal_lsn`은 데이터베이스의 현재 위치를 나타냅니다. `lsn_distance`가 0이라는 것은 최신 복제본까지 처리되었음을 의미합니다.

블루/그린 배포가 논리적 복제 또는 물리적 복제를 사용하는 상황에 대한 설명은 ??? 섹션을 참조하세요.

블루/그린 배포로의 전환

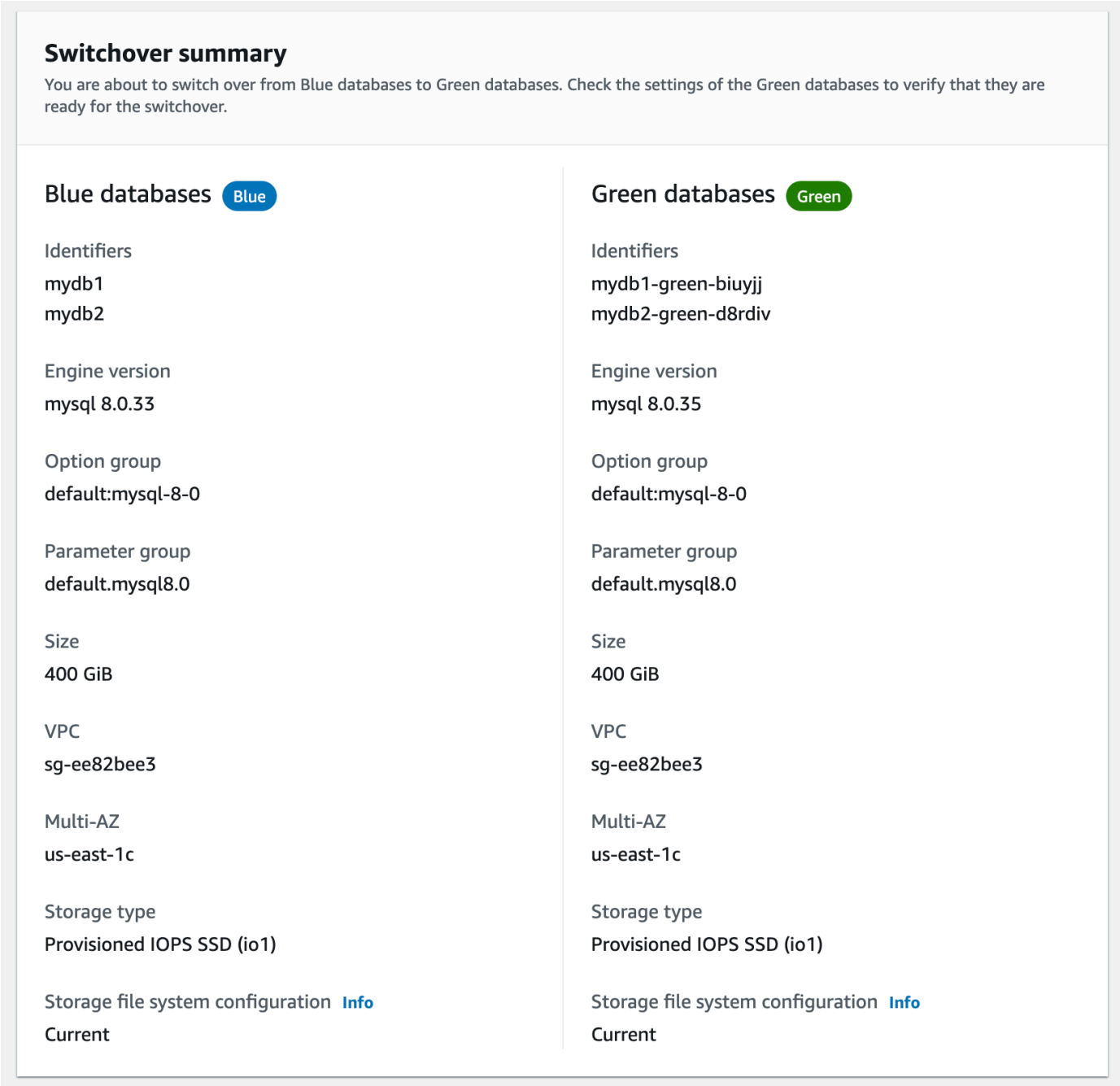
AWS Management Console, AWS CLI 또는 RDS API를 사용하여 블루/그린 배포로 전환할 수 있습니다.

콘솔

블루/그린 배포로 전환하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택한 후 전환할 블루/그린 배포를 선택합니다.
3. Actions(작업)에서 Switch over(전환)을 선택합니다.

   Switch over(전환) 페이지가 표시됩니다.



4. Switch over(전환) 페이지에서 전환 요약을 확인합니다. 두 환경의 리소스가 예상과 일치하는지 확인합니다. 일치하지 않는다면 Cancel(취소)를 선택합니다.
5. 제한 시간 설정에 전환 제한 시간을 입력합니다.
6. 인스턴스에서 RDS for PostgreSQL을 실행하는 경우 전환 전 권장 사항을 검토하고 확인하세요. 자세한 내용은 the section called “블루/그린 배포의 논리적 복제 관련 제한 사항” 단원을 참조하십시오.
7. Switch over(전환)을 선택합니다.

AWS CLI

AWS CLI를 사용하여 블루/그린 배포로 전환하려면 다음 옵션을 지정한 switchover-blue-green-deployment 명령을 사용해야 합니다.

- `--blue-green-deployment-identifier` - 블루/그린 배포의 리소스 ID를 지정합니다.
- `--switchover-timeout` - 전환의 제한 시간을 초 단위로 지정합니다. 기본값은 300입니다.

Example 블루/그린 배포로 전환

대상 LinuxmacOS, 또는Unix:

```
aws rds switchover-blue-green-deployment \
    --blue-green-deployment-identifier bgd-1234567890abcdef \
    --switchover-timeout 600
```

Windows의 경우:

```
aws rds switchover-blue-green-deployment ^
    --blue-green-deployment-identifier bgd-1234567890abcdef ^
    --switchover-timeout 600
```

RDS API

Amazon RDS API를 사용하여 블루/그린 배포로 전환하려면 다음 파라미터를 적용한 SwitchoverBlueGreenDeployment 작업을 사용해야 합니다.

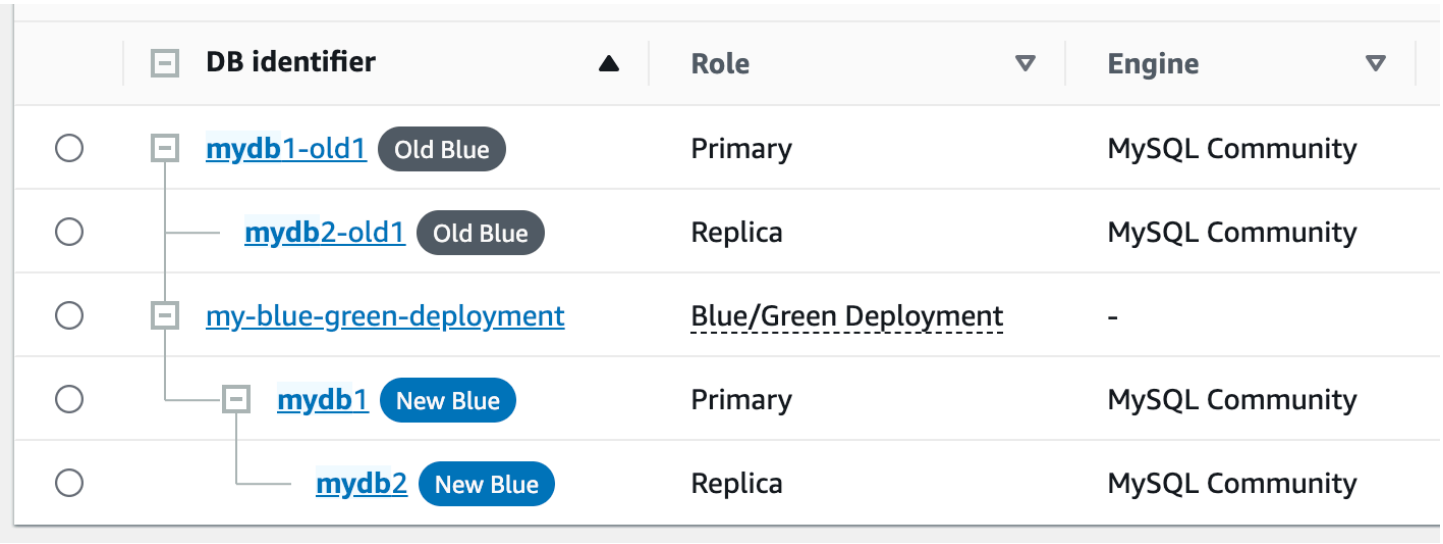
- `BlueGreenDeploymentIdentifier` - 블루/그린 배포의 리소스 ID를 지정합니다.
- `SwitchoverTimeout` - 전환의 제한 시간을 초 단위로 지정합니다. 기본값은 300입니다.

전환 후

전환한 후에는 이전 블루 환경의 DB 인스턴스 가 유지됩니다. 이러한 리소스에는 표준 요금이 적용됩니다. 블루와 그린 환경 간의 복제 및 이 중지됩니다.

RDS는 블루 환경의 DB 인스턴스의 현재 이름에 `-old`*n*을 추가하여 이름을 변경합니다(*n*은 숫자입니다). 이전 블루 환경의 DB 인스턴스는 `read_only` 파라미터(RDS for MySQL의 경우) 또는 `default_transaction_read_only` 파라미터(RDS for PostgreSQL의 경우)를 `0`으로 설정할 때까지 읽기 전용입니다. RDS는 그린 환경 `-new`*n*에서 DB 인스턴스의 이름을 지정합니다.

블루/그린 배포 리소스를 삭제하면 RDS는 `-old`*n* 및 `-new`*n* 리소스를 유지합니다.



소비자를 위한 상위 노드 업데이트

RDS는 완전 관리형 읽기 전용 복제본을 제공합니다. 그러나 외부 복제본이라고도 하는 자체 관리형 복제본을 설정하는 옵션도 제공합니다. 외부 복제본을 사용하면 타사 리소스를 복제 대상으로 사용할 수 있습니다.

RDS for MariaDB 또는 RDS for MySQL 블루/그린 배포로 전환한 후, 블루 DB 인스턴스 에 전환 이전의 외부 복제본 또는 이진 로그 소비자가 있었던 경우, 전환 후에 상위 노드를 업데이트해야 복제 연속성을 유지할 수 있습니다.

상위 노드를 업데이트하려면

1. 전환 후 이전에 그린 환경에 있었던 DB 인스턴스는 마스터 로그 파일 이름 및 마스터 로그 위치가 포함된 이벤트를 내보냅니다. 이벤트를 찾으려면 RDS 콘솔로 이동하여 왼쪽 탐색 창에서 이벤트를 선택합니다.
2. 전환하기 전에 소스가 이전 그린 DB 인스턴스의 이름인 이벤트를 기준으로 필터링합니다.
3. 이진 로그 좌표가 포함된 이벤트를 찾습니다. 이벤트 메시지는 `Binary log coordinates in green environment after switchover: file mysql-bin-changelog.`*`000003`* `and position` *`40134574`*와 유사합니다.
4. 소비자 또는 복제본이 이전 블루 환경의 모든 이진 로그를 적용했는지 확인하세요. 그런 다음 제공된 이진 로그 좌표를 사용하여 소비자에 대한 복제를 재개하세요. 예를 들어, EC2에서 MySQL 복제본을 실행하는 경우 다음 명령을 사용할 수 있습니다.

   MySQL 8.0.22 이하 메이저 및 마이너 버전

```
CHANGE MASTER TO MASTER_HOST='{new-writer-endpoint}', MASTER_LOG_FILE='mysql-bin-
changelog.000003', MASTER_LOG_POS=40134574;
```

MySQL 8.0.23 이상 메이저 및 마이너 버전

```
CHANGE REPLICATION SOURCE TO SOURCE_HOST='{new-writer-endpoint}',
 SOURCE_LOG_FILE='mysql-bin-changelog.000003', SOURCE_LOG_POS=40134574;
```

소비자가 다른 RDS for MySQL 또는 RDS for MariaDB DB 인스턴스인 경우, 다음 저장 프로시저를 순서대로 실행할 수 있습니다.

1. the section called “mysql.rds_stop_replication”
2. mysql.rds_reset_external_master(버전 8.0 이하의 경우) 또는 mysql_rds_reset_external_source(버전 8.4 이상의 경우)
3. mysql.rds_set_external_master(버전 8.0 이하의 경우) 또는 mysql_rds_set_external_source(버전 8.4 이상의 경우)
4. the section called “mysql.rds_start_replication”

Amazon RDS에서 블루/그린 배포 삭제

블루/그린 배포는 전환 전이나 후에 삭제할 수 있습니다.

전환하기 전에 블루/그린 배포를 삭제하면, Amazon RDS는 배포를 그린 환경에서 선택적으로 DB 인스턴스를 삭제합니다.

- 그린 환경(`--delete-target`)에서 DB 인스턴스를 삭제하기로 한 경우, 인스턴스에는 삭제 방지가 꺼져 있어야 합니다.
- 그린 환경(`--no-delete-target`)에서 DB 인스턴스를 삭제하지 않으면 인스턴스는 그대로 유지되지만, 더 이상 블루/그린 배포에 속하지 않게 됩니다. RDS for MySQL의 경우 환경 간에 복제가 계속됩니다. RDS for PostgreSQL의 경우 그린 환경이 독립 실행형 환경으로 승격되므로 복제가 중지됩니다.

녹색 데이터베이스를 삭제하는 옵션은 전환 후에는 콘솔에서 사용할 수 없습니다. AWS CLI를 사용하여 블루/그린 배포를 삭제할 때 배포 상태가 `SWITCHOVER_COMPLETED`인 경우 `--delete-target` 옵션을 지정할 수 없습니다.

> **Important**
>
> 블루/그린 배포를 삭제해도 블루 환경은 영향을 받지 않습니다.

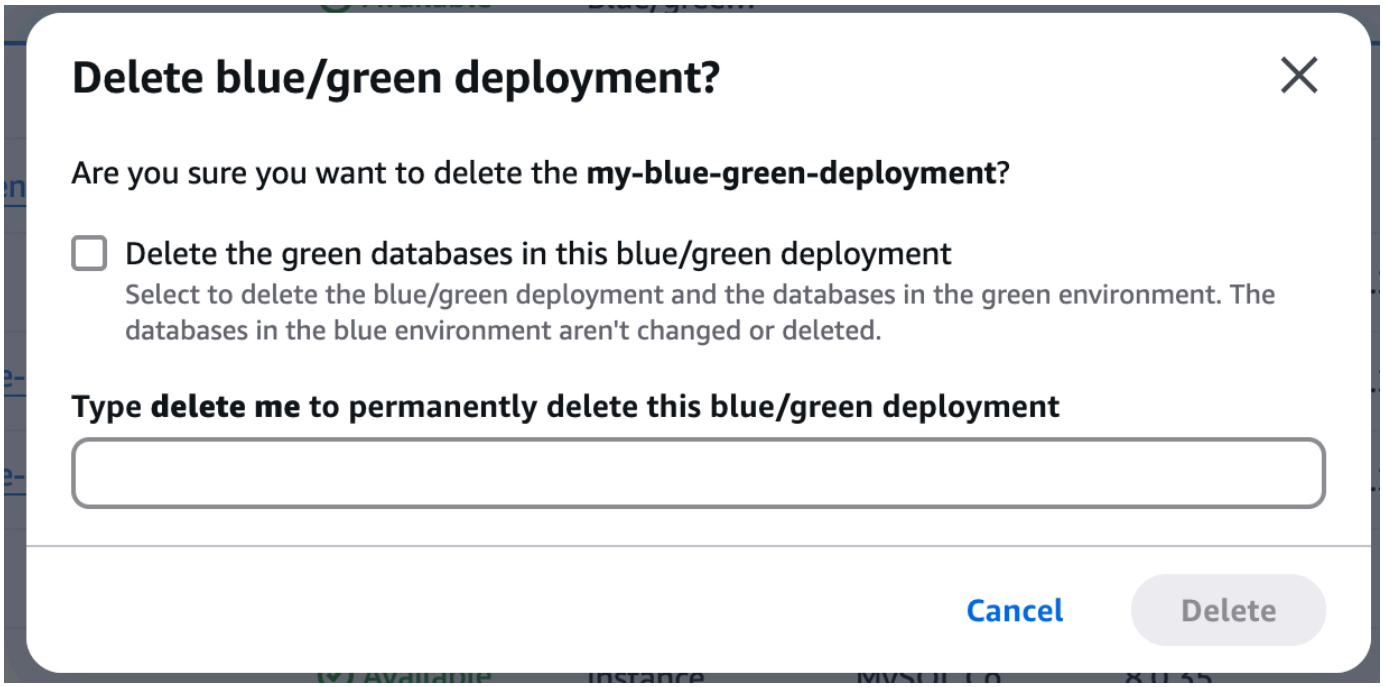
AWS Management Console, AWS CLI 또는 RDS API를 사용하여 블루/그린 배포를 삭제할 수 있습니다.

콘솔

블루/그린 배포를 삭제하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택한 후 삭제하려는 블루/그린 배포를 선택합니다.
3. 작업에 대해 삭제를 선택합니다.

   블루/그린 배포를 삭제하시겠습니까? 창이 나타납니다.



   그린 데이터베이스를 삭제하려면 이 블루/그린 배포에서 그린 데이터베이스 삭제를 선택합니다.
4. 상자에 **`delete me`**를 입력합니다.
5. 삭제를 선택합니다.

AWS CLI

AWS CLI를 사용하여 블루/그린 배포를 삭제하려면 다음 옵션을 적용한 delete-blue-green-deployment 명령을 사용해야 합니다.

- `--blue-green-deployment-identifier` - 삭제할 블루/그린 배포의 리소스 ID입니다.
- `--delete-target` – 그린 환경의 DB 인스턴스 가 삭제된다고 지정합니다. 블루/그린 배포가 `SWITCHOVER_COMPLETED` 상태인 경우 이 옵션을 지정할 수 없습니다.
- `--no-delete-target` – 그린 환경의 DB 인스턴스 가 유지된다고 지정합니다.

Example 그린 환경에서 블루/그린 배포 및 DB 인스턴스를 삭제합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds delete-blue-green-deployment \
    --blue-green-deployment-identifier bgd-1234567890abcdef \
    --delete-target
```

Windows의 경우:

```
aws rds delete-blue-green-deployment ^
    --blue-green-deployment-identifier bgd-1234567890abcdef ^
    --delete-target
```

Example 그린 환경에서 블루/그린 배포를 삭제하지만 DB 인스턴스는 유지합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds delete-blue-green-deployment \
    --blue-green-deployment-identifier bgd-1234567890abcdef \
    --no-delete-target
```

Windows의 경우:

```
aws rds delete-blue-green-deployment ^
    --blue-green-deployment-identifier bgd-1234567890abcdef ^
    --no-delete-target
```

RDS API

Amazon RDS API를 사용하여 블루/그린 배포를 삭제하려면 다음 파라미터를 적용한 `DeleteBlueGreenDeployment` 작업을 사용해야 합니다.

- `BlueGreenDeploymentIdentifier` - 삭제할 블루/그린 배포의 리소스 ID입니다.
- `DeleteTarget` – `TRUE`로 지정하여 그린 환경의 DB 인스턴스 를 삭제하거나 `FALSE`로 설정하여 유지합니다. 블루/그린 배포 상태가 `SWITCHOVER_COMPLETED`인 경우 `TRUE`일 수 없습니다.

데이터 백업, 복원 및 내보내기

이 섹션에서는 Amazon RDS DB 인스턴스 또는 다중 AZ DB 클러스터에서 데이터를 백업, 복원 및 내보내는 방법을 보여줍니다.

Amazon RDS DB 인스턴스 또는 다중 AZ DB 클러스터 백업에 대한 자세한 내용은 다음 주제를 참조하세요.

- 백업 소개
- 자동 백업 관리
- 수동 백업 관리

Amazon RDS DB 인스턴스 또는 다중 AZ DB 클러스터 복원에 대한 자세한 내용은 DB 인스턴스 복원 섹션을 참조하세요.

DB 스냅샷 복사, 공유 또는 내보내기에 대한 자세한 내용은 다음 주제를 참조하세요.

- Amazon RDS용 DB 스냅샷 복사
- Amazon RDS의 DB 스냅샷 공유
- Amazon S3 for Amazon RDS로 DB 스냅샷 데이터 내보내기

AWS Backup에서 관리하는 자동 백업에 대한 자세한 내용은 AWS Backup를 사용하여 Amazon RDS에 대한 자동 백업 관리 섹션을 참조하세요.

백업 소개

Amazon RDS는 DB 인스턴스 백업 기간 동안 DB 인스턴스 또는 다중 AZ DB 클러스터의 자동 백업을 생성하고 저장합니다. RDS는 개별 데이터베이스가 아닌 전체 DB 인스턴스를 백업하여 DB 인스턴스의 스토리지 볼륨 스냅샷을 생성합니다. RDS는 사용자가 지정한 백업 보존 기간에 따라 DB 인스턴스의 자동 백업을 저장합니다. 필요할 경우 백업 보존 기간 중 어느 특정 시점으로든 DB 인스턴스를 복구할 수 있습니다.

자동 백업은 이 규칙을 따릅니다.

- 자동 백업이 실행되려면 DB 인스턴스가 `available` 상태여야 합니다. DB 인스턴스가 `available` 외의 상태인 경우, 예컨대 `storage_full` 상태인 경우 자동 백업이 실행되지 않습니다.
- 동일한 데이터베이스에 대해 동일한 AWS 리전에서 DB 스냅샷 사본이 실행되는 동안에는 자동 백업이 발생하지 않습니다.

또한 DB 스냅샷을 생성하여 수동으로 DB 인스턴스를 백업할 수도 있습니다. DB 스냅샷 수동 생성에 대한 자세한 내용은 Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성 섹션을 참조하세요.

DB 인스턴스의 첫 번째 스냅샷에는 전체 데이터베이스의 데이터가 포함됩니다. 동일한 데이터베이스의 후속 스냅샷은 증분식이며, 마지막 스냅샷 이후 변경된 데이터만 저장됩니다.

자동 및 수동 DB 스냅샷을 모두 복사하고 수동 DB 스냅샷을 공유할 수 있습니다. DB 스냅샷 복사에 대한 자세한 내용은 Amazon RDS용 DB 스냅샷 복사 단원을 참조하십시오. DB 스냅샷 공유에 대한 자세한 내용은 Amazon RDS의 DB 스냅샷 공유 단원을 참조하십시오.

백업 스토리지

각 AWS 리전의 Amazon RDS 백업 스토리지는 해당 리전의 자동 백업 및 수동 DB 스냅샷으로 구성됩니다. 전체 백업 스토리지 공간은 해당 리전에 있는 모든 백업용 스토리지의 합계와 같습니다. DB 스냅샷을 다른 리전으로 이동하면 대상 리전의 백업 스토리지가 증가합니다. 백업은 Amazon S3에 저장됩니다.

백업 스토리지 비용에 대한 자세한 내용은 Amazon RDS 요금을 참조하십시오.

DB 인스턴스를 삭제할 때 자동 백업을 유지하기로 선택하는 경우, 자동 백업이 최대 보존 기간 동안 저장됩니다. DB 인스턴스를 삭제할 때 자동 백업 보존을 선택하지 않은 경우, 모든 자동 백업이 DB 인스턴스와 함께 삭제됩니다. 자동 백업은 삭제된 후에는 복구할 수 없습니다. Amazon RDS가 DB 인스턴스를 삭제하기 전에 최종 DB 스냅샷을 생성하도록 선택한 경우 이 스냅샷을 사용하여 DB 인스턴스를

복구할 수 있습니다. 원한다면 이전에 생성한 수동 스냅샷을 사용할 수 있습니다. 수동 스냅샷은 삭제되지 않습니다. 리전당 최대 100개의 수동 스냅샷을 보유할 수 있습니다.

자동 백업 관리

이 섹션에서는 DB 인스턴스 및 다중 AZ DB 클러스터의 자동 백업을 관리하는 방법을 보여줍니다.

주제

- 백업 기간
- 백업 보관 기간
- 자동 백업 활성화
- 자동 백업 보존
- 보관된 자동 백업 삭제
- 지원되지 않는 MySQL 스토리지 엔진에 대한 자동 백업
- 지원되지 않는 MariaDB 스토리지 엔진에 대한 자동 백업
- 다른 AWS 리전에 자동 백업 복제

백업 기간

자동 백업은 기본 백업 기간 동안 매일 실행됩니다. 백업 시간이 백업 기간에 할당된 시간보다 오래 걸릴 경우 백업은 백업 기간이 종료된 후에도 완료 시까지 계속 실행됩니다. 백업 기간은 해당 DB 인스턴스 또는 다중 AZ DB 클러스터에 대한 주간 유지 보수 기간과 겹칠 수 없습니다.

자동 백업 기간 중에 백업 프로세스가 시작될 때 스토리지 I/O가 일시적으로 중단될 수 있습니다(일반적으로 몇 초). 다중 AZ 배포에 대한 백업 시 지연 시간이 몇 분으로 증가할 수도 있습니다. MariaDB, MySQL, Oracle 및 PostgreSQL의 경우, 다중 AZ 배포에 대한 백업 시 기본 AZ에서는 I/O 작업이 중단되지 않습니다. 백업이 예비 복제본으로부터 수행되기 때문입니다. SQL Server의 경우, 단일 AZ 배포 및 다중 AZ 배포에 대한 백업 도중 I/O 작업이 일시적으로 중단됩니다. 백업을 기본 AZ에서 가져오기 때문입니다. Db2의 경우, I/O 작업이 일시적으로 중단됩니다. 백업을 예비 복제본으로부터 수행하더라도 백업 도중 I/O 작업이 일시적으로 중단됩니다.

백업이 시작되어야 할 시점에 DB 인스턴스 또는 클러스터에 워크로드가 많은 경우 자동 백업을 건너뛰는 경우가 있습니다. 백업을 건너뛰어도 특정 시점으로 복구(PITR)를 수행할 수 있으며 다음 백업 기간에도 백업이 시도됩니다. PITR에 대한 자세한 내용은 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 섹션을 참조하세요.

DB 인스턴스 또는 다중 AZ DB 클러스터를 생성할 때 원하는 백업 기간을 지정하지 않을 경우에는 Amazon RDS이 기본 30분 백업 기간을 할당합니다. 이 기간은 각 AWS 리전에 대해 8시간의 시간 블

록 중에서 임의로 선택됩니다. 다음 테이블은 기본 백업 기간이 할당된 각 AWS 리전별 시간 블록 목록입니다.

리전 이름	리전	시간 블록
미국 동부(버지니아 북부)	us-east-1	03:00~11:00 UTC
미국 동부(오하이오)	us-east-2	03:00~11:00 UTC
미국 서부(캘리포니아 북부 지역)	us-west-1	06:00~14:00 UTC
미국 서부(오리건)	us-west-2	06:00~14:00 UTC
Africa (Cape Town)	af-south-1	03:00~11:00 UTC
Asia Pacific (Hong Kong)	ap-east-1	06:00~14:00 UTC
아시아 태평양(하이데라바드)	ap-south-2	06:30~14:30 UTC
아시아 태평양(자카르타)	ap-southeast-3	08:00~16:00 UTC
아시아 태평양(말레이시아)	ap-southeast-5	09:00~17:00 UTC
아시아 태평양(멜버른)	ap-southeast-4	11:00~19:00 UTC
아시아 태평양(뭄바이)	ap-south-1	16:30~00:30 UTC
Asia Pacific (Osaka)	ap-northeast-3	00:00~08:00 UTC
Asia Pacific (Seoul)	ap-northeast-2	13:00~21:00 UTC

리전 이름	리전	시간 블록
아시아 태평양(싱가포르)	ap-southeast-1	14:00~22:00 UTC
아시아 태평양(시드니)	ap-southeast-2	12:00~20:00 UTC
아시아 태평양(도쿄)	ap-northeast-1	13:00~21:00 UTC
Canada (Central)	ca-central-1	03:00~11:00 UTC
캐나다 서부(캘거리)	ca-west-1	18:00~02:00 UTC
중국(베이징)	cn-north-1	06:00~14:00 UTC
China (Ningxia)	cn-northwest-1	06:00~14:00 UTC
Europe (Frankfurt)	eu-central-1	20:00~04:00 UTC
유럽(아일랜드)	eu-west-1	22:00~06:00 UTC
Europe (London)	eu-west-2	22:00~06:00 UTC
유럽(밀라노)	eu-south-1	02:00~10:00 UTC
유럽(파리)	eu-west-3	07:29~14:29 UTC
유럽(스페인)	eu-south-2	02:00~10:00 UTC
Europe (Stockholm)	eu-north-1	23:00~07:00 UTC
유럽(취리히)	eu-central-2	02:00~10:00 UTC
이스라엘(텔아비브)	il-central-1	03:00~11:00 UTC
중동(바레인)	me-south-1	06:00~14:00 UTC
중동(UAE)	me-central-1	05:00~13:00 UTC
남아메리카(상파울루)	sa-east-1	23:00~07:00 UTC

리전 이름	리전	시간 블록
AWS GovCloud(미국 동부)	us-gov-east-1	17:00~01:00 UTC
AWS GovCloud(미국 서부)	us-gov-west-1	06:00~14:00 UTC

백업 보관 기간

또한 DB 인스턴스 또는 다중 AZ DB 클러스터를 생성하면서 백업 보존 기간도 설정할 수 있습니다. Amazon RDS API 또는 AWS CLI를 사용하여 DB 인스턴스를 생성하고 백업 보존 기간을 설정하지 않으면 경우 기본 백업 보존 기간은 1일입니다. 콘솔을 사용하여 DB 인스턴스를 생성하는 경우 기본 백업 보존 기간은 7일입니다.

DB 인스턴스 또는 클러스터를 생성한 후 백업 보존 기간을 수정할 수 있습니다. DB 인스턴스의 백업 보존 기간은 0일에서 35일 사이로 설정할 수 있습니다. 백업 보존 기간을 0으로 설정하면 자동 백업이 비활성화됩니다. 다중 AZ DB 클러스터의 경우 백업 보존 기간을 1일에서 35일로 설정할 수 있습니다. 수동 스냅샷 한도(리전당 100개)는 자동 백업에 적용되지 않습니다.

> **Important**
>
> DB 인스턴스의 백업 보존 기간을 0에서 0이 아닌 값으로 또는 0이 아닌 값에서 0으로 변경할 경우 중단됩니다.

RDS는 백업 보존 기간 계산 시에 `stopped` 상태에서 경과된 시간을 포함하지 않습니다. DB 인스턴스 또는 클러스터가 중지된 동안에는 자동 백업이 생성되지 않습니다. DB 인스턴스가 중지된 경우 백업 보존 기간보다 오래 백업을 보존할 수 있습니다.

자동 백업 활성화

DB 인스턴스에 자동 백업이 활성화되어 있지 않더라도 언제든지 활성화할 수 있습니다. 백업 보존 기간을 0이 아닌 양수 값으로 설정하여 자동 백업을 활성화합니다. 자동 백업을 활성화하면 DB 인스턴스가 오프라인으로 설정되고 백업이 즉시 생성됩니다.

> **Note**
>
> AWS Backup에서 백업을 관리하는 경우 자동 백업을 활성화할 수 없습니다. 자세한 내용은 AWS Backup를 사용하여 Amazon RDS에 대한 자동 백업 관리을 참조하세요.

콘솔

자동 백업을 즉시 활성화하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 다음 수정하려는 DB 인스턴스 또는 다중 AZ DB 클러스터를 선택합니다.
3. 수정을 선택합니다.
4. 백업 보존 기간으로 0이 아닌 양수 값(예: 3일)을 선택합니다.
5. [Continue]를 선택합니다.
6. 즉시 적용을 선택합니다.
7. DB 인스턴스 수정 또는 클러스터 수정을 선택하여 변경 내용을 저장하고 자동 백업을 활성화합니다.

AWS CLI

자동 백업을 활성화하려면 AWS CLI modify-db-instance 또는 modify-db-cluster 명령을 사용합니다.

다음 파라미터를 포함합니다.

- `--db-instance-identifier`(또는 다중 AZ DB 클러스터의 경우 `--db-cluster-identifier`)
- `--backup-retention-period`
- `--apply-immediately` 또는 `--no-apply-immediately`

다음 예에서는 백업 보존 기간을 3일로 설정하여 자동 백업을 활성화합니다. 변경이 바로 적용됩니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier my_db_instance  \
    --backup-retention-period 3 \
    --apply-immediately
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier my_db_instance  ^
    --backup-retention-period 3 ^
    --apply-immediately
```

RDS API

자동 백업을 활성화하려면 RDS API ModifyDBInstance 또는 ModifyDBCluster 작업을 다음 필수 파라미터와 함께 사용합니다.

- `DBInstanceIdentifier` 또는 `DBClusterIdentifier`
- `BackupRetentionPeriod`

자동 백업 보기

자동 백업을 보려면 탐색 창에서 자동 백업(Automated backups)을 선택합니다. 자동 백업에 연결된 개별 스냅샷을 보려면 탐색 창에서 스냅샷(Snapshots)을 선택합니다. 또는 자동 백업과 연결된 개별 스냅샷을 설명할 수 있습니다. 그런 다음 이러한 스냅샷 중 하나에서 DB 인스턴스를 직접 복원할 수 있습니다.

자동 스냅샷 이름은 `rds:<database-name>-yyyy-mm-dd-hh-mm` 패턴을 따릅니다. 여기서 `yyyy-mm-dd-hh-mm`은 스냅샷이 만들어진 날짜 및 시간을 나타냅니다.

AWS CLI를 사용하여 기존 DB 인스턴스의 자동 백업을 설명하려면 다음 명령 중 하나를 사용합니다.

```
aws rds describe-db-instance-automated-backups --db-instance-
identifier DBInstanceIdentifier
```

or

```
aws rds describe-db-instance-automated-backups --dbi-resource-id DbiResourceId
```

RDS API를 사용하여 보존된 기존 DB 인스턴스의 자동 백업을 설명하려면 다음 파라미터 중 하나를 사용하여 `DescribeDBInstanceAutomatedBackups` 작업을 호출합니다.

- `DBInstanceIdentifier`
- `DbiResourceId`

자동 백업 보존

> **Note**
>
> 다중 AZ DB 클러스터가 아닌 DB 클러스터의 자동 백업만 유지할 수 있습니다.

DB 인스턴스를 삭제할 때 자동 백업을 보존하도록 선택할 수 있습니다. 자동 백업은 삭제 시점에 DB 인스턴스에 구성된 백업 보존 기간과 동일한 일수 동안 보존될 수 있습니다.

보존된 자동 백업에는 DB 인스턴스의 시스템 스냅샷 및 트랜잭션 로그가 포함되어 있습니다. 여기에는 할당된 스토리지 및 DB 인스턴스 클래스와 같은 DB 인스턴스 속성도 포함되며 이러한 속성은 활성 인스턴스로 복원하는 데 필요합니다.

보존된 자동 백업 및 수동 스냅샷은 삭제될 때까지 청구 요금이 부과됩니다. 자세한 내용은 보존 비용 단원을 참조하십시오.

Db2, MariaDB, PostgreSQL, Oracle 및 Microsoft SQL Server 엔진을 실행 중인 RDS 인스턴스에 대한 자동 백업을 보존할 수 있습니다.

AWS Management Console, RDS API 및 AWS CLI를 사용하여 보존된 자동 백업을 복원하거나 제거할 수 있습니다.

주제

- 보존 기간
- 보존된 백업 보기
- 복원
- 보존 비용
- 제한 사항

보존 기간

보존된 자동 백업의 시스템 스냅샷 및 트랜잭션 로그는 원본 DB 인스턴스에 대해 만료될 때와 동일한 방식으로 만료됩니다. 이 인스턴스에 대해 생성된 새 스냅샷이나 로그가 없으므로 보존 자동 백업은 결국 완전히 만료됩니다. 사실상 보존된 자동 백업은 삭제 시 원본 인스턴스에 지정되었던 보존 기간에 대한 설정에 따라 마지막 시스템 스냅샷이 수행될 동안 유지됩니다. 보존된 자동 백업은 마지막 시스템 스냅샷이 만료된 후 시스템에 의해 제거됩니다.

DB 인스턴스를 삭제할 때와 동일한 방식으로 보존된 자동 백업을 제거할 수 있습니다. 콘솔이나 RDS API 작업 `DeleteDBInstanceAutomatedBackup`을 사용하여 보존된 자동 백업을 제거할 수 있습니다.

마지막 스냅샷은 보존된 자동 백업과 무관합니다. 보존된 자동 백업도 결국 만료되므로 자동 백업을 보존하더라도 가급적이면 최종 스냅샷을 생성하는 것이 좋습니다 최종 스냅샷은 만료되지 않습니다.

보존된 백업 보기

보존된 자동 백업을 보려면 탐색 창에서 [자동 백업(Automated backups)]을 선택한 다음 [보존됨(Retained)]을 선택합니다. 보존된 자동 백업에 연결된 개별 스냅샷을 보려면 탐색 창에서 [스냅샷(Snapshots)]을 선택합니다. 또는 보존된 자동 백업과 연결된 개별 스냅샷을 설명할 수 있습니다. 그런 다음 이러한 스냅샷 중 하나에서 DB 인스턴스를 직접 복원할 수 있습니다.

AWS CLI를 사용하여 보존된 자동 백업을 설명하려면 다음 명령 중 하나를 사용합니다.

```
aws rds describe-db-instance-automated-backups --dbi-resource-id DbiResourceId
```

RDS API를 사용하여 보존된 자동 백업을 설명하려면 `DbiResourceId` 파라미터로 `DescribeDBInstanceAutomatedBackups` 작업을 호출합니다.

복원

자동 백업을 사용하여 DB 인스턴스를 복원하는 방법에 대한 자세한 내용은 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 섹션을 참조하세요.

보존 비용

보존된 자동 백업의 비용은 연결된 시스템 스냅샷의 총 스토리지 비용입니다. 트랜잭션 로그나 인스턴스 메타데이터에 대해 추가 요금이 부과되지 않습니다. 백업에 대한 기타 모든 요금 규칙이 복원 가능한 인스턴스에 적용됩니다.

예를 들어 실행 중인 인스턴스에 할당된 총 스토리지가 100GB라고 가정하겠습니다. 또한 수동 스냅샷 50GB와 보존된 자동 백업과 연결된 시스템 스냅샷 75GB가 있다고 가정하겠습니다. 이 경우 백업 스토리지의 추가 25GB에 대해서만 요금이 부과됩니다. 즉, (50GB + 75GB) – 100GB = 25GB입니다.

제한 사항

다음 제한은 보존된 자동 백업에 적용됩니다.

- 한 AWS 리전에서 보존된 자동 백업의 최대 개수는 40개입니다. 이 개수는 DB 인스턴스 할당량에 포함되지 않습니다. 실행 중인 DB 인스턴스 40개와 추가로 보존된 자동 백업 40개를 동시에 보유할 수 있습니다.
- 보존된 자동 백업에는 파라미터나 옵션 그룹에 대한 정보가 포함되지 않습니다.
- 삭제된 인스턴스를 삭제할 때의 보존 기간 내 특정 시점으로 복원할 수 있습니다.
- 보존된 자동 백업은 수정할 수 없습니다. 보존된 자동 백업은 원본 인스턴스를 삭제할 때 존재했던 시스템 백업, 트랜잭션 로그 및 DB 인스턴스 속성으로 구성되기 때문입니다.

보관된 자동 백업 삭제

보관된 자동 백업이 더 이상 필요하지 않으면 삭제할 수 있습니다.

콘솔

보관된 자동 백업을 삭제하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 백업 자동화를 선택합니다.
3. 보존됨(Retained) 탭에서 삭제하려는 보존된 자동 백업을 선택합니다.
4. [Actions]에 대해 [Delete]를 선택합니다.
5. 확인 페이지에서 **`delete me`**를 입력하고 삭제를 선택합니다.

AWS CLI

delete-db-instance-automated-backup이라는 AWS CLI 명령을 다음 옵션과 함께 사용하여 보관된 자동 백업을 삭제할 수 있습니다.

- `--dbi-resource-id` – 소스 DB 인스턴스의 리소스 식별자입니다.

describe-db-instance-automated-backups이라는 AWS CLI 명령을 실행하여 보관된 자동 백업의 소스 DB 인스턴스에 대한 리소스 식별자를 찾을 수 있습니다.

Example

다음 예시에서는 소스 DB 인스턴스 리소스 식별자가 db-123ABCEXAMPLE인 보관된 자동 백업을 삭제합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds delete-db-instance-automated-backup \
    --dbi-resource-id db-123ABCEXAMPLE
```

Windows의 경우:

```
aws rds delete-db-instance-automated-backup ^
    --dbi-resource-id db-123ABCEXAMPLE
```

RDS API

DeleteDBInstanceAutomatedBackup이라는 Amazon RDS API 작업을 다음 파라미터와 함께 사용하여 보관된 자동 백업을 삭제할 수 있습니다.

- DbiResourceId – 소스 DB 인스턴스의 리소스 식별자입니다.

  Amazon RDS API 작업인 DescribeDBInstanceAutomatedBackups를 사용하여 보관된 자동 백업의 소스 DB 인스턴스에 대한 리소스 식별자를 찾을 수 있습니다.

자동 백업 비활성화

대량의 데이터를 로드하는 등 특정 상황에서는 자동 백업을 일시적으로 비활성화해야 하는 경우가 있습니다.

> **Important**
>
> 특정 시점으로 복구가 어렵기 때문에 자동 복구의 비활성화는 최대한 자제하는 것이 좋습니다. DB 인스턴스 또는 다중 AZ DB 클러스터 자동 백업을 비활성화하면 해당 데이터베이스에 대한

기존 자동 백업이 모두 삭제됩니다. 자동 백업을 비활성화한 후 다시 활성화하면 자동 백업을 다시 활성화한 시점부터만 복구할 수 있습니다.

콘솔

자동 백업을 즉시 비활성화하려면

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 다음 수정하려는 DB 인스턴스 또는 다중 AZ DB 클러스터를 선택합니다.
3. 수정을 선택합니다.
4. 백업 보존 기간으로 0일을 선택합니다.
5. [Continue]를 선택합니다.
6. 즉시 적용을 선택합니다.
7. DB 인스턴스 수정 또는 클러스터 수정을 선택하여 변경 내용을 저장하고 자동 백업을 비활성화합니다.

AWS CLI

자동 백업을 즉시 비활성화하려면, modify-db-instance 또는 modify-db-cluster 명령을 사용하고 `--apply-immediately`를 사용하여 백업 보존 기간을 0으로 설정합니다.

Example

다음 예시에서는 다중 AZ DB 클러스터에서 자동 백업을 즉시 비활성화합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --backup-retention-period 0 \
    --apply-immediately
```

Windows의 경우:

```
aws rds modify-db-cluster ^
```

```
    --db-cluster-identifier mydbcluster ^
    --backup-retention-period 0 ^
    --apply-immediately
```

수정 사항이 적용되는 시점을 알아보려면 백업 보존 기간 값이 0이 되고, mydbcluster가 이용 가능한 상태가 될 때까지 DB 인스턴스에 대해 describe-db-instances를 호출합니다(다중 AZ DB 클러스터의 경우 describe-db-clusters).

```
aws rds describe-db-clusters --db-cluster-identifier mydcluster
```

RDS API

자동 백업을 즉시 비활성화하려면 다음 파라미터를 사용하여 ModifyDBInstance 또는 ModifyDBCluster 작업을 호출합니다.

- DBInstanceIdentifier = mydbinstance(또는 DBClusterIdentifier = mydbcluster)
- BackupRetentionPeriod = 0

Example

```
https://rds.amazonaws.com/
    ?Action=ModifyDBInstance
    &DBInstanceIdentifier=mydbinstance
    &BackupRetentionPeriod=0
    &SignatureVersion=2
    &SignatureMethod=HmacSHA256
    &Timestamp=2009-10-14T17%3A48%3A21.746Z
    &AWSAccessKeyId=<&AWS; Access Key ID>
    &Signature=<Signature>
```

지원되지 않는 MySQL 스토리지 엔진에 대한 자동 백업

MySQL DB 엔진의 경우, 자동 백업은 InnoDB 스토리지 엔진에만 지원됩니다. MyISAM 등의 다른 MySQL 스토리지 엔진에서 이러한 기능을 사용하는 경우 백업 복원 시에 작동이 불안정할 수 있습니다. 특히 MyISAM과 같은 스토리지 엔진은 안정적인 충돌 복구를 지원하지 않으므로 충돌 시 테이블이 손상될 수 있습니다. 이 같은 이유로 InnoDB 스토리지 엔진을 사용할 것을 권장합니다.

- 기존 MyISAM 테이블을 InnoDB 테이블로 변환하려면 ALTER TABLE 명령을 사용하면 됩니다(예: ALTER TABLE *table_name* ENGINE=innodb, ALGORITHM=COPY;).

- MyISAM을 사용하는 경우 REPAIR 명령을 사용하여 충돌 후 손상된 테이블을 수동으로 복구할 수 있습니다. 자세한 내용은 MySQL 설명서의 REPAIR TABLE 문을 참조하십시오. 그러나 MySQL 문서에 기재된 바와 같이 데이터를 전부 복원하지 못할 수도 있습니다.
- MyISAM 테이블을 복원하기 전에 테이블의 스냅샷을 생성하려면 다음 단계를 따릅니다.

  1. MyISAM 테이블의 모든 활동을 중지합니다(모든 세션 닫기).

     SHOW FULL PROCESSLIST 명령에서 반환되는 각 프로세스에 대해 mysql.rds_kill 명령을 호출하여 모든 세션을 닫을 수 있습니다.

  2. 각 MyISAM 테이블에 잠금 및 플러시를 수행합니다. 예를 들어, 다음 명령은 myisam_table1 및 myisam_table2라는 두 테이블을 잠그고 플러시합니다.

     ```
     mysql> FLUSH TABLES myisam_table, myisam_table2 WITH READ LOCK;
     ```

  3. DB 인스턴스 또는 다중 AZ DB 클러스터 스냅샷을 생성합니다. 스냅샷을 완료하면 MyISAM 테이블에서 잠금을 해제하고 활동을 다시 시작합니다. 다음 명령을 사용하여 테이블의 잠금을 해제할 수 있습니다.

     ```
     mysql> UNLOCK TABLES;
     ```

  이러한 단계를 통해 MyISAM에서 메모리에 저장된 데이터를 디스크로 강제 플러시하여 DB 스냅샷 복원 시 깨끗한 상태에서 시작할 수 있습니다. DB 스냅샷을 만드는 방법에 대한 자세한 내용은 Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성 단원을 참조하십시오.

지원되지 않는 MariaDB 스토리지 엔진에 대한 자동 백업

MariaDB DB 엔진의 경우 자동 백업은 InnoDB 스토리지 엔진에만 지원됩니다. Aria 등의 다른 MariaDB 스토리지 엔진에서 이러한 기능을 사용하는 경우 백업 복원 시에 작동이 불안정할 수 있습니다. Aria가 충돌 안정성을 개선한 MyISAM 대체 스토리지 엔진이지만, 충돌 이벤트가 발생하는 경우 여전히 테이블이 손상될 수 있습니다. 이 같은 이유로 InnoDB 스토리지 엔진을 사용할 것을 권장합니다.

- 기존 Aria 테이블을 InnoDB 테이블로 변환하려면 ALTER TABLE 명령을 사용합니다. 예: ALTER TABLE *table_name* ENGINE=innodb, ALGORITHM=COPY;
- Aria를 사용하는 경우 REPAIR TABLE 명령을 사용하여 충돌 후 손상된 테이블을 수동으로 복구할 수 있습니다. 자세한 내용은 http://mariadb.com/kb/en/mariadb/repair-table/ 단원을 참조하십시오.
- Aria 테이블을 복원하기 전에 테이블의 스냅샷을 생성하려면 다음 단계를 따릅니다.

  1. Aria 테이블의 모든 활동을 중지합니다(모든 세션 닫기).

2. 각 Aria 테이블에 잠금 및 플러시를 수행합니다.
3. DB 인스턴스 또는 다중 AZ DB 클러스터 스냅샷을 생성합니다. 스냅샷을 완료하면 Aria 테이블에서 잠금을 해제하고 활동을 다시 시작합니다. 이러한 단계를 통해 Aria에서 메모리에 저장된 데이터를 디스크로 강제 플러시하여 DB 스냅샷 복원 시 깨끗한 상태에서 시작할 수 있습니다.

다른 AWS 리전에 자동 백업 복제

추가 재해 복구 기능의 경우 원하는 대상 AWS 리전으로 스냅샷 및 트랜잭션 로그를 복제하도록 Amazon RDS 데이터베이스 인스턴스를 구성할 수 있습니다. DB 인스턴스에 대해 백업 복제가 구성된 경우 RDS는 DB 인스턴스에서 준비되는 즉시 모든 스냅샷 및 트랜잭션 로그의 교차 리전 복사를 시작합니다.

DB 스냅샷 복사 요금은 데이터 전송에 적용됩니다. DB 스냅샷을 복사한 후 대상 리전의 스토리지에 표준 요금이 적용됩니다. 자세한 내용은 RDS 요금을 참조하세요.

백업 복제 사용의 예는 AWS 온라인 기술 대화 Amazon RDS for Oracle 교차 리전 자동 백업을 통한 관리형 재해 복구를 참조하세요.

> **Note**
>
> 다중 AZ DB 클러스터에서는 자동 백업 복제가 지원되지 않습니다.

주제

- 리전 및 버전 사용 가능 여부
- 원본 및 대상 AWS 리전 지원
- 교차 리전 자동 백업 활성화
- 복제된 백업에 대한 정보 찾기
- 복제된 백업에서 지정된 시간으로 복구
- 자동 백업 복제 중지
- 복제된 백업 삭제

리전 및 버전 사용 가능 여부

기능 가용성 및 해당 지원은 각 데이터베이스 엔진의 특정 버전 및 AWS 리전에 따라 다릅니다. 교차 리전 복제를 통한 버전 및 리전 가용성에 관한 자세한 내용은 Amazon RDS에서 크로스 리전 자동 백업을 지원하는 리전 및 DB 엔진 단원을 참조하세요.

원본 및 대상 AWS 리전 지원

백업 복제는 다음 AWS 리전 간에 지원됩니다.

소스 리전	사용 가능한 대상 리전
Asia Pacific (Mumbai)	아시아 태평양(싱가포르) 미국 동부(버지니아 북부), 미국 동부(오하이오), 미국 서부(오레곤)
Asia Pacific (Osaka)	아시아 태평양(도쿄)
Asia Pacific (Seoul)	아시아 태평양(싱가포르), 아시아 태평양(도쿄) 미국 동부(버지니아 북부), 미국 동부(오하이오), 미국 서부(오레곤)
아시아 태평양(싱가포르)	아시아 태평양(뭄바이), 아시아 태평양(서울), 아시아 태평양(시드니), 아시아 태평양(도쿄) 미국 동부(버지니아 북부), 미국 동부(오하이오), 미국 서부(오레곤)
아시아 태평양(시드니)	아시아 태평양(싱가포르) 미국 동부(버지니아 북부), 미국 서부(캘리포니아 북부), 미국 서부(오레곤)
아시아 태평양(도쿄)	아시아 태평양(오사카), 아시아 태평양(서울), 아시아 태평양(싱가포르) 미국 동부(버지니아 북부), 미국 동부(오하이오), 미국 서부(오레곤)
Canada (Central)	유럽(아일랜드) 미국 동부(버지니아 북부), 미국 동부(오하이오), 미국 서부(캘리포니아 북부), 미국 서부(오레곤)
중국(베이징)	중국(닝샤)
중국(닝샤)	중국(베이징)
유럽(프랑크푸르트)	유럽(아일랜드), 유럽(런던), 유럽(파리), 유럽(스톡홀름) 미국 동부(버지니아 북부), 미국 동부(오하이오), 미국 서부(오레곤)
유럽(아일랜드)	Canada (Central)

소스 리전	사용 가능한 대상 리전
	유럽(프랑크푸르트), 유럽(런던), 유럽(파리), 유럽(스톡홀름) 미국 동부(버지니아 북부), 미국 동부(오하이오), 미국 서부(캘리포니아 북부), 미국 서부(오레곤)
Europe (London)	유럽(프랑크푸르트), 유럽(아일랜드), 유럽(파리), 유럽(스톡홀름) 미국 동부(버지니아 북부)
Europe (Paris)	유럽(프랑크푸르트), 유럽(아일랜드), 유럽(런던), 유럽(스톡홀름) 미국 동부(버지니아 북부)
Europe (Stockholm)	유럽(프랑크푸르트), 유럽(아일랜드), 유럽(런던), 유럽(파리) 미국 동부(버지니아 북부)
남아메리카(상파울루)	미국 동부(버지니아 북부) 미국 동부(오하이오)(us-east-1)
AWS GovCloud(미국 동부)	AWS GovCloud(미국 서부)
AWS GovCloud(미국 서부)	AWS GovCloud(미국 동부)
미국 동부(버지니아 북부)	아시아 태평양(뭄바이), 아시아 태평양(서울), 아시아 태평양(싱가포르), 아시아 태평양(시드니), 아시아 태평양(도쿄) Canada (Central) 유럽(프랑크푸르트), 유럽(아일랜드), 유럽(런던), 유럽(파리), 유럽(스톡홀름) 남아메리카(상파울루) 미국 동부(오하이오), 미국 서부(캘리포니아 북부), 미국 서부(오레곤)

소스 리전	사용 가능한 대상 리전
미국 동부(오하이오)	아시아 태평양(뭄바이), 아시아 태평양(서울), 아시아 태평양(싱가포르), 아시아 태평양(도쿄) Canada (Central) 유럽(프랑크푸르트), 유럽(아일랜드) 남아메리카(상파울루) 미국 동부(버지니아 북부), 미국 서부(캘리포니아 북부), 미국 서부(오레곤)
미국 서부(캘리포니아 북부)	아시아 태평양(시드니) Canada (Central) 유럽(아일랜드) 미국 동부(버지니아 북부), 미국 동부(오하이오), 미국 서부(오레곤)
미국 서부(오리건)	아시아 태평양(뭄바이), 아시아 태평양(서울), 아시아 태평양(싱가포르), 아시아 태평양(시드니), 아시아 태평양(도쿄) Canada (Central) 유럽(프랑크푸르트), 유럽(아일랜드) 미국 동부(버지니아 북부), 미국 동부(오하이오), 미국 서부(캘리포니아 북부)))

`describe-source-regions` AWS CLI 명령을 사용하여 서로 복제할 수 있는 AWS 리전을 확인할 수도 있습니다. 자세한 내용은 복제된 백업에 대한 정보 찾기을 참조하세요.

교차 리전 자동 백업 활성화

Amazon RDS 콘솔을 사용하여 신규 또는 기존 DB 인스턴스에 대해 백업 복제를 활성화할 수 있습니다. `start-db-instance-automated-backups-replication` AWS CLI 명령 또는

`StartDBInstanceAutomatedBackupsReplication` RDS API 작업을 사용할 수도 있습니다. 각 AWS 계정의 대상 AWS 리전에 대해 최대 20개의 백업을 복제할 수 있습니다.

> **Note**
>
> 자동 백업을 복제하려면 자동 백업을 활성화해야 합니다. 자세한 내용은 자동 백업 활성화을 참조하세요.

콘솔

신규 또는 기존 DB 인스턴스에 대해 백업 복제를 활성화할 수 있습니다.

- 새 DB 인스턴스의 경우 인스턴스를 시작할 때 활성화합니다. 자세한 내용은 DB 인스턴스에 대한 설정 섹션을 참조하세요.
- 기존 DB 인스턴스의 경우 다음 절차를 따릅니다.

기존 DB 인스턴스에 대한 백업 복제를 활성화하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 백업 자동화를 선택합니다.
3. 현재 리전(Current Region) 탭에서 백업 복제를 활성화할 DB 인스턴스를 선택합니다.
4. 작업에서 교차 리전 복제 관리를 선택합니다.
5. 백업 복제(Backup replication)에서 다른 AWS 리전으로의 복제 활성화(Enable replication to another AWS 리전)를 선택합니다.
6. 대상 리전을 선택합니다.
7. 복제된 백업 보존 기간을 선택합니다.
8. 소스 DB 인스턴스에서 암호화를 활성화한 경우 백업을 암호화하는 데 사용할 AWS KMS key를 선택하거나 키 ARN을 입력합니다.
9. Save(저장)를 선택합니다.

소스 리전에서, 복제된 백업은 자동 백업(Automated backups) 페이지의 현재 리전(Current Region) 탭에 나열됩니다. 대상 리전에서, 복제된 백업은 자동 백업(Automated backups) 페이지의 복제된 백업(Replicated backups) 탭에 나열됩니다.

AWS CLI

start-db-instance-automated-backups-replication AWS CLI 명령을 사용하여 백업 복제를 사용 설정합니다.

다음 CLI 예제는 미국 서부(오레곤) 지역의 DB 인스턴스에서 미국 동부(버지니아 북부) 지역로 자동 백업을 복제합니다. 또한 대상 리전의 AWS KMS key을(를) 사용하여 복제된 백업을 암호화합니다.

백업 복제를 활성화하려면

- 다음 명령 중 하나를 실행합니다.

  대상 LinuxmacOS, 또는Unix:

```
aws rds start-db-instance-automated-backups-replication \
--region us-east-1 \
--source-db-instance-arn "arn:aws:rds:us-west-2:123456789012:db:mydatabase" \
--kms-key-id "arn:aws:kms:us-east-1:123456789012:key/AKIAIOSFODNN7EXAMPLE" \
--backup-retention-period 7
```

  Windows의 경우:

```
aws rds start-db-instance-automated-backups-replication ^
--region us-east-1 ^
--source-db-instance-arn "arn:aws:rds:us-west-2:123456789012:db:mydatabase" ^
--kms-key-id "arn:aws:kms:us-east-1:123456789012:key/AKIAIOSFODNN7EXAMPLE" ^
--backup-retention-period 7
```

  이 `--source-region` 옵션은 AWS GovCloud(미국 동부) 및 AWS GovCloud(미국 서부) 리전 간에 백업을 암호화할 때 필요합니다. `--source-region`의 경우 소스 DB 인스턴스의 AWS 리전을 지정합니다.

  `--source-region`이 지정되지 않은 경우에는 `--pre-signed-url` 값을 지정해야 합니다. 미리 서명된 URL은 소스 AWS 리전에서 호출되는 `start-db-instance-automated-backups-replication` 명령에 대한 서명 버전 4의 서명된 요청이 포함된 URL입니다. `pre-signed-url` 옵션에 대해 자세히 알아보려면 AWS CLI 명령 참조에서 start-db-instance-automated-backups-replication을 참조하세요.

RDS API

다음 파라미터와 함께 StartDBInstanceAutomatedBackupsReplication RDS API 작업을 사용하여 백업 복제를 활성화합니다.

- Region(대상 리전에서 API 작업을 직접 호출하지 않는 경우)
- SourceDBInstanceArn
- BackupRetentionPeriod
- KmsKeyId(선택 사항)
- PreSignedUrl (를 사용하는 경우 필수)KmsKeyId

> **Note**
>
> 백업을 암호화하는 경우 미리 서명된 URL도 포함해야 합니다. 미리 서명된 URL에 대한 자세한 내용은 Amazon Simple Storage Service API 참조의 요청 인증: 쿼리 파라미터 사용(AWS 서명 버전 4) 및 AWS 일반 참조의 서명 버전 4 서명 프로세스를 참조하세요.

복제된 백업에 대한 정보 찾기

다음 CLI 명령을 사용하여 복제된 백업에 대한 정보를 찾을 수 있습니다.

- describe-source-regions
- describe-db-instances
- describe-db-instance-automated-backups

다음 describe-source-regions 예에서는 자동 백업을 미국 서부(오레곤) 대상 리전에 복제할 수 있는 소스 AWS 리전을 나열합니다.

소스 리전에 대한 정보를 표시하려면

- 다음 명령을 실행합니다.

```
aws rds describe-source-regions --region us-west-2
```

이 출력은 백업을 US East (N. Virginia)에서 미국 서부(오레곤)로 복제할 수 있지만 미국 동부(오하이오) 또는 미국 서부(캘리포니아 북부 지역)에서는 복제할 수는 없다는 것을 보여줍니다.

```
{
    "SourceRegions": [
        ...
        {
            "RegionName": "us-east-1",
            "Endpoint": "https://rds.us-east-1.amazonaws.com",
            "Status": "available",
            "SupportsDBInstanceAutomatedBackupsReplication": true
        },
        {
            "RegionName": "us-east-2",
            "Endpoint": "https://rds.us-east-2.amazonaws.com",
            "Status": "available",
            "SupportsDBInstanceAutomatedBackupsReplication": false
        },
            "RegionName": "us-west-1",
            "Endpoint": "https://rds.us-west-1.amazonaws.com",
            "Status": "available",
            "SupportsDBInstanceAutomatedBackupsReplication": false
        }
    ]
}
```

다음 `describe-db-instances` 예에서는 DB 인스턴스의 자동 백업을 보여 줍니다.

DB 인스턴스의 복제된 백업을 표시하려면

- 다음 명령 중 하나를 실행합니다.

  대상 LinuxmacOS, 또는Unix:

  ```
  aws rds describe-db-instances \
  --db-instance-identifier mydatabase
  ```

  Windows의 경우:

  ```
  aws rds describe-db-instances ^
  --db-instance-identifier mydatabase
  ```

이 출력에는 복제된 백업이 포함됩니다.

```
{
    "DBInstances": [
        {
            "StorageEncrypted": false,
            "Endpoint": {
                "HostedZoneId": "Z1PVIF0B656C1W",
                "Port": 1521,
            ...

            "BackupRetentionPeriod": 7,
            "DBInstanceAutomatedBackupsReplications":
 [{"DBInstanceAutomatedBackupsArn": "arn:aws:rds:us-east-1:123456789012:auto-backup:ab-
L2IJCEXJP7XQ7HOJ4SIEXAMPLE"}]
        }
    ]
}
```

다음 `describe-db-instance-automated-backups` 예에서는 DB 인스턴스의 자동 백업을 보여 줍니다.

DB 인스턴스의 자동 백업을 표시하려면

- 다음 명령 중 하나를 실행합니다.

  대상 LinuxmacOS, 또는Unix:

  ```
  aws rds describe-db-instance-automated-backups \
  --db-instance-identifier mydatabase
  ```

  Windows의 경우:

  ```
  aws rds describe-db-instance-automated-backups ^
  --db-instance-identifier mydatabase
  ```

이 출력에는 US East (N. Virginia)에 복제된 백업과 함께 미국 서부(오레곤)의 소스 DB 인스턴스 및 자동 백업이 표시됩니다 .

```
{
```

```
    "DBInstanceAutomatedBackups": [
        {
            "DBInstanceArn": "arn:aws:rds:us-west-2:868710585169:db:mydatabase",
            "DbiResourceId": "db-L2IJCEXJP7XQ7HOJ4SIEXAMPLE",
            "DBInstanceAutomatedBackupsArn": "arn:aws:rds:us-west-2:123456789012:auto-
backup:ab-L2IJCEXJP7XQ7HOJ4SIEXAMPLE",
            "BackupRetentionPeriod": 7,
            "DBInstanceAutomatedBackupsReplications":
 [{"DBInstanceAutomatedBackupsArn": "arn:aws:rds:us-east-1:123456789012:auto-backup:ab-
L2IJCEXJP7XQ7HOJ4SIEXAMPLE"}]
            "Region": "us-west-2",
            "DBInstanceIdentifier": "mydatabase",
            "RestoreWindow": {
                "EarliestTime": "2020-10-26T01:09:07Z",
                "LatestTime": "2020-10-31T19:09:53Z",
            }
            ...
        }
    ]
}
```

다음 `describe-db-instance-automated-backups` 예제에서는 `--db-instance-automated-backups-arn` 옵션을 사용하여 대상 리전에 복제된 백업을 표시합니다.

복제된 백업을 표시하려면

- 다음 명령 중 하나를 실행합니다.

  대상 LinuxmacOS, 또는Unix:

  ```
  aws rds describe-db-instance-automated-backups \
  --db-instance-automated-backups-arn "arn:aws:rds:us-east-1:123456789012:auto-
  backup:ab-L2IJCEXJP7XQ7HOJ4SIEXAMPLE"
  ```

  Windows의 경우:

  ```
  aws rds describe-db-instance-automated-backups ^
  --db-instance-automated-backups-arn "arn:aws:rds:us-east-1:123456789012:auto-
  backup:ab-L2IJCEXJP7XQ7HOJ4SIEXAMPLE"
  ```

이 출력에는 US East (N. Virginia)의 복제된 백업과 함께 미국 서부(오레곤)의 소스 DB 인스턴가 표시됩니다 .

```
{
    "DBInstanceAutomatedBackups": [
        {
            "DBInstanceArn": "arn:aws:rds:us-west-2:868710585169:db:mydatabase",
            "DbiResourceId": "db-L2IJCEXJP7XQ7HOJ4SIEXAMPLE",
            "DBInstanceAutomatedBackupsArn": "arn:aws:rds:us-east-1:123456789012:auto-
backup:ab-L2IJCEXJP7XQ7HOJ4SIEXAMPLE",
            "Region": "us-west-2",
            "DBInstanceIdentifier": "mydatabase",
            "RestoreWindow": {
                "EarliestTime": "2020-10-26T01:09:07Z",
                "LatestTime": "2020-10-31T19:01:23Z"
            },
            "AllocatedStorage": 50,
            "BackupRetentionPeriod": 7,
            "Status": "replicating",
            "Port": 1521,
            ...
        }
    ]
}
```

복제된 백업에서 지정된 시간으로 복구

Amazon RDS 콘솔을 사용하여 복제된 백업에서 DB 인스턴스를 특정 시점으로 복원할 수 있습니다. `restore-db-instance-to-point-in-time` AWS CLI 명령 또는 `RestoreDBInstanceToPointInTime` RDS API 작업을 사용할 수도 있습니다.

특정 시점으로 복구(PITR)에 대한 일반 정보는 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 섹션을 참조하세요.

> **Note**
>
> RDS for SQL Server에서 자동 백업이 복제될 때 옵션 그룹이 여러 AWS 리전 간에 복사되지 않습니다. 사용자 지정 옵션 그룹을 RDS for SQL Server DB 인스턴스와 연결한 경우 대상 리전에서 해당 옵션 그룹을 다시 생성할 수 있습니다. 그런 다음 대상 리전에서 DB 인스턴스를 복원하고 사용자 지정 옵션 그룹을 이 인스턴스에 연결합니다. 자세한 내용은 옵션 그룹 작업을 참조하세요.

콘솔

복제된 백업에서 DB 인스턴스를 지정된 시간으로 복원하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 리전 선택기에서 대상 리전(백업이 복제될 대상 리전)을 선택합니다.
3. 탐색 창에서 백업 자동화를 선택합니다.
4. [복제된 백업(Replicated backups)] 탭에서 복원할 DB 인스턴스를 선택합니다.
5. 작업에서 특정 시점으로 복구를 선택합니다.
6. 최근 복원 가능 시간을 선택하여 가능한 최근 시간으로 복원하거나, 사용자 지정을 선택하여 시간을 선택합니다.

   사용자 지정을 선택한 경우 인스턴스를 복원하려는 날짜와 시간을 입력합니다.

   > **Note**
   >
   > 시간은 현지 시간대로 표시됩니다. 즉, 협정 세계시(UTC)에서 오프셋으로 표시됩니다. 예를 들어 UTC-5는 동부 표준시/하절기 중부 표준시입니다.

7. DB 인스턴스 식별자에 대상 복원된 DB 인스턴스의 이름을 입력합니다.
8. (선택 사항) 필요에 따라 Auto Scaling 활성화와 같은 다른 옵션을 선택합니다.
9. 특정 시점으로 복구를 선택합니다.

AWS CLI

`restore-db-instance-to-point-in-time` AWS CLI 명령을 사용하여 새 DB 인스턴스를 생성합니다.

복제된 백업에서 DB 인스턴스를 지정된 시간으로 복원하려면

- 다음 명령 중 하나를 실행합니다.

  대상 LinuxmacOS, 또는Unix:

```
aws rds restore-db-instance-to-point-in-time \
    --source-db-instance-automated-backups-arn "arn:aws:rds:us-
east-1:123456789012:auto-backup:ab-L2IJCEXJP7XQ7HOJ4SIEXAMPLE" \
```

```
    --target-db-instance-identifier mytargetdbinstance \
    --restore-time 2020-10-14T23:45:00.000Z
```

Windows의 경우:

```
aws rds restore-db-instance-to-point-in-time ^
    --source-db-instance-automated-backups-arn "arn:aws:rds:us-
east-1:123456789012:auto-backup:ab-L2IJCEXJP7XQ7HOJ4SIEXAMPLE" ^
    --target-db-instance-identifier mytargetdbinstance ^
    --restore-time 2020-10-14T23:45:00.000Z
```

RDS API

DB 인스턴스를 특정 시간으로 복원하려면, `RestoreDBInstanceToPointInTime` Amazon RDS API 작업을 다음 파라미터와 함께 호출합니다.

- `SourceDBInstanceAutomatedBackupsArn`
- `TargetDBInstanceIdentifier`
- `RestoreTime`

자동 백업 복제 중지

Amazon RDS 콘솔을 사용하여 DB 인스턴스에 대한 백업 복제를 중지할 수 있습니다. `stop-db-instance-automated-backups-replication` AWS CLI 명령 또는 `StopDBInstanceAutomatedBackupsReplication` RDS API 작업을 사용할 수도 있습니다.

복제된 백업은 생성 시 설정된 백업 보존 기간에 따라 보존됩니다.

콘솔

소스 리전의 자동 백업 페이지에서 백업 복제를 중지합니다.

AWS 리전으로의 백업 복제를 중지하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. [리전 선택기(Region selector)]에서 소스 리전을 선택합니다.
3. 탐색 창에서 백업 자동화를 선택합니다.

4. [현재 리전(Current Region)] 탭에서 백업 복제를 중지할 DB 인스턴스를 선택합니다.
5. 작업(Actions)에서 교차 리전 복제 관리(Manage cross-Region replication)를 선택합니다.
6. 백업 복제(Backup replication)에서 다른 AWS 리전으로의 복제 활성화(Enable replication to another AWS 리전) 확인란 선택을 취소합니다.
7. Save(저장)를 선택합니다.

복제된 백업은 대상 지역의 [자동 백업(Automated backups)] 페이지의 [보존됨(Retained)] 탭에 나열됩니다.

AWS CLI

stop-db-instance-automated-backups-replication AWS CLI 명령을 사용하여 백업 복제를 중지합니다.

다음 CLI 예제는 DB 인스턴스의 자동 백업이 미국 서부(오레곤) 리전에서 복제되는 것을 중지합니다.

백업 복제를 중지하려면

- 다음 명령 중 하나를 실행합니다.

  대상 LinuxmacOS, 또는Unix:

```
aws rds stop-db-instance-automated-backups-replication \
--region us-east-1 \
--source-db-instance-arn "arn:aws:rds:us-west-2:123456789012:db:mydatabase"
```

  Windows의 경우:

```
aws rds stop-db-instance-automated-backups-replication ^
--region us-east-1 ^
--source-db-instance-arn "arn:aws:rds:us-west-2:123456789012:db:mydatabase"
```

RDS API

다음 파라미터와 함께 StopDBInstanceAutomatedBackupsReplication RDS API 작업을 사용하여 백업 복제를 중지합니다.

- `Region`
- `SourceDBInstanceArn`

복제된 백업 삭제

Amazon RDS 콘솔을 사용하여 DB 인스턴스의 복제된 백업을 삭제할 수 있습니다. `delete-db-instance-automated-backups` AWS CLI 명령 또는 `DeleteDBInstanceAutomatedBackup` RDS API 작업을 사용할 수도 있습니다.

콘솔

자동 백업] 페이지에서 대상 리전의 복제된 백업을 삭제합니다.

복제된 백업을 삭제하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. [리전 선택기(Region selector)]에서 대상 리전을 선택합니다.
3. 탐색 창에서 백업 자동화를 선택합니다.
4. [복제된 백업(Replicated backups)] 탭에서 복제된 백업을 삭제할 DB 인스턴스를 선택합니다.
5. [Actions]에 대해 [Delete]를 선택합니다.
6. 확인 페이지에서 **`delete me`**를 입력하고 삭제를 선택합니다.

AWS CLI

`delete-db-instance-automated-backup` AWS CLI 명령을 사용하여 복제된 백업을 삭제합니다.

`describe-db-instances` CLI 명령을 사용하여 복제된 백업의 Amazon 리소스 이름(ARN)을 찾을 수 있습니다. 자세한 내용은 복제된 백업에 대한 정보 찾기 섹션을 참조하세요.

복제된 백업을 삭제하려면

- 다음 명령 중 하나를 실행합니다.

  대상 LinuxmacOS, 또는Unix:

  ```
  aws rds delete-db-instance-automated-backup \
  --db-instance-automated-backups-arn "arn:aws:rds:us-east-1:123456789012:auto-
  backup:ab-L2IJCEXJP7XQ7HOJ4SIEXAMPLE"
  ```

  Windows의 경우:

```
aws rds delete-db-instance-automated-backup ^
--db-instance-automated-backups-arn "arn:aws:rds:us-east-1:123456789012:auto-
backup:ab-L2IJCEXJP7XQ7HOJ4SIEXAMPLE"
```

RDS API

`DeleteDBInstanceAutomatedBackup` 파라미터와 함께 `DBInstanceAutomatedBackupsArn` RDS API 작업을 사용하여 복제된 백업을 삭제합니다.

수동 백업 관리

이 섹션에서는 DB 인스턴스 및 DB 클러스터의 수동 백업을 관리하는 방법을 보여줍니다.

Single-AZ DB 인스턴스에 대한 DB 스냅샷 생성에 대한 자세한 내용은 Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성 섹션을 참조하세요.

다중 AZ DB 클러스터에 대한 DB 스냅샷 생성에 대한 자세한 내용은 Amazon RDS용 다중 AZ DB 클러스터 스냅샷 생성 섹션을 참조하세요.

DB 스냅샷 삭제에 대한 자세한 내용은 Amazon RDS용 DB 스냅샷 삭제 섹션을 참조하세요.

Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성

Amazon RDS는 개별 데이터베이스가 아닌 전체 DB 인스턴스를 백업하여 DB 인스턴스의 스토리지 볼륨 스냅샷을 생성합니다. 단일 AZ DB 인스턴스에서 이 DB 스냅샷을 생성하면 잠시 I/O가 중단되는데, 해당 DB 인스턴스의 크기 및 클래스에 따라 대체로 몇 초에서 몇 분 정도 지속됩니다. MariaDB, MySQL, Oracle 및 PostgreSQL의 경우, 다중 AZ 배포에 대한 백업 시 기본 AZ에서는 I/O 작업이 중단되지 않습니다. 백업이 예비 복제본으로부터 수행되기 때문입니다. SQL Server의 경우, 다중 AZ 배포에 대한 백업 도중 I/O 작업이 일시적으로 중단됩니다.

DB 스냅샷을 생성할 때는 백업할 DB 인스턴스를 구분한 다음 나중에 복구할 수 있도록 DB 스냅샷을 명명해야 합니다. 스냅샷을 생성하는 데 걸리는 시간은 데이터베이스 크기에 따라 다릅니다. 스냅샷에는 전체 스토리지 볼륨이 포함되기 때문에 임시 파일 같은 파일들의 크기가 스냅샷을 생성하는 데 걸리는 시간에 영향을 미치기도 합니다.

> **Note**
>
> DB 스냅샷을 생성하려면 DB 인스턴스가 `available` 상태여야 합니다.
> PostgreSQL DB 인스턴스의 경우 로그되지 않은 테이블의 데이터가 스냅샷에서 복원되지 않을 수 있습니다. 자세한 내용은 PostgreSQL로 작업하기 위한 모범 사례 섹션을 참조하세요.

자동 백업과 달리 수동 스냅샷에는 백업 보존 기간이 적용되지 않습니다. 스냅샷이 만료되지 않습니다.

MariaDB, MySQL 및 PostgreSQL 데이터를 매우 장기간 백업하려면 스냅샷 데이터를 Amazon S3로 내보내는 것이 좋습니다. DB 엔진의 메이저 버전이 더 이상 지원되지 않는 경우에는 스냅샷에서 해당 버전으로 복원할 수 없습니다. 자세한 내용은 Amazon S3 for Amazon RDS로 DB 스냅샷 데이터 내보내기 섹션을 참조하세요.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB 스냅샷을 생성할 수 있습니다.

콘솔

DB 스냅샷을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.

   [수동 스냅샷(Manual snapshots)] 목록이 나타납니다.

3. [스냅샷 생성(Take Snapshot)]을 선택합니다.

   [DB 스냅샷 생성(Take DB Snapshot)] 창이 나타납니다.
4. 스냅샷을 생성할 DB 인스턴스를 선택합니다.
5. 스냅샷 이름을 입력합니다.
6. [스냅샷 생성(Take Snapshot)]을 선택합니다.

수동 스냅샷 목록이 나타나고 새로운 DB 스냅샷 상태가 `Creating`으로 표시됩니다. 스냅샷 상태가 `Available`이 되면 스냅샷 생성 시간을 볼 수 있습니다.

AWS CLI

AWS CLI를 사용하여 DB 스냅샷을 생성할 때는 백업할 DB 인스턴스를 식별한 후 나중에 복구할 수 있도록 DB 스냅샷에 이름을 지정해야 합니다. 이를 위해서는 AWS CLI `create-db-snapshot` 명령을 다음 파라미터와 함께 사용하면 됩니다.

- `--db-instance-identifier`
- `--db-snapshot-identifier`

이번 예에서는 *`mydbinstance`*라는 이름의 DB 인스턴스에서 *`mydbsnapshot`*라는 이름의 DB 스냅샷을 생성합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-snapshot \
    --db-instance-identifier mydbinstance \
    --db-snapshot-identifier mydbsnapshot
```

Windows의 경우:

```
aws rds create-db-snapshot ^
    --db-instance-identifier mydbinstance ^
    --db-snapshot-identifier mydbsnapshot
```

RDS API

Amazon RDS API를 사용하여 DB 스냅샷을 생성할 때는 백업할 DB 인스턴스를 식별한 후 나중에 복구할 수 있도록 DB 스냅샷에 이름을 지정해야 합니다. 이를 위해서는 Amazon RDS API `CreateDBSnapshot` 명령을 다음 파라미터와 함께 사용하면 됩니다.

- `DBInstanceIdentifier`
- `DBSnapshotIdentifier`

Amazon RDS용 다중 AZ DB 클러스터 스냅샷 생성

다중 AZ DB 클러스터 스냅샷을 생성할 때 백업할 다중 AZ DB 클러스터를 확인한 후 나중에 복원할 수 있도록 DB 클러스터 스냅샷의 이름을 지정해야 합니다. 다중 AZ DB 클러스터 스냅샷을 공유할 수도 있습니다. 지침은 the section called “DB 스냅샷 공유” 단원을 참조하십시오.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 다중 AZ DB 클러스터 스냅샷을 생성할 수 있습니다.

매우 장기간 백업하려면 스냅샷 데이터를 Amazon S3로 내보내는 것이 좋습니다. DB 엔진의 메이저 버전이 더 이상 지원되지 않는 경우에는 스냅샷에서 해당 버전으로 복원할 수 없습니다. 자세한 내용은 Amazon S3 for Amazon RDS로 DB 스냅샷 데이터 내보내기 단원을 참조하십시오.

콘솔

DB 클러스터 스냅샷을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
3. 목록에서 스냅샷을 생성할 다중 AZ DB 클러스터를 선택합니다.
4. 작업에서 스냅샷 만들기를 선택합니다.

   [DB 스냅샷 생성(Take DB Snapshot)] 창이 나타납니다.
5. 스냅샷 이름(Snapshot name)에 스냅샷 이름을 입력합니다.
6. [스냅샷 생성(Take Snapshot)]을 선택합니다.

스냅샷(Snapshots) 페이지가 나타나고 새로운 다중 AZ DB 클러스터 스냅샷 상태가 `Creating`으로 표시됩니다. 스냅샷 상태가 `Available`이 되면 스냅샷 생성 시간을 볼 수 있습니다.

AWS CLI

다음 옵션과 함께 AWS CLI create-db-cluster-snapshot 명령을 사용하여 다중 AZ DB 클러스터 스냅샷을 생성할 수 있습니다.

- `--db-cluster-identifier`
- `--db-cluster-snapshot-identifier`

이 예제에서는 *mymultiazdbcluster*라는 DB 클러스터에 대해 *mymultiazdbclustersnapshot*이라는 다중 AZ DB 클러스터 스냅샷을 생성합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-cluster-snapshot \
    --db-cluster-identifier mymultiazdbcluster \
    --db-cluster-snapshot-identifier mymultiazdbclustersnapshot
```

Windows의 경우:

```
aws rds create-db-cluster-snapshot ^
    --db-cluster-identifier mymultiazdbcluster ^
    --db-cluster snapshot-identifier mymultiazdbclustersnapshot
```

RDS API

다음 파라미터를 사용하여 Amazon RDS API CreateDBClusterSnapshot 작업을 통해 다중 AZ DB 클러스터 스냅샷을 생성할 수 있습니다.

- `DBClusterIdentifier`
- `DBClusterSnapshotIdentifier`

다중 AZ DB 클러스터 스냅샷 삭제

Amazon RDS에서 관리하는 다중 AZ DB 스냅샷이 더 이상 필요하지 않으면 삭제할 수 있습니다. 지침은 the section called “DB 스냅샷 삭제” 단원을 참조하세요.

Amazon RDS용 DB 스냅샷 삭제

Amazon RDS에서 관리하는 DB 스냅샷이 더 이상 필요하지 않으면 삭제할 수 있습니다.

> **Note**
>
> AWS Backup에서 관리하는 백업을 삭제하려면 AWS Backup 콘솔을 사용하십시오. AWS Backup에 대한 자세한 내용은 AWS Backup 개발자 안내서를 참조하세요.

DB 스냅샷 삭제

AWS Management Console, AWS CLI 또는 RDS API를 사용해 수동, 공유 또는 퍼블릭 DB 스냅샷을 삭제할 수 있습니다.

공유 또는 퍼블릭 스냅샷을 삭제하려면 해당 스냅샷을 소유하는 AWS 계정에 로그인해야 합니다.

DB 인스턴스를 삭제하지 않은 채로 삭제하고 싶은 자동화된 DB 스냅샷이 있는 경우 DB 인스턴스의 백업 보존 기간을 0으로 변경하십시오. 자동화된 스냅샷은 변경 내용이 적용되면 삭제됩니다. 그다음 유지 관리 기간까지 기다리고 싶지 않다면 변경 내용을 즉시 적용할 수 있습니다. 변경이 완료된 후에는 백업 보존 기간을 0보다 큰 수로 설정하여 자동 백업을 다시 활성화할 수 있습니다. DB 인스턴스 수정에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

보존된 자동 백업 및 수동 스냅샷은 삭제될 때까지 청구 요금이 부과됩니다. 자세한 내용은 보존 비용 단원을 참조하십시오.

DB 인스턴스를 삭제한 경우 DB 인스턴스에 대한 자동 백업을 제거하여 자동화된 DB 스냅샷을 삭제할 수 있습니다. 자동 백업에 대한 자세한 내용은 백업 소개 단원을 참조하십시오.

콘솔

DB 스냅샷을 삭제하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.

   [수동 스냅샷(Manual snapshots)] 목록이 나타납니다.
3. 삭제하고 싶은 DB 스냅샷을 선택합니다.
4. 작업(Actions)에서 스냅샷 삭제(Delete snapshot)를 선택합니다.

5. 확인 페이지에서 삭제를 선택합니다.

AWS CLI

AWS CLI 명령 copy-db-snapshot을 사용하여 DB 스냅샷을 삭제할 수 있습니다.

다음 옵션을 사용하여 DB 스냅샷을 삭제할 수 있습니다.

- `--db-snapshot-identifier` – DB 스냅샷 식별자입니다.

Example

다음 코드는 `mydbsnapshot` DB 스냅샷을 삭제합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds delete-db-snapshot \
    --db-snapshot-identifier mydbsnapshot
```

Windows의 경우:

```
aws rds delete-db-snapshot ^
    --db-snapshot-identifier mydbsnapshot
```

RDS API

Amazon RDS API 연산 DeleteDBSnapshot을 사용하여 DB 스냅샷을 삭제할 수 있습니다.

다음 파라미터를 사용하여 DB 스냅샷을 삭제할 수 있습니다.

- `DBSnapshotIdentifier` – DB 스냅샷 식별자입니다.

DB 인스턴스 복원

이 섹션에서는 DB 인스턴스를 복원하는 방법에 대해 설명합니다. 이 페이지는 DB 스냅샷에서 Amazon RDS DB 인스턴스로 복원하는 방법을 보여줍니다.

Amazon RDS는 개별 데이터베이스가 아닌 전체 DB 인스턴스를 백업하여 DB 인스턴스의 스토리지 볼륨 스냅샷을 생성합니다. DB 스냅샷에서 복원하여 새 DB 인스턴스를 생성할 수 있습니다. 복원할 DB 스냅샷의 이름을 입력한 다음 복원으로부터 생성된 새 DB 인스턴스의 이름을 입력하면 됩니다. DB 스냅샷에서 기존 DB 인스턴스로 복원할 수 없습니다. 복원할 때 새 DB 인스턴스가 생성됩니다.

복원된 DB 인스턴스는 `available` 상태가 되면 바로 사용할 수 있습니다. DB 인스턴스는 백그라운드에서 데이터를 계속 로드합니다. 이를 지연 로딩이라고 합니다.

아직 로드되지 않은 데이터에 액세스하는 경우, DB 인스턴스는 Amazon S3에서 요청된 데이터를 즉시 다운로드한 후, 백그라운드에서 데이터의 나머지 로드를 계속 진행합니다. 자세한 내용은 Amazon EBS 스냅샷을 참조하세요.

빠른 액세스가 필요한 테이블에 대한 지연 로딩의 영향을 완화하기 위해 `SELECT *`와 같은 전체 테이블 스캔과 관련된 작업을 수행할 수 있습니다. 이를 통해 Amazon RDS는 S3에서 백업된 모든 테이블 데이터를 다운로드할 수 있습니다.

DB 인스턴스를 복원하고 원본 DB 스냅샷과 다른 스토리지 유형을 사용할 수 있습니다. 이 경우 데이터를 새로운 스토리지 유형으로 마이그레이션하기 위해 추가 작업이 필요하기 때문에 복원 프로세스가 오래 걸립니다. 마그네틱 스토리지로 또는 스토리지에서 복원할 때 마이그레이션 프로세스의 속도가 가장 느립니다. 이는 마그네틱 스토리지에는 프로비저닝된 IOPS 또는 범용(SSD) 스토리지의 IOPS 기능이 없기 때문입니다.

DB 인스턴스 스냅샷에서 DB 인스턴스를 복원하기 위해 AWS CloudFormation을 사용할 수 있습니다. 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS::RDS::DBInstance를 참조하세요.

> **Note**
>
> 공유되고 동시에 암호화된 DB 스냅샷에서는 DB 인스턴스를 복원할 수 없습니다. 대신 DB 스냅샷의 사본을 만들어 거기에서 DB 인스턴스를 복원할 수 있습니다. 자세한 내용은 Amazon RDS용 DB 스냅샷 복사 섹션을 참조하세요.

RDS 추가 지원 버전을 사용하여 DB 인스턴스를 복원하는 방법에 대한 자세한 내용은 Amazon RDS 추가 지원이 포함된 DB 인스턴스 또는 다중 AZ DB 클러스터 복원 섹션을 참조하세요.

스냅샷에서 복원

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB 스냅샷에서 DB 인스턴스를 복원할 수 있습니다.

> **Note**
>
> DB 인스턴스 복원 시 스토리지 양을 줄일 수 없습니다. 할당된 스토리지를 늘릴 경우, 10% 이상 늘려야 합니다. 이 값을 10% 미만으로 늘리면 오류가 발생합니다. RDS for SQL Server DB 인스턴스를 복원할 때는 할당된 스토리지를 늘릴 수 없습니다.

콘솔

DB 스냅샷에서 DB 인스턴스를 복원하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.
3. 복원 원본으로 사용할 DB 스냅샷을 선택합니다.
4. 작업에서 스냅샷 복원을 선택합니다.
5. 스냅샷 복원(Restore snapshot) 페이지의 DB 인스턴스 식별자(DB instance identifier)에 복원된 DB 인스턴스의 이름을 입력합니다.
6. 할당된 스토리지 크기와 같은 다른 설정을 지정합니다.

   각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하세요.
7. [DB 인스턴스 복원(Restore DB instance)]을 선택합니다.

AWS CLI

DB 스냅샷에서 DB 인스턴스를 복원하려면 AWS CLI 명령 restore-db-instance-from-db-snapshot을 사용합니다.

이 예에서는 `mydbsnapshot`이라는 이전에 생성된 DB 스냅샷에서 복원합니다. 그리고 `mynewdbinstance`라는 새 DB 인스턴스로 복원해야 합니다. 이 예에서는 할당된 스토리지 크기도 설정합니다.

다른 설정을 지정할 수 있습니다. 각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하세요.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds restore-db-instance-from-db-snapshot \
    --db-instance-identifier mynewdbinstance \
    --db-snapshot-identifier mydbsnapshot \
    --allocated-storage 100
```

Windows의 경우:

```
aws rds restore-db-instance-from-db-snapshot ^
    --db-instance-identifier mynewdbinstance ^
    --db-snapshot-identifier mydbsnapshot ^
    --allocated-storage 100
```

다음과 비슷한 출력이 반환됩니다.

```
DBINSTANCE  mynewdbinstance  db.t3.small  MySQL     50       sa              creating
 3  n  8.0.28  general-public-license
```

RDS API

DB 스냅샷에서 DB 인스턴스를 복원하려면 Amazon RDS API 함수 RestoreDBInstanceFromDBSnapshot을 다음 파라미터와 함께 호출합니다.

- `DBInstanceIdentifier`
- `DBSnapshotIdentifier`

고려 사항

DB 스냅샷에서 DB 인스턴스로 복원할 때 고려해야 할 사항은 다음 주제를 참조하세요.

주제

- 파라미터 그룹 고려 사항
- 보안 그룹 고려 사항
- 옵션 그룹 고려 사항
- 리소스 태깅 고려 사항
- Db2 고려 사항
- Microsoft SQL Server 고려 사항
- MySQL 고려 사항
- Oracle Database 고려 사항

파라미터 그룹 고려 사항

복원된 DB 인스턴스를 올바른 파라미터 그룹과 연결할 수 있도록 생성한 DB 스냅샷에 대한 DB 파라미터 그룹을 유지하는 것이 좋습니다.

기본 DB 파라미터 그룹은 다른 그룹을 선택하지 않는 한 복원된 인스턴스와 연결됩니다. 기본 파라미터 그룹에서는 사용자 정의 파라미터 설정을 사용할 수 없습니다.

DB 인스턴스를 복원할 때 파라미터 그룹을 지정할 수 있습니다.

DB 파라미터 그룹에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.

보안 그룹 고려 사항

DB 인스턴스를 복원할 때 다른 항목을 선택하지 않는 한 기본 Virtual Private Cloud(VPC), DB 서브넷 그룹 및 VPC 보안 그룹이 복원된 인스턴스와 연결됩니다.

- Amazon RDS 콘솔을 사용하는 경우 사용자 지정 VPC 보안 그룹을 지정하여 인스턴스와 연결하거나 새 VPC 보안 그룹을 생성할 수 있습니다.
- AWS CLI를 사용 중이라면 `restore-db-instance-from-db-snapshot` 명령에 `--vpc-security-group-ids` 옵션을 포함하여 인스턴스에 연결할 사용자 지정 VPC 보안 그룹을 지정할 수 있습니다.
- Amazon RDS API를 사용 중이라면 `VpcSecurityGroupIds.VpcSecurityGroupId.N` 작업에 `RestoreDBInstanceFromDBSnapshot` 파라미터를 포함할 수 있습니다.

복원이 완료되고 새 DB 인스턴스를 사용할 수 있게 되면 DB 인스턴스를 수정하여 VPC 설정을 변경할 수도 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

옵션 그룹 고려 사항

DB 인스턴스를 복원할 경우 기본 DB 옵션 그룹은 대부분의 경우 복원된 DB 인스턴스와 연결됩니다.

원본 DB 인스턴스가 영구 또는 영구 옵션이 포함된 옵션 그룹과 연결된 경우는 예외입니다. 예를 들어 원본 DB 인스턴스가 Oracle Transparent Data Encryption(TDE)을 사용하는 경우 복원된 DB 인스턴스는 TDE 옵션이 있는 옵션 그룹을 사용해야 합니다.

DB 인스턴스를 다른 VPC로 복원하는 경우 다음 중 하나를 수행하여 DB 옵션 그룹을 할당해야 합니다.

- 해당 VPC 그룹의 기본 옵션 그룹을 인스턴스에 할당합니다.
- 해당 VPC에 연결된 다른 옵션 그룹을 할당합니다.
- 새 옵션 그룹을 생성하여 해당 DB 인스턴스에 배정. Oracle TDE와 같은 지속적 또는 영구적 옵션의 경우 지속적 또는 영구적 옵션을 포함하는 새 옵션 그룹을 생성해야 합니다.

DB 옵션 그룹에 대한 자세한 내용은 옵션 그룹 작업 섹션을 참조하세요.

리소스 태깅 고려 사항

DB 스냅샷에서 DB 인스턴스를 복원할 때 RDS는 새 태그를 지정하는지 여부를 확인합니다. 그렇다면 새 태그가 복원된 DB 인스턴스에 추가됩니다. 새 태그가 없는 경우 RDS는 스냅샷 생성 시 원본 DB 인스턴스의 태그를 복원된 DB 인스턴스에 추가합니다.

자세한 내용은 DB 스냅샷에 태그 복사 섹션을 참조하세요.

Db2 고려 사항

BYOL 모델에서는 Amazon RDS for Db2 DB 인스턴스를 사용자 IBM Site ID 및 사용자 IBM Customer ID가 포함된 사용자 지정 파라미터 그룹과 연결해야 합니다. 그렇지 않으면 스냅샷에서 DB 인스턴스를 복원하려는 시도가 실패합니다. Amazon RDS for Db2 DB 인스턴스도 AWS License Manager 자체 관리형 라이선스와 연결되어야 합니다. 자세한 내용은 Db2에 기존 보유 라이선스 사용 섹션을 참조하세요.

AWS Marketplace를 통한 Db2 라이선스 모델을 사용하려면 사용하려는 특정 IBM Db2 에디션에 대한 활성 AWS Marketplace 구독이 필요합니다. 아직 구독이 없는 경우 해당 IBM Db2 에디션에 대해 AWS Marketplace에서 Db2를 구독합니다. 자세한 내용은 AWS Marketplace를 통한 Db2 라이선스 섹션을 참조하세요.

Microsoft SQL Server 고려 사항

RDS for Microsoft SQL Server DB 스냅샷을 새 인스턴스로 복원할 때에는 항상 스냅샷과 동일한 버전으로 복원할 수 있습니다. 경우에 따라서는 DB 인스턴스의 버전을 변경할 수도 있습니다. 다음은 버전을 변경할 때 적용되는 제한 사항입니다.

- DB 스냅샷에는 새 버전에 할당되는 스토리지가 충분히 있어야 합니다.
- 다음 버전 변경만이 지원됩니다.
  - Standard Edition에서 Enterprise Edition으로 변경
  - Web Edition에서 Standard Edition 또는 Enterprise Edition으로 변경
  - Express Edition에서 Web Edition, Standard Edition 또는 Enterprise Edition으로 변경

스냅샷을 복원하여 지원되지 않는 새 버전으로 변경할 경우, 기본 백업과 복원 기능을 사용할 수 있습니다. SQL Server는 데이터베이스에서 활성화된 SQL Server 기능을 기반으로 데이터베이스가 새 버전과 호환되는지 여부를 확인합니다. 자세한 내용은 기본 백업 및 복원 기능을 사용하여 SQL Server 데이터베이스 가져오기 및 내보내기 섹션을 참조하세요.

MySQL 고려 사항

지원되지 않는 엔진 버전을 사용하여 RDS for MySQL DB 스냅샷에서 복원하려면 DB 스냅샷을 두 번 이상 업그레이드해야 할 수 있습니다. 업그레이드 옵션에 대한 자세한 내용은 RDS for MySQL에서 지원되지 않는 엔진 버전이 있는 DB 스냅샷에 대한 업그레이드 옵션 섹션을 참조하세요.

RDS for MySQL DB 스냅샷 엔진 버전을 업그레이드하는 방법에 대한 자세한 내용은 MySQL DB 스냅샷 엔진 버전 업그레이드 섹션을 참조하세요.

Oracle Database 고려 사항

DB 스냅샷에서 Oracle 데이터베이스를 복원할 때는 다음 사항을 고려하세요.

- DB 스냅샷을 복원하기 전에 이후 Oracle 데이터베이스 릴리스로 업그레이드할 수 있습니다. 자세한 내용은 Oracle DB 스냅샷 업그레이드 섹션을 참조하세요.
- 단일 테넌트 구성을 사용하는 CDB 인스턴스의 스냅샷을 복원하는 경우 PDB 이름을 변경할 수 있습니다. CDB 인스턴스가 다중 테넌트 구성을 사용하는 경우 PDB 이름을 변경할 수 없습니다. 자세한 내용은 CDB 백업 및 복원 섹션을 참조하세요.
- CDB 이름은 항상 `RDSCDB`이며 변경할 수 없습니다. 이 CDB 이름은 모든 CDB 인스턴스에서 동일합니다.

- DB 스냅샷에서는 테넌트 데이터베이스와 직접 상호 작용할 수 없습니다. 다중 테넌트 구성을 사용하는 CDB 인스턴스의 스냅샷을 복원하는 경우 포함된 테넌트 데이터베이스를 모두 복원하는 것입니다. describe-db-snapshot-tenant-database를 사용하여 복원하기 전에 DB 스냅샷 내의 테넌트 데이터베이스를 검사할 수 있습니다.
- Oracle GoldenGate를 사용하는 경우 항상 `compatible` 파라미터를 포함하는 파라미터 그룹을 유지하십시오. DB 스냅샷에서 DB 인스턴스를 복원하는 경우 `compatible` 값 이상을 가진 파라미터 그룹을 지정합니다.
- DB 스냅샷을 복원할 때 데이터베이스 이름을 변경하도록 선택할 수 있습니다. 온라인 redo 로그의 총 크기가 20GB를 초과하는 경우 RDS는 온라인 redo 로그 크기를 기본 설정인 512MB(4x128MB)로 재설정할 수 있습니다. 크기가 작을수록 적절한 시간 내에 복원 작업을 완료할 수 있습니다. 나중에 온라인 redo 로그를 다시 생성하고 크기를 변경할 수 있습니다.
- AWS Secrets Manager에서 마스터 사용자 암호를 관리할 수 있습니다. 자세한 내용은 AWS Secrets Manager를 통한 마스터 사용자 암호 관리 개요 섹션을 참조하세요.

Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원

소스 DB 인스턴스를 수정하지 않고 DB 인스턴스를 특정 시점으로 복원하여 새 DB 인스턴스를 생성할 수 있습니다.

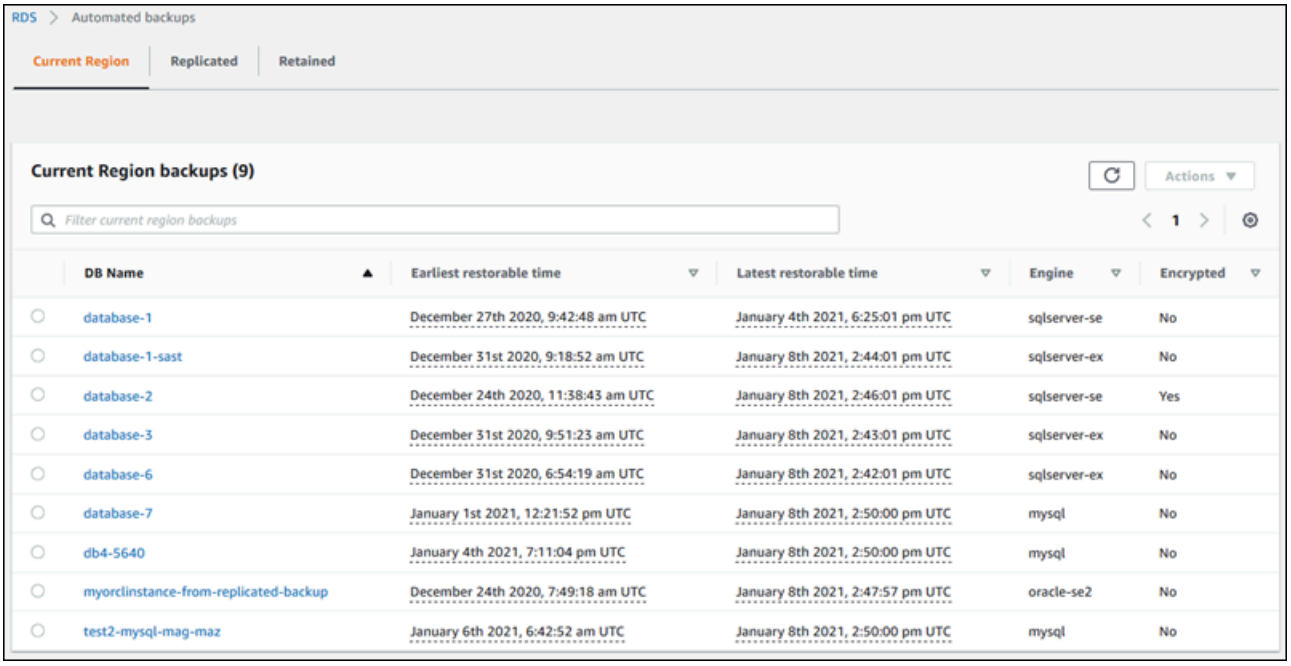
특정 시점으로 DB 인스턴스를 복원할 때 기본 Virtual Private Cloud(VPC) 보안 그룹을 선택할 수 있습니다. 또는 DB 인스턴스에 사용자 정의 VPC 보안 그룹을 적용할 수 있습니다.

복원된 DB 인스턴스는 기본 DB 파라미터 및 옵션 그룹과 자동으로 연결됩니다. 하지만 복원 중에 사용자 지정 파라미터 그룹 및 옵션 그룹을 지정하여 적용할 수 있습니다.

원본 DB 인스턴스에 리소스 태그가 있는 경우 RDS는 복원된 DB 인스턴스에 최신 태그를 추가합니다.

RDS는 DB 인스턴스에 대한 트랜잭션 로그를 Amazon S3에 5분마다 업로드합니다. DB 인스턴스의 최근 복원 가능 시간을 확인하려면 AWS CLIdescribe-db-instances 명령을 사용한 후 DB 인스턴스의 `LatestRestorableTime` 필드에 반환되는 값을 살펴봅니다. Amazon RDS 콘솔에서 각 DB 인스턴스의 복원 가능한 최신 시간을 보려면 [자동 백업을 선택합니다.

백업 보존 기간 중 어느 특정 시점으로든 복원할 수 있습니다. 각 DB 인스턴스의 복원 가능한 가장 빠른 시간을 보려면 Amazon RDS 콘솔에서 자동 백업을 선택합니다.



> **Note**
>
> 프로비저닝된 IOPS 스토리지를 원본 DB 인스턴스로 사용하는 경우 동일하거나 유사한 DB 인스턴스 크기와 IOPS로 복원하는 것이 좋습니다. 예를 들어, 호환되지 않는 IOPS 값이 있는 DB 인스턴스 크기를 선택하면 오류가 발생할 수 있습니다.

RDS 추가 지원 버전을 사용하여 DB 인스턴스를 복원하는 방법에 대한 자세한 내용은 Amazon RDS 추가 지원이 포함된 DB 인스턴스 또는 다중 AZ DB 클러스터 복원 섹션을 참조하세요.

일부 Amazon RDS 데이터베이스 엔진에는 특정 시점에서 복원할 때 다음과 같이 특별히 고려할 사항이 있습니다.

- Amazon RDS for Db2 DB 인스턴스에서 암호 인증을 사용하는 경우, `rdsadmin.add_user`를 포함한 사용자 관리 작업은 로그에 캡처되지 않습니다. 이러한 작업을 수행하려면 전체 스냅샷 백업이 필요합니다.

  BYOL 모델에서는 RDS for Db2 DB 인스턴스를 사용자 IBM Site ID 및 사용자 IBM Customer ID가 포함된 사용자 지정 파라미터 그룹과 연결해야 합니다. 그렇지 않으면 DB 인스턴스를 특정 시점으로 복원하려는 시도가 실패합니다. Amazon RDS for Db2 DB 인스턴스도 AWS License Manager 자체 관리형 라이선스와 연결되어야 합니다. 자세한 내용은 Db2에 기존 보유 라이선스 사용 섹션을 참조하세요.

  AWS Marketplace를 통한 Db2 라이선스 모델을 사용하려면 사용하려는 특정 IBM Db2 에디션에 대한 활성 AWS Marketplace 구독이 필요합니다. 아직 구독이 없는 경우 해당 IBM Db2 에디션에 대해 AWS Marketplace에서 Db2를 구독합니다. 자세한 내용은 AWS Marketplace를 통한 Db2 라이선스 섹션을 참조하세요.
- RDS for Oracle DB 인스턴스를 특정 시점으로 복원할 때, 복원된 DB 인스턴스를 위한 다른 DB 엔진, 라이선스 모델 및 DBName(SID)을 지정할 수 있습니다. RDS가 AWS Secrets Manager에 마스터 사용자 암호 관리를 저장하도록 지정할 수도 있습니다. 자세한 내용은 AWS Secrets Manager를 통한 마스터 사용자 암호 관리 개요 섹션을 참조하세요.
- Microsoft SQL Server DB 인스턴스를 특정 시점으로 복원할 때 그 인스턴스 내의 각 데이터베이스는 인스턴스 내에 있는 각각의 다른 데이터베이스와 1초 이내의 시점으로 복원됩니다. 인스턴스 내에 있는 여러 데이터베이스에 걸쳐 이루어지는 트랜잭션은 일관되게 복원되지 않을 수 있습니다.
- SQL Server DB 인스턴스의 경우 `OFFLINE`, `EMERGENCY` 및 `SINGLE_USER` 모드는 지원되지 않습니다. 데이터베이스를 이들 모드 중 하나로 설정하면 최근 복원 가능 시간이 전체 인스턴스를 앞서 가는 동작이 중지됩니다.

- SQL Server 데이터베이스의 복구 모델 변경과 같은 동작으로 특정 시점으로 복구에 사용되는 로그 시퀀스가 중단될 수 있습니다. 경우에 따라 Amazon RDS는 이 문제를 감지하고 최근 복원 가능 시간이 앞으로 진행되지 않도록 합니다. SQL Server 데이터베이스에서 `BULK_LOGGED` 복구 모델을 사용하는 등의 경우에는 로그 시퀀스의 중단이 감지되지 않습니다. 로그 시퀀스가 중단될 경우 SQL Server DB 인스턴스를 특정 시점으로 복원하지 못할 수도 있습니다. 이런 이유로, Amazon RDS는 SQL Server 데이터베이스의 복구 모델 변경을 지원하지 않습니다.

또한 AWS Backup을 사용해 Amazon RDS DB 인스턴스의 백업을 관리하는 방법도 있습니다. DB 인스턴스가 AWS Backup의 백업 계획에 연결되어 있는 경우, 해당 백업 계획이 특정 시점으로 복구에 사용됩니다. AWS Backup으로 생성된 백업은 이름이 `awsbackup:`*`AWS-Backup-job-number`*로 끝납니다. AWS Backup에 대한 자세한 내용은 AWS Backup 개발자 안내서를 참조하세요.

> **Note**
>
> 이 주제의 정보는 Amazon RDS에 적용됩니다. Amazon Aurora DB 클러스터 복원에 대한 자세한 내용은 지정된 시간으로 DB 클러스터 복원을 참조하세요.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB 인스턴스를 특정 시점으로 복원할 수 있습니다.

> **Note**
>
> DB 인스턴스 복원 시 스토리지 양을 줄일 수 없습니다. 할당된 스토리지를 늘릴 경우, 10% 이상 늘려야 합니다. 이 값을 10% 미만으로 늘리면 오류가 발생합니다. RDS for SQL Server DB 인스턴스를 복원할 때는 할당된 스토리지를 늘릴 수 없습니다.

콘솔

지정된 시간으로 DB 인스턴스 복원

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 자동 백업(Automated backups)을 선택합니다.

   자동 백업은 현재 리전(Current Region) 탭에 표시됩니다.
3. 복원하려는 DB 인스턴스를 선택합니다.

4. 작업에서 특정 시점으로 복구를 선택합니다.

   특정 시점으로 복구 창이 나타납니다.

5. 최근 복원 가능 시간을 선택하여 가능한 최근 시간으로 복원하거나, 사용자 지정을 선택하여 시간을 선택합니다.

   사용자 지정(Custom)을 선택한 경우 인스턴스를 복원하려는 날짜와 시간을 입력합니다.

   > **Note**
   >
   > 시간은 현지 시간대로 표시됩니다. 즉, 협정 세계시(UTC)에서 오프셋으로 표시됩니다. 예를 들어 UTC-5는 동부 표준시/하절기 중부 표준시입니다.

6. DB 인스턴스 식별자에 대상 복원된 DB 인스턴스의 이름을 입력합니다. 이름은 고유해야 합니다.
7. 필요에 따라 DB 인스턴스 클래스, 스토리지, 스토리지 자동 크기 조정 사용 여부 등의 다른 옵션을 선택합니다.

   각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하세요.

8. 특정 시점으로 복구를 선택합니다.

AWS CLI

지정된 시간으로 DB 인스턴스를 복원하려면 AWS CLI 명령 restore-db-instance-to-point-in-time을 사용하여 새로운 DB 인스턴스를 만듭니다. 또한 이 예에서는 할당된 스토리지 크기를 설정하고 스토리지 AutoScaling을 활성화합니다.

이 작업에는 리소스 태깅이 지원됩니다. `--tags` 옵션을 사용하면 소스 DB 인스턴스 태그가 무시되고 제공된 태그가 사용됩니다. 이 옵션을 사용하지 않으면 소스 인스턴스의 최신 태그가 사용됩니다.

다른 설정을 지정할 수 있습니다. 각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하세요.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds restore-db-instance-to-point-in-time \
    --source-db-instance-identifier mysourcedbinstance \
    --target-db-instance-identifier mytargetdbinstance \
    --restore-time 2017-10-14T23:45:00.000Z \
```

```
    --allocated-storage 100 \
    --max-allocated-storage 1000
```

Windows의 경우:

```
aws rds restore-db-instance-to-point-in-time ^
    --source-db-instance-identifier mysourcedbinstance ^
    --target-db-instance-identifier mytargetdbinstance ^
    --restore-time 2017-10-14T23:45:00.000Z ^
    --allocated-storage 100 ^
    --max-allocated-storage 1000
```

RDS API

DB 인스턴스를 특정 시간으로 복원하려면, Amazon RDS API `RestoreDBInstanceToPointInTime` 작업을 다음 파라미터와 함께 호출합니다.

- `SourceDBInstanceIdentifier`
- `TargetDBInstanceIdentifier`
- `RestoreTime`

다중 AZ DB 클러스터를 특정 시점으로 복원

다중 AZ DB 클러스터를 특정 시점으로 복원하여 새 다중 AZ DB 클러스터를 생성할 수 있습니다.

RDS는 다중 AZ DB 클러스터의 트랜잭션 로그를 Amazon S3에 지속적으로 업로드합니다. 백업 보존 기간 중 어느 특정 시점으로든 복원할 수 있습니다. 다중 AZ DB 클러스터에서 복원 가능한 가장 빠른 시간을 보려면 AWS CLI describe-db-clusters 명령을 사용하세요. DB 클러스터의 `EarliestRestorableTime` 필드에 반환된 값을 확인합니다. 다중 AZ DB 클러스터의 최근 복원 가능 시간을 확인하려면 DB 클러스터의 `LatestRestorableTime` 필드에 반환되는 값을 살펴봅니다.

다중 AZ DB 클러스터를 특정 시점으로 복원할 때 다중 AZ DB 클러스터의 기본 VPC 보안 그룹을 선택하거나 다중 AZ DB 클러스터에 사용자 지정 VPC 보안 그룹을 적용할 수 있습니다.

복원된 다중 AZ DB 클러스터는 기본 DB 클러스터 파라미터 그룹과 자동으로 연결됩니다. 그러나 사용자 지정 DB 클러스터 파라미터 그룹을 복원 중에 지정하여 적용할 수 있습니다.

소스 DB 클러스터에 리소스 태그가 있는 경우 RDS는 복원된 DB 클러스터에 최신 태그를 추가합니다.

> **Note**
>
> 소스 DB 클러스터와 동일하거나 유사한 다중 AZ DB 클러스터 크기로 복원하는 것이 좋습니다. 또한 프로비저닝된 IOPS 스토리지를 사용하는 경우 동일하거나 유사한 IOPS 값으로 복원하는 것이 좋습니다. 예를 들어, 호환되지 않는 IOPS 값을 가진 DB 클러스터 크기를 선택하는 경우 오류가 발생할 수 있습니다.
> 소스 다중 AZ DB 클러스터에서 범용 SSD(gp3) 스토리지를 사용하고 할당된 스토리지가 400GiB 미만인 경우에는 복원된 DB 클러스터의 프로비저닝된 IOPS를 수정할 수 없습니다.

RDS 추가 지원 버전을 사용하여 다중 AZ DB 클러스터를 복원하는 방법에 대한 자세한 내용은 Amazon RDS 추가 지원이 포함된 DB 인스턴스 또는 다중 AZ DB 클러스터 복원 섹션을 참조하세요.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 다중 AZ DB 클러스터를 특정 시점으로 복원할 수 있습니다.

콘솔

다중 AZ DB 클러스터를 지정된 시점으로 복원하는 방법

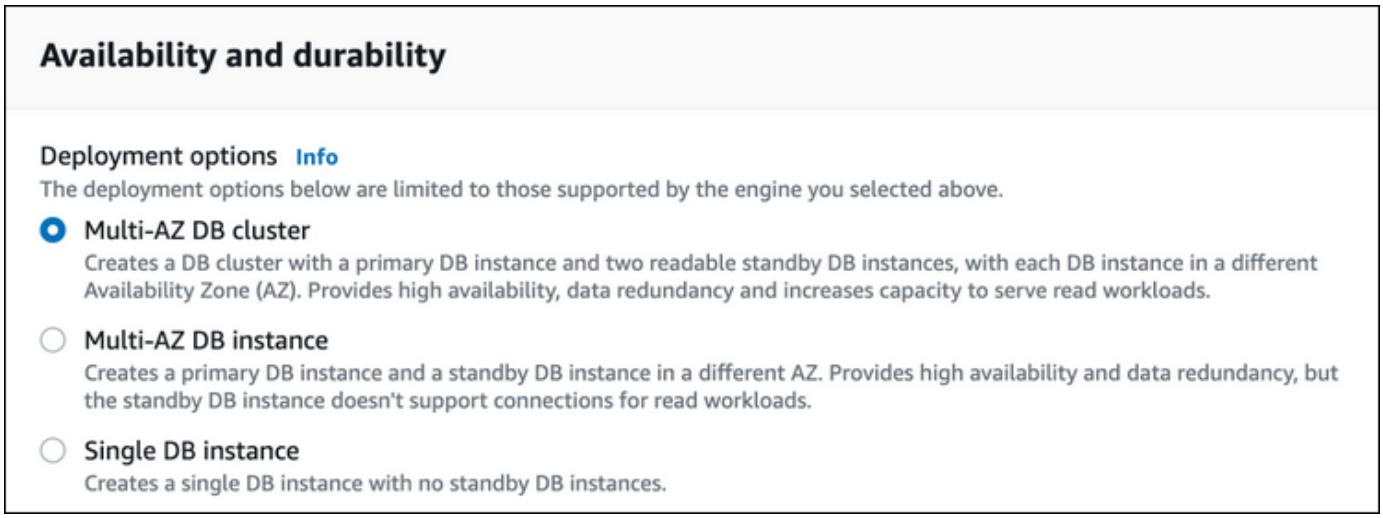
1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.

2. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
3. 복원하려는 다중 AZ DB 클러스터를 선택합니다.
4. 작업에서 특정 시점으로 복구를 선택합니다.

   특정 시점으로 복구 창이 나타납니다.
5. 최근 복원 가능 시간을 선택하여 가능한 최근 시간으로 복원하거나, 사용자 지정을 선택하여 시간을 선택합니다.

   사용자 지정(Custom)을 선택한 경우 다중 AZ DB 클러스터를 복원할 날짜 및 시간을 입력합니다.

   > **Note**
   >
   > 시간은 현지 시간대로 표시됩니다. 즉, 협정 세계시(UTC)에서 오프셋으로 표시됩니다. 예를 들어 UTC-5는 동부 표준시/하절기 중부 표준시입니다.
6. DB 클러스터 식별자(DB cluster identifier)로 복원된 다중 AZ DB 클러스터의 이름을 입력합니다.
7. 가용성 및 지속성(Availability and durability)에서 다중 AZ DB 클러스터(Multi-AZ DB cluster)를 선택합니다.

8. DB 인스턴스 클래스(DB instance class)에서 DB 인스턴스 클래스를 선택합니다.

   현재 다중 AZ DB 클러스터는 db.m6gd 및 db.r6gd DB 인스턴스 클래스만 지원합니다. DB 인스턴스 클래스에 대한 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요.
9. 나머지 섹션에서 DB 클러스터 설정을 지정합니다. 각 설정에 대한 자세한 내용은 다중 AZ DB 클러스터를 생성하기 위한 설정 단원을 참조하세요.
10. 특정 시점으로 복구를 선택합니다.

AWS CLI

다중 AZ DB 클러스터를 지정된 시간으로 복원하려면 AWS CLI 명령 restore-db-cluster-to-point-in-time을 사용하여 다중 AZ DB 클러스터를 새로 생성합니다.

현재 다중 AZ DB 클러스터는 db.m6gd 및 db.r6gd DB 인스턴스 클래스만 지원합니다. DB 인스턴스 클래스에 대한 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-identifier mysourcemultiazdbcluster \
    --db-cluster-identifier mytargetmultiazdbcluster \
    --restore-to-time 2021-08-14T23:45:00.000Z \
    --db-cluster-instance-class db.r6gd.xlarge
```

Windows의 경우:

```
aws rds restore-db-cluster-to-point-in-time ^
    --source-db-cluster-identifier mysourcemultiazdbcluster ^
    --db-cluster-identifier mytargetmultiazdbcluster ^
    --restore-to-time 2021-08-14T23:45:00.000Z ^
    --db-cluster-instance-class db.r6gd.xlarge
```

RDS API

DB 클러스터를 지정된 시간으로 복원하려면 다음 파라미터를 사용하여 Amazon RDS API RestoreDBClusterToPointInTime 작업을 호출합니다.

- `SourceDBClusterIdentifier`
- `DBClusterIdentifier`
- `RestoreToTime`

스냅샷에서 다중 AZ DB 클러스터로 복원

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 다중 AZ DB 클러스터로 스냅샷을 복원할 수 있습니다. 다음과 같은 각 스냅샷 유형을 다중 AZ DB 클러스터로 복원할 수 있습니다.

- 단일 AZ 배포의 스냅샷
- 단일 DB 인스턴스가 포함된 다중 AZ DB 클러스터 배포의 스냅샷
- 다중 AZ DB 클러스터의 스냅샷

다중 AZ 배포에 대한 정보는 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 섹션을 참조하세요.

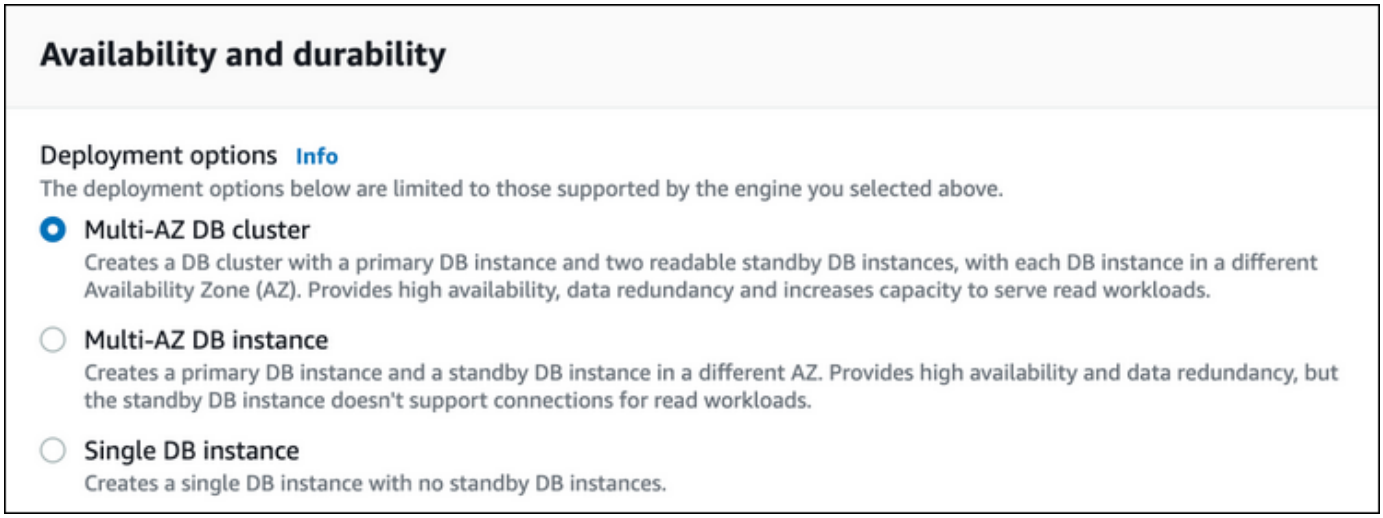
> **Tip**
>
> 스냅샷을 복원하여 단일 AZ 배포 또는 다중 AZ DB 클러스터 배포를 다중 AZ DB 클러스터 배포로 마이그레이션할 수 있습니다.

RDS 추가 지원 버전을 사용하여 다중 AZ DB 클러스터를 복원하는 방법에 대한 자세한 내용은 Amazon RDS 추가 지원이 포함된 DB 인스턴스 또는 다중 AZ DB 클러스터 복원 섹션을 참조하세요.

콘솔

다중 AZ DB 클러스터로 스냅샷을 복원하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.
3. 복원 원본으로 사용할 스냅샷을 선택합니다.
4. 작업에서 스냅샷 복원을 선택합니다.
5. 스냅샷 복원 페이지의 가용성 및 지속성(Availability and durability)에서 다중 AZ DB 클러스터(Multi-AZ DB cluster)를 선택합니다.



6. DB 클러스터 식별자(DB cluster identifier)로 복원된 다중 AZ DB 클러스터의 이름을 입력합니다.
7. 나머지 섹션에서 DB 클러스터 설정을 지정합니다. 각 설정에 대한 자세한 내용은 다중 AZ DB 클러스터를 생성하기 위한 설정 단원을 참조하세요.
8. DB 인스턴스 복원을 선택합니다.

AWS CLI

스냅샷을 다중 AZ DB 클러스터로 복원하려면 AWS CLI 명령 restore-db-cluster-from-snapshot을 사용합니다.

다음 예제에서는 이전에 생성된 `mysnapshot`이라는 스냅샷에서 복원합니다. `mynewmultiazdbcluster`라는 새로운 다중 AZ DB 클러스터로 복원합니다. 다중 AZ DB 클러스터의 DB 인스턴스에서 사용하는 DB 인스턴스 클래스도 지정합니다. `mysql` 또는 `postgres`를 DB 엔진으로 지정합니다.

`--snapshot-identifier` 옵션에서 DB 클러스터 스냅샷을 지정할 때는 이름 또는 Amazon 리소스 이름(ARN)을 사용할 수 있습니다. 그러나 DB 스냅샷을 지정할 때는 ARN만 사용해야 합니다.

`--db-cluster-instance-class` 옵션의 경우 새 다중 AZ DB 클러스터에 대한 DB 인스턴스 클래스를 지정합니다. 다중 AZ DB 클러스터는 db.m6gd 및 db.r6gd와 같은 특정 DB 인스턴스 클래스만 지원합니다. DB 인스턴스 클래스에 대한 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요.

또한 다른 옵션도 지정할 수 있습니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mynewmultiazdbcluster \
    --snapshot-identifier mysnapshot \
    --engine mysql|postgres \
    --db-cluster-instance-class db.r6gd.xlarge
```

Windows의 경우:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier mynewmultiazdbcluster ^
    --snapshot-identifier mysnapshot ^
    --engine mysql|postgres ^
    --db-cluster-instance-class db.r6gd.xlarge
```

DB 클러스터를 복원한 후에는 스냅샷을 생성하는 데 사용한 DB 클러스터 또는 DB 인스턴스와 연결된 보안 그룹에 다중 AZ DB 클러스터를 추가해야 합니다. 이 작업을 완료하면 이전 DB 클러스터 또는 DB 인스턴스와 동일한 기능이 제공됩니다.

RDS API

스냅샷을 다중 AZ DB 클러스터로 복원하려면 다음 파라미터를 사용하여 RDS API 작업 RestoreDBClusterFromSnapshot을 호출하면 됩니다.

- `DBClusterIdentifier`
- `SnapshotIdentifier`
- `Engine`

필요한 경우 다른 파라미터를 지정할 수도 있습니다.

DB 클러스터를 복원한 후에는 스냅샷을 생성하는 데 사용한 DB 클러스터 또는 DB 인스턴스와 연결된 보안 그룹에 다중 AZ DB 클러스터를 추가해야 합니다. 이 작업을 완료하면 이전 DB 클러스터 또는 DB 인스턴스와 동일한 기능이 제공됩니다.

다중 AZ DB 클러스터 스냅샷에서 DB 인스턴스로 복원

다중 AZ DB 클러스터 스냅샷은 개별 데이터베이스가 아닌 전체 DB 클러스터를 백업하여 DB 클러스터의 스토리지 볼륨 스냅샷을 생성합니다. 다중 AZ DB 클러스터 스냅샷을 단일 AZ 배포 또는 다중 AZ DB 인스턴스 배포에 복원할 수 있습니다. 다중 AZ 배포에 대한 정보는 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 섹션을 참조하세요.

> **Note**
>
> 다중 AZ DB 클러스터 스냅샷을 새 다중 AZ DB 클러스터로 복원할 수도 있습니다. 지침은 스냅샷에서 다중 AZ DB 클러스터로 복원 단원을 참조하십시오.

RDS 추가 지원 버전을 사용하여 다중 AZ DB 클러스터를 복원하는 방법에 대한 자세한 내용은 Amazon RDS 추가 지원이 포함된 DB 인스턴스 또는 다중 AZ DB 클러스터 복원 섹션을 참조하세요.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 다중 AZ DB 클러스터 스냅샷을 단일 AZ 배포 또는 다중 AZ DB 인스턴스 배포에 복원할 수 있습니다.

콘솔

다중 AZ DB 클러스터 스냅샷을 단일 AZ 배포 또는 다중 AZ DB 인스턴스 배포에 복원

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.
3. 복원 원본으로 사용할 다중 AZ DB 클러스터 스냅샷을 선택합니다.
4. 작업에서 스냅샷 복원을 선택합니다.
5. Restore snapshot(스냅샷 복원) 페이지의 Availability and durability(가용성 및 내구성)에서 다음 중 하나를 선택합니다.
   - Single DB instance(단일 DB 인스턴스) – 대기 DB 인스턴스가 없는 하나의 DB 인스턴스로 스냅샷을 복원합니다.
   - Multi-AZ DB instance(다중 AZ DB 인스턴스) – 스냅샷을 하나의 기본 DB 인스턴스와 하나의 대기 DB 인스턴스가 포함된 다중 AZ DB 인스턴스 배포로 복원합니다.
6. DB instance identifier(DB 인스턴스 식별자)에 복원된 DB 인스턴스의 이름을 입력합니다.
7. 나머지 섹션에서 DB 인스턴스 설정을 지정합니다. 각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하세요.

8. DB 인스턴스 복원을 선택합니다.

AWS CLI

DB 인스턴스 배포로 다중 AZ DB 클러스터를 복원하려면 AWS CLI 명령 restore-db-instance-from-db-snapshot을 사용합니다.

다음 예제에서는 이전에 생성된 myclustersnapshot이라는 다중 AZ DB 클러스터 스냅샷에서 복원합니다. mynewdbinstance라는 기본 DB 인스턴스가 포함된 다중 AZ DB 인스턴스 배포로 복원합니다. --db-cluster-snapshot-identifier 옵션의 경우 다중 AZ DB 클러스터 스냅샷의 이름을 지정합니다.

--db-instance-class 옵션의 경우 새 DB 인스턴스 배포에 DB 인스턴스 클래스를 지정합니다. DB 인스턴스 클래스에 대한 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요.

또한 다른 옵션도 지정할 수 있습니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds restore-db-instance-from-db-snapshot \
    --db-instance-identifier mynewdbinstance \
    --db-cluster-snapshot-identifier myclustersnapshot \
    --engine mysql \
    --multi-az \
    --db-instance-class db.r6g.xlarge
```

Windows의 경우:

```
aws rds restore-db-instance-from-db-snapshot ^
    --db-instance-identifier mynewdbinstance ^
    --db-cluster-snapshot-identifier myclustersnapshot ^
    --engine mysql ^
    --multi-az ^
    --db-instance-class db.r6g.xlarge
```

DB 인스턴스를 복원한 후에는 스냅샷을 생성하는 데 사용한 다중 AZ DB 클러스터와 연결된 보안 그룹에 DB 인스턴스를 추가해야 합니다(해당하는 경우). 이 작업을 완료하면 이전 다중 AZ DB 클러스터와 동일한 기능이 제공됩니다.

RDS API

다중 AZ DB 클러스터 스냅샷을 DB 인스턴스 배포로 복원하려면 다음 파라미터를 사용하여 RDS API 작업 RestoreDBInstanceFromDBSnapshot을 호출합니다.

- `DBInstanceIdentifier`
- `DBClusterSnapshotIdentifier`
- `Engine`

필요한 경우 다른 파라미터를 지정할 수도 있습니다.

DB 인스턴스를 복원한 후에는 스냅샷을 생성하는 데 사용한 다중 AZ DB 클러스터와 연결된 보안 그룹에 DB 인스턴스를 추가해야 합니다(해당하는 경우). 이 작업을 완료하면 이전 다중 AZ DB 클러스터와 동일한 기능이 제공됩니다.

자습서: DB 스냅샷에서 Amazon RDS DB 인스턴스 복원

Amazon RDS를 사용하여 작업할 때, 때때로 사용하지만 늘 필요하지는 않은 DB 인스턴스가 있는 경우가 많습니다. 예를 들어, Amazon EC2 인스턴스를 사용하여 고객 설문 웹 사이트를 호스트하는 분기별 고객 설문 작업이 있다고 합시다. 설문조사 결과를 저장하는 데 사용되는 DB 인스턴스도 있습니다. 이러한 시나리오에서 비용을 절약하는 한 가지 방법은 보고서 설문조사가 완료된 후 DB 인스턴스의 DB 스냅샷을 생성하는 것입니다. 그런 다음 DB 인스턴스를 삭제하고 설문조사를 다시 수행해야 할 때 복원합니다.

DB 인스턴스를 복원할 때 복원할 DB 스냅샷의 이름을 제공합니다. 그런 다음 복원 작업에서 생성되는 새 DB 인스턴스의 이름을 입력하면 됩니다.

스냅샷에서 DB 인스턴스를 복원하는 방법에 대한 자세한 내용은 DB 인스턴스 복원 섹션을 참조하세요.

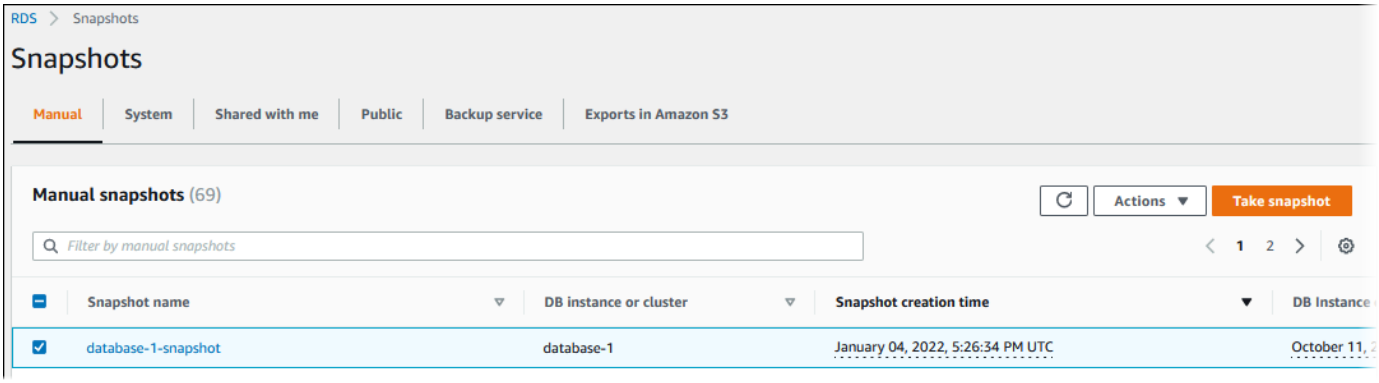
Amazon RDS의 AWS KMS 키 관리에 대한 내용은 AWS KMS key 관리 섹션을 참조하세요.

DB 스냅샷에서 DB 인스턴스 복원

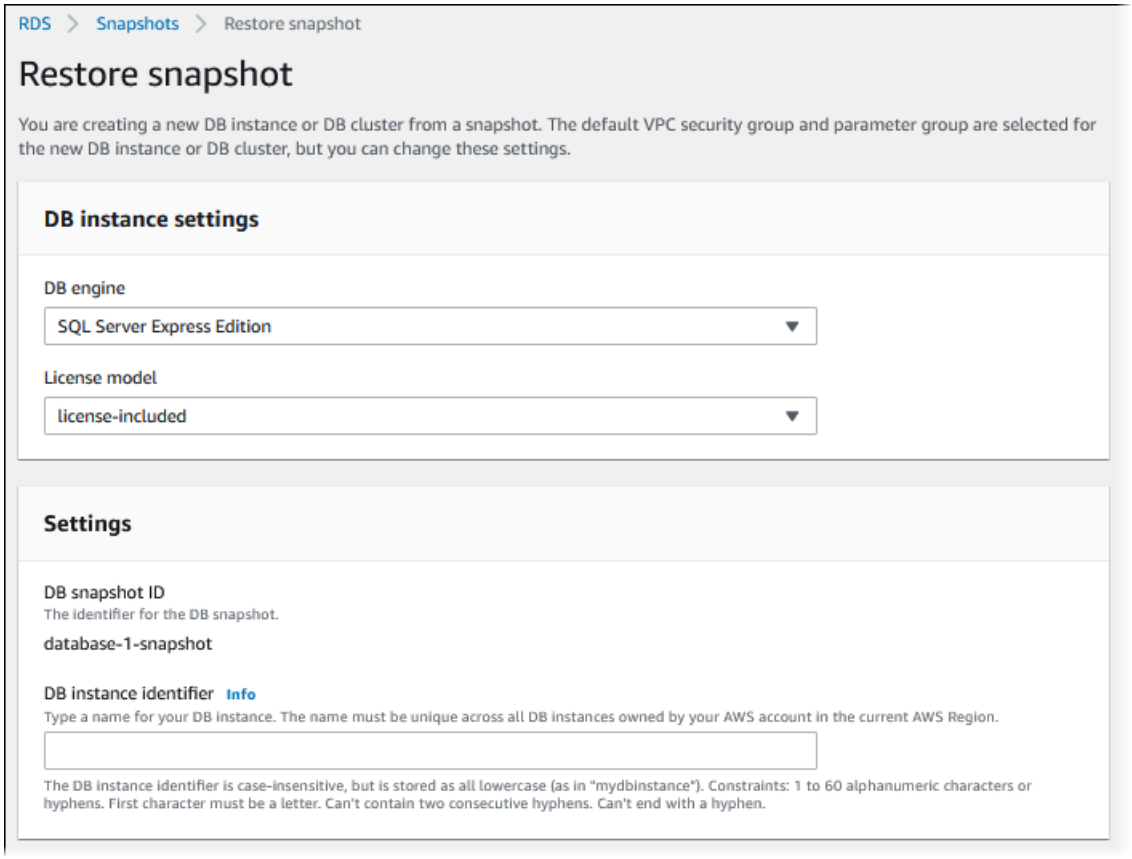
AWS Management Console에서 스냅샷에서 복원하려면 다음 절차를 사용하세요.

DB 스냅샷에서 DB 인스턴스를 복원하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.
3. 복원 원본으로 사용할 DB 스냅샷을 선택합니다.
4. 작업에서 스냅샷 복원을 선택합니다.



스냅샷 복원 페이지가 표시됩니다.



5. DB 인스턴스 설정(DB instance settings)에서 DB 엔진(DB engine) 및 라이선스 모델(License model)(Oracle 또는 Microsoft SQL Server용)에 대한 기본 설정을 사용합니다.
6. 설정(Settings) 아래의 DB 인스턴스 식별자(DB instance identifier)에서 복원된 DB 인스턴스에 사용할 고유 이름(예: **`mynewdbinstance`**)을 입력합니다.

   DB 스냅샷을 만든 후 삭제한 DB 인스턴스로부터 복원하는 경우 해당 DB 인스턴스의 이름을 사용할 수 있습니다.
7. 가용성 및 내구성 아래에서 다른 가용 영역에 대기 인스턴스를 생성할지 여부를 선택합니다.

   이 자습서에서는 대기 인스턴스를 생성하지 마세요.
8. 연결 아래에서 다음에 대해 기본 설정을 사용합니다.

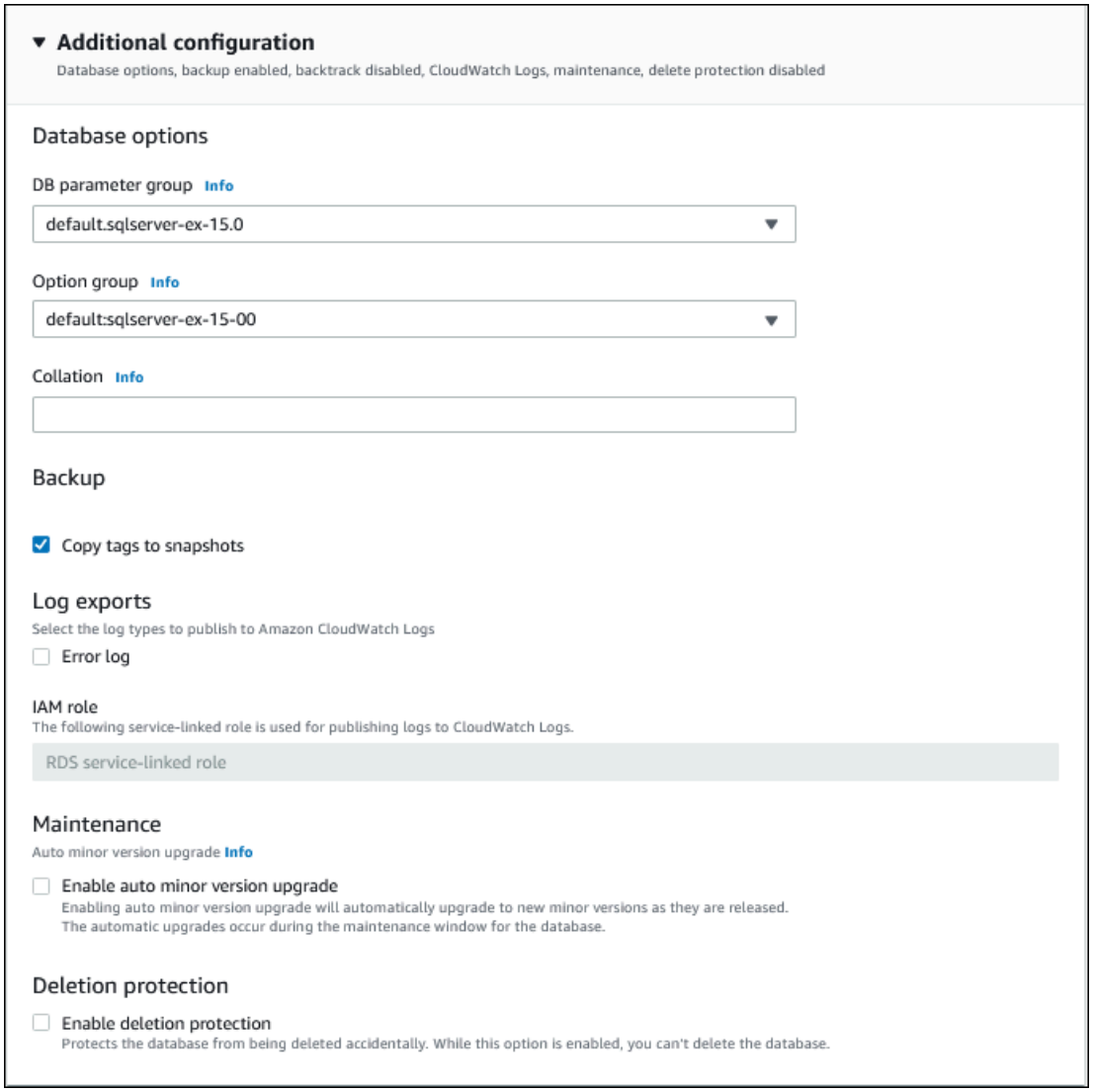
   - Virtual private cloud(VPC)
   - DB 서브넷 그룹
   - 공개 액세스(Public access)
   - VPC 보안 그룹(방화벽)
9. DB 인스턴스 클래스(DB instance class)를 선택합니다.

이 자습서에서는 버스트 가능한 클래스(t 클래스 포함)(Burstable classes (includes t classes))를 선택한 다음 db.t3.small을 선택합니다.

10. 암호화(Encryption)에 기본 설정을 사용합니다.

    스냅샷의 원본 DB 인스턴스가 암호화된 경우 복원된 DB 인스턴스도 암호화됩니다. 암호화되지 않은 상태로 만들 수는 없습니다.

11. 페이지 하단에서 추가 구성(Additional configuration)을 확장합니다.



12. 데이터베이스 옵션(Database options) 아래에서 다음을 수행합니다.

    a. DB 파라미터 그룹(DB parameter group)을 선택합니다.

       이 자습서에서는 기본 파라미터 그룹을 사용합니다.

    b. 옵션 그룹(Option group)을 선택합니다.

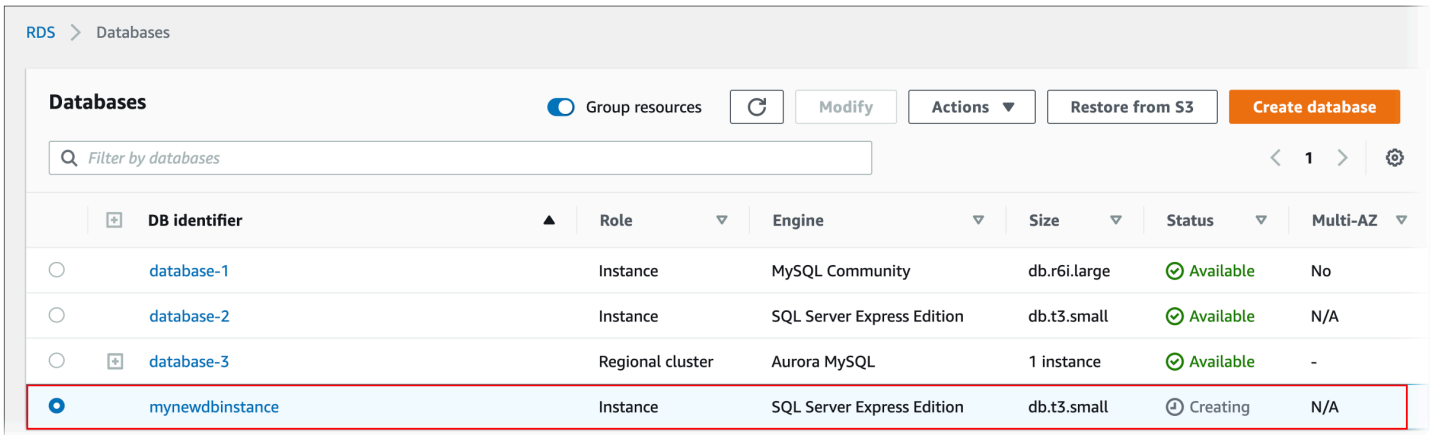
이 자습서에서는 기본 옵션 그룹을 사용합니다.

> **Important**
>
> 어떤 경우에는 지속 또는 영구 옵션을 사용하는 DB 인스턴스의 DB 스냅샷에서 복원할 수 있습니다. 그렇다면 동일한 옵션을 사용하는 옵션 그룹을 선택해야 합니다.

c. 삭제 방지(Deletion protection)에서 삭제 방지 활성화(Enable deletion protection) 확인란을 선택합니다.

13. DB 인스턴스 복원을 선택합니다.

데이터베이스(Databases) 페이지에 복원된 DB 인스턴스가 `Creating` 상태로 표시됩니다.



Amazon RDS용 DB 스냅샷 복사

Amazon RDS에서는 자동 백업 또는 수동으로 DB 스냅샷을 복사할 수 있습니다. 스냅샷을 복사한 후 사본은 수동 스냅샷입니다. 자동 백업 또는 수동 스냅샷의 복사본을 여러 개 만들 수 있지만 각 복사본에는 고유한 식별자가 있어야 합니다.

동일한 AWS 리전 내의 스냅샷을 복사할 수 있고, AWS 리전 간에 스냅샷을 복사할 수 있고, 공유 스냅샷을 복사할 수 있습니다.

DB 스냅샷 복사

각 AWS 계정에 대해 AWS 리전 간에 DB 스냅샷을 한 번에 20개까지 복사할 수 있습니다. DB 스냅샷을 다른 AWS 리전으로 복사하면 그 AWS 리전에 유지되는 수동 DB 스냅샷이 생성됩니다. 소스 AWS 리전 외부로 DB 스냅샷을 복사하면 Amazon RDS 데이터 전송 요금이 발생합니다.

데이터 전송 요금에 대한 자세한 정보는 Amazon RDS 요금을 참조하십시오.

새 AWS 리전에 DB 스냅샷 사본이 생성된 후 DB 스냅샷 사본은 해당 AWS 리전의 다른 모든 DB 스냅샷과 똑같이 동작합니다.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB 스냅샷을 복사할 수 있습니다.

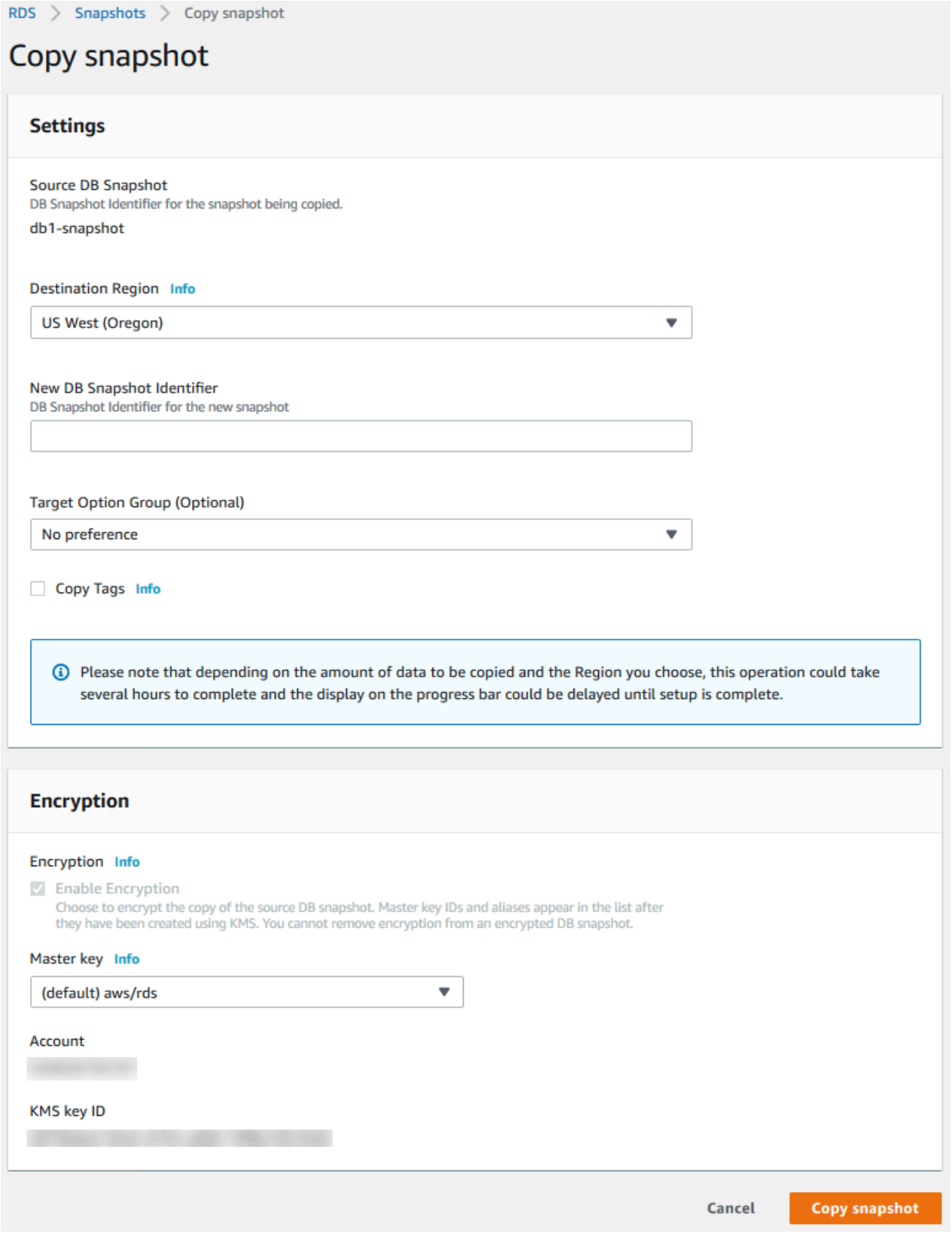
콘솔

다음 절차에서는 AWS Management Console을 사용하여 암호화된 DB 스냅샷 또는 암호화되지 않은 DB 스냅샷을 동일한 AWS 리전에서 또는 리전 간에 복사합니다.

DB 스냅샷을 복사하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.
3. 복사하려는 DB 스냅샷을 선택합니다.
4. [작업(Actions)]에서 [스냅샷 복사(Copy snapshot)]를 선택합니다.

   [스냅샷 복사(Copy snapshot)] 페이지가 나타납니다.



5. 원하는 경우 대상 옵션 그룹(선택 사항)에서 새로운 옵션 그룹을 선택합니다.

한 AWS 리전에서 다른 리전으로 스냅샷을 복사하고 DB 인스턴스가 기본값이 아닌 옵션 그룹을 사용하는 경우 이 옵션을 지정합니다.

원본 DB 인스턴스가 Oracle 또는 Microsoft SQL Server용 TDE(Transparent Data Encryption)를 사용하는 경우 리전 간 복사 시 이 옵션을 지정해야 합니다. 자세한 내용은 옵션 그룹에 대한 고려 사항 섹션을 참조하세요.

6. (선택 사항) DB 스냅샷을 다른 AWS 리전에 복사하려면 대상 리전(Destination Region)에서 새 AWS 리전을 선택합니다.

> **Note**
>
> 대상 AWS 리전에는 소스 AWS 리전과 동일한 가용 데이터베이스 엔진 버전이 있어야 합니다.

7. 새 DB 스냅샷 식별자에 DB 스냅샷 사본의 이름을 입력합니다.

   자동 백업 또는 수동 스냅샷의 복사본을 여러 개 만들 수 있지만 각 복사본에는 고유한 식별자가 있어야 합니다.

8. (선택 사항) [Copy Tags]를 선택하여 스냅샷의 태그와 값을 스냅샷 사본에 복사합니다.
9. (선택 사항) [암호화(Encryption)]에서 다음을 수행합니다.

   a. DB 스냅샷이 암호화되지 않았지만 사본을 암호화하려면 [암호화 활성화(Enable Encryption)]를 선택합니다.

   > **Note**
   >
   > DB 스냅샷이 암호화된 경우 복사본을 암호화해야 하므로 해당 확인란이 이미 선택되어 있습니다.

   b. AWS KMS key에 대해 DB 스냅샷 복사본을 암호화하는 데 사용할 KMS 키 식별자를 지정합니다.

10. 스냅샷 복사를 선택합니다.

AWS CLI

AWS CLI 명령 copy-db-snapshot을 사용하여 DB 스냅샷을 복사할 수 있습니다. 스냅샷을 새 AWS 리전으로 복사하는 경우 새 AWS 리전에서 명령을 실행합니다.

다음 옵션을 사용하여 DB 스냅샷을 복사할 수 있습니다. 시나리오에 따라 필요하지 않은 옵션도 있습니다. 다음의 설명 및 예제를 사용하여 어느 옵션을 사용할지 결정하십시오.

- `--source-db-snapshot-identifier` – 원본 DB 스냅샷의 식별자입니다.
  - 소스 스냅샷이 사본과 동일한 AWS 리전에 있는 경우 유효한 DB 스냅샷 식별자를 지정합니다. 예를 들어 `rds:mysql-instance1-snapshot-20130805`입니다.
  - 원본 스냅샷이 사본과 동일한 AWS 리전에 있고 사용자의 AWS 계정과 공유된 경우 유효한 DB 스냅샷 ARN을 지정합니다. 예를 들어 `arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20130805`입니다.
  - 소스 스냅샷이 사본과 다른 AWS 리전에 있는 경우 유효한 DB 스냅샷 ARN을 지정합니다. 예를 들어 `arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20130805`입니다.
  - 공유된 수동 DB 스냅샷에서 복사하는 경우에는 이 파라미터가 공유된 DB 스냅샷의 Amazon 리소스 이름(ARN)이어야 합니다.
  - 암호화된 스냅샷을 복사하는 경우 이 파라미터가 소스 AWS 리전의 ARN 형식이어야 하며 `PreSignedUrl` 파라미터의 `SourceDBSnapshotIdentifier`와 일치해야 합니다.
- `--target-db-snapshot-identifier` – 암호화된 DB 스냅샷의 새 사본의 식별자입니다.
- `--copy-option-group` - AWS 계정과 공유된 스냅샷에서 옵션 그룹을 복사합니다.
- `--copy-tags` – 스냅샷의 태그와 값들을 스냅샷 사본에 복사하기 위한 태그 복사 옵션을 포함합니다.
- `--option-group-name` – 스냅샷 사본에 연결할 옵션 그룹입니다.

  한 AWS 리전에서 다른 리전으로 스냅샷을 복사하고 DB 인스턴스가 기본값이 아닌 옵션 그룹을 사용하는 경우 이 옵션을 지정합니다.

  원본 DB 인스턴스가 Oracle 또는 Microsoft SQL Server용 TDE(Transparent Data Encryption)를 사용하는 경우 리전 간 복사 시 이 옵션을 지정해야 합니다. 자세한 내용은 옵션 그룹에 대한 고려 사항 단원을 참조하십시오.
- `--kms-key-id` – 암호화된 DB 스냅샷의 KMS 키 식별자입니다. KMS 키 식별자는 KMS 키의 Amazon 리소스 이름(ARN), 키 식별자 또는 키 별칭입니다.
  - AWS 계정에서 암호화된 DB 스냅샷을 복사하는 경우 이 파라미터에 값을 지정하여 새 KMS 키를 사용하여 사본을 암호화할 수 있습니다. 이 파라미터에 값을 지정하지 않을 경우 DB 스냅샷 사본이 원본 DB 스냅샷과 동일한 KMS 키를 사용하여 암호화됩니다.
  - 다른 AWS 계정에서 공유되는 암호화된 DB 스냅샷을 복사하는 경우 이 파라미터에 값을 지정해야 합니다.

- 암호화되지 않은 스냅샷을 복사할 때 이 파라미터를 지정할 경우 사본이 암호화됩니다.
- 암호화된 스냅샷을 다른 AWS 리전에 복사하는 경우 대상 AWS 리전에 대해 KMS 키를 지정해야 합니다. KMS 키는 해당 키를 만든 AWS 리전에 고유하며, 한 AWS 리전의 암호화 키를 다른 AWS 리전에서 사용할 수는 없습니다.

Example 암호화되지 않은 스냅샷을 동일한 리전으로 복사

다음 코드는 소스 스냅샷과 동일한 AWS 리전에 mydbsnapshotcopy라는 새 이름으로 스냅샷의 사본을 생성합니다. 복사가 완료되면 원래 스냅샷의 DB 옵션 그룹과 태그가 스냅샷 사본에 복사됩니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds copy-db-snapshot \
    --source-db-snapshot-identifier arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20130805 \
    --target-db-snapshot-identifier mydbsnapshotcopy \
    --copy-option-group \
    --copy-tags
```

Windows의 경우:

```
aws rds copy-db-snapshot ^
    --source-db-snapshot-identifier arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20130805 ^
    --target-db-snapshot-identifier mydbsnapshotcopy ^
    --copy-option-group ^
    --copy-tags
```

Example 암호화되지 않은 스냅샷을 다른 리전으로 복사

다음 코드는 명령이 실행되는 AWS 리전에서 mydbsnapshotcopy라는 새 이름으로 스냅샷의 사본을 생성합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds copy-db-snapshot \
    --source-db-snapshot-identifier arn:aws:rds:us-east-1:123456789012:snapshot:mysql-instance1-snapshot-20130805 \
    --target-db-snapshot-identifier mydbsnapshotcopy
```

Windows의 경우:

```
aws rds copy-db-snapshot ^
    --source-db-snapshot-identifier arn:aws:rds:us-east-1:123456789012:snapshot:mysql-instance1-snapshot-20130805 ^
    --target-db-snapshot-identifier mydbsnapshotcopy
```

Example 암호화된 스냅샷을 다른 리전으로 복사

다음 코드 예시에서는 암호화된 DB 스냅샷을 미국 서부(오레곤) 리전에서 US East (N. Virginia) 리전으로 복사합니다. 대상(us-east-1) 리전에서 명령을 실행합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds copy-db-snapshot \
    --source-db-snapshot-identifier arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20161115 \
    --target-db-snapshot-identifier mydbsnapshotcopy \
    --kms-key-id my-us-east-1-key \
    --option-group-name custom-option-group-name
```

Windows의 경우:

```
aws rds copy-db-snapshot ^
    --source-db-snapshot-identifier arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20161115 ^
    --target-db-snapshot-identifier mydbsnapshotcopy ^
    --kms-key-id my-us-east-1-key ^
    --option-group-name custom-option-group-name
```

이 `--source-region` 파라미터는 AWS GovCloud(미국 동부) 및 AWS GovCloud(미국 서부) 리전 간에 암호화된 스냅샷을 복사할 때 필요합니다. `--source-region`의 경우 소스 DB 인스턴스의 AWS 리전을 지정합니다.

`--source-region`이 지정되지 않은 경우에는 `--pre-signed-url` 값을 지정합니다. 미리 서명된 URL은 소스 AWS 리전에서 호출되는 `copy-db-snapshot` 명령에 대한 서명 버전 4의 서명된 요청이 포함된 URL입니다. `pre-signed-url` 옵션에 대한 자세한 정보는 AWS CLI 명령 참조에서 copy-db-snapshot을 참조하세요.

RDS API

Amazon RDS API 작업 CopyDBSnapshot을 사용하여 DB 스냅샷을 복사할 수 있습니다. 스냅샷을 새 AWS 리전으로 복사하는 경우 새 AWS 리전에서 작업을 수행합니다.

다음 파라미터를 사용하여 DB 스냅샷을 복사할 수 있습니다. 시나리오에 따라 필요하지 않은 파라미터도 있습니다. 다음의 설명 및 예제를 사용하여 어느 파라미터를 사용할지 결정하십시오.

- `SourceDBSnapshotIdentifier` – 원본 DB 스냅샷의 식별자입니다.
  - 소스 스냅샷이 사본과 동일한 AWS 리전에 있는 경우 유효한 DB 스냅샷 식별자를 지정합니다. 예를 들어 `rds:mysql-instance1-snapshot-20130805`입니다.
  - 원본 스냅샷이 사본과 동일한 AWS 리전에 있고 사용자의 AWS 계정과 공유된 경우 유효한 DB 스냅샷 ARN을 지정합니다. 예를 들어 `arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20130805`입니다.
  - 소스 스냅샷이 사본과 다른 AWS 리전에 있는 경우 유효한 DB 스냅샷 ARN을 지정합니다. 예를 들어 `arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20130805`입니다.
  - 공유된 수동 DB 스냅샷에서 복사하는 경우에는 이 파라미터가 공유된 DB 스냅샷의 Amazon 리소스 이름(ARN)이어야 합니다.
  - 암호화된 스냅샷을 복사하는 경우 이 파라미터가 소스 AWS 리전의 ARN 형식이어야 하며 `PreSignedUrl` 파라미터의 `SourceDBSnapshotIdentifier`와 일치해야 합니다.
- `TargetDBSnapshotIdentifier` – 암호화된 DB 스냅샷의 새 사본의 식별자입니다.
- `CopyOptionGroup` – 공유된 스냅샷에서 스냅샷 사본으로 옵션 그룹을 복사하려면 이 파라미터를 `true`로 설정합니다. 기본값은 `false`입니다.
- `CopyTags` – 스냅샷의 태그와 값을 스냅샷 사본에 복사하려면 이 파라미터를 `true`로 설정합니다. 기본값은 `false`입니다.
- `OptionGroupName` – 스냅샷 사본에 연결할 옵션 그룹입니다.

  한 AWS 리전에서 다른 리전으로 스냅샷을 복사하고 DB 인스턴스가 기본값이 아닌 옵션 그룹을 사용하는 경우 이 파라미터를 지정합니다.

  원본 DB 인스턴스가 Oracle 또는 Microsoft SQL Server용 TDE(Transparent Data Encryption)를 사용하는 경우 리전 간 복사 시 이 파라미터를 지정해야 합니다. 자세한 내용은 옵션 그룹에 대한 고려 사항 단원을 참조하십시오.
- `KmsKeyId` – 암호화된 DB 스냅샷의 KMS 키 식별자입니다. KMS 키 식별자는 KMS 키의 Amazon 리소스 이름(ARN), 키 식별자 또는 키 별칭입니다.
  - AWS 계정에서 암호화된 DB 스냅샷을 복사하는 경우 이 파라미터에 값을 지정하여 새 KMS 키를 사용하여 사본을 암호화할 수 있습니다. 이 파라미터에 값을 지정하지 않을 경우 DB 스냅샷 사본이 원본 DB 스냅샷과 동일한 KMS 키를 사용하여 암호화됩니다.

  - 다른 AWS 계정에서 공유되는 암호화된 DB 스냅샷을 복사하는 경우 이 파라미터에 값을 지정해야 합니다.
  - 암호화되지 않은 스냅샷을 복사할 때 이 파라미터를 지정할 경우 사본이 암호화됩니다.
  - 암호화된 스냅샷을 다른 AWS 리전에 복사하는 경우 대상 AWS 리전에 대해 KMS 키를 지정해야 합니다. KMS 키는 해당 키를 만든 AWS 리전에 고유하며, 한 AWS 리전의 암호화 키를 다른 AWS 리전에서 사용할 수는 없습니다.
- `PreSignedUrl` - 복사할 소스 DB 스냅샷이 위치하는 소스 AWS 리전에서 `CopyDBSnapshot` API 작업에 대한 서명 버전 4의 서명된 요청이 포함된 URL입니다.

  Amazon RDS API를 사용하여 다른 AWS 리전으로부터 암호화된 DB 스냅샷을 복사하는 경우 이 파라미터를 지정합니다. AWS CLI를 사용하여 다른 AWS 리전으로부터 암호화된 DB 스냅샷을 복사하는 경우 이 파라미터 대신 소스 리전 옵션을 지정할 수 있습니다.

  미리 서명된 URL은 복사할 암호화된 DB 스냅샷이 위치하는 소스 AWS 리전에서 실행할 수 있는 `CopyDBSnapshot` API 작업에 대한 유효한 요청이어야 합니다. 미리 서명된 URL 요청은 다음 파라미터 값을 포함해야 합니다.

  - `DestinationRegion` - 암호화된 DB 스냅샷이 복사될 AWS 리전입니다. AWS 리전은 이 미리 서명된 URL을 포함하며 `CopyDBSnapshot` 작업이 호출되는 리전과 동일합니다.

    예를 들어 us-west-2 리전에서 us-east-1 리전으로 암호화된 DB 스냅샷을 복사한다고 가정합니다. 이 경우 us-east-1 리전에서 `CopyDBSnapshot` 작업을 호출하고, us-east-1 리전에서 `CopyDBSnapshot` 작업에 대한 호출이 포함된 미리 서명된 URL을 제공합니다. 이 예제의 경우 미리 서명된 URL에서 `DestinationRegion`이 us-east-1 리전으로 설정되어야 합니다.
  - `KmsKeyId` - 대상 AWS 리전에서 DB 스냅샷의 복사본을 암호화하는 데 사용할 키에 대한 KMS 키 식별자입니다. 이 식별자는 대상 AWS 리전에서 호출되는 `CopyDBSnapshot` 작업과 미리 서명된 URL에 포함된 작업 모두에 동일합니다.
  - `SourceDBSnapshotIdentifier` – 복사할 암호화된 DB 스냅샷의 DB 스냅샷 식별자입니다. 이 식별자는 소스 AWS 리전용 Amazon 리소스 이름(ARN) 형식이어야 합니다. 예를 들어 us-west-2 리전에서 암호화된 DB 스냅샷을 복사하는 경우 `SourceDBSnapshotIdentifier`는 다음 예와 같습니다. `arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20161115`.

  서명 버전 4로 서명한 요청에 대한 자세한 내용은 다음을 참조하십시오.

  - Amazon Simple Storage Service API 참조의 요청 인증: 쿼리 파라미터 사용(AWS 서명 버전 4)
  - AWS 일반 참조의 서명 버전 4 서명 프로세스

Example 암호화되지 않은 스냅샷을 동일한 리전으로 복사

다음 코드는 소스 스냅샷과 동일한 AWS 리전에 mydbsnapshotcopy라는 새 이름으로 스냅샷의 사본을 생성합니다. 복사가 완료되면 원래 스냅샷의 모든 태그가 스냅샷 사본에 복사됩니다.

```
https://rds.us-west-1.amazonaws.com/
 ?Action=CopyDBSnapshot
 &CopyTags=true
 &SignatureMethod=HmacSHA256
 &SignatureVersion=4
 &SourceDBSnapshotIdentifier=mysql-instance1-snapshot-20130805
 &TargetDBSnapshotIdentifier=mydbsnapshotcopy
 &Version=2013-09-09
 &X-Amz-Algorithm=AWS4-HMAC-SHA256
 &X-Amz-Credential=AKIADQKE4SARGYLE/20140429/us-west-1/rds/aws4_request
 &X-Amz-Date=20140429T175351Z
 &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
 &X-Amz-Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Example 암호화되지 않은 스냅샷을 다른 리전으로 복사

다음 코드는 미국 서부(캘리포니아 북부 지역) 리전에 mydbsnapshotcopy라는 새 이름으로 스냅샷 복사본을 생성합니다.

```
https://rds.us-west-1.amazonaws.com/
 ?Action=CopyDBSnapshot
 &SignatureMethod=HmacSHA256
 &SignatureVersion=4
 &SourceDBSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-east-1%3A123456789012%3Asnapshot
%3Amysql-instance1-snapshot-20130805
 &TargetDBSnapshotIdentifier=mydbsnapshotcopy
 &Version=2013-09-09
 &X-Amz-Algorithm=AWS4-HMAC-SHA256
 &X-Amz-Credential=AKIADQKE4SARGYLE/20140429/us-west-1/rds/aws4_request
 &X-Amz-Date=20140429T175351Z
 &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
 &X-Amz-Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Example 암호화된 스냅샷을 다른 리전으로 복사

다음 코드는 US East (N. Virginia) 리전에 mydbsnapshotcopy라는 새 이름으로 스냅샷 복사본을 생성합니다.

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CopyDBSnapshot
    &KmsKeyId=my-us-east-1-key
    &OptionGroupName=custom-option-group-name
    &PreSignedUrl=https%253A%252F%252Frds.us-west-2.amazonaws.com%252F
         %253FAction%253DCopyDBSnapshot
         %2526DestinationRegion%253Dus-east-1
         %2526KmsKeyId%253Dmy-us-east-1-key
         %2526SourceDBSnapshotIdentifier%253Darn%25253Aaws%25253Ards%25253Aus-
west-2%25253A123456789012%25253Asnapshot%25253Amysql-instance1-snapshot-20161115
         %2526SignatureMethod%253DHmacSHA256
         %2526SignatureVersion%253D4
         %2526Version%253D2014-10-31
         %2526X-Amz-Algorithm%253DAWS4-HMAC-SHA256
         %2526X-Amz-Credential%253DAKIADQKE4SARGYLE%252F20161117%252Fus-west-2%252Frds
%252Faws4_request
         %2526X-Amz-Date%253D20161117T215409Z
         %2526X-Amz-Expires%253D3600
         %2526X-Amz-SignedHeaders%253Dcontent-type%253Bhost%253Buser-agent%253Bx-amz-
content-sha256%253Bx-amz-date
         %2526X-Amz-Signature
%253D255a0f17b4e717d3b67fad163c3ec26573b882c03a65523522cf890a67fca613
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &SourceDBSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-west-2%3A123456789012%3Asnapshot
%3Amysql-instance1-snapshot-20161115
    &TargetDBSnapshotIdentifier=mydbsnapshotcopy
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
    &X-Amz-Date=20161117T221704Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=da4f2da66739d2e722c85fcfd225dc27bba7e2b8dbea8d8612434378e52adccf
```

제한 사항

다음은 스냅샷을 복사할 때 적용되는 몇몇 제한 사항입니다.

- 중국(베이징) 리전이나 중국(닝샤) 리전에 또는 이들 리전에서 스냅샷을 복사할 수 없습니다.
- AWS GovCloud(미국 동부) 및 AWS GovCloud(미국 서부) 간에 스냅샷을 복사할 수 있습니다. 하지만 이들 GovCloud(미국) 리전과 GovCloud(미국) 이외 리전 간에는 스냅샷을 복사할 수 없습니다.

- 대상 스냅샷이 제공되기 전에 소스 스냅샷을 삭제하면 스냅샷 복사가 실패할 수 있습니다. 원본 스냅샷을 삭제하기 전에 대상 스냅샷의 상태가 AVAILABLE인지 확인하십시오.
- 계정당 단일 대상 리전으로 최대 20개의 스냅샷 복사 요청이 진행될 수 있습니다.
- 동일한 소스 DB 인스턴스에 대해 여러 스냅샷 복사본을 요청하면 내부적으로 대기열에 들어갑니다. 나중에 요청한 복사본은 이전 스냅샷 복사본이 완료될 때까지 시작되지 않습니다. 자세한 내용은 AWS 지식 센터의 EC2 AMI 또는 EBS 스냅샷 생성 속도가 느린 이유는 무엇입니까? 단원을 참조하세요.
- 관련된 AWS 리전과 복사할 데이터 양에 따라 교차 리전 스냅샷 복사를 완료하는 데 몇 시간이 걸릴 수 있습니다. 경우에 따라 지정된 소스 리전으로부터 대량의 리전 간 스냅샷 복사 요청이 있을 수 있습니다. 이러한 경우 진행 중인 복사본 중 일부가 완료될 때까지 Amazon RDS가 해당 소스 리전의 새로운 교차 리전 복사 요청을 대기열에 넣을 수 있습니다. 대기열에 있는 복사 요청에 대한 진행 정보는 표시되지 않습니다. 복사가 시작되면 진행 정보가 표시됩니다.
- 다른 복사 작업을 시작할 때 이전 복사 작업이 대기 중인 경우 첫 번째 복사 작업이 완료될 때까지 두 번째 복사 작업이 시작되지 않습니다.
- 다중 AZ DB 클러스터의 스냅샷을 복사할 수 없습니다.
- io2 볼륨을 사용하는 DB 인스턴스의 스냅샷만 io2 Block Express 볼륨을 사용할 수 있는 AWS 리전으로 복사할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 섹션을 참조하세요.

고려 사항

DB 스냅샷을 복사할 때 고려해야 할 사항은 다음 주제를 참조하세요.

주제

- 스냅샷 보관
- 공유 스냅샷 복사 시 고려 사항
- 암호화된 스냅샷 복사 시 고려 사항
- 증분 스냅샷 복사 시 고려 사항
- 리전 간 스냅샷 복사 시 고려 사항
- 옵션 그룹에 대한 고려 사항
- 파라미터 그룹에 대한 고려 사항

스냅샷 보관

Amazon RDS는 다음과 같은 여러 상황에서 자동화된 스냅샷을 삭제합니다.

- 보관 기간 종료 시
- DB 인스턴스에서 자동 백업을 비활성화하는 경우
- DB 인스턴스를 삭제하는 경우

자동 백업을 더 오랜 기간 동안 유지하려면 자동 백업을 복사하여 수동 스냅샷을 만듭니다. 그러면 사용자가 삭제할 때까지 스냅샷이 보관됩니다. Amazon RDS 스토리지 비용이 기본 스토리지 공간을 초과할 경우 수동 스냅샷에 적용될 수 있습니다.

백업 스토리지 비용에 대한 자세한 내용은 Amazon RDS 요금을 참조하십시오.

공유 스냅샷 복사 시 고려 사항

다른 AWS 계정이 공유하는 스냅샷을 복사할 수 있습니다. 경우에 따라 다른 AWS 계정에서 공유된 암호화된 스냅샷을 복사할 수 있습니다. 이 경우 스냅샷을 암호화하는 데 사용된 AWS KMS key에 대한 액세스 권한이 있어야 합니다.

> **Note**
>
> Amazon RDS 스토리지 비용은 복사하는 공유 스냅샷에 적용됩니다. Amazon RDS는 소스 DB 인스턴스의 ARN을 복사한 스냅샷에 연결할 수 있습니다.

스냅샷이 암호화되지 않은 경우 AWS 리전 간에 공유 DB 스냅샷을 복사할 수 있습니다. 하지만 공유 DB 스냅샷이 암호화된 경우 이를 동일한 리전에서만 복사할 수 있습니다.

> **Note**
>
> 동일한 AWS 리전에서의 공유된 증분 스냅샷 복사는 암호화되지 않았거나 초기 전체 스냅샷과 동일한 KMS 키를 사용하여 암호화된 경우에 지원됩니다. 후속 스냅샷을 복사할 때 다른 KMS 키를 사용하여 암호화하는 경우 해당 공유 스냅샷은 전체 스냅샷입니다. 자세한 내용은 증분 스냅샷 복사 시 고려 사항 섹션을 참조하세요.

암호화된 스냅샷 복사 시 고려 사항

KMS 키를 사용하여 암호화된 스냅샷을 복사할 수 있습니다. 암호화된 스냅샷을 복사할 경우 스냅샷의 사본도 암호화해야 합니다. 동일한 AWS 리전 내에서 암호화된 스냅샷을 복사하는 경우 원본 스냅샷과 동일한 KMS 키를 사용하여 사본을 암호화할 수 있습니다. 또는 다른 KMS 키를 지정할 수 있습니다.

리전 간에 암호화된 스냅샷을 복사하는 경우 대상 AWS 리전에서 유효한 KMS 키를 지정해야 합니다. 리전별 KMS 키 또는 다중 리전 키일 수 있습니다. 다중 리전 KMS 키에 대한 자세한 내용은 AWS KMS에서 다중 리전 키 사용을 참조하세요.

소스 스냅샷은 복사 프로세스 전체에서 암호화를 유지합니다. 자세한 내용은 Amazon RDS 암호화된 DB 인스턴스의 제한 섹션을 참조하세요.

암호화되지 않은 스냅샷의 사본을 암호화할 수도 있습니다. 이렇게 하면 이전의 암호화되지 않은 DB 인스턴스에 신속히 암호화를 추가할 수 있습니다. 이렇게 하려면 DB 인스턴스를 암호화할 준비가 될 때 DB 인스턴스의 스냅샷을 생성합니다. 그런 다음 해당 스냅샷의 복사본을 생성하고 해당 스냅샷 복사본을 암호화하는 KMS 키를 지정합니다. 그런 다음 암호화된 스냅샷에서 암호화된 DB 인스턴스를 복원할 수 있습니다.

> **Important**
>
> Amazon RDS와 같은 관리형 서비스에서 AWS KMS 키 정책의 모든 리소스(*)에 대해 명시적 거부 문을 사용하는 경우 리소스 소유 계정을 허용하는 조건을 지정해야 합니다. 거부 규칙에 IAM 사용자에 대한 예외가 포함되어 있더라도 이 조건이 없다면 복사 작업이 실패할 수 있습니다.

Amazon RDS의 AWS KMS 키 관리에 대한 자세한 내용은 AWS KMS key 관리 섹션을 참조하세요.

증분 스냅샷 복사 시 고려 사항

증분 스냅샷에는 동일한 DB 인스턴스의 마지막 스냅샷 이후 변경된 데이터만 포함됩니다. 증분 스냅샷 복사는 전체 스냅샷 복사에 비해 속도가 더욱 빠르고 스토리지 비용이 낮습니다.

스냅샷 복사가 증분 복사인지는 가장 최근 완료된 스냅샷 복사본과 소스 스냅샷에 따라 결정됩니다. 가장 최근 스냅샷 복사본이 삭제된 경우 다음 복사본은 전체 복사이며, 증분 복사가 아닙니다. 스냅샷 복사본은 소스 스냅샷과 동일한 유형입니다. 소스 스냅샷이 증분 스냅샷인 경우 스냅샷 복사본은 증분 스냅샷이 됩니다.

AWS 계정 간에 스냅샷을 복사할 때 다음 조건이 모두 충족되어야 증분 복사가 실행됩니다.

- 가장 최근 스냅샷 복사는 소스 DB 인스턴스가 같으며 여전히 대상 계정에 있습니다.
- 대상 계정에 있는 스냅샷의 모든 복사본이 암호화되지 않거나 동일한 KMS 키를 사용하여 암호화되었습니다.
- 원본 DB 인스턴스가 다중 AZ 인스턴스인 경우 마지막 스냅샷을 생성한 이후로 다른 AZ로 장애 조치된 적이 없습니다.

다음 예는 전체 스냅샷과 증분 스냅샷의 차이점을 보여줍니다. 공유 스냅샷과 공유되지 않은 스냅샷에 모두 적용됩니다.

스냅샷	암호화 키	전체 또는 증분
S1	K1	전체
S2	K1	S1의 증분
S3	K1	S2의 증분
S4	K1	S3 증분
S1의 복사본(S1C)	K2	전체
S2의 복사본(S2C)	K3	전체
S3의 복사본(S3C)	K3	S2C의 증분
S4의 복사본(S4C)	K3	S3C의 증분
S4의 복사본2(S4C2)	K4	전체

> **Note**
>
> 이 예에서 이전 스냅샷이 여전히 존재하는 경우에만 스냅샷 S2, S3 및 S4는 증분입니다. 복사본도 마찬가지입니다. 스냅샷 복사본 S3C 및 S4C는 이전 복사본이 존재하는 경우에만 증분입니다.

AWS 리전 간에 증분 스냅샷을 복사하는 방법에 대한 자세한 내용은 전체 및 증분 복사 섹션을 참조하세요.

리전 간 스냅샷 복사 시 고려 사항

AWS 리전 간에 DB 스냅샷을 복사할 수 있습니다. 하지만 리전 간 스냅샷 복사와 관련한 특정 제약 및 고려 사항이 있습니다.

리전 간 DB 스냅샷 복사 요청

소스 리전과 통신하여 리전 간 DB 스냅샷 복사를 요청하려면 요청자(IAM 역할 또는 IAM 사용자)가 소스 DB 스냅샷과 소스 리전에 액세스할 수 있어야 합니다.

요청자의 IAM 정책에 있는 특정 조건으로 인해 요청이 실패할 수 있습니다. 다음 예제에서는 DB 스냅샷을 미국 동부(오하이오)에서 US East (N. Virginia)로 복사한다고 가정합니다. 이 예에서는 요청을 실패하게 하는 요청자의 IAM 정책에 있는 조건을 보여 줍니다.

- 요청자의 정책에 `aws:RequestedRegion`에 대한 조건이 있습니다.

```
...
"Effect": "Allow",
"Action": "rds:CopyDBSnapshot",
"Resource": "*",
"Condition": {
    "StringEquals": {
        "aws:RequestedRegion": "us-east-1"
    }
}
```

  정책이 소스 리전에 대한 액세스를 허용하지 않기 때문에 요청이 실패합니다. 요청이 성공하려면 소스 리전과 대상 리전을 모두 지정합니다.

```
...
"Effect": "Allow",
"Action": "rds:CopyDBSnapshot",
"Resource": "*",
"Condition": {
    "StringEquals": {
        "aws:RequestedRegion": [
            "us-east-1",
            "us-east-2"
        ]
    }
}
```

- 요청자의 정책이 소스 DB 스냅샷에 대한 액세스를 허용하지 않습니다.

```
...
"Effect": "Allow",
"Action": "rds:CopyDBSnapshot",
```

```
"Resource": "arn:aws:rds:us-east-1:123456789012:snapshot:target-snapshot"
...
```

요청이 성공하려면 소스 스냅샷과 대상 스냅샷을 모두 지정합니다.

```
...
"Effect": "Allow",
"Action": "rds:CopyDBSnapshot",
"Resource": [
    "arn:aws:rds:us-east-1:123456789012:snapshot:target-snapshot",
    "arn:aws:rds:us-east-2:123456789012:snapshot:source-snapshot"
]
...
```

- 요청자의 정책이 `aws:ViaAWSService`를 거부합니다.

```
...
"Effect": "Allow",
"Action": "rds:CopyDBSnapshot",
"Resource": "*",
"Condition": {
    "Bool": {"aws:ViaAWSService": "false"}
}
```

소스 리전과의 통신은 요청자를 대신하여 RDS가 수행합니다. 요청이 성공하려면 AWS 서비스로부터의 호출을 거부하지 않아야 합니다.

- 요청자의 정책에 `aws:SourceVpc` 또는 `aws:SourceVpce`에 대한 조건이 있습니다.

  RDS가 원격 리전을 호출할 때 지정된 VPC 또는 VPC 엔드포인트에서 수신된 요청이 아니기 때문에 요청이 실패할 수 있습니다.

요청을 실패하게 하는 이전 조건 중 하나를 사용해야 하는 경우 `aws:CalledVia`를 사용한 두 번째 문을 포함하여 요청이 성공하도록 할 수 있습니다. 예를 들어 다음과 같이 `aws:CalledVia`와 함께 `aws:SourceVpce`를 사용할 수 있습니다.

```
...
"Effect": "Allow",
"Action": "rds:CopyDBSnapshot",
"Resource": "*",
"Condition": {
```

```
    "Condition" : {
        "ForAnyValue:StringEquals" : {
          "aws:SourceVpce": "vpce-1a2b3c4d"
        }
     }
},
{
    "Effect": "Allow",
    "Action": [
        "rds:CopyDBSnapshot"
    ],
    "Resource": "*",
    "Condition": {
        "ForAnyValue:StringEquals": {
            "aws:CalledVia": [
                "rds.amazonaws.com"
            ]
        }
    }
}
```

자세한 내용은 IAM 사용 설명서에서 IAM의 정책 및 권한을 참조하세요.

스냅샷 복사 승인

리전 간 DB 스냅샷 복사 요청에서 success가 반환되고 나면, RDS가 백그라운드에서 복사를 시작합니다. RDS가 소스 스냅샷에 액세스할 수 있는 권한이 생성됩니다. 이 승인은 소스 DB 스냅샷을 대상 DB 스냅샷에 연결하고 RDS가 지정된 대상 스냅샷에만 복사할 수 있도록 합니다.

승인은 서비스 연결 IAM 역할에서 rds:CrossRegionCommunication 권한을 사용하여 RDS에 의해 확인됩니다. 복사가 승인되면 RDS가 소스 리전과 통신하고 복사를 완료합니다.

RDS는 이전에 CopyDBSnapshot 요청에 의해 승인되지 않은 DB 스냅샷에 액세스할 권한이 없습니다. 복사가 완료되면 승인이 취소됩니다.

RDS는 서비스 연결 역할을 사용하여 소스 리전에서 승인을 확인합니다. 복사 프로세스 중에 서비스 연결 역할을 삭제하면 복사가 실패합니다.

자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 사용을 참조하십시오.

AWS Security Token Service 보안 인증 사용

전역 AWS Security Token Service(AWS STS) 엔드포인트의 세션 토큰은 기본적으로 활성화된 AWS 리전(상용 리전)에서만 유효합니다. assumeRole에서 AWS STS API 작업을 통해 얻은 자격 증명을

사용하는 경우 소스 리전이 옵트인 리전이면 리전별 엔드포인트를 사용합니다. 그렇지 않으면 요청이 실패합니다. 이는 자격 증명이 두 리전 모두에서 유효해야 하기 때문입니다. 옵트인 리전의 경우 리전별 AWS STS 엔드포인트를 사용할 때만 두 리전 모두에서 유효합니다.

전역 엔드포인트를 사용하려면 작업의 두 리전 모두에 대해 전역 엔드포인트를 사용하도록 설정되어 있어야 합니다. `Valid in all AWS ##` 계정 설정에서 AWS STS에 대해 전역 엔드포인트를 설정합니다.

미리 서명된 URL 파라미터의 보안 인증에도 동일한 규칙이 적용됩니다.

자세한 내용은 IAM 사용 설명서의 AWS 리전에서 AWS STS 관리를 참조하세요.

지연 시간 및 여러 복제 요청

관련된 AWS 리전과 복사할 데이터 양에 따라 리전 간 스냅샷 복사를 완료하는 데 몇 시간이 걸릴 수 있습니다.

경우에 따라 지정된 소스 AWS 리전으로부터 대량의 교차 리전 스냅샷 복사 요청이 있을 수 있습니다. 이러한 경우 진행 중인 복사본 중 일부가 완료될 때까지 Amazon RDS가 해당 소스 AWS 리전의 새로운 교차 리전 복사 요청을 대기열에 넣을 수 있습니다. 대기열에 있는 복사 요청에 대한 진행 정보는 표시되지 않습니다. 복사가 시작되면 진행 정보가 표시됩니다.

전체 및 증분 복사

소스 스냅샷과 다른 AWS 리전에 스냅샷을 복사할 때 증분 스냅샷을 복사하는 경우에도 첫 번째 복사는 전체 스냅샷 복사입니다. 전체 스냅샷 복사에는 DB 인스턴스 복원에 필요한 모든 데이터 및 메타데이터가 포함됩니다. 첫 번째 스냅샷 복사 후 동일한 DB 인스턴스의 증분 스냅샷을 동일한 AWS 계정 내의 동일한 대상 리전에 복사할 수 있습니다. 증분 스냅샷에 대한 자세한 내용은 증분 스냅샷 복사 시 고려 사항 단원을 참조하십시오.

AWS 리전 간 증분 스냅샷 복사는 비암호화 및 암호화된 스냅샷 모두에 대해 지원됩니다.

AWS 리전 간에 스냅샷을 복사할 때 다음 조건이 충족되면 복사는 증분 복사입니다.

- 이전에 스냅샷이 대상 리전에 복사되었습니다.
- 가장 최근 스냅샷 복사가 여전히 대상 리전에 있습니다.
- 대상 리전에 있는 스냅샷의 모든 복사본이 암호화되지 않거나 동일한 KMS 키를 사용하여 암호화되었습니다.

옵션 그룹에 대한 고려 사항

DB 옵션 그룹은 그룹이 생성된 AWS 리전에 고유하며, 한 AWS 리전의 옵션 그룹을 다른 AWS 리전에서 사용할 수 없습니다.

Oracle 데이터베이스의 경우 AWS CLI 또는 RDS API를 사용하여 사용자의 AWS 계정와 공유된 스냅샷에서 사용자 지정 DB 옵션 그룹을 복사할 수 있습니다. 옵션 그룹은 동일한 AWS 리전 내에서만 복사할 수 있습니다. 옵션 그룹이 대상 계정에 이미 복사되었고 복사된 이후에 변경 사항이 없는 경우 옵션 그룹은 복사되지 않습니다. 원본 옵션 그룹이 이전에 복사되었지만 복사된 이후에 변경된 경우 RDS는 새 버전을 대상 계정에 복사합니다. 기본 옵션 그룹은 복사되지 않습니다.

리전 간에 스냅샷을 복사할 경우 스냅샷에 새 옵션 그룹을 지정할 수 있습니다. 스냅샷을 복사하기 전에 새 옵션 그룹을 준비하는 것이 좋습니다. 대상 AWS 리전에서 원래 DB 인스턴스와 동일한 설정으로 옵션 그룹을 생성합니다. 새 AWS 리전에 이미 옵션 그룹이 있는 경우 이 그룹을 사용할 수 있습니다.

경우에 따라 스냅샷을 복사하고 스냅샷에 대한 새 옵션 그룹을 지정하지 않을 수 있습니다. 이 경우 스냅샷을 복원할 때 DB 인스턴스에 기본 옵션 그룹이 적용됩니다. 새 DB 인스턴스에 원본과 같은 옵션을 지정하려면 다음을 수행합니다.

1. 대상 AWS 리전에서 원래 DB 인스턴스와 동일한 설정으로 옵션 그룹을 생성합니다. 새 AWS 리전에 이미 옵션 그룹이 있는 경우 이 그룹을 사용할 수 있습니다.
2. 대상 AWS 리전에 스냅샷을 복원한 후 새 DB 인스턴스를 수정하여 이전 단계의 새로운 또는 기존 옵션 그룹을 추가합니다.

파라미터 그룹에 대한 고려 사항

리전 간에 스냅샷을 복사하는 경우 복사에는 원래 DB 인스턴스에서 사용된 파라미터 그룹이 포함되지 않습니다. 스냅샷을 복원하여 새 DB 인스턴스를 만들면 해당 DB 인스턴스가 생성된 AWS 리전에 대한 기본 파라미터 그룹을 얻습니다. 새 DB 인스턴스에 원본과 같은 파라미터를 지정하려면 다음을 수행합니다.

1. 대상 AWS 리전에서 원래 DB 인스턴스와 동일한 설정으로 DB 파라미터 그룹을 생성합니다. 새 AWS 리전에 이미 옵션 그룹이 있는 경우 이 그룹을 사용할 수 있습니다.
2. 대상 AWS 리전에 스냅샷을 복원한 후 새 DB 인스턴스를 수정하여 이전 단계의 새로운 또는 기존 파라미터 그룹을 추가합니다.

Amazon RDS의 DB 스냅샷 공유

Amazon RDS를 사용하면 다음 방법으로 수동 DB 스냅샷을 공유할 수 있습니다.

- 암호화되었거나 암호화되지 않은 수동 DB 스냅샷을 공유하면 권한이 있는 AWS 계정에서 해당 스냅샷을 복사할 수 있습니다.
- 암호화되지 않은 수동 DB 스냅샷을 공유하면 권한이 있는 AWS 계정에서 스냅샷의 복사본을 만든 후 복원하는 대신에 해당 스냅샷에서 DB 인스턴스를 직접 복원할 수 있습니다. 그러나 공유되고 동시에 암호화된 DB 스냅샷에서는 DB 인스턴스를 복원할 수 없습니다. 대신 DB 스냅샷의 사본을 만들어 거기에서 DB 인스턴스를 복원할 수 있습니다.

> **Note**
>
> 자동 DB 스냅샷을 공유하려면 자동 스냅샷을 복사하여 수동 DB 스냅샷을 생성한 다음 해당 복사본을 공유합니다. 이 프로세스는 AWS Backup에서 생성된 리소스에도 적용됩니다.

스냅샷 복사에 대한 자세한 정보는 Amazon RDS용 DB 스냅샷 복사 단원을 참조하십시오. DB 스냅샷에서 DB 인스턴스를 복원하는 방법에 대한 자세한 내용은 DB 인스턴스 복원 섹션을 참조하세요.

최대 20개의 다른 AWS 계정 계정과 수동 스냅샷을 공유할 수 있습니다.

수동 스냅샷을 다른 AWS 계정과 공유할 경우 다음과 같은 제한이 적용됩니다.

- AWS Command Line Interface(AWS CLI) 또는 Amazon RDS API를 사용하여 공유 스냅샷에서 DB 인스턴스를 복원할 때는 공유 스냅샷의 Amazon 리소스 이름(ARN)을 스냅샷 식별자로 지정해야 합니다.
- 영구적 또는 지속적 옵션이 포함된 옵션 그룹을 사용하는 DB 스냅샷은 공유할 수 없습니다. 단, `Timezone` 또는 `OLS` 옵션(또는 둘 다)이 있는 Oracle DB 인스턴스는 제외됩니다.

  영구적 옵션은 옵션 그룹에서 제거할 수 없습니다. 옵션 그룹이 DB 인스턴스에 할당된 경우 영구적 옵션을 포함하는 옵션 그룹을 DB 인스턴스에서 제거할 수 없습니다.

  아래 표에 영구적 옵션과 지속적 옵션 및 관련된 DB 엔진이 나열되어 있습니다.

옵션 이름	지속적	영구적	DB 엔진
TDE	예	아니요	Microsoft SQL Server Enterprise Edition
TDE	예	예	Oracle Enterprise Edition
시간대	예	예	Oracle Enterprise Edition Oracle Standard Edition Oracle Standard Edition One Oracle Standard Edition 2

Oracle DB 인스턴스의 경우 `Timezone` 또는 `OLS` 옵션(또는 둘 다)이 있는 공유 DB 스냅샷을 복사할 수 있습니다. 이렇게 하려면 DB 스냅샷을 복사할 때 이 옵션이 포함된 대상 옵션 그룹을 지정하십시오. OLS 옵션은 Oracle 버전 12.2 이상을 실행하는 Oracle DB 인스턴스에 대해서만 영구적이고 지속적입니다. 이러한 옵션에 대한 자세한 내용은 Oracle 시간대 및 Oracle 레이블 보안 단원을 참조하세요.

- 다중 AZ DB 클러스터의 스냅샷을 공유할 수 없습니다.

퍼블릭 스냅샷 공유, 암호화된 스냅샷 공유 및 스냅샷 공유 중지에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- Amazon RDS의 퍼블릭 스냅샷 공유
- Amazon RDS의 암호화된 스냅샷 공유
- Amazon RDS에 대한 스냅샷 공유 중지

스냅샷 공유

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB 스냅샷을 공유할 수 있습니다.

콘솔

Amazon RDS 콘솔을 사용하여 최대 20개의 AWS 계정과 수동 DB 스냅샷을 공유할 수 있습니다. 콘솔을 사용하여 하나 이상의 계정에 대한 수동 스냅샷 공유를 중지할 수도 있습니다.

Amazon RDS 콘솔을 사용하여 수동 DB 스냅샷 공유

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.
3. 공유할 수동 스냅샷을 선택합니다.
4. Actions(작업)에서 Share snapshot(스냅샷 공유)을 선택합니다.
5. DB 스냅샷 공개 여부에 대해 다음 옵션 중 하나를 선택합니다.

   - 소스가 암호화되어 있지 않은 경우 퍼블릭을 선택하여 모든 AWS 계정이 수동 DB 스냅샷에서 DB 인스턴스를 복원하도록 허용하거나, 프라이빗을 선택하여 지정한 AWS 계정만 수동 DB 스냅샷에서 DB 인스턴스를 복원하도록 허용합니다.

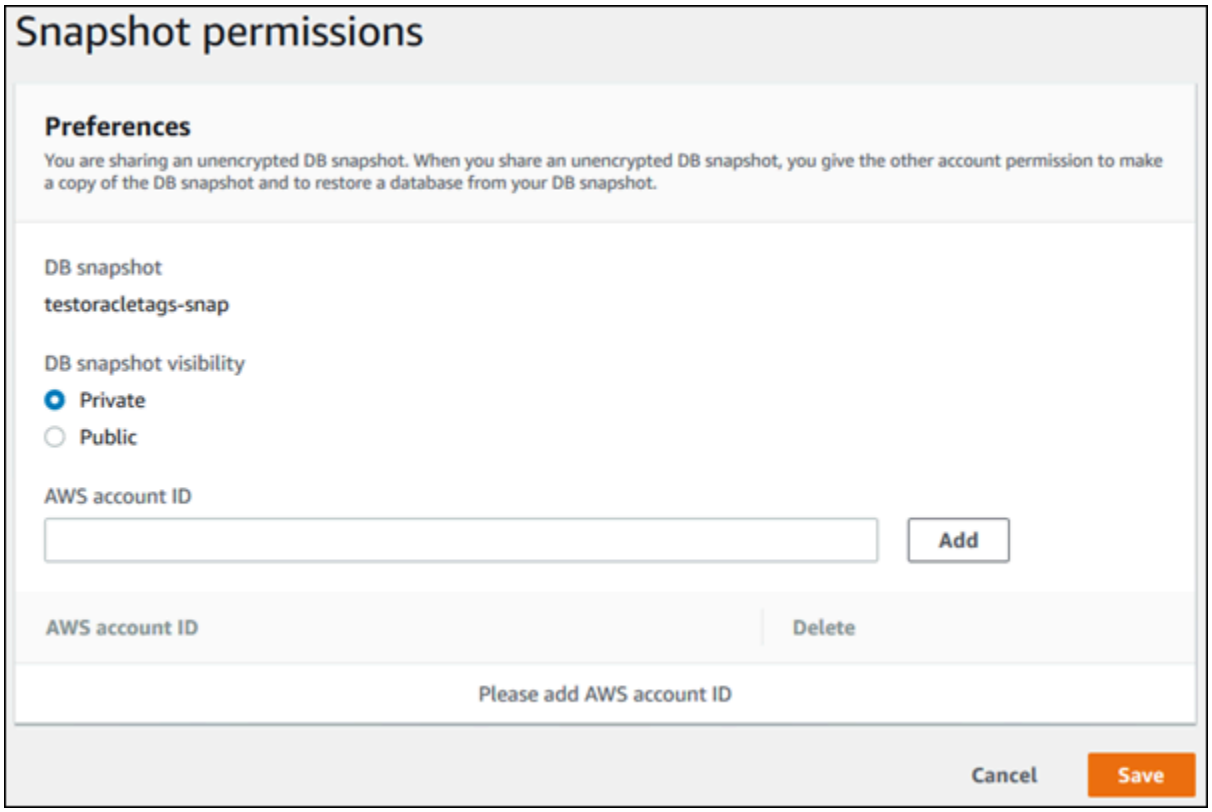
     > **Warning**
     >
     > DB 스냅샷 가시성을 퍼블릭으로 설정한 경우 모든 AWS 계정은 수동 DB 스냅샷에서 DB 인스턴스를 복원하고 사용자 데이터에 액세스할 수 있습니다. 프라이빗 정보가 포함된 수동 DB 스냅샷을 퍼블릭(Public)으로 공유하지 마세요.
     > 자세한 내용은 Amazon RDS의 퍼블릭 스냅샷 공유 단원을 참조하십시오.

   - 원본 DB 클러스터가 암호화되어 있는 경우 암호화된 스냅샷을 퍼블릭으로 공유할 수 없으므로 DB snapshot visibility(DB 스냅샷 가시성)는 Private(프라이빗)으로 설정됩니다.

     > **Note**
     >
     > 기본 AWS KMS key로 암호화된 스냅샷은 공유할 수 없습니다. 이 문제를 해결하는 방법에 대한 자세한 내용은 Amazon RDS의 암호화된 스냅샷 공유 섹션을 참조하세요.

6. AWS 계정 D에 수동 스냅샷에서 DB 인스턴스를 복원하도록 허용할 계정의 AWS 계정 계정 식별자를 입력한 다음 추가를 선택합니다. 이 과정을 반복하여 최대 20개의 AWS 계정까지 AWS 계정 식별자를 추가합니다.

허용된 계정 목록에 AWS 계정 식별자를 추가하다가 실수하는 경우, 잘못된 AWS 계정 식별자의 오른쪽에 있는 삭제를 선택하여 목록에서 해당 식별자를 삭제할 수 있습니다.



7. 수동 스냅샷을 복원할 수 있도록 허용할 모든 AWS 계정의 식별자를 추가한 후 저장을 선택하여 변경 내용을 저장합니다.

AWS CLI

DB 스냅샷을 공유하려면 `aws rds modify-db-snapshot-attribute` 명령을 사용합니다. `--values-to-add` 파라미터를 사용하여 수동 스냅샷을 복원할 권한이 있는 AWS 계정 계정의 ID 목록을 추가합니다.

Example 단일 계정과 스냅샷 공유

다음 예에서는 AWS 계정 식별자 `123456789012`가 `db7-snapshot`이라는 DB 스냅샷을 복원할 수 있습니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-snapshot-attribute \
--db-snapshot-identifier db7-snapshot \
```

```
--attribute-name restore \
--values-to-add 123456789012
```

Windows의 경우:

```
aws rds modify-db-snapshot-attribute ^
--db-snapshot-identifier db7-snapshot ^
--attribute-name restore ^
--values-to-add 123456789012
```

Example 여러 계정과 스냅샷 공유

다음 예에서는 111122223333 및 444455556666이라는 두 개의 AWS 계정 식별자가 `manual-snapshot1`이라는 DB 스냅샷을 복원할 수 있습니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-snapshot-attribute \
--db-snapshot-identifier manual-snapshot1 \
--attribute-name restore \
--values-to-add {"111122223333","444455556666"}
```

Windows의 경우:

```
aws rds modify-db-snapshot-attribute ^
--db-snapshot-identifier manual-snapshot1 ^
--attribute-name restore ^
--values-to-add "[\"111122223333\",\"444455556666\"]"
```

> **Note**
>
> Windows 명령 프롬프트를 사용하는 경우 백슬래시(\)를 접두사로 추가하여 JSON 코드에서 큰 따옴표(")를 이스케이프해야 합니다.

스냅샷을 복원할 수 있는 AWS 계정를 나열하려면 `describe-db-snapshot-attributes` AWS CLI 명령을 사용합니다.

RDS API

Amazon RDS API를 사용하여 수동 DB 스냅샷을 다른 AWS 계정과 공유할 수도 있습니다. 이렇게 하려면 `ModifyDBSnapshotAttribute` 작업을 호출합니다. `AttributeName`에 대해 `restore`를 지정하고 `ValuesToAdd` 파라미터를 사용하여 수동 스냅샷의 복원 권한이 있는 AWS 계정를 위한 ID 목록을 추가합니다.

수동 스냅샷을 퍼블릭으로 설정하고 모든 AWS 계정에서 복원할 수 있도록 하려면 `all` 값을 사용합니다. 단, 일부 AWS 계정에만 제공할 비공개 정보가 포함된 수동 스냅샷에 대해 `all` 값을 추가하지 않도록 유의합니다. 또한 암호화된 스냅샷에 대해 `all`을 지정하지 마십시오. 그러한 스냅샷을 퍼블릭으로 설정하는 것은 지원되지 않습니다.

스냅샷을 복원할 수 있도록 허용된 모든 AWS 계정를 나열하려면 `DescribeDBSnapshotAttributes` API 작업을 사용합니다.

Amazon RDS의 퍼블릭 스냅샷 공유

암호화되지 않은 수동 스냅샷을 퍼블릭으로 공유할 수 있습니다. 그러면 모든 AWS 계정에서 해당 스냅샷을 사용할 수 있습니다. 스냅샷을 퍼블릭으로 공유할 경우, 퍼블릭 스냅샷에 비공개 정보가 포함되지 않도록 유의하세요.

스냅샷이 공개적으로 공유될 경우 스냅샷을 복사하고 해당 스냅샷에서 DB 인스턴스를 생성할 수 있는 권한을 모든 AWS 계정에 부여합니다.

다른 계정이 소유한 퍼블릭 스냅샷의 백업 스토리지에 대해서는 요금이 청구되지 않습니다. 본인이 소유한 스냅샷에 대해서만 요금이 청구됩니다.

퍼블릭 스냅샷을 복사할 경우 해당 복사본을 소유하게 됩니다. 스냅샷 복사본의 백업 스토리지에 대한 요금이 청구됩니다. 퍼블릭 스냅샷에서 DB 인스턴스를 생성할 경우 해당 DB 인스턴스에 대한 요금이 청구됩니다. Amazon RDS 요금에 대한 내용은 Amazon RDS 제품 페이지를 참조하세요.

본인이 소유한 퍼블릭 스냅샷만 삭제할 수 있습니다. 공유 또는 퍼블릭 스냅샷을 삭제하려면 해당 스냅샷을 소유한 AWS 계정에 로그인해야 합니다.

다른 AWS 계정이 소유한 퍼블릭 스냅샷 보기

Amazon RDS 콘솔의 [스냅샷(Snapshots)] 페이지에 있는 [퍼블릭(Public)] 탭에서 특정 AWS 리전의 다른 계정이 소유한 퍼블릭 스냅샷을 볼 수 있습니다. 사용 중인 계정이 소유한 스냅샷은 이 탭에 표시되지 않습니다.

퍼블릭 스냅샷을 보려면

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.
3. [퍼블릭(Public)] 탭을 선택합니다.

   퍼블릭 스냅샷이 표시됩니다. [소유자(Owner)] 열에서 퍼블릭 스냅샷을 소유한 계정을 확인할 수 있습니다.

   > **Note**
   >
   > 이 열을 보려면 [퍼블릭 스냅샷(Public snapshots)] 목록의 오른쪽 상단에 있는 톱니바퀴 아이콘을 선택하여 페이지 기본 설정을 수정해야 할 수도 있습니다.

본인이 소유한 퍼블릭 스냅샷 보기

다음 AWS CLI 명령(Unix에만 해당)을 사용하여 특정 AWS 리전의 AWS 계정이 소유한 퍼블릭 스냅샷을 볼 수 있습니다.

```
aws rds describe-db-snapshots --snapshot-type public --include-public |
 grep account_number
```

퍼블릭 스냅샷이 있는 경우 반환되는 출력은 다음 예제와 유사합니다.

```
"DBSnapshotArn": "arn:aws:rds:us-east-1:123456789012:snapshot:mysnapshot1",
"DBSnapshotArn": "arn:aws:rds:us-east-1:123456789012:snapshot:mysnapshot2",
```

> **Note**
>
> `DBSnapshotIdentifier` 또는 `SourceDBSnapshotIdentifier`에 대한 중복 항목이 표시될 수 있습니다.

지원 중단된 DB 엔진 버전의 퍼블릭 스냅샷 공유

더 이상 사용되지 않는 DB 엔진 버전에서 퍼블릭 스냅샷을 복원하거나 복사하는 것은 지원되지 않습니다.

RDS for Oracle과 RDS for PostgreSQL DB 엔진에서는 DB 스냅샷 엔진 버전을 직접 업그레이드할 수 있습니다. 스냅샷을 업그레이드한 다음 공개적으로 다시 공유할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- Oracle DB 스냅샷 업그레이드
- PostgreSQL DB 스냅샷 엔진 버전 업그레이드

다른 DB 엔진의 경우 다음 단계를 수행하여 지원되지 않는 기존 퍼블릭 스냅샷을 복원하거나 복사할 수 있도록 하세요.

1. 스냅샷을 프라이빗으로 표시합니다.
2. 스냅샷을 복원합니다.
3. 복원된 DB 인스턴스를 지원되는 엔진 버전으로 업그레이드합니다.
4. 새로운 스냅샷을 생성합니다.
5. 스냅샷을 공개적으로 다시 공유합니다.

Amazon RDS의 암호화된 스냅샷 공유

Amazon RDS 리소스 암호화에 설명된 대로 AES-256 암호화 알고리즘을 사용하여 '저장된 상태'에서 암호화된 DB 스냅샷을 공유할 수 있습니다.

다음 제한은 암호화된 스냅샷 공유에 적용됩니다.

- 암호화된 스냅샷을 퍼블릭으로는 공유할 수 없습니다.
- TDE(Transparent Data Encryption)를 사용하여 암호화된 Oracle 또는 Microsoft SQL Server 스냅샷은 공유할 수 없습니다.
- 스냅샷을 공유한 AWS 계정의 기본 KMS 키를 사용하여 암호화된 스냅샷은 공유할 수 없습니다.

  Amazon RDS의 AWS KMS 키 관리에 대한 자세한 내용은 AWS KMS key 관리 섹션을 참조하세요.

기본 KMS 키 문제를 해결하려면 다음 작업을 수행하세요.

1. 고객 관리 키를 생성하고 액세스 권한 부여.
2. 소스 계정에서 스냅샷 복사 및 공유.
3. 공유된 스냅샷을 대상 계정에 복사.

고객 관리 키를 생성하고 액세스 권한 부여

먼저 암호화된 DB 스냅샷과 동일한 AWS 리전에 있는 사용자 지정 KMS 키를 생성합니다. 고객 관리 키를 생성할 때는 다른 AWS 계정에 키에 대한 액세스 권한을 부여합니다.

> **Note**
>
> 키 정책이 소스 및 대상 계정에 대한 액세스 권한을 부여할 때 다른 AWS 계정의 KMS 키를 사용할 수도 있습니다.

고객 관리 키를 생성하고 액세스 권한을 부여하는 방법

1. AWS 계정 소스에서 AWS Management Console에 로그인합니다.
2. AWS KMS 콘솔(https://console.aws.amazon.com/kms)을 엽니다.
3. AWS 리전을 변경하려면 페이지의 오른쪽 상단 모서리에 있는 리전 선택기를 사용합니다.
4. 탐색 창에서 고객 관리형 키를 선택합니다.
5. 키 생성을 선택합니다.
6. 키 구성 페이지에서 다음 단계를 따릅니다.
   a. 키 유형에서 대칭을 선택합니다.
   b. 키 사용에서 암호화 및 해독을 선택합니다.
   c. 고급 옵션을 확장합니다.
   d. 키 구성 요소 원본에서 KMS를 선택합니다.
   e. 리전 특성에서 단일 리전 키를 선택합니다.
   f. 다음을 선택합니다.
7. 레이블 추가 페이지에서 다음 단계를 따릅니다.
   a. 별칭에 KMS 키의 표시 이름을 입력합니다(예: **`share-snapshot`**).
   b. (선택 사항) KMS 키에 대한 설명을 입력합니다.
   c. (선택 사항) KMS 키에 태그를 추가합니다.
   d. 다음을 선택합니다.
8. 키 관리 권한 정의 페이지에서 다음을 선택합니다.
9. 키 사용 권한 정의 페이지에서 다음 단계를 따릅니다.

a. 기타 AWS 계정에서 다른 AWS 계정 추가를 선택합니다.

b. 액세스 권한을 부여할 AWS 계정 ID를 입력합니다.

   여러 AWS 계정에 액세스 권한을 부여할 수 있습니다.

c. 다음을 선택합니다.

10. KMS 키를 검토한 다음 완료를 선택합니다.

소스 계정에서 스냅샷 복사 및 공유

다음으로 고객 관리 키를 사용하여 소스 DB 스냅샷을 새 스냅샷에 복사합니다. 그런 다음 대상 AWS 계정에 공유합니다.

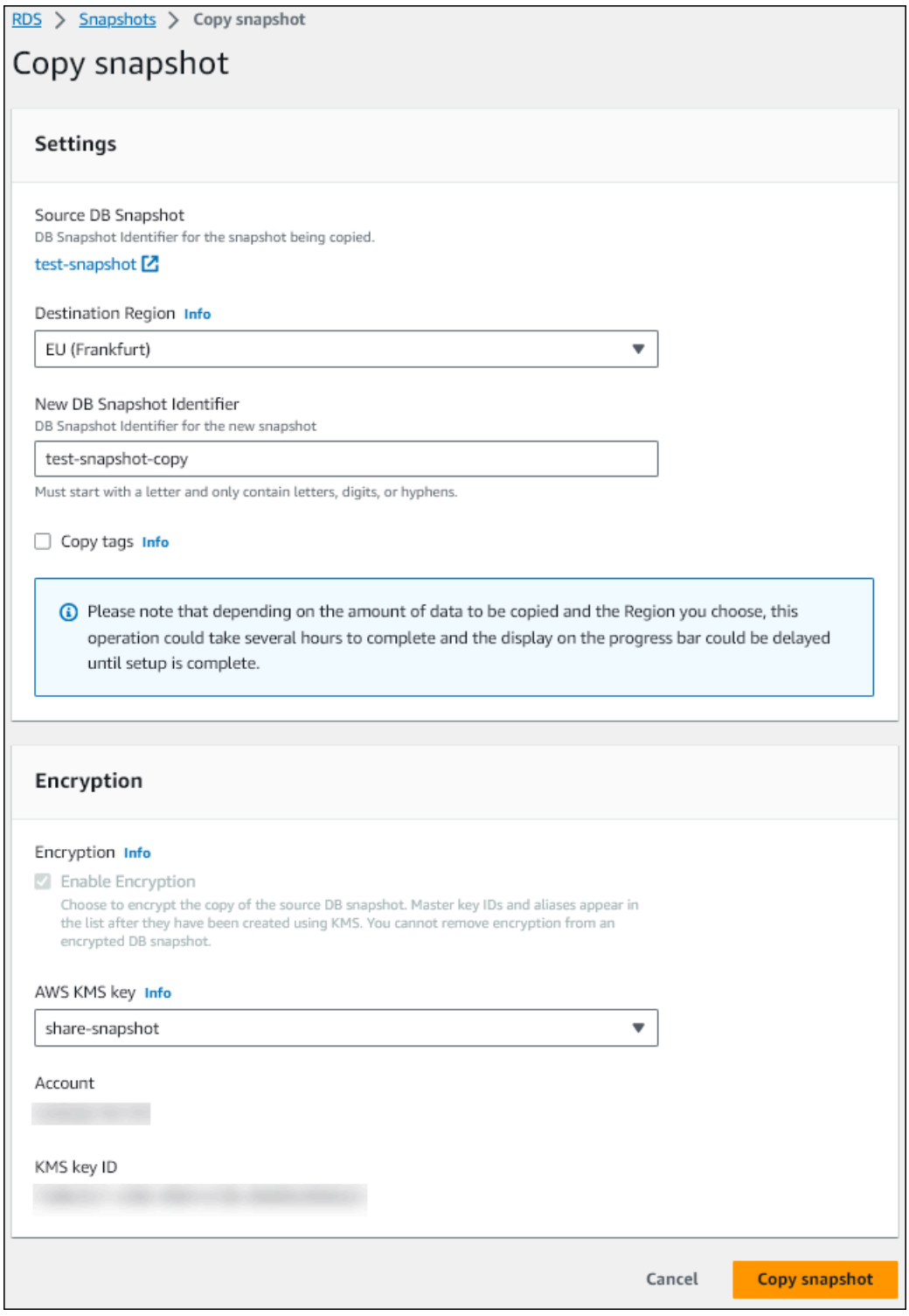
스냅샷을 복사하여 공유하는 방법

1. AWS 계정 소스에서 AWS Management Console에 로그인합니다.
2. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
3. 탐색 창에서 스냅샷을 선택합니다.
4. 복사하려는 DB 스냅샷을 선택합니다.
5. [작업(Actions)]에서 [스냅샷 복사(Copy snapshot)]를 선택합니다.
6. 스냅샷 복사 페이지에서 다음 단계를 따릅니다.

   a. 대상 리전에 이전 프로시저에서 고객 관리 키를 생성한 AWS 리전을 선택합니다.

   b. 새 DB 스냅샷 식별자에 DB 스냅샷 복사본의 이름을 입력합니다.

   c. AWS KMS key에 생성한 고객 관리 키를 선택합니다.



d. 스냅샷 복사를 선택합니다.

7. 스냅샷 사본을 사용할 수 있는 경우 해당 복사본을 선택합니다.
8. Actions(작업)에서 Share snapshot(스냅샷 공유)을 선택합니다.
9. 스냅샷 권한 페이지에서 다음 단계를 따릅니다.

a. 스냅샷 사본을 공유하는 데 사용할 AWS 계정 ID를 입력한 다음 추가를 선택합니다.
b. 저장을 선택합니다.

스냅샷이 복사됩니다.

공유된 스냅샷을 대상 계정에 복사

이제 대상 AWS 계정에서 공유 스냅샷을 복사할 수 있습니다.

공유된 스냅샷을 복사하는 방법

1. 대상 AWS 계정에서 AWS Management Console에 로그인합니다.
2. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
3. 탐색 창에서 스냅샷을 선택합니다.
4. 나와 공유됨 탭을 선택합니다.
5. 공유된 스냅샷을 선택합니다.
6. [작업(Actions)]에서 [스냅샷 복사(Copy snapshot)]를 선택합니다.
7. 이전 절차에서와 같이 스냅샷 복사 설정을 선택하고 대상 계정에 속한 AWS KMS key를 사용합니다.

스냅샷 복사를 선택합니다.

Amazon RDS에 대한 스냅샷 공유 중지

DB 스냅샷 공유를 중지하려면 대상 AWS 계정에서 권한을 제거해야 합니다.

콘솔

AWS 계정과 수동 DB 스냅샷 공유를 중지하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.
3. 공유를 중지할 수동 스냅샷을 선택합니다.
4. Actions(작업)를 선택한 다음 Share snapshot(스냅샷 공유)을 선택합니다.

5. AWS 계정의 권한을 제거하려면 권한이 있는 계정 목록에서 해당 계정의 AWS 계정 식별자에 대해 삭제를 선택합니다.
6. 저장을 선택하여 변경 사항을 저장합니다.

CLI

목록에서 AWS 계정 식별자를 제거하려면 `--values-to-remove` 파라미터를 사용합니다.

Example 스냅샷 공유 중지

다음은 AWS 계정 ID 444455556666의 스냅샷 복원을 방지하는 예제입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-snapshot-attribute \
--db-snapshot-identifier manual-snapshot1 \
--attribute-name restore \
--values-to-remove 444455556666
```

Windows의 경우:

```
aws rds modify-db-snapshot-attribute ^
--db-snapshot-identifier manual-snapshot1 ^
--attribute-name restore ^
--values-to-remove 444455556666
```

RDS API

AWS 계정의 공유 권한을 제거하려면 `AttributeName`을 `restore`로 설정한 `ModifyDBSnapshotAttribute` 작업과 `ValuesToRemove` 파라미터를 사용합니다. 수동 스냅샷을 비공개로 하려면 `all` 속성에 대한 값 목록에서 `restore` 값을 제거합니다.

Amazon S3 for Amazon RDS로 DB 스냅샷 데이터 내보내기

DB 스냅샷 데이터를 Amazon S3 버킷으로 내보낼 수 있습니다. 내보내기 프로세스는 백그라운드에서 실행되며 활성 데이터베이스의 성능에는 영향을 주지 않습니다.

DB 스냅샷을 내보낼 때 Amazon RDS는 스냅샷에서 데이터를 추출하여 Amazon S3 버킷에 저장합니다. 데이터는 압축되고 일관된 Apache Parquet 형식으로 저장됩니다.

수동 스냅샷, 자동화된 시스템 스냅샷, AWS Backup 서비스에서 생성된 수동 스냅샷을 포함하여 모든 유형의 DB 스냅샷을 내보낼 수 있습니다. 기본적으로 스냅샷의 모든 데이터가 내보내집니다. 그러나 특정한 데이터베이스, 스키마 또는 테이블 집합을 내보내도록 선택할 수 있습니다.

데이터를 내보낸 후에는 Amazon Athena 또는 Amazon Redshift Spectrum 같은 도구를 통해 직접 내보낸 데이터를 분석할 수 있습니다. Parquet 데이터를 읽는 데 Athena를 사용하는 방법에 대한 자세한 내용은 Amazon Athena 사용 설명서의 Parquet SerDe를 참조하세요. Parquet 데이터를 읽는 데 Redshift Spectrum을 사용하는 방법에 대한 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서에서 열 기반 데이터 형식의 COPY를 참조하세요.

> **Warning**
>
> S3에서 내보낸 스냅샷 데이터를 새 DB 인스턴스로 복원하거나 S3의 스냅샷 데이터를 기존 DB 인스턴스로 가져올 수 없습니다. 그러나 분석을 위해 Amazon Athena 또는 Redshift Spectrum을 사용하여 데이터를 처리할 수 있습니다. 또한 AWS Glue를 사용하여 데이터를 변환한 다음 AWS DMS 또는 사용자 지정 스크립트와 같은 도구를 사용하여 Amazon RDS로 가져올 수 있습니다.

DB 스냅샷을 Amazon S3로 내보내는 방법에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- Amazon RDS의 스냅샷 내보내기 모니터링
- Amazon RDS에 대한 스냅샷 내보내기 작업 취소
- Amazon RDS의 Amazon S3 내보내기 작업에 대한 실패 메시지
- RDS for PostgreSQL 권한 오류 문제 해결
- Amazon S3 for Amazon RDS로 내보내기 위한 파일 이름 지정 규칙
- Amazon RDS용 Amazon S3 버킷으로 내보낼 때 데이터 변환

스냅샷 데이터 내보내기 개요

다음 프로세스를 사용하여 DB 스냅샷 데이터를 Amazon S3 버킷으로 내보냅니다. 자세한 내용은 다음 섹션을 참조하십시오.

1. 내보낼 스냅샷을 식별합니다.

   기존의 자동 또는 수동 스냅샷을 사용하거나 DB 인스턴스 또는 다중 AZ DB 클러스터의 수동 스냅샷을 생성합니다.

2. Amazon S3 버킷에 대한 액세스를 설정합니다.

   버킷은 Amazon S3 객체 또는 파일에 대한 컨테이너입니다. 버킷에 액세스하기 위한 정보를 제공하려면 다음 단계를 수행하십시오.

   a. 스냅샷을 내보낼 S3 버킷을 식별합니다. S3 버킷은 스냅샷과 같은 AWS 리전에 있어야 합니다. 자세한 내용은 내보낼 Amazon S3 버킷 식별 단원을 참조하십시오.

   b. 스냅샷 내보내기 태스크에 S3 버킷에 대한 액세스 권한을 부여하는 AWS Identity and Access Management(IAM) 역할을 생성합니다. 자세한 내용은 IAM 역할을 사용하여 Amazon S3 버킷에 대한 액세스 권한 제공 단원을 참조하십시오.

3. 서버 측 암호화를 위한 대칭 암호화 AWS KMS key를 생성합니다. KMS 키는 스냅샷 내보내기 작업에서 S3에 내보내기 데이터를 기록할 때 AWS KMS 서버 측 암호화를 설정하는 데 사용됩니다.

   KMS 키 정책에는 `kms:CreateGrant` 및 `kms:DescribeKey` 권한이 모두 포함되어야 합니다. Amazon RDS에서 KMS 키를 사용하는 자세한 방법은 AWS KMS key 관리 단원을 참조하십시오.

   KMS 키 정책에 거부 문이 있는 경우 AWS 서비스 보안 주체 `export.rds.amazonaws.com`을 명시적으로 제외해야 합니다.

   AWS 계정 안에서 KMS 키를 사용할 수 있으며, 또는 교차 계정 KMS 키를 사용할 수 있습니다. 자세한 내용은 Amazon S3 내보내기 암호화를 위한 교차 계정 AWS KMS key 사용하기 단원을 참조하십시오.

4. 콘솔 또는 `start-export-task` CLI 명령을 사용하여 Amazon S3로 스냅샷을 내보냅니다. 자세한 내용은 Amazon S3 버킷으로 DB 스냅샷 내보내기 단원을 참조하십시오.

5. Amazon S3 버킷에서 내보낸 데이터에 액세스하려면 Amazon Simple Storage Service 사용 설명서의 객체 업로드, 다운로드 및 관리를 참조하세요.

Amazon S3 버킷에 대한 액세스 권한 설정

DB 스냅샷 데이터를 Amazon S3 파일로 내보내려면 먼저 스냅샷에 Amazon S3 버킷에 액세스할 수 있는 권한을 부여합니다. 그런 다음 Amazon RDS 서비스가 Amazon S3 버킷에 쓰기 작업을 할 수 있게 허용하는 IAM 역할을 생성합니다.

주제

- 내보낼 Amazon S3 버킷 식별
- IAM 역할을 사용하여 Amazon S3 버킷에 대한 액세스 권한 제공
- 교차 계정 Amazon S3 버킷 사용하기
- Amazon S3 내보내기 암호화를 위한 교차 계정 AWS KMS key 사용하기

내보낼 Amazon S3 버킷 식별

DB 스냅샷을 내보낼 Amazon S3 버킷을 식별합니다. 기존 S3 버킷을 사용하거나 새 S3 버킷을 생성합니다.

> **Note**
>
> 내보낼 S3 버킷은 스냅샷과 동일한 AWS 리전에 있어야 합니다.

Amazon S3 버킷 작업에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서에서 다음을 참조하세요.

- S3 버킷에 대한 속성을 보려면 어떻게 해야 합니까?
- Amazon S3 버킷의 기본 암호화를 활성화하려면 어떻게 해야 합니까?
- S3 버킷을 생성하려면 어떻게 해야 합니까?

IAM 역할을 사용하여 Amazon S3 버킷에 대한 액세스 권한 제공

DB 스냅샷 데이터를 Amazon S3로 내보내기 전에 스냅샷 내보내기 작업에 Amazon S3 버킷에 대한 쓰기-액세스 권한을 부여하십시오.

이 권한을 부여하려면 버킷에 대한 액세스 권한을 부여하는 IAM 정책을 생성한 다음 IAM 역할을 생성하고 정책을 역할에 연결해야 합니다. 나중에 스냅샷 내보내기 작업에 IAM 역할을 할당합니다.

다른 Amazon S3 액세스 관리 도구에 대한 내용은 Amazon S3 사용 설명서의 Amazon S3의 액세스 제어를 참조하세요.

> **Important**
>
> AWS Management Console을 사용하여 스냅샷을 내보내려는 경우 스냅샷을 내보낼 때 IAM 정책 및 역할을 자동으로 생성하도록 선택할 수 있습니다. 지침은 Amazon S3 버킷으로 DB 스냅샷 내보내기 섹션을 참조하세요.

DB 스냅샷 작업에 Amazon S3에 대한 액세스 권한을 부여하려면

1. IAM 정책을 생성합니다. 이 정책은 스냅샷 내보내기 작업에서 Amazon S3 액세스를 허용하는 버킷 및 객체 권한을 제공합니다.

   정책에서 Amazon RDS에서 S3 버킷으로의 파일 전송을 허용하려면 다음과 같은 필수 작업을 포함시킵니다.

   - `s3:PutObject*`
   - `s3:GetObject*`
   - `s3:ListBucket`
   - `s3:DeleteObject*`
   - `s3:GetBucketLocation`

   정책에 다음 리소스를 포함하여 버킷에 있는 S3 버킷과 객체를 식별합니다. 다음 리소스 목록은 Amazon S3에 액세스하기 위한 Amazon 리소스 이름(ARN) 형식을 보여 줍니다.

   - `arn:aws:s3:::amzn-s3-demo-bucket`
   - `arn:aws:s3:::amzn-s3-demo-bucket/*`

   Amazon RDS에 대한 IAM 정책을 생성하는 방법에 대한 자세한 내용은 IAM 데이터베이스 액세스를 위한 IAM 정책 생성 및 사용 섹션을 참조하세요. IAM 사용 설명서의 자습서: 첫 번째 고객 관리형 정책 생성 및 연결도 참조하십시오.

   다음 AWS CLI 명령은 이 옵션으로 `ExportPolicy`라는 IAM 정책을 만듭니다. *amzn-s3-demo-bucket*이라는 버킷에 대한 액세스 권한을 부여합니다.

> **Note**
>
> 정책을 생성한 후 정책의 ARN을 기록해 둡니다. IAM 역할에 정책을 연결할 때 이후 단계에 ARN이 필요합니다.

```
aws iam create-policy  --policy-name ExportPolicy --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ExportPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject*",
                "s3:ListBucket",
                "s3:GetObject*",
                "s3:DeleteObject*",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}'
```

2. Amazon RDS가 사용자 대신 이 IAM 역할을 수임하여 Amazon S3 버킷에 액세스할 수 있도록 IAM 역할을 생성합니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자에게 권한을 위임하기 위한 역할 생성을 참조하세요.

   다음 예제에서는 AWS CLI 명령을 사용해 `rds-s3-export-role`이라는 역할을 생성하는 방법을 보여줍니다.

```
aws iam create-role  --role-name rds-s3-export-role  --assume-role-policy-document
 '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
```

```
        "Principal": {
           "Service": "export.rds.amazonaws.com"
         },
        "Action": "sts:AssumeRole"
      }
    ]
  }'
```

3. 생성한 IAM 역할에 생성한 IAM 정책을 연결합니다.

   다음 AWS CLI 명령은 앞서 생성한 정책을 `rds-s3-export-role`이라는 역할에 연결합니다. *`your-policy-arn`*을 이전 단계에서 기록해 둔 정책 ARN으로 바꿉니다.

```
aws iam attach-role-policy  --policy-arn your-policy-arn  --role-name rds-s3-
export-role
```

교차 계정 Amazon S3 버킷 사용하기

Amazon S3 버킷 교차 AWS 계정을 사용할 수 있습니다. 교차 계정 버킷을 사용하려면 S3 내보내기에 사용 중인 IAM 역할에 대한 액세스를 허용하는 버킷 정책을 추가합니다. 자세한 내용은 예제 2: 버킷 소유자가 교차 계정 버킷 권한 부여를 참조하세요.

다음 예에 표시된 대로 버킷에 버킷 정책을 연결합니다.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/Admin"
            },
            "Action": [
                "s3:PutObject*",
                "s3:ListBucket",
                "s3:GetObject*",
                "s3:DeleteObject*",
                "s3:GetBucketLocation"
            ],
```

```
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket",
                "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
            ]
        }
    ]
}
```

Amazon S3 내보내기 암호화를 위한 교차 계정 AWS KMS key 사용하기

Amazon S3 내보내기 암호화를 위해 교차 계정 AWS KMS key를 사용할 수 있습니다. 먼저 로컬 계정에 키 정책을 추가한 다음 외부 계정에 IAM 정책을 추가합니다. 자세한 내용은 다른 계정의 사용자가 KMS 키를 사용하도록 허용을 참조하세요.

교차 계정 KMS 키 사용

1. 로컬 계정에 키 정책을 추가합니다.

   다음 예는 로컬 계정 123456789012의 권한을 외부 계정 444455556666의 `ExampleRole`과 `ExampleUser`에 부여합니다.

```
{
    "Sid": "Allow an external account to use this KMS key",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::444455556666:role/ExampleRole",
            "arn:aws:iam::444455556666:user/ExampleUser"
        ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:CreateGrant",
        "kms:DescribeKey",
        "kms:RetireGrant"
    ],
    "Resource": "*"
```

```
}
```

2. 외부 계정에서 IAM 정책 추가

   다음 IAM 정책 예시는 보안 주체가 계정 123456789012의 KMS 키를 암호화 작업에 사용하도록 허용합니다. 이 권한을 계정 444455556666의 `ExampleRole` 및 `ExampleUser`에 부여하려면 계정에서 정책을 연결합니다.

```
{
    "Sid": "Allow use of KMS key in account 123456789012",
    "Effect": "Allow",
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:CreateGrant",
        "kms:DescribeKey",
        "kms:RetireGrant"
    ],
    "Resource": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

Amazon S3 버킷으로 DB 스냅샷 내보내기

AWS 계정당 최대 5개의 DB 스냅샷 내보내기 작업을 동시에 수행할 수 있습니다.

> **Note**
>
> RDS 스냅샷 내보내기는 데이터베이스 유형 및 크기에 따라 다소 시간이 걸릴 수 있습니다. 내보내기 작업은 먼저 전체 데이터베이스를 복원하고 크기를 조정하여 Amazon S3로 데이터를 추출합니다. 이 단계 동안의 작업 진행 상황은 Starting으로 표시됩니다. 작업이 S3로 데이터 내보내기로 전환되면 진행 상황이 진행 중(In progress)으로 표시됩니다.
> 내보내기를 완료하는 데 걸리는 시간은 데이터베이스에 저장된 데이터에 따라 다릅니다. 예를 들어 숫자로 된 기본 키 또는 인덱스 열이 잘 분산되어 있는 테이블은 가장 빠르게 내보냅니다. 분할에 적합한 열을 포함하지 않는 테이블과 문자열 기반 열에 인덱스가 하나만 있는 테이블은 더 오래 걸립니다. 내보내기가 느린 단일 스레드 프로세스를 사용하기 때문에 이렇게 더 긴 내보내기 시간이 발생합니다.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB 스냅샷을 Amazon S3로 내보낼 수 있습니다. DB 스냅샷을 교차 계정 Amazon S3 버킷으로 내보내려면 AWS CLI 또는 RDS API를 사용합니다.

Lambda 함수를 사용하여 스냅샷을 내보내는 경우 Lambda 함수 정책에 `kms:DescribeKey` 작업을 추가합니다. 자세한 내용은 AWS Lambda 권한을 참조하세요.

콘솔

Amazon S3로 내보내기 옵션은 Amazon S3로 내보낼 수 있는 스냅샷에 대해서만 나타납니다. 다음과 같은 이유로 스냅샷을 내보내기에 사용하지 못할 수 있습니다.

- DB 엔진이 S3 내보내기를 지원하지 않습니다.
- DB 엔진 버전이 S3 내보내기를 지원하지 않습니다.
- 스냅샷이 생성된 AWS 리전에서 S3 내보내기가 지원되지 않습니다.

DB 스냅샷을 내보내려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.
3. 이 탭에서 내보낼 스냅샷 유형을 선택합니다.
4. 스냅샷 목록에서 내보낼 스냅샷을 선택합니다.
5. 작업에서 Export to Amazon S3(Amazon S3로 내보내기)를 선택합니다.

   Export to Amazon S3(Amazon S3로 내보내기) 창이 나타납니다.
6. Export identifier(내보내기 식별자)에 내보내기 작업을 식별할 이름을 입력합니다. 이 값은 S3 버킷에 생성된 파일의 이름에도 사용됩니다.
7. 내보낼 데이터를 선택합니다.

   - 스냅샷의 모든 데이터를 내보내려면 모두를 선택합니다.
   - 스냅샷의 특정 부분을 내보내려면 Partial(부분)을 선택합니다. 스냅샷에서 내보낼 부분을 식별하려면 [식별자(Identifiers)]에 공백으로 구분된 하나 이상의 데이터베이스, 스키마 또는 테이블을 입력합니다.

     다음 형식을 사용합니다.

```
database[.schema][.table] database2[.schema2][.table2] ... databasen[.schemaN]
[.tablen]
```

예:

```
mydatabase mydatabase2.myschema1 mydatabase2.myschema2.mytable1
 mydatabase2.myschema2.mytable2
```

8. S3 버킷에서 내보낼 버킷을 선택합니다.

   내보낸 데이터를 S3 버킷의 폴더 경로에 할당하려면 S3 prefix(S3 접두사)에 선택적 경로를 입력합니다.

9. IAM 역할에서 선택한 S3 버킷에 대한 쓰기 액세스 권한을 부여하는 역할을 선택하거나 새 역할을 생성합니다.

   - IAM 역할을 사용하여 Amazon S3 버킷에 대한 액세스 권한 제공의 단계에 따라 역할을 생성한 경우에는 해당 역할을 선택합니다.
   - 선택한 S3 버킷에 대한 쓰기 액세스 권한을 부여하는 역할을 생성하지 않은 경우 Create a new role(새 역할 생성)을 선택하여 역할을 자동으로 생성합니다. 그런 다음 IAM 역할 이름에 해당 역할의 이름을 입력합니다.

10. AWS KMS key에 내보낸 데이터를 암호화하는 데 사용할 키의 ARN을 입력합니다.
11. Amazon S3로 내보내기를 선택합니다.

AWS CLI

AWS CLI를 사용하여 DB 스냅샷을 Amazon S3로 내보내려면 start-export-task 명령을 다음과 같은 필수 옵션과 함께 사용합니다.

- `--export-task-identifier`
- `--source-arn`
- `--s3-bucket-name`
- `--iam-role-arn`
- `--kms-key-id`

다음 예제에서 스냅샷 내보내기 작업의 이름은 *my_snapshot_export*이고, 이 작업은 스냅샷을 *amzn-s3-demo-bucket*이라는 S3 버킷으로 내보냅니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds start-export-task \
    --export-task-identifier my-snapshot-export \
    --source-arn arn:aws:rds:AWS_Region:123456789012:snapshot:snapshot-name \
    --s3-bucket-name amzn-s3-demo-bucket \
    --iam-role-arn iam-role \
    --kms-key-id my-key
```

Windows의 경우:

```
aws rds start-export-task ^
    --export-task-identifier my-snapshot-export ^
    --source-arn arn:aws:rds:AWS_Region:123456789012:snapshot:snapshot-name ^
    --s3-bucket-name amzn-s3-demo-bucket ^
    --iam-role-arn iam-role ^
    --kms-key-id my-key
```

샘플 출력은 다음과 같습니다.

```
{
    "Status": "STARTING",
    "IamRoleArn": "iam-role",
    "ExportTime": "2019-08-12T01:23:53.109Z",
    "S3Bucket": "my-export-bucket",
    "PercentProgress": 0,
    "KmsKeyId": "my-key",
    "ExportTaskIdentifier": "my-snapshot-export",
    "TotalExtractedDataInGB": 0,
    "TaskStartTime": "2019-11-13T19:46:00.173Z",
    "SourceArn": "arn:aws:rds:AWS_Region:123456789012:snapshot:snapshot-name"
}
```

스냅샷 내보내기를 위해 S3 버킷에 폴더 경로를 제공하려면 start-export-task 명령에 `--s3-prefix` 옵션을 포함시킵니다.

RDS API

Amazon RDS API를 사용하여 DB 스냅샷을 Amazon S3로 내보내려면 다음 필수 파라미터와 함께 StartExportTask 작업을 사용합니다.

- `ExportTaskIdentifier`
- `SourceArn`
- `S3BucketName`
- `IamRoleArn`
- `KmsKeyId`

리전 및 버전 사용 가능 여부

기능 가용성 및 해당 지원은 각 데이터베이스 엔진의 특정 버전 및 AWS 리전에 따라 다릅니다. 스냅샷을 S3로 내보낼 때 사용할 수 있는 버전 및 리전 가용성에 대한 자세한 내용은 Amazon RDS에서 S3로 스냅샷 내보내기가 지원되는 리전 및 DB 엔진을 참조하십시오.

제한 사항

DB 스냅샷 데이터를 Amazon S3로 내보내는 데는 다음과 같은 제한 사항이 적용됩니다.

- 동일한 DB 스냅샷에 대해 여러 내보내기 작업을 동시에 실행할 수 없습니다. 이는 전체 및 부분 내보내기에 모두 적용됩니다.
- 마그네틱 스토리지를 사용하는 데이터베이스에서 스냅샷을 내보내는 것은 지원되지 않습니다.
- S3로 내보내는 경우 콜론(:)이 포함된 S3 접두사를 지원하지 않습니다.
- 내보내는 동안 S3 파일 경로의 다음 문자는 밑줄(_)로 변환됩니다.

```
\ ` " (space)
```

- 데이터베이스, 스키마 또는 테이블의 이름에 다음 문자가 아닌 문자가 있으면 부분 내보내기가 지원되지 않습니다. 그러나 전체 DB 스냅샷을 내보낼 수는 있습니다.
  - 라틴 문자(A–Z)
  - 숫자(0–9)
  - 달러 기호($)
  - 밑줄(_)

- 공백( )과 특정 문자는 데이터베이스 테이블 열 이름에서 지원되지 않습니다. 열 이름에 다음 문자가 포함되어 있는 테이블은 내보내기를 수행하는 동안 건너뛰기가 됩니다.

  ```
  , ; { } ( ) \n \t = (space)
  ```

- 이름에 슬래시(/)가 포함되어 있는 테이블은 내보내기를 수행하는 동안 생략됩니다.
- RDS for PostgreSQL 임시 테이블과 로깅되지 않는 테이블은 내보내기 중에 건너뛰게 됩니다.
- 데이터에 500MB에 근접하거나 이보다 큰 객체(예: BLOB 또는 CLOB)가 포함되어 있으면 내보내기가 실패합니다.
- 테이블에 2GB에 가깝거나 그보다 큰 행이 있으면 내보내기 중 테이블을 건너뜁니다.
- 부분 내보내기의 경우 `ExportOnly` 목록의 최대 크기는 200KB입니다.
- 각 내보내기 작업에 고유한 이름을 사용하는 것이 좋습니다. 고유한 작업 이름을 사용하지 않으면 다음 오류 메시지가 표시될 수 있습니다.

  ExportTaskAlreadyExistsFault: StartExportTask 작업을 호출하는 동안 오류 (ExportTaskAlreadyExists)가 발생했습니다. ID가 *xxxxx*인 내보내기 작업이 이미 존재합니다.
- 데이터를 S3로 내보내는 동안 스냅샷을 삭제할 수 있지만, 내보내기 태스크가 완료될 때까지 해당 스냅샷에 대한 스토리지 비용은 계속 청구됩니다.
- S3에서 내보낸 스냅샷 데이터를 새 DB 인스턴스로 복원하거나 S3의 스냅샷 데이터를 기존 DB 인스턴스로 가져올 수 없습니다.
- AWS 계정당 최대 5개의 DB 스냅샷 내보내기 작업을 동시에 수행할 수 있습니다.
- DB 스냅샷을 교차 계정 Amazon S3 버킷으로 내보내려면 AWS CLI 또는 RDS API를 사용해야 합니다.
- Amazon RDS가 내보내기 작업을 완료한 후 동일한 DB 스냅샷에서 다른 내보내기 작업을 시작하려면 잠시 기다려야 할 수 있습니다.
- 뷰 또는 구체화된 뷰는 내보낼 수 없습니다.
- RDS Export to S3는 GuardDuty Malware Protection for S3에 대한 태그 기반 액세스 제어를 지원하지 않습니다.

Amazon RDS의 스냅샷 내보내기 모니터링

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB 스냅샷 내보내기를 모니터링할 수 있습니다.

콘솔

DB 스냅샷 내보내기를 모니터링하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.
3. 스냅샷 내보내기 목록을 보려면 Amazon S3에서 내보내기 탭을 선택합니다.
4. 특정 스냅샷 내보내기에 대한 정보를 보려면 해당 내보내기 작업을 선택합니다.

AWS CLI

AWS CLI을 사용하여 DB 스냅샷 내보내기를 모니터링하려면 describe-export-tasks 명령을 사용합니다.

다음 예제에서는 모든 스냅샷 내보내기에 대한 현재 정보를 표시하는 방법을 보여 줍니다.

Example

```
aws rds describe-export-tasks

{
    "ExportTasks": [
        {
            "Status": "CANCELED",
            "TaskEndTime": "2019-11-01T17:36:46.961Z",
            "S3Prefix": "something",
            "ExportTime": "2019-10-24T20:23:48.364Z",
            "S3Bucket": "amzn-s3-demo-bucket",
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/K7MDENG/
bPxRfiCYEXAMPLEKEY",
            "ExportTaskIdentifier": "anewtest",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "TaskStartTime": "2019-10-25T19:10:58.885Z",
            "SourceArn": "arn:aws:rds:AWS_Region:123456789012:snapshot:parameter-
groups-test"
        },
{
            "Status": "COMPLETE",
```

```
            "TaskEndTime": "2019-10-31T21:37:28.312Z",
            "WarningMessage": "{\"skippedTables\":[],\"skippedObjectives\":[],\"general
\":[{\"reason\":\"FAILED_TO_EXTRACT_TABLES_LIST_FOR_DATABASE\"}]}",
            "S3Prefix": "",
            "ExportTime": "2019-10-31T06:44:53.452Z",
            "S3Bucket": "amzn-s3-demo-bucket1",
            "PercentProgress": 100,
            "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "thursday-events-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 263,
            "TaskStartTime": "2019-10-31T20:58:06.998Z",
            "SourceArn":
 "arn:aws:rds:AWS_Region:123456789012:snapshot:rds:example-1-2019-10-31-06-44"
        },
        {
            "Status": "FAILED",
            "TaskEndTime": "2019-10-31T02:12:36.409Z",
            "FailureCause": "The S3 bucket edgcuc-export isn't located in the current
 AWS Region. Please, review your S3 bucket name and retry the export.",
            "S3Prefix": "",
            "ExportTime": "2019-10-30T06:45:04.526Z",
            "S3Bucket": "amzn-s3-demo-bucket2",
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "wednesday-afternoon-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "TaskStartTime": "2019-10-30T22:43:40.034Z",
            "SourceArn":
 "arn:aws:rds:AWS_Region:123456789012:snapshot:rds:example-1-2019-10-30-06-45"
        }
    ]
}
```

특정 스냅샷 내보내기에 대한 정보를 표시하려면 `--export-task-identifier` 명령과 함께 `describe-export-tasks` 옵션을 포함시킵니다. 출력을 필터링하려면 `--Filters` 옵션을 포함시킵니다. 자세한 옵션은 describe-export-tasks 명령을 참조하십시오.

RDS API

Amazon RDS API를 사용하여 DB 스냅샷 내보내기에 대한 정보를 표시하려면 DescribeExportTasks 작업을 사용합니다.

내보내기 워크플로우의 완료를 추적하거나 다른 워크플로우를 시작하려면 Amazon Simple Notification Service 주제를 구독하십시오. Amazon SNS에 대한 자세한 내용은 Amazon RDS 이벤트 알림 작업 단원을 참조하십시오.

Amazon RDS에 대한 스냅샷 내보내기 작업 취소

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB 스냅샷 내보내기 작업을 취소할 수 있습니다.

> **Note**
>
> 스냅샷 내보내기 작업을 취소해도 Amazon S3로 내보낸 데이터는 제거되지 않습니다. 콘솔을 사용하여 데이터를 삭제하는 방법에 대한 자세한 내용은 S3 버킷에서 객체를 삭제하려면 어떻게 해야 합니까?를 참조하십시오. CLI를 사용하여 데이터를 삭제하려면 delete-object 명령을 사용합니다.

콘솔

스냅샷 내보내기 작업을 취소하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.
3. Exports in Amazon S3(Amazon S3에서 내보내기) 탭을 선택합니다.
4. 취소할 스냅샷 내보내기 작업을 선택합니다.
5. 취소를 선택합니다.
6. 확인 페이지에서 Cancel export task(내보내기 작업 취소)를 선택합니다.

AWS CLI

AWS CLI를 사용하여 스냅샷 내보내기 작업을 취소하려면 cancel-export-task 명령을 사용합니다. 이 명령에는 `--export-task-identifier` 옵션이 필요합니다.

Example

```
aws rds cancel-export-task --export-task-identifier my_export
{
    "Status": "CANCELING",
    "S3Prefix": "",
    "ExportTime": "2019-08-12T01:23:53.109Z",
    "S3Bucket": "amzn-s3-demo-bucket",
    "PercentProgress": 0,
    "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "ExportTaskIdentifier": "my_export",
    "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
    "TotalExtractedDataInGB": 0,
    "TaskStartTime": "2019-11-13T19:46:00.173Z",
    "SourceArn": "arn:aws:rds:AWS_Region:123456789012:snapshot:export-example-1"
}
```

RDS API

Amazon RDS API를 사용하여 스냅샷 내보내기 작업을 취소하려면 `ExportTaskIdentifier` 파라미터와 함께 CancelExportTask 작업을 사용합니다.

Amazon RDS의 Amazon S3 내보내기 작업에 대한 실패 메시지

다음 표에서는 Amazon S3 내보내기 작업이 실패할 때 반환되는 메시지에 대해 설명합니다.

오류 메시지	설명
알 수 없는 내부 오류가 발생했습니다.	알 수 없는 오류, 예외 또는 장애 때문에 요청 처리가 실패했습니다.
내보내기 작업의 메타데이터를 S3 버킷 [버킷 이름]에 쓰는 중에 알 수 없는 내부 오류가 발생했습니다.	알 수 없는 오류, 예외 또는 장애 때문에 요청 처리가 실패했습니다.
RDS 내보내기가 IAM 역할 [역할 ARN]을 수임할 수 없기 때문에 내보내기 작업의 메타데이터를 작성하지 못했습니다.	내보내기 작업은 S3 버킷에 메타데이터를 쓸 수 있는지 여부를 검증하는 IAM 역할을 가정합니다. 작업이 IAM 역할을 맡을 수 없으면 실패합니다.

오류 메시지	설명
RDS 내보내기가 KMS 키 [키 ID] 와 함께 IAM 역할[역할 ARN]을 사용하여 S3 버킷 [버킷 이름]에 내보내기 작업의 메타데이터를 쓰지 못했습니다. 오류 코드: [오류 코드]	하나 이상의 권한이 누락되어 내보내기 작업이 S3 버킷에 액세스할 수 없습니다. 이 실패 메시지는 다음 오류 코드 중 하나를 수신할 때 발생합니다. • `AWSSecurityTokenServiceException` , 오류 코드 `AccessDenied` • `AmazonS3Exception` , 오류 코드 `NoSuchBucket` , `AccessDenied` , `KMS.KMSInvalidStateException` , `403 Forbidden` 또는 `KMS.DisabledException` 이러한 오류 코드는 IAM 역할, S3 버킷 또는 KMS 키에 대한 설정이 잘못 구성되었음을 의미합니다.
IAM 역할 [역할 ARN]은 S3 버킷 [버킷 이름]에서 [S3 작업]을 호출할 권한이 없습니다. 권한을 검토하고 내보내기를 다시 시도해 주세요.	IAM 정책이 잘못 구성되었습니다. S3 버킷의 특정 S3 작업에 대한 권한이 없어 내보내기 작업이 실패하게 됩니다.
KMS 키 확인이 실패했습니다. KMS 키의 자격 증명을 확인하고 다시 시도해 주세요.	KMS 키 자격 증명 확인에 실패했습니다.
S3 자격 증명 확인이 실패했습니다. S3 버킷 및 IAM 정책에서 권한을 확인합니다.	S3 자격 증명 확인이 실패했습니다.
S3 버킷 [버킷 이름]이 유효하지 않습니다. 현재 AWS 리전에 위치하지 않거나 존재하지 않습니다. S3 버킷 이름을 검토하고 내보내기를 다시 시도해 주세요.	S3 버킷이 유효하지 않습니다.
S3 버킷 [버킷 이름]이 현재 AWS 리전에 위치하지 않습니다. S3 버킷 이름을 검토하고 내보내기를 다시 시도해 주세요.	S3 버킷이 잘못된 AWS 리전에 있습니다.

RDS for PostgreSQL 권한 오류 문제 해결

PostgreSQL 데이터베이스를 Amazon S3으로 내보낼 때 특정 테이블을 건너뛰었다는 PERMISSIONS_DO_NOT_EXIST 오류가 표시될 수 있습니다. 이 오류는 대부분 데이터베이스를 생성할 때 지정한 수퍼유저에게 이러한 테이블에 액세스할 권한이 없을 때 발생합니다.

이 오류를 해결하려면 다음 명령을 실행합니다.

```
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA schema_name TO superuser_name
```

수퍼유저 권한에 대한 자세한 내용은 마스터 사용자 계정 권한 단원을 참조하십시오.

Amazon S3 for Amazon RDS로 내보내기 위한 파일 이름 지정 규칙

특정 테이블에 대해 내보낸 데이터는 *base_prefix*/*files* 형식으로 저장됩니다. 여기서 기본 접두사는 다음과 같습니다.

```
export_identifier/database_name/schema_name.table_name/
```

예:

```
export-1234567890123-459/rdststdb/rdststdb.DataInsert_7ADB5D19965123A2/
```

파일의 명명 규칙은 두 가지가 있습니다.

- 현행 규칙:

  ```
  batch_index/part-partition_index-random_uuid.format-based_extension
  ```

  배치 인덱스는 테이블에서 읽은 데이터 배치를 나타내는 시퀀스 번호입니다. 테이블을 병렬로 내보낼 작은 청크로 파티셔닝할 수 없는 경우에는 배치 인덱스가 여러 개 생성됩니다. 테이블이 여러 테이블로 파티셔닝된 경우에도 마찬가지입니다. 기본 테이블의 각 테이블 파티션마다 하나씩, 배치 인덱스가 여러 개 생성됩니다.

  테이블을 병렬로 읽을 작은 청크로 파티셔닝할 수 있는 경우에는 배치 인덱스 1 폴더 하나만 생성됩니다.

배치 인덱스 폴더에는 테이블 데이터를 포함하는 하나 이상의 Parquet 파일이 있습니다. Parquet 파일 이름의 접두사는 `part-`*`partition_index`*입니다. 테이블이 파티셔닝된 경우 파티션 인덱스 `00000`으로 시작되는 여러 개의 파일이 생성됩니다.

파티션 인덱스 시퀀스에 간격이 있을 수 있습니다. 이러한 간격은 테이블의 범위 지정 쿼리에서 각 파티션을 가져오기 때문에 발생합니다. 해당 파티션 범위에 데이터가 없는 경우 해당 시퀀스 번호는 건너뛰게 됩니다.

예를 들어 `id`열이 테이블의 프라이머리 키이고 최소값과 최대값이 `100` 및 `1000`이라고 가정해 보겠습니다. 9개의 파티션이 있는 이 테이블을 내보내려고 하면 다음과 같은 병렬 쿼리를 사용하여 테이블을 읽습니다.

```
SELECT * FROM table WHERE id <= 100 AND id < 200
SELECT * FROM table WHERE id <= 200 AND id < 300
```

여기에서 `part-00000-`*`random_uuid`*`.gz.parquet~part-00008-`*`random_uuid`*`.gz.parquet`에 해당하는 9개의 파일이 생성됩니다. 그러나 `200~350` 사이의 ID가 있는 행이 없으면 완료된 파티션 중 하나가 비어 있게 되고 해당 파티션에 대한 파일이 생성되지 않습니다. 이전 예제에서는 `part-00001-`*`random_uuid`*`.gz.parquet`이 생성되지 않았습니다.

- 이전 규칙:

```
part-partition_index-random_uuid.format-based_extension
```

이는 현재 규칙과 동일하지만 *`batch_index`* 접두사는 없습니다. 예를 들면 다음과 같습니다.

```
part-00000-c5a881bb-58ff-4ee6-1111-b41ecff340a3-c000.gz.parquet
part-00001-d7a881cc-88cc-5ab7-2222-c41ecab340a4-c000.gz.parquet
part-00002-f5a991ab-59aa-7fa6-3333-d41eccd340a7-c000.gz.parquet
```

파일 명명 규칙은 변경될 수 있습니다. 따라서 대상 테이블을 읽을 때 테이블의 기본 접두사 내에 있는 모든 내용을 읽는 것이 좋습니다.

Amazon RDS용 Amazon S3 버킷으로 내보낼 때 데이터 변환

DB 스냅샷을 Amazon S3 버킷으로 내보낼 때 Amazon RDS는 데이터를 Parquet 형식으로 변환하고, 데이터를 Parquet 형식으로 내보내며, 데이터를 Parquet 형식으로 저장합니다. Parquet에 대한 자세한 내용은 Apache Parquet 웹 사이트를 참조하십시오.

Apache Parquet은 다음과 같은 프리미티브 유형 중 하나로 모든 데이터를 저장합니다.

- BOOLEAN
- INT32
- INT64
- INT96
- FLOAT
- DOUBLE
- BYTE_ARRAY – 가변 길이 바이트 배열(이진수라고도 함)
- FIXED_LEN_BYTE_ARRAY – 값이 일정한 크기를 가질 때 사용되는 고정 길이 바이트 배열

Parquet 데이터 유형은 형식을 읽고 쓸 때 복잡성을 줄이기 위한 것입니다. Parquet은 프리미티브 유형을 확장할 수 있도록 논리적 유형을 제공합니다. 논리적 유형은 `LogicalType` 메타 데이터 필드의 데이터를 이용으로 주석으로 구현됩니다. 논리적 유형 주석은 프리미티브 유형을 해석하는 방법을 설명합니다.

`STRING` 논리적 유형에 `BYTE_ARRAY` 유형이 주석으로 달려 있으면 바이트 배열이 UTF-8로 인코딩된 문자열로 해석되어야 한다는 뜻입니다. 내보내기 태스크가 완료되면 Amazon RDS에서 문자열 변환이 발생했는지를 알려줍니다. 내보낸 기초 데이터는 항상 소스의 데이터와 동일합니다. 그러나 UTF-8에서의 인코딩 차이로 인해 일부 문자는 Athena 같은 도구에서 읽을 때 소스와 다르게 나타날 수 있습니다.

자세한 내용은 Parquet 설명서의 Parquet 논리적 유형 정의를 참고하십시오.

주제

- Parquet로 MySQL 및 MariaDB 데이터 유형 매핑
- Parquet로 PostgreSQL 데이터 유형 매핑

Parquet로 MySQL 및 MariaDB 데이터 유형 매핑

다음 표는 데이터를 변환해 Amazon S3로 내보낼 때 MySQL 및 MariaDB 데이터 유형에서 Parquet 데이터 유형으로의 매핑을 보여줍니다.

소스 데이터 유형	Parquet 프리미티브 유형	논리적 유형 주석	변환 노트
숫자 데이터 유형			
BIGINT	INT64		
BIGINT UNSIGNED	FIXED_LEN _BYTE_ARRAY(9)	DECIMAL(20,0)	Parquet는 서명된 유형만 지원하므로 매핑에는 BIGINT_UNSIGNED 유형을 저장하기 위해 추가 바이트(8 + 1)가 필요합니다.
BIT	BYTE_ARRAY		
DECIMAL	INT32	DECIMAL (p,s)	소스 값이 2^{31} 미만이면 INT32로 저장됩니다.
	INT64	DECIMAL (p,s)	소스 값이 2^{31} 이상이지만 2^{63} 미만인 경우 INT64로 저장됩니다.
	FIXED_LEN _BYTE_ARRAY(N)	DECIMAL (p,s)	소스 값이 2^{63} 이상이면 FIXED_LEN _BYTE_ARRAY(N)로 저장됩니다.
	BYTE_ARRAY	STRING	Parquet는 38보다 큰 십진수 정밀도를 지원하지 않습니다. 십진수 값은 BYTE_ARRAY 유형의 문자열로 변

소스 데이터 유형	Parquet 프리미티브 유형	논리적 유형 주석	변환 노트
			환되고 UTF8로 인코딩됩니다.
DOUBLE	DOUBLE		
FLOAT	DOUBLE		
INT	INT32		
INT UNSIGNED	INT64		
MEDIUMINT	INT32		
MEDIUMINT UNSIGNED	INT64		
NUMERIC	INT32	DECIMAL (p,s)	소스 값이 2^{31} 미만이면 INT32로 저장됩니다.
	INT64	DECIMAL (p,s)	소스 값이 2^{31} 이상이지만 2^{63} 미만인 경우 INT64로 저장됩니다.
	FIXED_LEN_ARRAY(N)	DECIMAL (p,s)	소스 값이 2^{63} 이상이면 FIXED_LEN_BYTE_ARRAY(N)로 저장됩니다.
	BYTE_ARRAY	STRING	Parquet는 38보다 큰 숫자 정밀도를 지원하지 않습니다. 이 숫자 값은 BYTE_ARRAY 유형의 문자열로 변환되고 UTF8로 인코딩됩니다.

소스 데이터 유형	Parquet 프리미티브 유형	논리적 유형 주석	변환 노트
SMALLINT	INT32		
SMALLINT UNSIGNED	INT32		
TINYINT	INT32		
TINYINT UNSIGNED	INT32	INT(16, true)	
문자열 데이터 유형			
BINARY	BYTE_ARRAY		
BLOB	BYTE_ARRAY		
CHAR	BYTE_ARRAY		
ENUM	BYTE_ARRAY	STRING	
LINESTRING	BYTE_ARRAY		
LONGBLOB	BYTE_ARRAY		
LONGTEXT	BYTE_ARRAY	STRING	
MEDIUMBLOB	BYTE_ARRAY		
MEDIUMTEXT	BYTE_ARRAY	STRING	
MULTILINESTRING	BYTE_ARRAY		
SET	BYTE_ARRAY	STRING	
TEXT	BYTE_ARRAY	STRING	
TINYBLOB	BYTE_ARRAY		
TINYTEXT	BYTE_ARRAY	STRING	

소스 데이터 유형	Parquet 프리미티브 유형	논리적 유형 주석	변환 노트
VARBINARY	BYTE_ARRAY		
VARCHAR	BYTE_ARRAY	STRING	
날짜 및 시간 데이터 유형			
DATE	BYTE_ARRAY	STRING	날짜는 BYTE_ARRAY 유형의 문자열로 변환되고 UTF8로 인코딩됩니다.
DATETIME	INT64	TIMESTAMP_MICROS	
TIME	BYTE_ARRAY	STRING	TIME 유형은 BYTE_ARRAY의 문자열로 변환되고 UTF8로 인코딩됩니다.
TIMESTAMP	INT64	TIMESTAMP_MICROS	
YEAR	INT32		
기하학적 데이터 유형			
GEOMETRY	BYTE_ARRAY		
GEOMETRYCOLLECTION	BYTE_ARRAY		
MULTIPOINT	BYTE_ARRAY		
MULTIPOLYGON	BYTE_ARRAY		
POINT	BYTE_ARRAY		

소스 데이터 유형	Parquet 프리미티브 유형	논리적 유형 주석	변환 노트
POLYGON	BYTE_ARRAY		
JSON 데이터 유형			
JSON	BYTE_ARRAY	STRING	

Parquet로 PostgreSQL 데이터 유형 매핑

다음 표는 데이터를 변환해 Amazon S3로 내보낼 때 PostgreSQL 데이터 유형에서 Parquet 데이터 유형으로의 매핑을 보여줍니다.

PostgreSQL 데이터 형식	Parquet 프리미티브 유형	논리적 유형 주석	매핑 노트
숫자 데이터 유형			
BIGINT	INT64		
BIGSERIAL	INT64		
DECIMAL	BYTE_ARRAY	STRING	DECIMAL 유형은 BYTE_ARRAY 유형의 문자열로 변환되고 UTF8로 인코딩됩니다. 이 변환은 데이터 정밀도와 숫자(NaN)가 아닌 데이터 값으로 인해 복잡해지는 것을 피하기 위한 것입니다.
DOUBLE PRECISION	DOUBLE		
INTEGER	INT32		

PostgreSQL 데이터 형식	Parquet 프리미티브 유형	논리적 유형 주석	매핑 노트
MONEY	BYTE_ARRAY	STRING	
REAL	FLOAT		
SERIAL	INT32		
SMALLINT	INT32	INT(16, true)	
SMALLSERIAL	INT32	INT(16, true)	
문자열 및 관련 데이터 유형			
ARRAY	BYTE_ARRAY	STRING	배열은 문자열로 변환되고 BINARY(UTF8)로 인코딩됩니다. 이 변환은 데이터 정밀도와 숫자(NaN)가 아닌 데이터 값, 그리고 시간 데이터 값으로 인해 복잡해지는 것을 피하기 위한 것입니다.
BIT	BYTE_ARRAY	STRING	
BIT VARYING	BYTE_ARRAY	STRING	
BYTEA	BINARY		
CHAR	BYTE_ARRAY	STRING	
CHAR(N)	BYTE_ARRAY	STRING	
ENUM	BYTE_ARRAY	STRING	
NAME	BYTE_ARRAY	STRING	

PostgreSQL 데이터 형식	Parquet 프리미티브 유형	논리적 유형 주석	매핑 노트
TEXT	BYTE_ARRAY	STRING	
TEXT SEARCH	BYTE_ARRAY	STRING	
VARCHAR(N)	BYTE_ARRAY	STRING	
XML	BYTE_ARRAY	STRING	
날짜 및 시간 데이터 유형			
DATE	BYTE_ARRAY	STRING	
INTERVAL	BYTE_ARRAY	STRING	
TIME	BYTE_ARRAY	STRING	
TIME WITH TIME ZONE	BYTE_ARRAY	STRING	
TIMESTAMP	BYTE_ARRAY	STRING	
TIMESTAMP(시간대 사용)	BYTE_ARRAY	STRING	
기하학적 데이터 유형			
BOX	BYTE_ARRAY	STRING	
CIRCLE	BYTE_ARRAY	STRING	
LINE	BYTE_ARRAY	STRING	
LINESEGMENT	BYTE_ARRAY	STRING	
PATH	BYTE_ARRAY	STRING	
POINT	BYTE_ARRAY	STRING	
POLYGON	BYTE_ARRAY	STRING	

PostgreSQL 데이터 형식	Parquet 프리미티브 유형	논리적 유형 주석	매핑 노트
JSON 데이터 유형			
JSON	BYTE_ARRAY	STRING	
JSONB	BYTE_ARRAY	STRING	
기타 데이터 유형			
BOOLEAN	BOOLEAN		
CIDR	BYTE_ARRAY	STRING	네트워크 데이터 유형
COMPOSITE	BYTE_ARRAY	STRING	
DOMAIN	BYTE_ARRAY	STRING	
INET	BYTE_ARRAY	STRING	네트워크 데이터 유형
MACADDR	BYTE_ARRAY	STRING	
OBJECT IDENTIFIER	해당 사항 없음		
PG_LSN	BYTE_ARRAY	STRING	
RANGE	BYTE_ARRAY	STRING	
UUID	BYTE_ARRAY	STRING	

AWS Backup를 사용하여 Amazon RDS에 대한 자동 백업 관리

AWS Backup은 클라우드 및 온프레미스에서 AWS 서비스 전반에 걸친 데이터 백업을 쉽게 중앙 집중화하고 자동화할 수 있는 종합 관리형 백업 서비스입니다. AWS Backup에서 Amazon RDS 데이터베이스의 백업을 관리할 수 있습니다.

> **Note**
>
> AWS Backup에서 관리하는 백업은 수동 DB 스냅샷으로 간주되지만 RDS의 DB 스냅샷 할당량에 포함되지 않습니다. AWS Backup으로 생성된 백업은 이름이 `awsbackup:`*`backup-job-number`*로 끝납니다.

AWS Backup에 대한 자세한 내용은 AWS Backup 개발자 안내서를 참조하세요.

AWS Backup에 의해 관리되는 백업을 보려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.
3. [백업 서비스(Backup service)] 탭을 선택합니다.

   AWS Backup 백업이 [백업 서비스 스냅샷(Backup service snapshots)]에 나열됩니다.

Amazon RDS 인스턴스에서 지표 모니터링

다음 섹션에서는 Amazon RDS 모니터링의 개요와 지표 액세스 방법에 대한 설명을 확인할 수 있습니다. 이벤트, 로그 및 데이터베이스 활동 스트림을 모니터링하는 방법에 대한 자세한 내용은 Amazon RDS DB 인스턴스에서 이벤트, 로그 및 스트림 모니터링 섹션을 참조하세요.

주제

- 모니터링 계획
- 성능 기준
- 성능 지침
- Amazon RDS의 모니터링 도구
- 인스턴스 상태 조회
- Amazon RDS의 권장 사항
- Amazon RDS 콘솔에서 지표 보기
- 성능 개선 도우미 대시보드와 결합된 지표 보기
- Amazon CloudWatch로 Amazon RDS 지표 모니터링
- CloudWatch Database Insights를 사용하여 Amazon RDS 데이터베이스 모니터링
- 성능 개선 도우미를 통한 Amazon RDS 모니터링
- Amazon DevOps Guru for Amazon RDS로 성능 이상 분석
- Enhanced Monitoring을 사용하여 OS 지표 모니터링
- Amazon RDS용 지표 참조

모니터링 계획

Amazon RDS 모니터링을 시작하기 전에 모니터링 계획을 생성합니다. 이 계획에서는 다음과 같은 의문 사항을 해결합니다.

- 모니터링의 목표
- 모니터링할 리소스
- 이러한 리소스를 모니터링하는 빈도
- 사용할 모니터링 도구

- 모니터링 작업을 수행할 사람
- 문제 발생 시 알려야 할 대상

성능 기준

모니터링 목표를 달성하려면 기준을 설정해야 합니다. 이렇게 하려면 Amazon RDS 환경에서 다양한 시간과 다양한 부하 조건으로 성능을 측정해야 합니다. 다음과 같은 지표를 모니터링할 수 있습니다.

- 네트워크 처리량
- 클라이언트 연결
- 읽기, 쓰기 또는 메타데이터 작업의 I/O
- DB 인스턴스의 버스트 크레딧 밸런스

Amazon RDS에 대한 성능 이력 데이터를 저장하는 것이 좋습니다. 저장된 데이터를 사용하여 현재 성능을 과거 추세와 비교할 수 있습니다. 또한 정상적인 성능 패턴과 이상 현상을 구별하고 문제를 해결하는 기술을 고안할 수 있습니다.

성능 지침

일반적으로 성능 지표에 허용되는 값은 기준이 무엇인지 그리고 애플리케이션 무엇을 수행하는지에 따라 다릅니다. 기준과의 일관된 차이 또는 추세를 조사하십시오. 다음과 같은 지표가 성능 문제의 원인인 경우가 많습니다.

- CPU 또는 RAM 사용량이 많음 – CPU 또는 RAM 사용량이 많을 경우 해당 애플리케이션의 목표(처리량 또는 동시성)와 일치하고 예상되는 결과라면 문제가 되지 않을 수 있습니다.
- 디스크 공간 사용량 – 총 디스크 용량의 85퍼센트 이상이 계속 사용될 경우 디스크 공간 사용량을 검사합니다. 인스턴스에서 데이터를 삭제할 수 있는지 또는 다른 시스템에 데이터를 아카이브하여 공간을 확보할 수 있는지 확인합니다.
- 네트워크 트래픽 – 네트워크 트래픽의 경우 시스템 관리자에게 문의하여 해당 도메인 네트워크 및 인터넷 연결의 기대 처리량을 확인합니다. 처리량이 기대값보다 항상 낮으면 네트워크 트래픽을 검사합니다.
- 데이터베이스 연결 – 사용자 연결 수가 많고 인스턴스 성능 및 응답 시간이 저하되는 경우 데이터베이스 연결 제한을 고려합니다. DB 인스턴스에 대한 최적의 사용자 연결 수는 해당 인스턴스 클래스와, 수행하는 작업의 복잡성에 따라 다릅니다. 데이터베이스 연결 수를 지정하려면 DB 인스턴스를 `User Connections` 파라미터가 0(무제한)이 아닌 다른 값으로 설정된 파라미터 그룹과 연결합니

다. 기존 파라미터 그룹을 사용하거나 새로 하나 만들 수 있습니다. 자세한 내용은 Amazon RDS의 파라미터 그룹 섹션을 참조하세요.

- IOPS 지표 – IOPS 지표의 기대값은 디스크 사양 및 서버 구성에 따라 다르므로 해당 기준에 일반적인 값을 파악합니다. 값이 기준과 계속 차이가 나는지 검사합니다. 최적의 IOPS 성능을 위해, 일반적인 작업 세트가 메모리에 적합하고 읽기 및 쓰기 작업을 최소화하는지 확인합니다.

성능이 설정된 기준을 벗어나면 워크로드에 맞게 데이터베이스 가용성을 최적화하기 위해 변경해야 할 수 있습니다. 예를 들어 DB 인스턴스의 인스턴스 클래스를 변경해야 할 수 있습니다. 또는 클라이언트에 사용할 수 있는 DB 인스턴스 및 읽기 전용 복제본의 수를 변경해야 할 수도 있습니다.

Amazon RDS의 모니터링 도구

모니터링은 Amazon RDS 및 사용자의 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 있어서 중요한 부분입니다. AWS는 Amazon RDS을 모니터링하고, 이상이 있을 때 이를 보고하고, 적절할 경우 자동 조치를 취할 수 있도록 모니터링 도구를 제공합니다.

주제

- 자동 모니터링 도구
- 수동 모니터링 도구

자동 모니터링 도구

모니터링 작업은 최대한 자동화하는 것이 좋습니다.

주제

- Amazon RDS 인스턴스상태 및 권장 사항
- Amazon RDS에 대한 Amazon CloudWatch 지표
- Amazon RDS 성능 개선 도우미 및 운영 체제 모니터링
- 통합 서비스

Amazon RDS 인스턴스상태 및 권장 사항

다음과 같은 자동화된 도구를 사용하여 Amazon RDS을 관찰하고 문제 발생 시 보고할 수 있습니다.

- Amazon RDS 인스턴스 상태 - Amazon RDS 콘솔, AWS CLI 또는 RDS API를 사용하여 인스턴스의 현재 상태에 대한 세부 정보를 봅니다.
- Amazon RDS 권장 사항 - DB 인스턴스, 읽기 복제본, DB 파라미터 그룹 등의 데이터베이스 리소스에 대한 자동화된 권장 사항에 응답합니다. 자세한 내용은 Amazon RDS의 권장 사항 단원을 참조하십시오.

Amazon RDS에 대한 Amazon CloudWatch 지표

Amazon RDS는 추가 모니터링 기능을 위해 Amazon CloudWatch, Amazon EventBridge 및 과 통합됩니다.

- Amazon CloudWatch – 이 서비스는 AWS 리소스와 AWS에서 실시간으로 실행 중인 애플리케이션을 모니터링합니다. 다음 Amazon CloudWatch 기능을 Amazon RDS에 사용할 수 있습니다.
  - Amazon CloudWatch 지표 – Amazon RDS는 각각의 활성 데이터베이스에 대해 자동으로 1분마다 CloudWatch로 지표를 전송합니다. CloudWatch에서 Amazon RDS 지표에 대한 추가 요금은 표시되지 않습니다. 자세한 내용은 Amazon CloudWatch로 Amazon RDS 지표 모니터링 섹션을 참조하세요.
  - Amazon CloudWatch 경보 – 특정 기간 동안 단일 Amazon RDS 지표를 볼 수 있습니다. 그런 다음 설정한 임계값과 지표 값을 비교하여 하나 이상의 작업을 수행할 수 있습니다. 자세한 내용은 Amazon CloudWatch로 Amazon RDS 지표 모니터링 단원을 참조하세요.

Amazon RDS 성능 개선 도우미 및 운영 체제 모니터링

다음과 같은 자동화된 도구를 사용하여 Amazon RDS 성능을 모니터링할 수 있습니다.

- Amazon RDS 성능 개선 도우미 - 데이터베이스의 로드를 평가하고 조치를 취할 시점과 위치를 결정합니다. 자세한 내용은 성능 개선 도우미를 통한 Amazon RDS 모니터링 단원을 참조하세요.
- Amazon RDS 확장된 모니터링 - 운영 체제에 대한 지표를 실시간으로 확인합니다. 자세한 내용은 Enhanced Monitoring을 사용하여 OS 지표 모니터링 단원을 참조하세요.

통합 서비스

다음의 AWS 서비스는 Amazon RDS와 통합됩니다.

- Amazon EventBridge: 애플리케이션을 다양한 소스의 데이터와 쉽게 연결할 수 있는 서버리스 이벤트 버스 서비스입니다. 자세한 내용은 Amazon RDS 이벤트 모니터링 단원을 참조하세요.
- Amazon CloudWatch Logs로 Amazon RDS 인스턴스, CloudTrail, 기타 소스의 로그 파일을 모니터링, 저장 및 액세스할 수 있습니다. 자세한 내용은 Amazon RDS 로그 파일 모니터링 단원을 참조하세요.
- AWS CloudTrail은 직접 수행하거나 AWS 계정을 대신하여 수행한 API 직접 호출 및 관련 이벤트를 캡처하고 지정한 Amazon S3 버킷에 로그 파일을 전송합니다. 자세한 내용은 AWS CloudTrail에서 Amazon RDS API 호출 모니터링 단원을 참조하세요.
- 데이터베이스 활동 스트림은 Oracle DB 인스턴스에서 거의 실시간에 가까운 활동 스트림을 제공하는 Amazon RDS 기능입니다. 자세한 내용은 데이터베이스 활동 스트림을 사용하여 Amazon RDS 모니터링 단원을 참조하세요.

수동 모니터링 도구

CloudWatch 경보가 다루지 않는 항목은 수동으로 모니터링해야 합니다. Amazon RDS, CloudWatch, AWS Trusted Advisor 및 기타 AWS 콘솔 대시보드에서는 AWS 환경의 상태를 한 눈에 볼 수 있습니다. 또한 DB 인스턴스에서 로그 파일을 확인하는 것이 좋습니다.

- Amazon RDS 콘솔에서 리소스에 대해 다음과 같은 항목을 모니터링할 수 있습니다.
  - DB 인스턴스에 대한 연결 수
  - DB 인스턴스에 대한 읽기 및 쓰기 작업량
  - DB 인스턴스에서 현재 사용 중인 스토리지 양
  - DB 인스턴스에 대해 사용 중인 메모리 및 CPU 양
  - DB 인스턴스에서 주고 받는 네트워크 트래픽 양
- Trusted Advisor 대시보드에서는 다음과 같은 비용 최적화, 보안, 내결함성과 성능 개선 확인을 살펴볼 수 있습니다.
  - Amazon RDS 유휴 DB 인스턴스
  - Amazon RDS 보안 그룹 액세스 위험
  - Amazon RDS 백업
  - Amazon RDS 다중 AZ

  이러한 사항에 대한 자세한 정보를 알고 싶다면 Trusted Advisor Best Practices (Checks) 단원을 참조하십시오.
- CloudWatch 홈 페이지에 표시되는 항목은 다음과 같습니다.
  - 현재 경보 및 상태
  - 경보 및 리소스 그래프
  - 서비스 상태

  또한 CloudWatch를 사용하여 다음을 수행할 수 있습니다.
  - 사용자 정의 대시보드를 생성하여 관심 있는 서비스를 모니터링
  - 지표 데이터를 그래프로 작성하여 문제를 해결하고 추세 파악.
  - 모든 AWS 리소스 지표 검색 및 찾아보기
  - 문제에 대해 알려주는 경보 생성 및 편집.

인스턴스 상태 조회

Amazon RDS 콘솔을 사용하여 빠르게 DB 인스턴스 상태에 액세스할 수 있습니다.

주제

- Amazon RDS DB 인스턴스 상태 보기

Amazon RDS DB 인스턴스 상태 보기

의 DB 인스턴스 상태는 DB 인스턴스의 상태를 나타냅니다. 다음 절차를 사용하여 Amazon RDS 콘솔, AWS CLI 명령 또는 API 작업에서 의 DB 인스턴스 상태를 볼 수 있습니다.

> **Note**
>
> 또한 Amazon RDS는 유지 관리 상태라는 상태를 한 가지 더 사용하며, 이는 Amazon RDS 콘솔의 유지 관리 열에 표시됩니다. 이 값은 DB 인스턴스에 적용해야 하는 모든 유지 관리 패치 상태를 나타냅니다. 유지 관리 상태는 DB 인스턴스 상태와 무관합니다. 유지 관리 상태에 대한 자세한 내용은 DB 인스턴스에 업데이트 적용 단원을 참조하세요.

다음 표에서 DB 인스턴스에 사용할 수 있는 상태 값을 찾습니다. 이 표에는 DB 인스턴스와 스토리지에만 요금이 청구되는지, 스토리지에만 청구되는지, 혹은 청구되지 않는지 표시됩니다. 모든 DB 인스턴스 상태의 백업 사용에는 항상 청구됩니다.

DB 인스턴스 상태	청구	설명
available	청구	DB 인스턴스에 문제가 없으며 사용할 수 있습니다.
backing-up	청구	현재 DB 인스턴스를 백업 중입니다.
configuring-enhanced-monitoring	청구	이 DB 인스턴스에 대해 확장 모니터링이 활성화 또는 비활성화 상태입니다.
configuring-iam-database-auth	청구	이 DB 인스턴스에 대해 AWS Identity and Access Management(IAM) 데이터베이스 인증이 활성화 또는 비활성화 상태입니다.
configuring-log-exports	청구	Amazon CloudWatch Logs에 로그 파일을 게시하는 기능이 이 DB 인스턴스에 대해 활성화 또는 비활성화 상태입니다.
converting-to-vpc	청구	DB 인스턴스를 Amazon Virtual Private Cloud(Amazon VPC)에 없는 DB 인스턴스에서 Amazon VPC에 있는 DB 인스턴스로 변환 중입니다.

DB 인스턴스 상태	청구	설명
creating	미청구(비 PITR) 청구(PITR만 해당)	DB 인스턴스를 생성 중입니다. 생성 중인 DB 인스턴스에는 액세스할 수 없습니다. 시점 복구(PITR) 시 데이터베이스를 복원하는 경우 데이터베이스가 생성 중 상태인 동안 요금이 청구됩니다. 이는 생성 중 상태에서 요금이 발생하는 유일한 시나리오입니다.
delete-precheck	미청구	Amazon RDS가 읽기 전용 복제본이 정상이며 삭제해도 안전한지 검증하고 있습니다.
deleting	미청구	DB 인스턴스를 삭제 중입니다.
failed	미청구	DB 인스턴스 오류가 발생하여 Amazon RDS로 복구할 수 없습니다. DB 인스턴스의 복원 가능한 가장 최근 시간을 기준으로 특정 시점으로 복원을 수행하여 데이터를 복구합니다.
inaccessible-encryption-credentials	미청구	DB 인스턴스를 암호화 또는 복호화하는 데 사용되는 AWS KMS key에 액세스하거나 복구할 수 없습니다.
inaccessible-encryption-credentials-recoverable	요금 부과 스토리지	DB 인스턴스를 암호화 또는 해독하는 데 사용되는 KMS 키는 액세스할 수 없습니다. 그러나 KMS 키가 활성 상태이면 DB 인스턴스를 다시 시작하면 복구할 수 있습니다. 자세한 내용은 DB 인스턴스 암호화 섹션을 참조하세요.
incompatible-create	미청구	Amazon RDS가 DB 인스턴스를 만들려고 하지만 리소스가 DB 인스턴스와 호환되지 않기 때문에 만들 수 없습니다. 예를 들어 DB 인스턴스의 인스턴스 프로파일에 올바른 권한이 없는 경우 이 상태가 발생할 수 있습니다.

DB 인스턴스 상태	청구	설명
incompatible-network	미청구	Amazon RDS가 DB 인스턴스에 대한 복구 작업을 시도했으나 VPC의 상태로 인하여 작업을 완료할 수 없어 복구하지 못했습니다. 예를 들어 서브넷에 사용 가능한 IP 주소가 모두 사용 중이어서 Amazon RDS가 DB 인스턴스에 대한 IP 주소를 받을 수 없는 경우 이러한 상태가 발생할 수 있습니다.
incompatible-option-group	청구	Amazon RDS 옵션 그룹 변경을 적용하려고 시도했으나 적용하지 못했습니다. 또한 Amazon RDS는 이전 옵션 그룹 상태를 롤백하지 못했습니다. 자세한 내용은 DB 인스턴스에 대한 최근 이벤트 목록을 참조하십시오. 예를 들어, 옵션 그룹은 TDE 등의 옵션을 포함하는데 DB 인스턴스는 암호화된 정보를 포함하지 않는 경우 이러한 상태가 발생할 수 있습니다.
incompatible-parameters	청구	Amazon RDS가 DB 인스턴스의 DB 파라미터 그룹에서 지정된 파라미터가 DB 인스턴스와 호환되지 않아 DB 인스턴스를 시작할 수 없습니다. DB 인스턴스에 대한 액세스 권한을 다시 얻으려면 변경된 파라미터를 되돌리거나 DB 인스턴스와 호환되는 파라미터를 정의해야 합니다. 호환되지 않는 파라미터에 대한 자세한 내용은 DB 인스턴스에 대한 최근 이벤트 목록을 참조하십시오.
incompatible-restore	미청구	Amazon RDS가 특정 시점으로 복원을 수행할 수 없습니다. 임시 테이블 사용, MySQL에서 MyISAM 테이블 사용 또는 MariaDB에서 Aria 테이블 사용이 일반적인 원인입니다.
insufficient-capacity	미청구	Amazon RDS가 현재 충분한 용량을 사용할 수 없기 때문에 인스턴스를 생성할 수 없습니다. 동일한 인스턴스 유형으로 동일한 AZ에 DB 인스턴스를 생성하려면 DB 인스턴스를 삭제하고 몇 시간 기다린 후 다시 생성해 봅니다. 또는 다른 인스턴스 클래스 또는 AZ를 사용하여 새 인스턴스를 생성합니다.

DB 인스턴스 상태	청구	설명
유지 관리	청구	Amazon RDS는 DB 인스턴스에 유지 관리 업데이트를 적용합니다. 이 상태는 RDS가 한참 전에 미리 예약하는 인스턴스 수준의 유지 관리에 사용됩니다.
modifying	청구	고객의 DB 인스턴스 수정 요청으로 인해 DB 인스턴스를 수정 중입니다.
moving-to-vpc	청구	DB 인스턴스가 새 Amazon Virtual Private Cloud(Amazon VPC)로 이동하는 중입니다.
rebooting	청구	고객의 요청 또는 DB 인스턴스 재부팅이 필요한 Amazon RDS 프로세스로 인해 DB 인스턴스를 재부팅 중입니다.
resetting-master-credentials	청구	고객의 재설정 요청으로 인해 DB 인스턴스 사용자 자격 증명을 재설정하는 중입니다.
renaming	청구	고객의 이름 바꾸기 요청으로 인해 DB 인스턴스 이름을 바꾸는 중입니다.
restore-error	청구	특정 시점으로 복원을 시도하거나 스냅샷에서 복원을 시도할 때 DB 인스턴스에서 오류가 발생했습니다.
starting	요금 부과 스토리지	DB 인스턴스를 시작하는 중입니다.
stopped	요금 부과 스토리지	DB 인스턴스가 중지되었습니다.
stopping	요금 부과 스토리지	DB 인스턴스를 중지 중입니다.
storage-config-upgrade	청구	DB 인스턴스의 스토리지 파일 시스템 구성이 업그레이드되고 있습니다. 이 상태는 블루/그린 배포 내의 그린 데이터베이스 또는 DB 인스턴스 읽기 전용 복제본에만 적용됩니다.

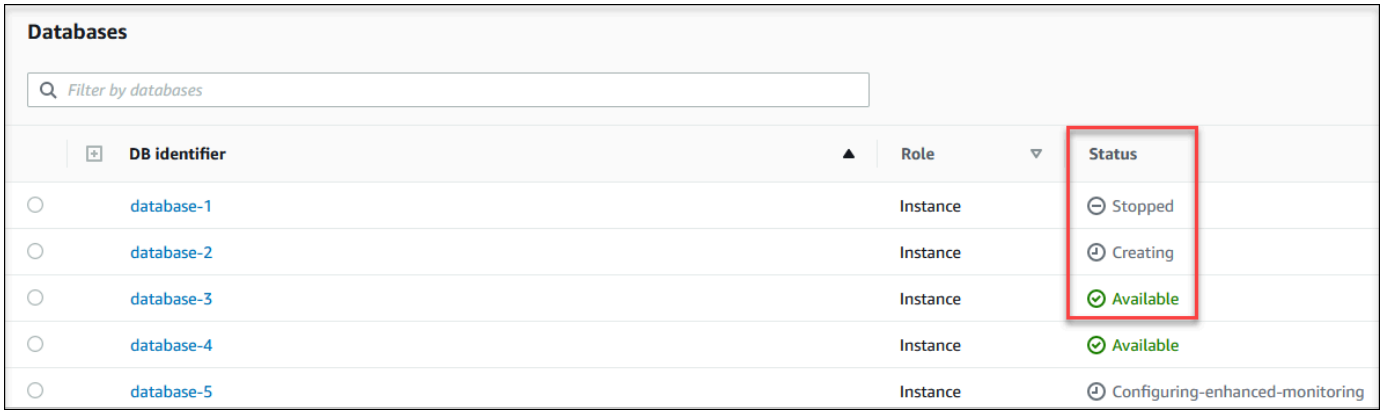
DB 인스턴스 상태	청구	설명
storage-full	청구	DB 인스턴스에 할당된 스토리지 용량이 거의 다 찼습니다. 이것은 중요한 상태이므로 즉각 이 문제를 수정하는 것이 좋습니다. 그러려면 DB 인스턴스를 수정하여 스토리지를 확장하십시오. 이러한 상황을 피하려면 스토리지 공간이 부족해지면 경고하도록 Amazon CloudWatch 경보를 설정하십시오.
storage-initialization	청구	DB 인스턴스는 스냅샷에서 복원된 후 볼륨 성능을 최적화하기 위해 Amazon S3에서 데이터 블록을 로드하고 있습니다. 작업에 사용할 수 있지만 초기화가 완료될 때까지 성능이 최대가 아닐 수 있습니다.
storage-optimization	청구	Amazon RDS가 DB 인스턴스의 스토리지를 최적화하고 있습니다. 스토리지 최적화 프로세스는 보통 짧지만, 가끔 최대 24시간까지 걸릴 수도 있습니다. 스토리지 최적화 중에는 DB 인스턴스를 계속 사용할 수 있습니다. 스토리지 최적화는 인스턴스의 가용성에 영향을 주지 않는 백그라운드 프로세스입니다.
upgrading	청구	데이터베이스 엔진 또는 운영 체제 버전은 현재 업그레이드 중입니다.

콘솔

DB 인스턴스의 상태를 보려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택합니다.

   DB 인스턴스 목록과 함께 데이터베이스 페이지가 표시됩니다. 각 DB 인스턴스에 대해 상태 값이 표시됩니다.



CLI

AWS CLI를 사용하여 DB 인스턴스 및 DB 인스턴스 상태 정보를 보려면 describe-db-instances 명령을 사용하십시오. 예를 들어 다음 AWS CLI 명령은 모든 DB 인스턴스의 정보를 나열합니다.

```
aws rds describe-db-instances
```

특정 DB 인스턴스 및 DB 인스턴스 상태를 보려면 다음 옵션을 지정한 describe-db-instances 명령을 호출하십시오.

- `DBInstanceIdentifier` – DB 인스턴스의 이름입니다.

```
aws rds describe-db-instances --db-instance-identifier mydbinstance
```

모든 DB 인스턴스의 상태만 확인하려면 AWS CLI에서 다음 쿼리를 사용하십시오.

```
aws rds describe-db-instances --query 'DBInstances[*].
[DBInstanceIdentifier,DBInstanceStatus]' --output table
```

API

Amazon RDS API를 사용하여 DB 인스턴스 상태를 보려면 DescribeDBInstances 작업을 호출하십시오.

Amazon RDS의 권장 사항

Amazon RDS에서는 DB 인스턴스, 읽기 전용 복제본, DB 파라미터 그룹과 같은 데이터베이스 리소스에 대한 자동화된 권장 사항을 제공합니다. 이러한 권장 사항은 DB 인스턴스 구성, 사용량 및 성능 데이터에 대한 분석을 통해 모범 사례 지침을 제공합니다.

Amazon RDS 성능 개선 도우미는 특정 지표를 모니터링하고 특정 리소스에 대해 문제가 될 수 있다고 간주되는 수준을 분석하여 임계값을 자동으로 생성합니다. 지정된 기간 동안 새 지표 값이 사전 정의된 임계값을 초과하면 성능 개선 도우미는 사전 예방적 권장 사항을 생성합니다. 이 권장 사항은 향후 데이터베이스 성능에 미치는 영향을 방지하는 데 도움이 됩니다. 예를 들어, 데이터베이스에 연결된 세션이 활성 작업을 수행하지 않지만 데이터베이스 리소스가 차단된 상태로 유지될 수 있는 경우 RDS for PostgreSQL 인스턴스에 대해 'Idle In Transaction' 권장 사항이 생성됩니다. 사전 예방적 권장 사항을 받으려면 유료 등급 보존 기간과 함께 성능 개선 도우미를 활성화해야 합니다. 성능 개선 도우미를 활성화하는 방법에 대한 자세한 내용은 Amazon RDS의 성능 개선 도우미 설정 및 해제 섹션을 참조하세요. 성능 개선 도우미의 요금 및 데이터 보존에 대한 자세한 내용은 성능 개선 도우미의 요금 및 데이터 보존 섹션을 참조하세요.

DevOps Guru for RDS는 특정 지표를 모니터링하여 지표의 행동이 매우 비정상적이거나 이례적으로 전환되는 시점을 탐지합니다. 이러한 이상 현상은 권장 사항이 포함된 사후 대응적 인사이트로 보고됩니다. 예를 들어, DevOps Guru for RDS는 CPU 용량을 늘리거나 DB 로드에 기여하는 대기 이벤트를 조사할 것을 권장할 수 있습니다. DevOps Guru for RDS는 임계값 기반 사전 예방적 권장 사항도 제공합니다. 이러한 권장 사항을 적용하려면 DevOps Guru for RDS를 켜야 합니다. DevOps Guru for RDS를 켜는 방법에 대한 자세한 내용은 DevOps Guru 활성화 및 리소스 적용 범위 지정 섹션을 참조하세요.

권장 사항은 활성, 무시, 보류 또는 해결됨 상태 중 하나로 나타납니다. 해결된 권장 사항은 365일 동안 제공됩니다.

권장 사항을 보거나 무시할 수 있습니다. 즉시 구성 기반 활성 권장 사항을 적용하거나, 다음 유지 관리 기간으로 예약하거나, 무시할 수 있습니다. 임계값 기반 사전 예방 및 기계 학습 기반 사후 대응 권장 사항의 경우 문제의 제안된 원인을 검토한 다음 권장 조치를 수행하여 문제를 해결해야 합니다.

권장 사항은 다음 AWS 리전에서 지원됩니다.

- 미국 동부(오하이오)
- 미국 동부(버지니아 북부)
- 미국 서부(캘리포니아 북부)
- 미국 서부(오레곤)

- 아시아 태평양(뭄바이)
- 아시아 태평양(서울)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(도쿄)
- 캐나다(중부)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(파리)
- 유럽(스톡홀름)
- 남아메리카(상파울루)

다음 섹션에서 Amazon RDS의 권장 사항을 보고, 적용하고, 무시하고, 수정하는 방법을 알아보세요.

주제

- Amazon RDS 권장 사항 보기
- Amazon RDS 권장 사항 적용
- Amazon RDS 권장 사항 무시
- 무시된 Amazon RDS 권장 사항을 활성 권장 사항으로 수정
- Amazon RDS 권장 사항 참조

Amazon RDS 권장 사항 보기

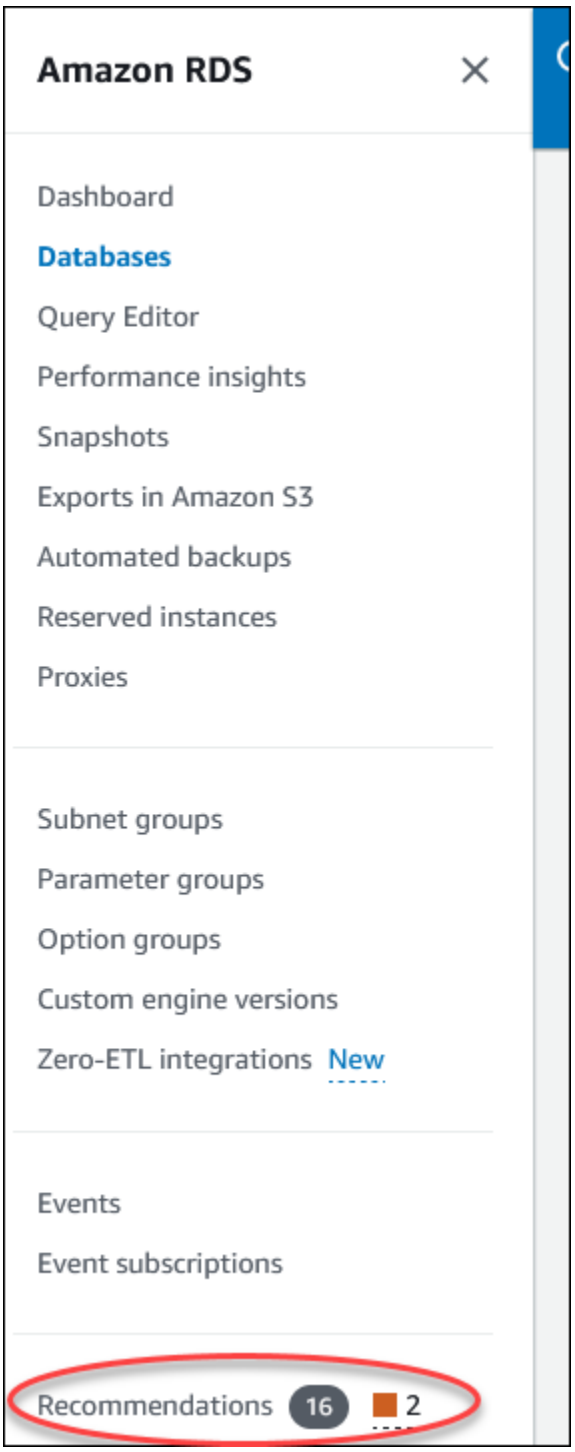
Amazon RDS 콘솔을 사용하면 데이터베이스 리소스에 대한 Amazon RDS 권장 사항을 볼 수 있습니다.

콘솔

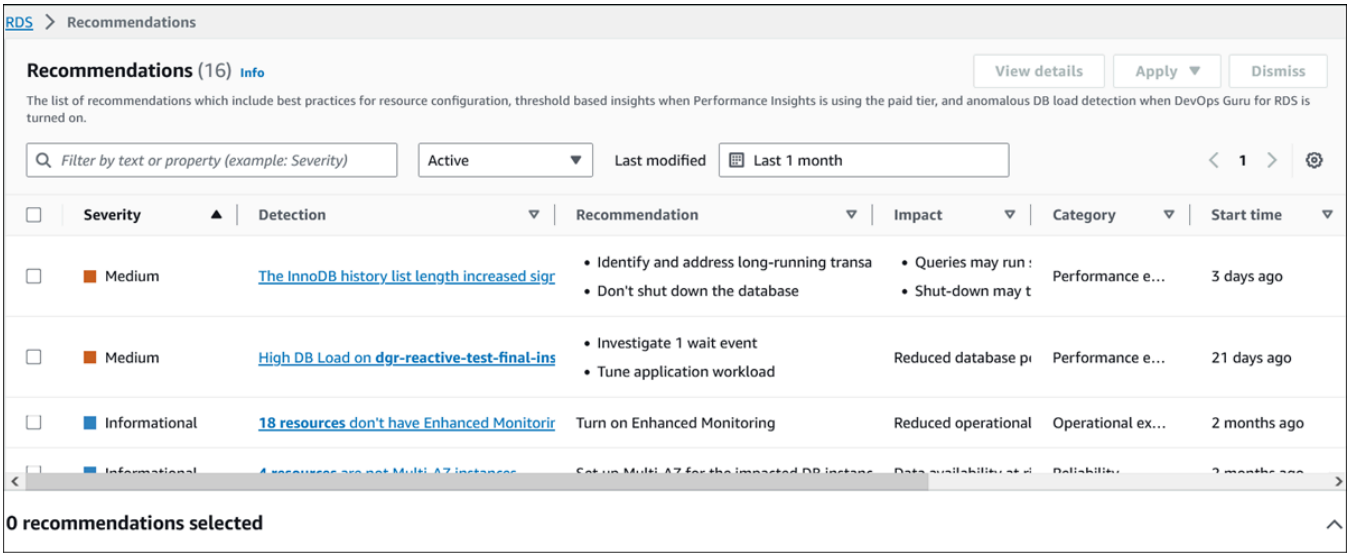
Amazon RDS 권장 사항을 확인하려면

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 다음 중 하나를 수행합니다.

   - 권장 사항을 선택합니다. 리소스에 대한 활성 권장 사항 수와 지난달 생성된 심각도가 가장 높은 권장 사항 수는 권장 사항 옆에서 확인할 수 있습니다. 각 심각도에 대한 활성 권장 사항 수를 찾으려면 가장 높은 심각도를 나타내는 숫자를 선택하면 됩니다.

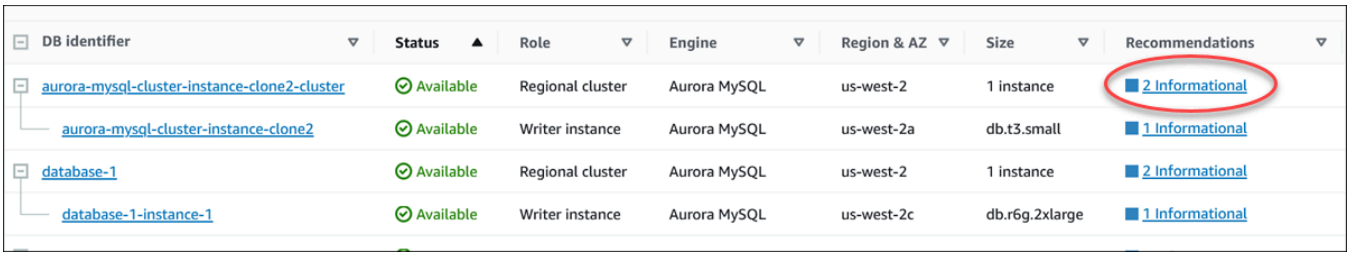


기본적으로 권장 사항 페이지에는 지난달의 새 권장 사항 목록이 표시됩니다. Amazon RDS는 계정의 모든 리소스에 대해 권장 사항을 제공하고 심각도별로 권장 사항을 정렬합니다.

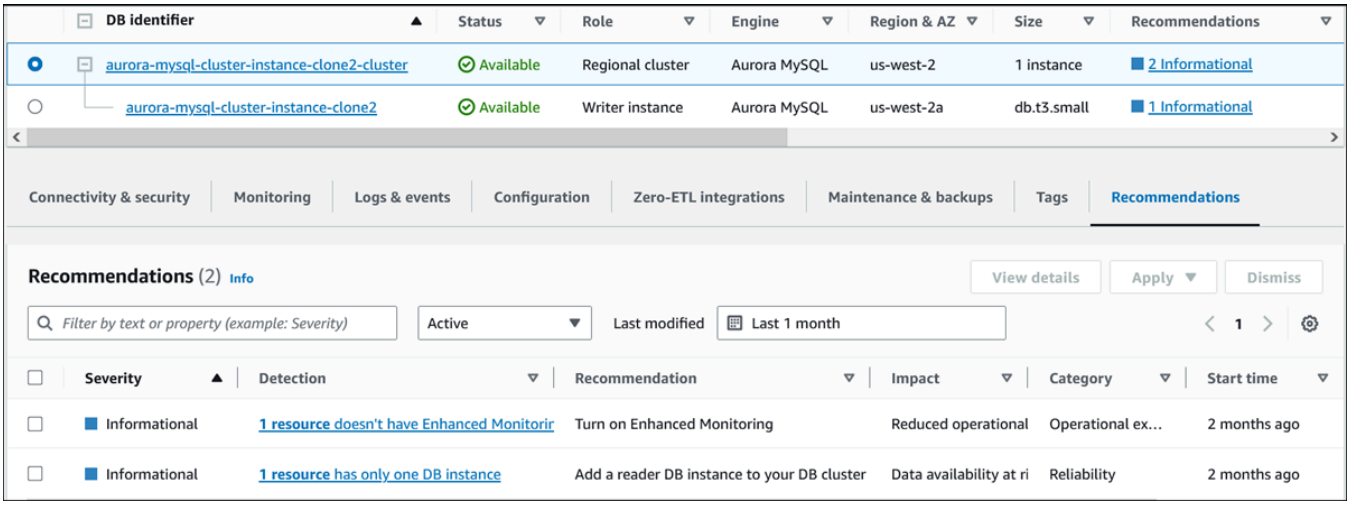


권장 사항을 선택하여 페이지 하단에서 영향을 받는 리소스와 권장 사항 적용 방법에 대한 세부 정보가 포함된 섹션을 볼 수 있습니다.

- 데이터베이스 페이지에서 리소스에 대한 권장 사항을 선택합니다.



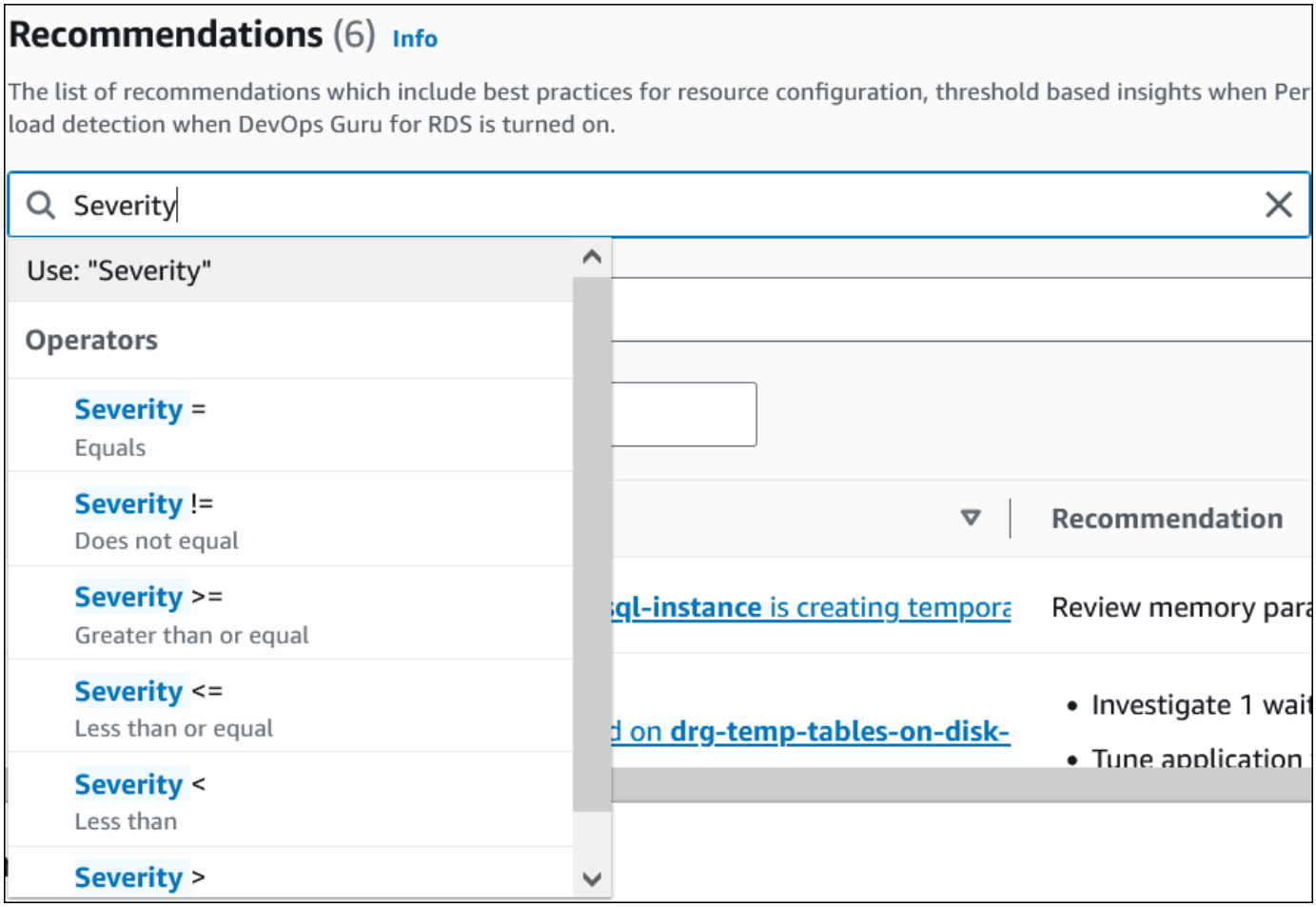
권장 사항 탭에는 선택한 리소스에 대한 권장 사항 및 세부 정보가 표시됩니다.



권장 사항에 대한 세부 정보는 다음과 같습니다.

- 심각도 - 문제의 영향 수준입니다. 심각도 수준은 높음, 보통, 낮음, 정보용으로 나뉩니다.
- 탐지 - 영향을 받는 리소스의 수와 문제에 대한 간략한 설명입니다. 권장 사항 및 분석 세부 정보를 보려면 이 링크를 선택하세요.
- 권장 사항 - 적용할 권장 조치에 대한 간략한 설명입니다.
- 영향 - 권장 사항이 적용되지 않을 때 발생할 수 있는 영향에 대한 간략한 설명입니다.
- 카테고리 - 권장 사항 유형입니다. 카테고리는 성능 효율성, 보안, 신뢰성, 비용 최적화, 운영 우수성, 지속 가능성입니다.
- 상태 - 현재 권장 사항의 상태입니다. 가능한 상태로는 모두, 활성, 무시, 해결됨, 보류 중이 있습니다.
- 시작 시간 - 문제가 시작된 시간입니다. 예를 들어, 18시간 전일 수 있습니다.
- 최종 수정 시간 - 심각도 변경으로 인해 시스템에서 권장 사항을 마지막으로 업데이트한 시간 또는 권장 사항에 응답한 시간입니다. 예를 들어, 10시간 전일 수 있습니다.
- 종료 시간 - 문제가 종료된 시간입니다. 계속되는 문제의 경우 시간이 표시되지 않습니다.
- 리소스 식별자 - 하나 이상의 리소스 이름입니다.

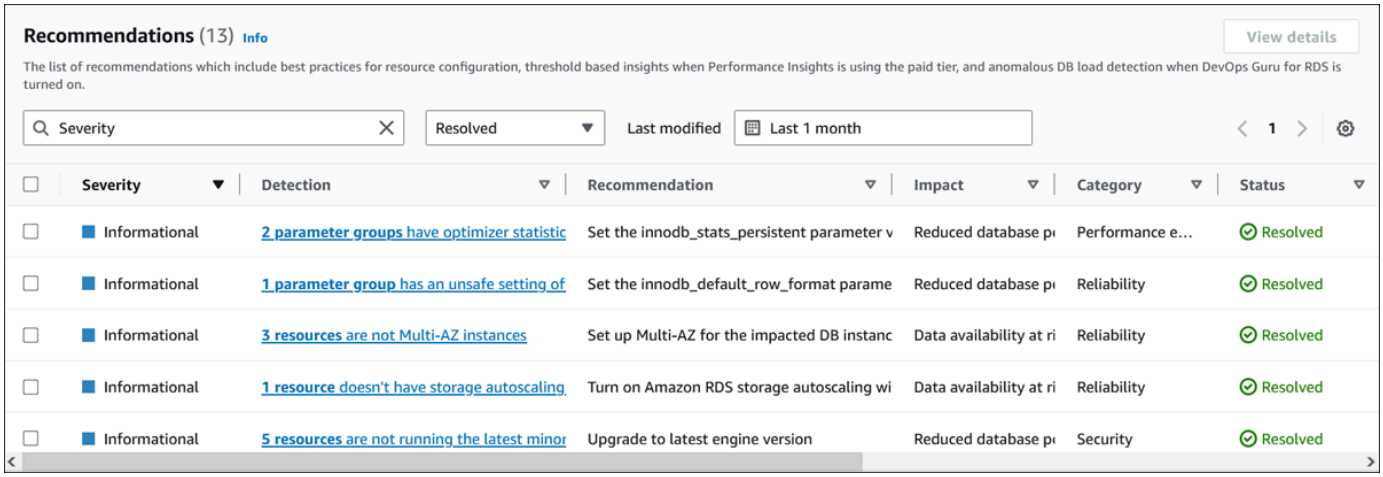
3. (선택 사항) 필드에서 심각도 또는 카테고리 연산자를 선택하여 권장 사항 목록을 필터링합니다.



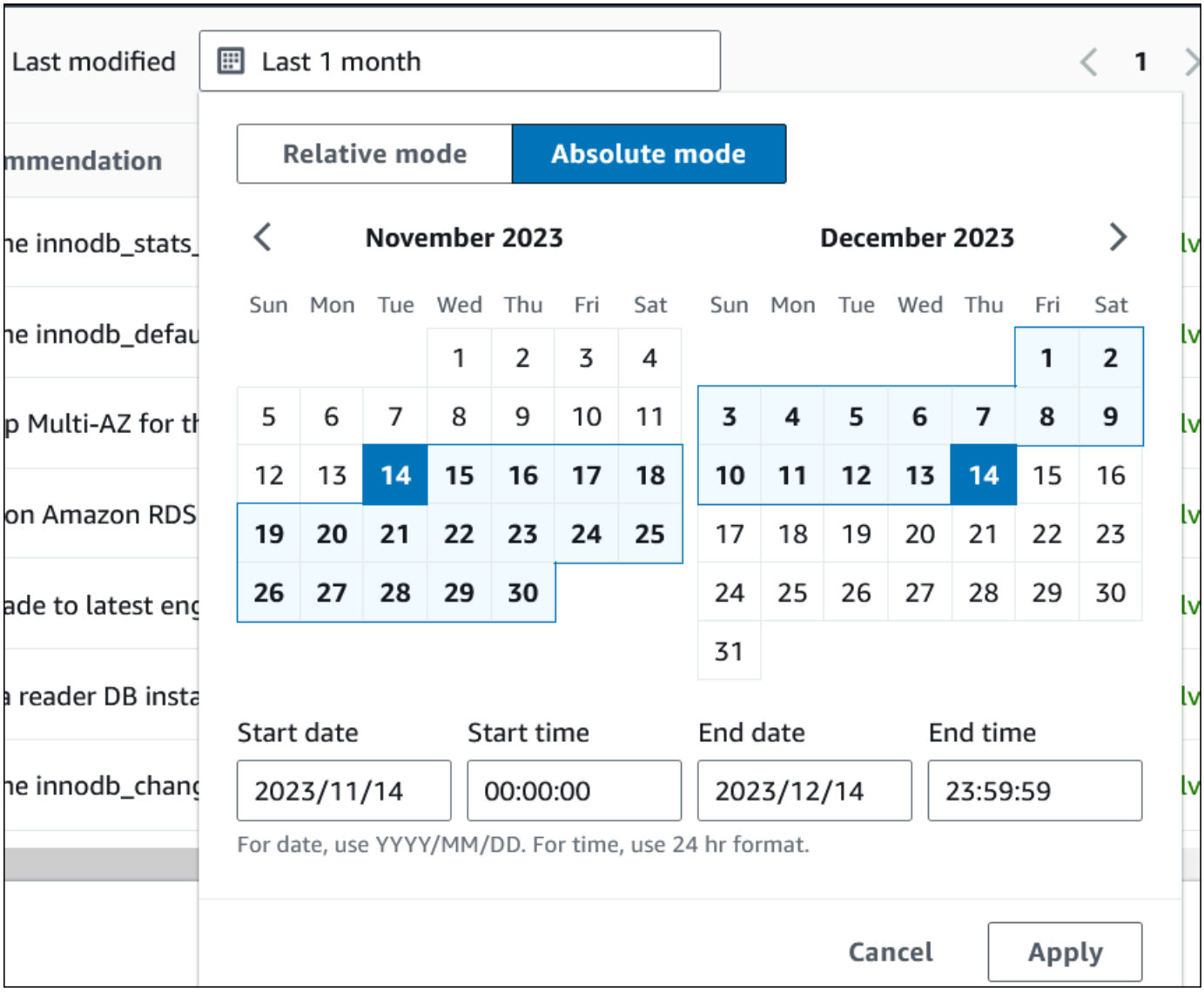
선택한 작업에 대한 권장 사항이 표시됩니다.

4. (선택 사항) 다음 권장 사항 상태 중 하나를 선택합니다.

   - 활성(기본값) – 적용하거나, 다음 유지 관리 기간에 예약하거나, 무시할 수 있는 현재 권장 사항을 표시합니다.
   - 모두 - 현재 상태의 모든 권장 사항을 표시합니다.
   - 무시 - 무시된 권장 사항을 표시합니다.
   - 해결됨 - 해결된 권장 사항을 표시합니다.
   - 보류 중 - 권장 조치가 진행 중이거나 다음 유지 관리 기간에 예정되어 있는 권장 사항을 표시합니다.



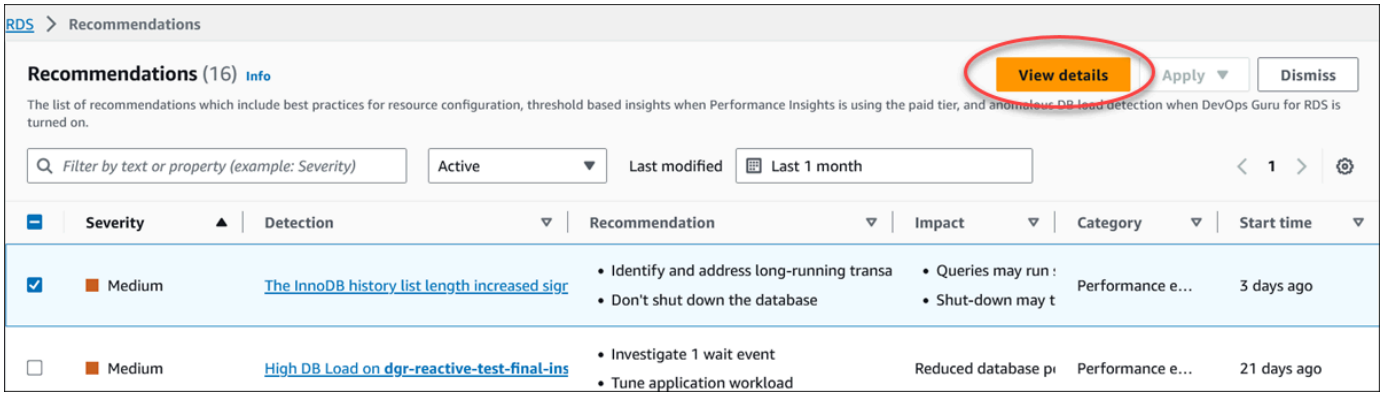
5. (선택 사항) 기간을 수정하려면 최종 수정 시간에서 상대 모드 또는 절대 모드를 선택합니다. 권장 사항 페이지에는 해당 기간에 생성된 권장 사항이 표시됩니다. 기본 기간은 지난달입니다. 절대 모드에서는 기간을 선택하거나 시작 날짜 및 종료 날짜 필드에 시간을 입력할 수 있습니다.



설정된 기간에 대한 권장 사항이 표시됩니다.

범위를 전체로 설정하면 계정 내 리소스에 대한 모든 권장 사항을 볼 수 있다는 점을 참고하세요.

6. (선택 사항) 오른쪽에서 기본 설정을 선택하여 표시할 세부 정보를 사용자 지정합니다. 페이지 크기를 선택하고, 텍스트 줄을 줄 바꿈하고, 열을 허용하거나 숨길 수 있습니다.
7. (선택 사항) 권장 사항을 선택한 다음 세부 정보 보기를 선택합니다.



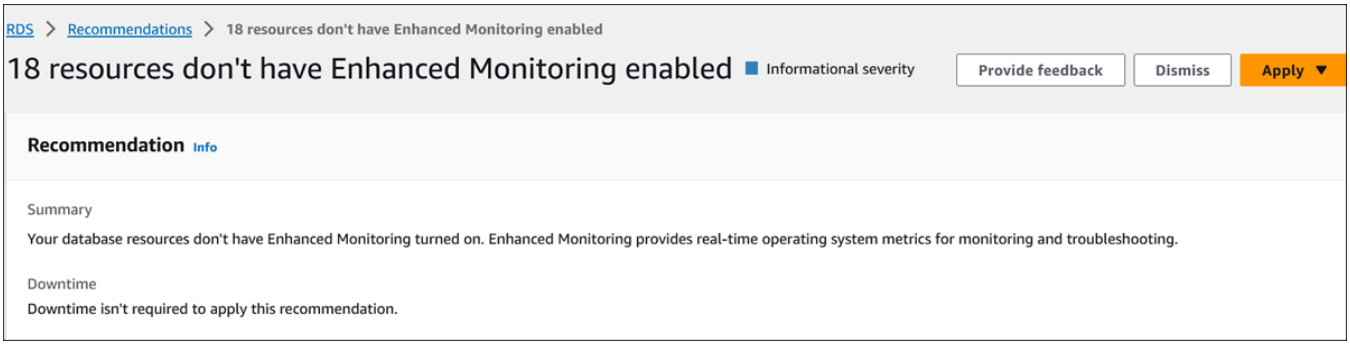
권장 사항 세부 정보 페이지가 표시됩니다. 제목은 발견된 문제가 있는 리소스의 총 개수와 심각도를 제공합니다.

이상 항목 기반 사후 대응 권장 사항의 세부 정보 페이지에 있는 구성 요소에 대한 자세한 내용은 Amazon DevOps Guru 사용 설명서의 Viewing reactive anomalies를 참조하세요.

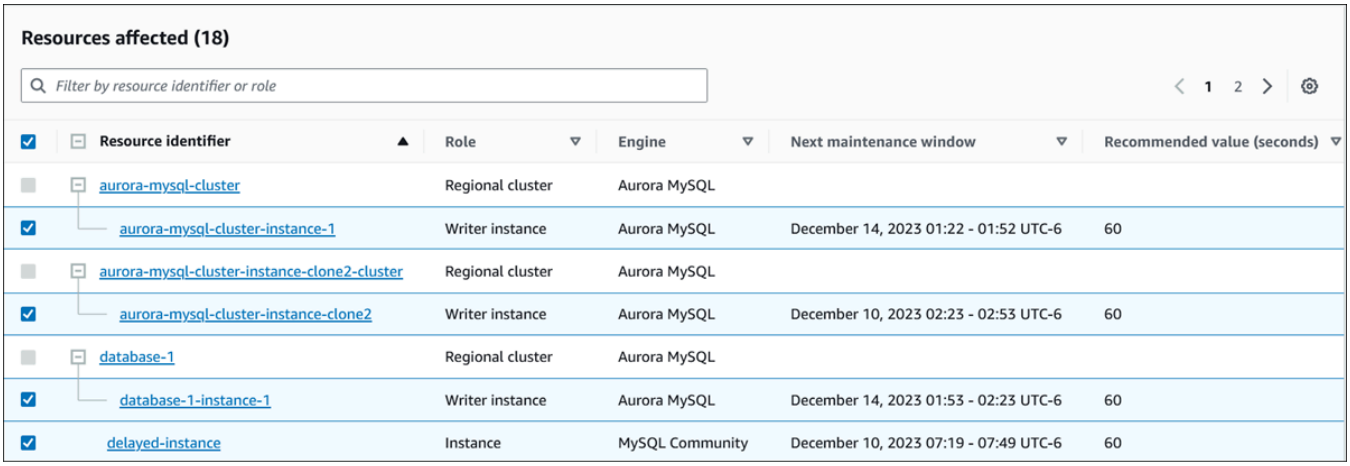
임계값 기반 사전 권장 사항의 세부 정보 페이지에 있는 구성 요소에 대한 자세한 내용은 성능 개선 도우미 사전 권장 사항 보기 섹션을 참조하세요.

다른 자동 권장 사항은 권장 사항 세부 정보 페이지에 다음 구성 요소를 표시합니다.

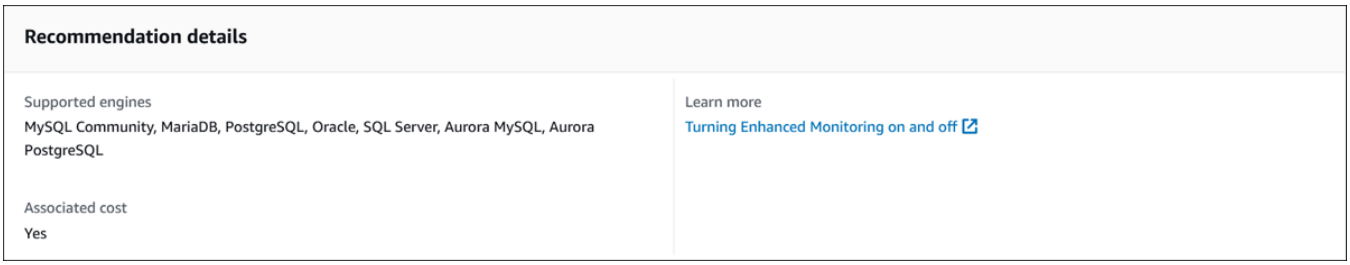
- 권장 사항 - 권장 사항에 대한 요약과 권장 사항을 적용하는 데 가동 중지가 필요한지 여부를 표시합니다.



- 영향을 받는 리소스 - 영향을 받는 리소스의 세부 정보입니다.



- 권장 사항 세부 정보 - 지원되는 엔진 정보, 권장 사항을 적용하는 데 필요한 모든 관련 비용, 자세히 알아볼 수 있는 설명서 링크가 나와 있습니다.



CLI

DB 인스턴스 의 Amazon RDS 권장 사항을 보려면 AWS CLI에서 다음 명령을 사용하세요.

```
aws rds describe-db-recommendations
```

RDS API

Amazon RDS API를 사용하여 Amazon RDS 권장 사항을 보려면 DescribeDBRecommendations 작업을 사용하세요.

Amazon RDS 권장 사항 적용

Amazon RDS 콘솔을 사용하여 Amazon RDS 권장 사항을 적용하려면 세부 정보 페이지에서 구성 기반 권장 사항 또는 영향을 받는 리소스를 선택하세요. 그런 다음 권장 사항을 즉시 적용하거나 다음 유지 관리 기간으로 미루도록 선택하세요. 변경 사항을 적용하려면 리소스를 다시 시작해야 할 수 있습니다. 일부 DB 파라미터 그룹 권장 사항의 경우 리소스를 다시 시작해야 할 수 있습니다.

임계값 기반 사전 예방적 권장 사항이나 이상 항목 기반 사후 권장 사항에는 적용 옵션이 없으므로, 추가 검토가 필요할 수 있습니다.

콘솔

구성 기반 권장 사항을 적용하려면

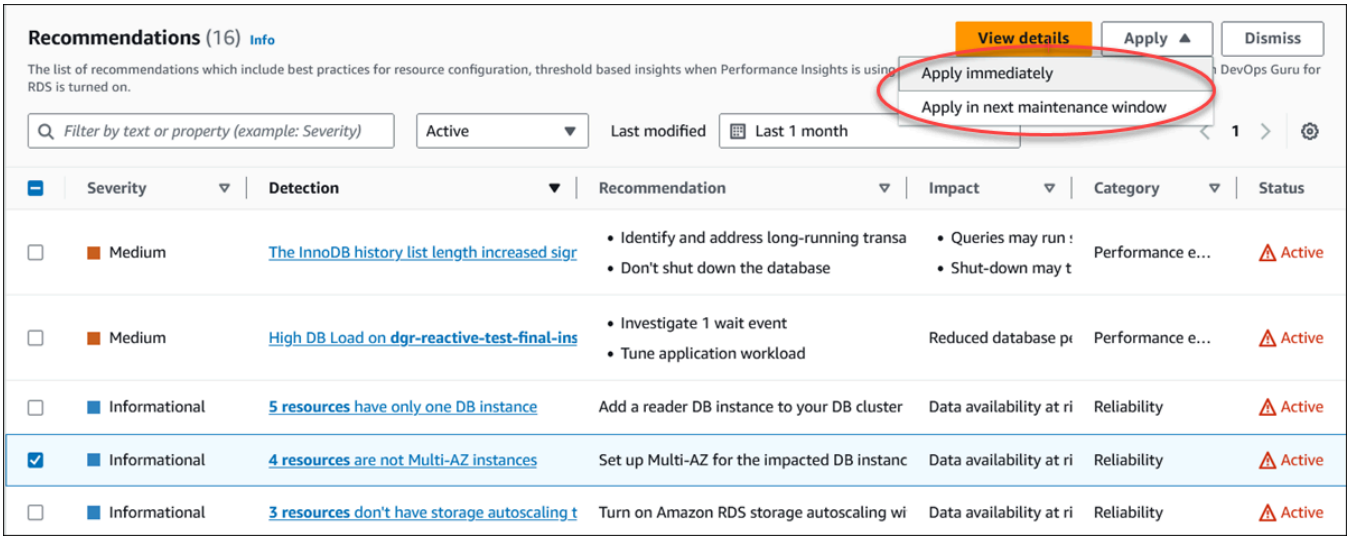
1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 다음 중 하나를 수행합니다.
   - 권장 사항을 선택합니다.

     모든 권장 사항 목록과 함께 권장 사항 페이지가 나타납니다.
   - 데이터베이스를 선택한 다음 데이터베이스 페이지에서 리소스에 대한 권장 사항을 선택합니다.

     선택한 권장 사항의 권장 사항 탭에 세부 정보가 표시됩니다.
   - 권장 사항 페이지 또는 데이터베이스 페이지의 권장 사항 탭에서 활성 권장 사항에 대한 탐지를 선택합니다.

     권장 사항 세부 정보 페이지가 표시됩니다.
3. 권장 사항 세부 정보 페이지에서 권장 사항 또는 영향을 받는 리소스를 하나 이상 선택하고 다음 중 하나를 수행합니다.
   - 적용을 선택한 다음 즉시 적용을 선택하여 권장 사항을 즉시 적용합니다.
   - 적용을 선택하고 다음 유지 관리 기간에 적용을 선택하여 다음 유지 관리 기간에 일정을 예약합니다.

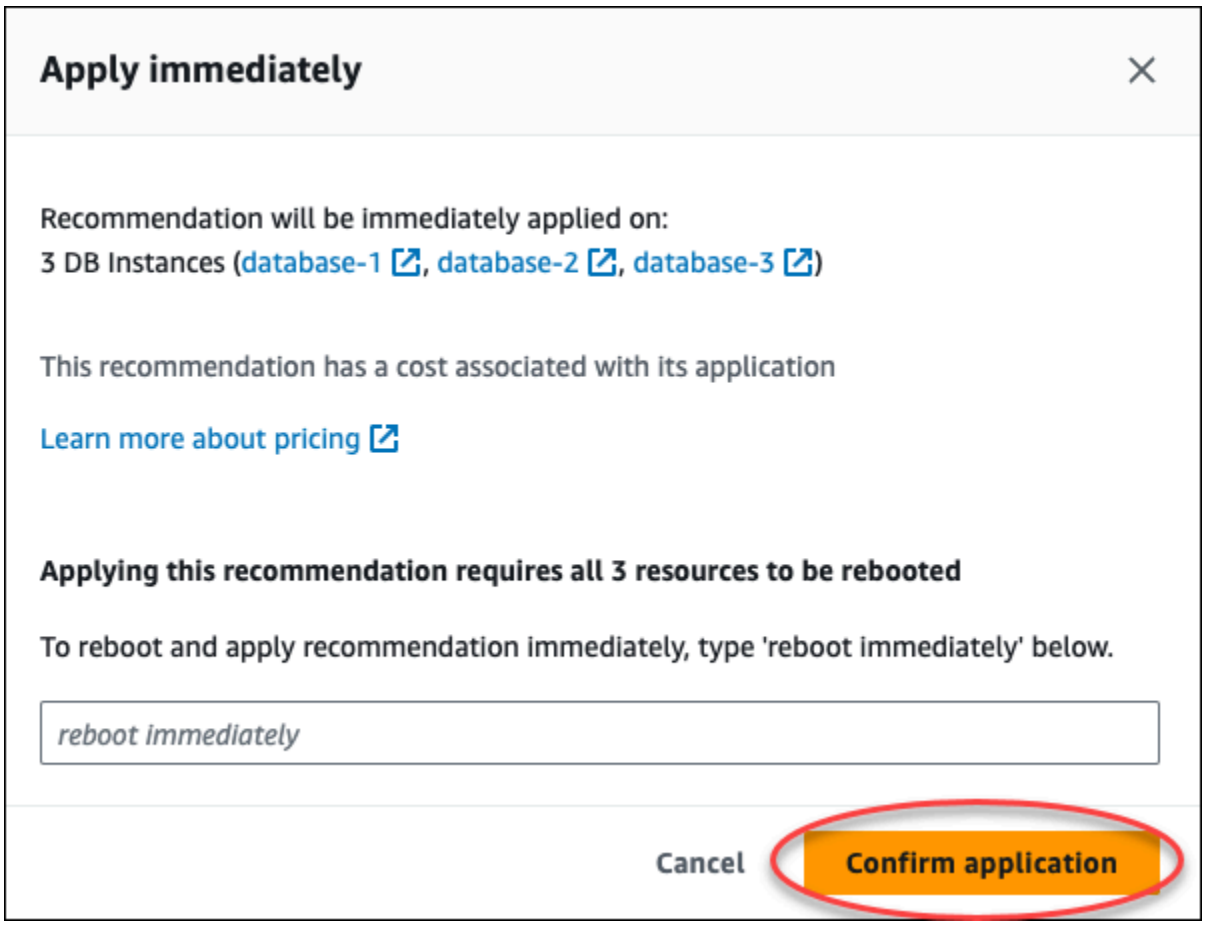
     선택한 권장 사항 상태는 다음 유지 관리 기간까지 보류 중으로 업데이트됩니다.



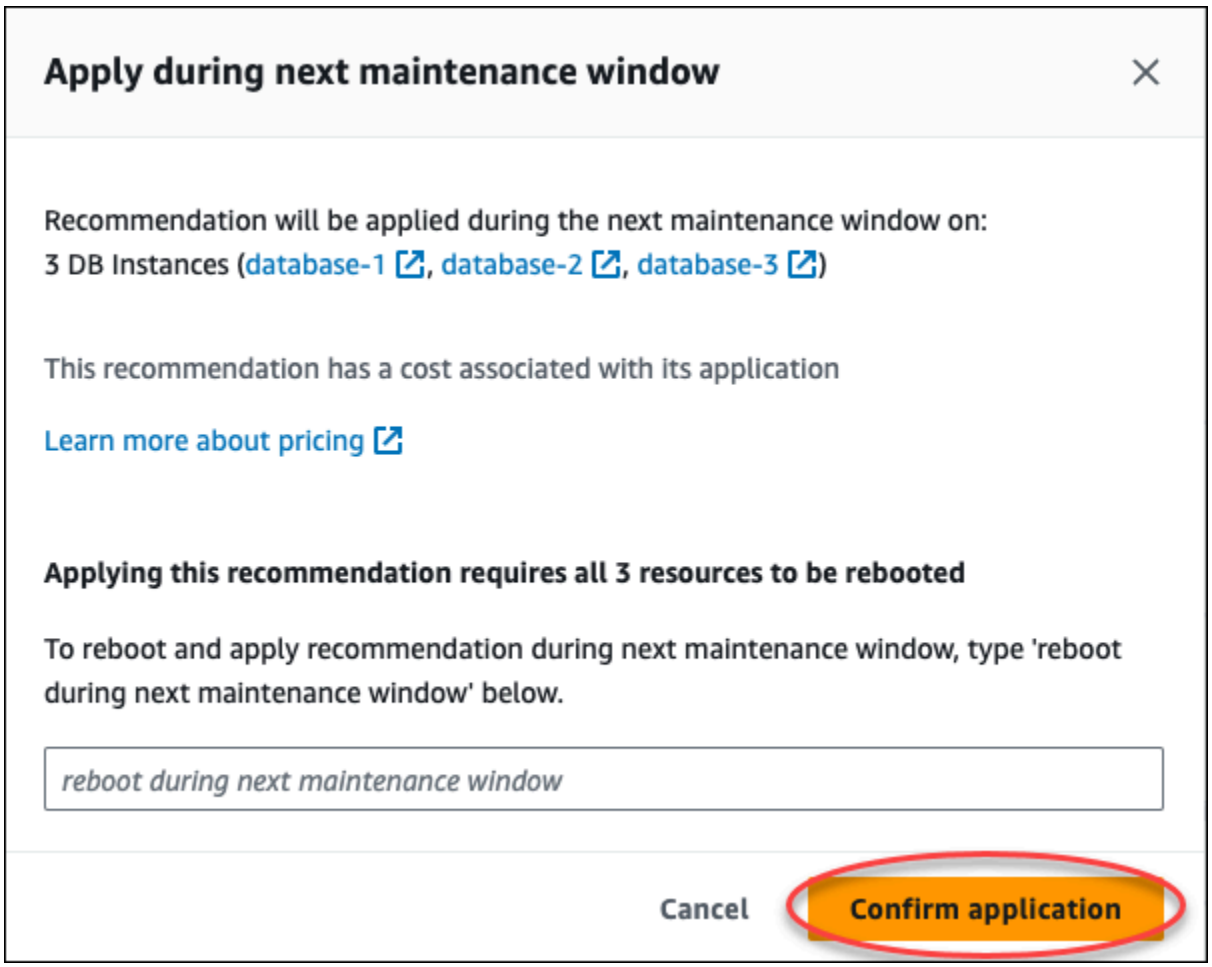
확인 창이 나타납니다.

4. 애플리케이션 확인을 선택하여 권장 사항을 적용합니다. 이 창은 변경 사항을 적용하기 위해 리소스를 자동으로 다시 시작해야 할지, 수동으로 다시 시작해야 할지 확인합니다.

   다음 예제에서는 권장 사항을 즉시 적용하기 위한 확인 창을 보여줍니다.

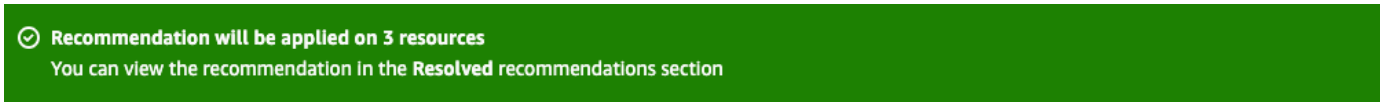


다음 예제는 다음 유지 관리 기간에 권장 사항을 적용하기로 예약하는 확인 창을 보여줍니다.



배너에 권장 사항이 적용되거나 적용에 실패하면 메시지가 표시됩니다.

다음 예제에서는 성공 메시지가 포함된 배너를 보여줍니다.



다음 예제에서는 실패 메시지가 포함된 배너를 보여줍니다.



RDS API

Amazon RDS API를 사용하여 구성 기반 RDS 권장 사항을 적용하려면

1. DescribeDBRecommendations 작업을 사용합니다. 출력의 RecommendedActions에 하나 이상의 권장 조치가 포함될 수 있습니다.
2. 1단계의 각 권장 조치에 대해 RecommendedAction 객체를 사용합니다. 출력에는 Operation 및 Parameters가 포함됩니다.

   다음 예제는 하나 이상의 권장 조치가 포함된 출력을 보여줍니다.

```
   "RecommendedActions": [
       {
           "ActionId": "0b19ed15-840f-463c-a200-b10af1b552e3",
           "Title": "Turn on auto backup", // localized
           "Description": "Turn on auto backup for my-mysql-instance-1", //
localized
           "Operation": "ModifyDbInstance",
           "Parameters": [
               {
                   "Key": "DbInstanceIdentifier",
                   "Value": "my-mysql-instance-1"
               },
               {
                   "Key": "BackupRetentionPeriod",
                   "Value": "7"
               }
           ],
           "ApplyModes": ["immediately", "next-maintenance-window"],
           "Status": "applied"
       },
       ... // several others
   ],
```

3. 2단계의 출력에서 각 권장 조치에 대해 operation을 사용하고 Parameters 값을 입력합니다.
4. 2단계의 작업이 성공하면 ModifyDBRecommendation 작업을 사용하여 권장 사항 상태를 수정합니다.

Amazon RDS 권장 사항 무시

Amazon RDS 콘솔, AWS CLI 또는 Amazon RDS API를 사용하여 하나 이상의 Amazon RDS 권장 사항을 무시할 수 있습니다.

콘솔

하나 이상의 권장 사항을 무시하려면

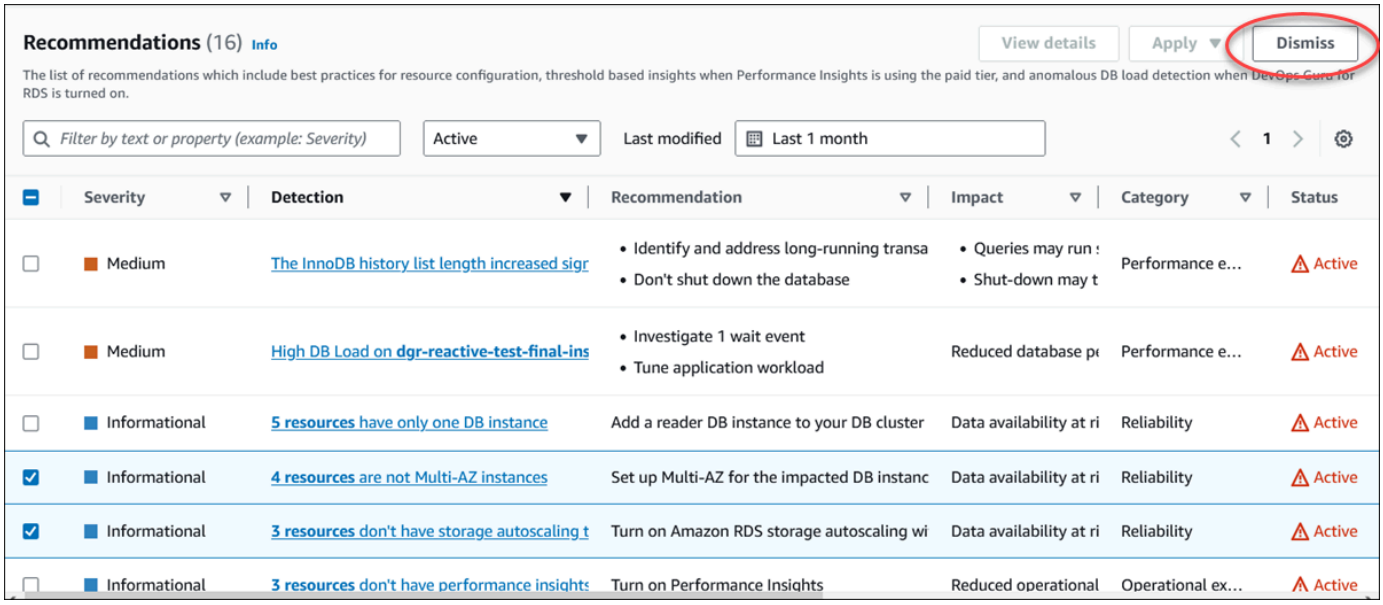
1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 다음 중 하나를 수행합니다.
   - 권장 사항을 선택합니다.

     모든 권장 사항 목록과 함께 권장 사항 페이지가 나타납니다.
   - 데이터베이스를 선택한 다음 데이터베이스 페이지에서 리소스에 대한 권장 사항을 선택합니다.

     선택한 권장 사항의 권장 사항 탭에 세부 정보가 표시됩니다.
   - 권장 사항 페이지 또는 데이터베이스 페이지의 권장 사항 탭에서 활성 권장 사항에 대한 탐지를 선택합니다.

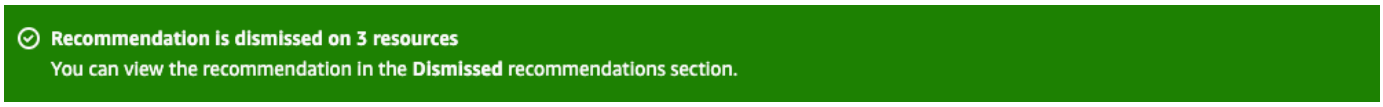
     권장 사항 세부 정보 페이지에는 영향을 받는 리소스 목록이 표시됩니다.
3. 권장 사항 세부 정보 페이지에서 하나 이상의 권장 사항 또는 영향을 받는 리소스를 선택한 다음 무시를 선택합니다.

   다음 예제는 무시하기로 선택한 활성 권장 사항이 여러 개 있는 권장 사항 페이지를 보여줍니다.



선택한 하나 이상의 권장 사항이 무시되면 배너에 메시지가 표시됩니다.

다음 예제에서는 성공 메시지가 포함된 배너를 보여줍니다.



다음 예제에서는 실패 메시지가 포함된 배너를 보여줍니다.



CLI

AWS CLI를 사용하여 RDS 권장 사항을 무시하려면

1. `aws rds describe-db-recommendations --filters "Name=status,Values=active"` 명령을 실행합니다.

   출력에서 `active` 상태에 있는 권장 사항의 목록을 제공합니다.
2. 1단계에서 무시하려는 권장 사항에 대한 `recommendationId`를 찾습니다.
3. 2단계에서 `recommendationId`와 함께 `>aws rds modify-db-recommendation --status dismissed --recommendationId <ID>` 명령을 실행하여 권장 사항을 무시합니다.

RDS API

Amazon RDS API를 사용하여 RDS 권장 사항을 무시하려면 ModifyDBRecommendation 작업을 사용합니다.

무시된 Amazon RDS 권장 사항을 활성 권장 사항으로 수정

Amazon RDS 콘솔, AWS CLI 또는 Amazon RDS API를 사용하여 무시된 Amazon RDS 권장 사항 하나 이상을 활성 권장 사항으로 변경할 수 있습니다.

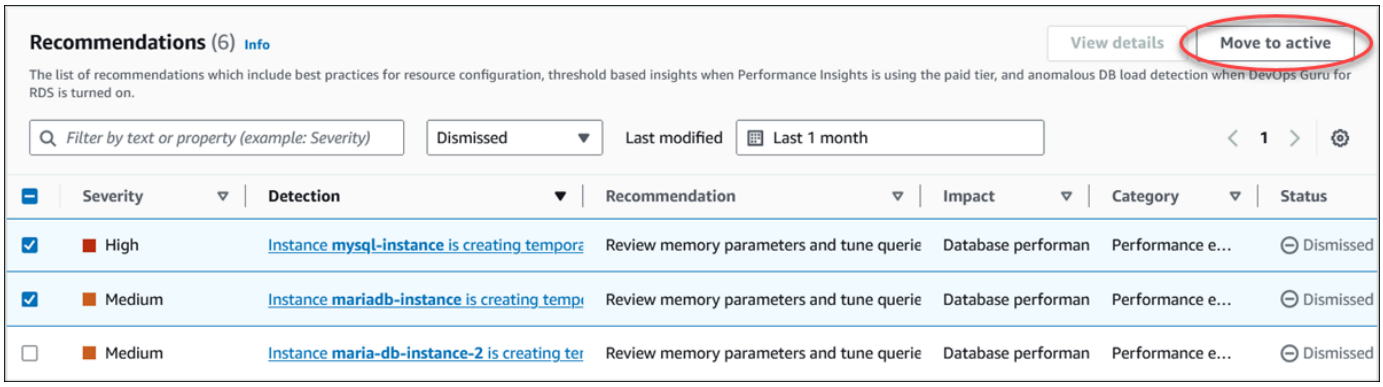
콘솔

하나 이상의 무시된 권장 사항을 활성 권장 사항으로 이동하려면

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 다음 중 하나를 수행합니다.
   - 권장 사항을 선택합니다.

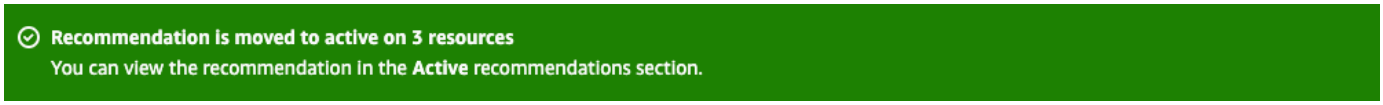
     권장 사항 페이지에는 계정 내 모든 리소스의 심각도별로 정렬된 권장 사항 목록이 표시됩니다.
   - 데이터베이스를 선택한 다음 데이터베이스 페이지에서 리소스에 대한 권장 사항을 선택합니다.

     권장 사항 탭에는 선택한 리소스에 대한 권장 사항 및 세부 정보가 표시됩니다.
3. 목록에서 무시된 권장 사항을 하나 이상 선택한 다음 활성으로 이동을 선택합니다.



선택한 권장 사항을 무시 상태에서 활성 상태로 이동하면 배너에 성공 또는 실패 메시지가 표시됩니다.

다음 예제에서는 성공 메시지가 포함된 배너를 보여줍니다.



다음 예제에서는 실패 메시지가 포함된 배너를 보여줍니다.



CLI

AWS CLI를 사용하여 무시된 RDS 권장 사항을 활성 권장 사항으로 변경하려면

1. `aws rds describe-db-recommendations --filters "Name=status,Values=dismissed"` 명령을 실행합니다.

   출력에서 `dismissed` 상태에 있는 권장 사항의 목록을 제공합니다.
2. 1단계에서 상태를 변경하려는 권장 사항에 대한 `recommendationId`를 찾습니다.
3. 2단계에서 `recommendationId`와 함께 `>aws rds modify-db-recommendation --status active --recommendationId <ID>` 명령을 실행하여 활성 권장 사항으로 변경합니다.

RDS API

Amazon RDS API를 사용하여 무시된 RDS 권장 사항을 활성 권장 사항으로 변경하려면 ModifyDBRecommendation 작업을 사용합니다.

Amazon RDS 권장 사항 참조

Amazon RDS는 리소스가 생성되거나 수정될 때 리소스에 대해 권장 사항을 생성합니다. 다음 표에서 Amazon RDS의 권장 사항 예제를 찾을 수 있습니다.

유형	설명	권장 사항	가동 중지 필요	추가 정보
사용 중인 마그네틱 볼륨	DB 인스턴스에서 마그네틱 스토리지를 사용 중입니다. 대부분 DB	범용(SSD) 또는 프로비저닝된 IOPS 등 다	예	Amazon EC2 설명서의 이전 세대 볼륨

유형	설명	권장 사항	가동 중지 필요	추가 정보
	인스턴스에 대해 마그네틱 스토리지는 권장되지 않습니다. 범용(SSD) 또는 프로비저닝된 IOPS 등 다른 스토리지 유형을 선택하세요.	른 스토리지 유형을 선택하세요.		
리소스 자동 백업이 비활성화되어 있습니다.	DB 인스턴스에 대해 자동 백업이 활성화되어 있지 않습니다. 자동 백업은 DB 인스턴스의 특정 시점 복구를 가능하게 하므로, 권장됩니다.	보존 기간이 최대 14일인 자동 백업을 활성화하세요.	예	자동 백업 활성화 AWS 데이터베이스 블로그의 Demystifying Amazon RDS backup storage costs
엔진 마이너 버전 업그레이드가 필요합니다.	데이터베이스 리소스가 최신 마이너 DB 엔진 버전을 실행하지 않습니다. 최신 마이너 버전에는 최신 보안 수정 및 기타 개선 사항이 포함되어 있습니다.	최신 엔진 버전으로 업그레이드하세요.	예	DB 인스턴스 엔진 버전 업그레이드
향상된 모니터링이 꺼졌습니다.	데이터베이스 리소스에 향상된 모니터링이 켜져 있지 않습니다. 확장된 모니터링은 모니터링 및 문제 해결을 위해 실시간 운영 체제 지표를 제공합니다.	향상된 모니터링을 켜세요.	No	Enhanced Monitoring을 사용하여 OS 지표 모니터링

유형	설명	권장 사항	가동 중지 필요	추가 정보
스토리지 암호화가 비활성화되어 있습니다.	Amazon RDS는 AWS Key Management Service(AWS KMS)에서 관리하는 키를 사용하여 모든 데이터베이스 엔진에 대한 저장 중 암호화를 지원합니다. Amazon RDS 암호화를 사용하는 활성 DB 인스턴스에서는 스토리지에 저장된 데이터가 자동 백업, 읽기 전용 복제본 및 스냅샷과 마찬가지로 암호화됩니다. DB 인스턴스를 만들 때 암호화가 켜져 있지 않으면 암호화를 켜기 전에 DB 인스턴스의 해독된 스냅샷에 대한 암호화된 사본을 만들어 복원해야 합니다.	DB 인스턴스에 저장된 데이터의 암호화를 활성화하세요.	예	Amazon RDS의 보안 Amazon RDS용 DB 스냅샷 복사
성능 개선 도우미가 비활성화되어 있습니다.	성능 개선 도우미는 데이터베이스 성능 문제를 분석하고 해결하는 데 도움이 되는 DB 인스턴스 로드를 모니터링합니다. 성능 개선 도우미를 켜는 것이 좋습니다.	성능 개선 도우미를 활성화합니다.	No	성능 개선 도우미를 통한 Amazon RDS 모니터링

유형	설명	권장 사항	가동 중지 필요	추가 정보
DB 인스턴스에 스토리지 자동 크기 조정이 해제되어 있습니다.	DB 인스턴스에 대해 스토리지 자동 크기 조정이 활성화되어 있지 않습니다. 데이터베이스 워크로드가 증가하면 RDS 스토리지 자동 크기 조정 기능은 가동 중지 시간 없이 스토리지 용량을 자동으로 확장합니다.	지정된 최대 스토리지 임계값으로 Amazon RDS 스토리지 자동 크기 조정을 활성화하세요.	No	Amazon RDS 스토리지 Autoscaling을 사용한 용량 자동 관리
RDS 리소스 메이저 버전 업데이트가 필요합니다.	현재 DB 엔진용 메이저 버전이 설치된 데이터베이스는 지원되지 않습니다. 새 기능과 개선 사항이 포함된 최신 메이저 버전으로 업그레이드하는 것이 좋습니다.	DB 엔진의 최신 메이저 버전으로 업그레이드하세요.	예	DB 인스턴스 엔진 버전 업그레이드 데이터베이스 업데이트에 Amazon RDS 블루/그린 배포 사용
RDS 리소스 인스턴스 클래스 업데이트가 필요합니다.	DB 인스턴스가 이전 세대 DB 인스턴스 클래스를 실행하고 있습니다. 이전 세대의 DB 인스턴스 클래스를 더 좋은 비용, 성능 또는 이 둘 모두를 갖춘 DB 인스턴스 클래스로 대체했습니다. 최신 세대의 DB 인스턴스 클래스로 DB 인스턴스를 실행하는 것이 좋습니다.	DB 인스턴스 클래스를 업그레이드하세요.	예	DB 인스턴스 클래스에 지원되는 DB 엔진

유형	설명	권장 사항	가동 중지 필요	추가 정보
RDS 리소스가 라이선스 포함 상태에서 지원 종료 엔진 에디션을 사용합니다.	현재 라이선스를 계속 지원받으려면 메이저 버전을 Amazon RDS에서 지원하는 최신 엔진 버전으로 업그레이드하는 것이 좋습니다. 데이터베이스의 엔진 버전이 현재 라이선스에서 지원되지 않습니다.	라이선스 모델을 계속 사용하려면 데이터베이스를 Amazon RDS에서 지원되는 최신 버전으로 업그레이드하는 것이 좋습니다.	예	Oracle 메이저 버전 업그레이드

유형	설명	권장 사항	가동 중지 필요	추가 정보
DB 인스턴스가 다중 AZ 배포를 사용하지 않습니다.	다중 AZ 배포를 사용하는 것이 좋습니다. 다중 AZ 배포는 DB 인스턴스의 가용성과 내구성을 향상합니다.	영향을 받는 DB 인스턴스에 대한 다중 AZ를 설정하세요.	No 이 변경 도중에는 가동 중지 시간이 발생하지 않습니다. 그러나 성능에 영향을 줄 수 있습니다. 자세한 내용은 DB 인스턴스를 Amazo RDS용 다중 AZ	Amazon RDS 다중 AZ 요금

유형	설명	권장 사항	가동 중지 필요	추가 정보
			배포로 변환 단원을 참조하세요.	
DB 메모리 파라미터가 기본값과 다릅니다.	DB 인스턴스의 메모리 파라미터가 기본값과 크게 다릅니다. 이러한 설정은 성능에 영향을 미치고 오류를 일으킬 수 있습니다. DB 인스턴스에 대한 사용자 지정 메모리 파라미터를 DB 파라미터 그룹의 기본값으로 재설정하는 것이 좋습니다.	메모리 파라미터를 기본값으로 재설정하세요.	No	AWS 데이터베이스 블로그의 Best practices for configuring performance parameters for Amazon RDS for MySQL

유형	설명	권장 사항	가동 중지 필요	추가 정보
최적 값보다 작은 값을 사용하는 `InnoDB_Change_Buffering` 파라미터	변경 버퍼링을 사용하면 MySQL DB 인스턴스가 보조 인덱스를 유지하는 데 필요한 몇 가지 쓰기를 연기할 수 있습니다. 이 기능은 디스크 속도가 느린 환경에서 유용했습니다. 변경 버퍼링 구성으로 인해 DB 성능이 약간 향상되었지만, 충돌 복구가 지연되고 업그레이드 중 종료 시간이 길어졌습니다. MySQL 버전 8.4에서 기본적으로 `OFF`로 설정되어 있습니다.	DB 파라미터 그룹의 `InnoDB_Change_Buffering` 파라미터를 `NONE`으로 설정하세요.	No	AWS 데이터베이스 블로그의 Best practices for configuring performance parameters for Amazon RDS for MySQL
쿼리 캐시 파라미터가 켜져 있습니다.	변경으로 인해 쿼리 캐시를 제거해야 하는 경우 DB 인스턴스가 정지된 것처럼 보입니다. 쿼리 캐시는 대부분의 워크로드에 이점이 되지 못합니다. 쿼리 캐시는 MySQL 8.0 이상 버전에서 제거되었습니다. query_cache_type 파라미터를 0으로 설정하는 것이 좋습니다.	DB 파라미터 그룹에서 `query_cache_type` 파라미터 값을 `0`으로 설정하세요.	예	AWS 데이터베이스 블로그의 Best practices for configuring performance parameters for Amazon RDS for MySQL

유형	설명	권장 사항	가동 중지 필요	추가 정보
log_output 파라미터가 표로 설정되었습니다.	log_output 을 TABLE로 설정하면 log_output 을 FILE로 설정한 경우보다 더 많은 스토리지가 사용됩니다. 스토리지 크기 제한에 도달하지 않도록 파라미터를 FILE로 설정하는 것이 좋습니다. MySQL 8.4 이상 버전에서 기본적으로 FILE로 설정되어 있습니다.	DB 파라미터 그룹에서 log_output 파라미터 값을 FILE으로 설정하세요.	No	MySQL 데이터베이스 로그 파일
파라미터 그룹이 대용량 페이지를 사용하지 않습니다.	대용량 페이지는 데이터베이스 확장성을 높일 수 있는데, DB 인스턴스가 대용량 페이지를 사용하지 않습니다. use_large_pages 파라미터 값을 DB 인스턴스의 DB 파라미터 그룹에서만 ONLY로 설정하는 것이 좋습니다.	DB 파라미터 그룹에서 use_large_pages 파라미터 값을 ONLY으로 설정하세요.	예	RDS for Oracle 인스턴스에 HugePages 활성화

유형	설명	권장 사항	가동 중지 필요	추가 정보
autovacuum 파라미터가 비활성화되어 있습니다.	DB 인스턴스의 autovacuum 파라미터가 비활성화되어 있습니다. autovacuum 기능을 비활성화하면 표 및 인덱스 팽창이 증가하고 성능에 영향을 미칩니다. DB 파라미터 그룹에서 autovacuum을 켜는 것이 좋습니다.	DB 파라미터 그룹에서 autovacuum 파라미터를 활성화하세요.	No	AWS 데이터베이스 블로그의 Understanding autovacuum in Amazon RDS for PostgreSQL environments
synchronous_commit 파라미터가 비활성화되어 있습니다.	synchronous_commit 파라미터를 끄면 데이터베이스 충돌로 인해 데이터가 손실될 수 있습니다. 데이터베이스의 내구성에 악영향을 미칠 수 있습니다. synchronous_commit 파라미터를 활성화하는 것이 좋습니다.	DB 파라미터 그룹에서 synchronous_commit 파라미터를 활성화하세요.	예	AWS 데이터베이스 블로그의 Amazon Aurora PostgreSQL parameters: Replication, security, and logging

유형	설명	권장 사항	가동 중지 필요	추가 정보
track_counts 파라미터가 비활성화되어 있습니다.	track_counts 파라미터를 비활성화하면 데이터베이스에서 데이터베이스 활동 통계를 수집하지 않습니다. Autovacuum을 사용하려면 이러한 통계가 제대로 작동해야 합니다. track_counts 파라미터를 1으로 설정하는 것이 좋습니다.	track_counts 파라미터를 1로 설정하세요.	No	PostgreSQL의 런타임 통계
enable_indexonlyscan 파라미터가 비활성화되어 있습니다.	인덱스 전용 스캔 계획 유형이 비활성화되어 있으면 쿼리 플래너 또는 옵티마이저가 인덱스 전용 스캔 계획 유형을 사용할 수 없습니다. enable_indexonlyscan 파라미터 값을 1으로 설정하는 것이 좋습니다.	enable_indexonlyscan 파라미터 값을 1로 설정하세요.	No	PostgreSQL용 플래너 메서드 구성

유형	설명	권장 사항	가동 중지 필요	추가 정보
enable_indexscan 파라미터가 비활성화되어 있습니다.	인덱스 스캔 계획 유형이 비활성화되어 있으면 쿼리 플래너 또는 옵티마이저가 인덱스 스캔 계획 유형을 사용할 수 없습니다. enable_indexscan 값을 1로 설정하는 것이 좋습니다.	enable_indexscan 파라미터 값을 1로 설정하세요.	No	PostgreSQL용 플래너 메서드 구성
innodb_flush_log_at_trx 파라미터가 비활성화되어 있습니다.	DB 인스턴스의 innodb_flush_log_at_trx 파라미터 값은 안전한 값이 아닙니다. 이 파라미터는 디스크에 대한 커밋 작업의 지속성을 제어합니다. innodb_flush_log_at_trx 파라미터를 1으로 설정하는 것이 좋습니다.	innodb_flush_log_at_trx 파라미터 값을 1로 설정하세요.	No	AWS 데이터베이스 블로그의 Best practices for configuring performance parameters for Amazon RDS for MySQL

유형	설명	권장 사항	가동 중지 필요	추가 정보
sync_binlog 파라미터가 비활성화되어 있습니다.	트랜잭션 커밋이 DB 인스턴스에서 확인되기 전에는 이진 로그를 디스크에 동기화하지 않습니다. sync_binlog 파라미터 값을 1으로 설정하는 것이 좋습니다.	sync_binlog 파라미터 값을 1로 설정하세요.	No	AWS 데이터베이스 블로그의 Best practices for configuring replication parameters for Amazon RDS for MySQL
innodb_stats_persistent 파라미터가 비활성화되어 있습니다.	DB 인스턴스가 InnoDB 통계를 디스크에 유지하도록 구성되지 않았습니다. 통계가 저장되지 않으면 인스턴스가 다시 시작되고 표에 액세스할 때마다 통계가 다시 계산됩니다. 이로 인해 쿼리 실행 계획이 달라질 수 있습니다. 테이블 수준에서 이 글로벌 파라미터의 값을 수정할 수 있습니다. innodb_stats_persistent 파라미터 값을 ON으로 설정하는 것이 좋습니다.	innodb_stats_persistent 파라미터 값을 ON로 설정하세요.	No	AWS 데이터베이스 블로그의 Best practices for configuring performance parameters for Amazon RDS for MySQL

유형	설명	권장 사항	가동 중지 필요	추가 정보
innodb_open_files 파라미터가 낮습니다.	innodb_open_files 파라미터는 InnoDB가 한 번에 열 수 있는 파일 수를 제어합니다. InnoDB는 mysqld가 실행될 때 모든 로그 및 시스템 테이블스페이스 파일을 엽니다. InnoDB가 한 번에 열 수 있는 최대 파일 수에 대한 DB 인스턴스 값이 낮습니다. innodb_open_files 파라미터 값을 65로 설정하는 것이 좋습니다.	innodb_open_files 파라미터를 최소값인 65로 설정합니다.	예	InnoDB에서 MySQL용 파일 열기
max_user_connections 파라미터가 낮습니다.	DB 인스턴스의 각 데이터베이스 계정에 대한 최대 동시 연결 수 값이 낮습니다. max_user_connections 파라미터 설정을 5보다 큰 숫자로 늘리는 것이 좋습니다.	max_user_connections 파라미터 값을 5보다 큰 수로 늘리세요.	예	MySQL용 계정 리소스 제한 설정

유형	설명	권장 사항	가동 중지 필요	추가 정보
읽기 전용 복제본이 쓰기 가능 모드에서 열립니다.	DB 인스턴스에 쓰기 가능 모드의 읽기 전용 복제본이 있어 클라이언트의 업데이트를 허용합니다. 읽기 전용 복제본이 쓰기 가능 모드가 되지 않도록 `read_only` 파라미터를 `TrueIfReplica` 로 설정하는 것이 좋습니다.	`read_only` 파라미터 값을 `TrueIfReplica` 로 설정하세요.	No	AWS 데이터베이스 블로그의 Best practices for configuring replication parameters for Amazon RDS for MySQL
`innodb_default_row_format` 파라미터 설정이 안전하지 않습니다.	DB 인스턴스에 8.0.26보다 낮은 MySQL 버전에서 `row_format` 이 `COMPACT` 또는 `REDUNDANT` 로 설정된 표에서 인덱스가 767바이트를 초과한 경우 액세스할 수 없고 복구할 수 없는 알려진 문제가 발생했습니다. `innodb_default_row_format` 파라미터 값을 `DYNAMIC`으로 설정하는 것이 좋습니다.	`innodb_default_row_format` 파라미터 값을 `DYNAMIC`로 설정하세요.	No	MySQL 8.0.26의 변경 사항

유형	설명	권장 사항	가동 중지 필요	추가 정보
general_logging 파라미터가 활성화됨	DB 인스턴스에 대해 일반 로깅이 설정됩니다. 이 설정은 데이터베이스 문제를 해결하는 데 유용합니다. 그러나 일반 로깅을 활성화하면 I/O 작업량과 할당된 스토리지 공간이 늘어나 경합이 발생하고 성능이 저하될 수 있습니다. 일반 로깅 사용에 대한 요구 사항을 확인하세요. general_logging 파라미터 값을 0으로 설정하는 것이 좋습니다.	일반 로깅 사용에 대한 요구 사항을 확인하세요. 필수가 아닌 경우 general_logging 파라미터 값을 0으로 설정하는 것이 좋습니다.	No	RDS for MySQL 데이터베이스 로그 개요
시스템 메모리 용량에 비해 충분히 프로비저닝되지 않은 RDS 인스턴스	더 적은 메모리를 사용하거나 할당된 메모리가 더 많은 DB 인스턴스 유형을 사용하도록 쿼리를 조정하는 것이 좋습니다. 인스턴스의 메모리가 부족하면 데이터베이스 성능이 영향을 받습니다.	메모리 용량이 더 큰 DB 인스턴스 사용	예	AWS 데이터베이스 블로그의 Scaling Your Amazon RDS Instance Vertically and Horizontally Amazon RDS 인스턴스 유형 Amazon RDS 요금

유형	설명	권장 사항	가동 중지 필요	추가 정보
시스템 CPU 용량에 비해 충분히 프로비저닝되지 않은 RDS 인스턴스	CPU 사용량을 줄이도록 쿼리를 조정하거나 DB 인스턴스를 수정하여 더 높은 vCPU가 할당된 DB 인스턴스 클래스를 사용하도록 DB 인스턴스를 수정하는 것이 좋습니다. DB 인스턴스의 CPU가 부족하면 데이터베이스 성능이 저하될 수 있습니다.	CPU 용량이 더 큰 DB 인스턴스 사용	예	AWS 데이터베이스 블로그의 Scaling Your Amazon RDS Instance Vertically and Horizontally Amazon RDS 인스턴스 유형 Amazon RDS 요금
RDS 리소스가 연결 풀링을 제대로 활용하지 못하는 경우	Amazon RDS 프록시를 활성화하여 기존 데이터베이스 연결을 효율적으로 풀링하고 공유하는 것이 좋습니다. 이미 데이터베이스용 프록시를 사용하고 있다면 프록시를 올바르게 구성하여 여러 DB 인스턴스 간의 연결 풀링과 로드 밸런싱을 개선합니다. RDS 프록시는 가용성과 확장성을 개선하는 동시에 연결 고갈 및 가동 중지 시간의 위험을 줄이는 데 유용할 수 있습니다.	RDS 프록시를 활성화하거나 기존 프록시 구성을 수정합니다.	No	AWS 데이터베이스 블로그의 Scaling Your Amazon RDS Instance Vertically and Horizontally Amazon RDS Proxy Amazon RDS 프록시 요금

유형	설명	권장 사항	가동 중지 필요	추가 정보
RDS 인스턴스가 과도한 임시 객체를 생성하고 있음	과도한 임시 객체를 생성하지 않도록 워크로드를 조정하거나 최적화된 읽기를 지원하는 RDS 인스턴스 클래스로 전환하는 것이 좋습니다. RDS 최적화된 읽기는 많은 수의 임시 객체 및/또는 대규모 임시 객체가 포함된 워크로드의 데이터베이스 성능을 개선합니다. 워크로드를 평가하여 RDS 최적화된 읽기가 포함된 인스턴스를 사용하는 것이 데이터베이스 워크로드에 도움이 되는지 확인하세요.	RDS 최적화된 읽기와 함께 DB 인스턴스 유형 사용	예	Amazon RDS 인스턴스 유형 Amazon RDS 최적화된 읽기로 RDS for MySQL 쿼리 성능 개선 Amazon RDS 최적화된 읽기로 RDS for MariaDB 쿼리 성능 개선 Amazon RDS 최적화된 읽기로 RDS for PostgreSQL 쿼리 성능 개선
시스템 IOPS 용량에 비해 RDS 인스턴스가 충분히 프로비저닝되지 않았음	데이터베이스 워크로드를 조정하여 IOPS를 줄이거나 기본 IOPS 제한이 더 높은 유형으로 DB 인스턴스를 확장하는 것이 좋습니다. 현재 DB 인스턴스가 프로비저닝된 IOPS를 지원할 수 없거나 데이터베이스 워크로드의 IOPS 사용률이 높습니다.	기본 IOPS 한도가 더 높은 DB 인스턴스 유형 사용	예	Amazon RDS 인스턴스 유형 Amazon RDS DB 인스턴스 스토리지 데이터베이스 부하

유형	설명	권장 사항	가동 중지 필요	추가 정보
RDS 인스턴스의 Amazon EBS 볼륨 프로비저닝이 부족함	데이터베이스 워크로드를 조정하여 IOPS를 줄이거나 데이터베이스의 프로비저닝된 IOPS를 늘리는 것이 좋습니다. IOPS 사용률이 프로비저닝된 IOPS에 가까워지면 데이터베이스 성능이 저하될 수 있습니다.	DB 인스턴스에 대한 더 많은 IOPS 프로비저닝	예	Amazon RDS 인스턴스 유형 Amazon RDS DB 인스턴스 스토리지 데이터베이스 부하
처리량 용량에 비해 RDS 인스턴스가 충분히 프로비저닝되지 않았음	데이터베이스 워크로드를 조정하여 처리량을 줄이거나 데이터베이스의 프로비저닝된 처리량을 늘리는 것이 좋습니다. 처리량 사용률이 프로비저닝된 처리량에 가까워지면 데이터베이스 성능에 영향을 미칠 수 있습니다.	DB 인스턴스에 대한 더 많은 처리량 프로비저닝	예	Amazon RDS 인스턴스 유형 Amazon RDS DB 인스턴스 스토리지 데이터베이스 부하

유형	설명	권장 사항	가동 중지 필요	추가 정보
EBS I/O에 비해 RDS 인스턴스가 과소 프로비저닝됨	데이터베이스 워크로드를 조정하여 I/O 작업을 줄이거나 고성능, 높은 처리량, 짧은 지연 시간이 필요한 데이터베이스 워크로드용으로 설계된 Amazon RDS io2 Block Express 볼륨을 사용하도록 DB 인스턴스를 수정하는 것이 좋습니다. 현재 워크로드로는 데이터베이스가 필요한 속도로 I/O 작업을 처리하지 못해 성능이 저하될 수 있습니다.	RDS 인스턴스에 Amazon RDS io2 Block Express 볼륨 사용	No	Amazon RDS DB 인스턴스 스토리지 Amazon RDS에 대한 Amazon CloudWatch 지표 Amazon EBS 사용 설명서의 프로비저닝된 IOPS SSD 볼륨

Amazon RDS 콘솔에서 지표 보기

Amazon RDS는 Amazon CloudWatch와 통합되어 RDS 콘솔의 RDS DB 인스턴스 지표의 다양한 기능을 표시합니다. 이 지표에 대한 설명은 Amazon RDS용 지표 참조 섹션을 참조하세요.

DB 인스턴스의 경우 다음의 지표 범주가 모니터링됩니다.

- CloudWatch - RDS 콘솔에서 액세스할 수 있는 RDS에 대한 Amazon CloudWatch 지표를 보여줍니다. CloudWatch 콘솔에서 이러한 지표에 액세스할 수 있습니다. 각 지표에는 특정 시간대에서 지표를 모니터링한 그래프도 포함되어 있습니다. 전체 CloudWatch 지표 목록은 Amazon RDS에 대한 Amazon CloudWatch 지표 섹션을 참조하세요.
- 향상된 모니터링 - RDS DB 인스턴스가 향상된 모니터링을 활성화할 때 운영 체제 지표의 요약을 보여줍니다. RDS는 지표를 향상된 모니터링에서 Amazon CloudWatch Logs 계정으로 전달합니다. 각 지표에는 특정 시간대에서 지표를 모니터링한 그래프도 포함되어 있습니다. 개요는 Enhanced Monitoring을 사용하여 OS 지표 모니터링 섹션을 참조하세요. 향상된 모니터링 지표 목록은 향상된 모니터링의 OS 지표 섹션을 참조하세요.
- OS 프로세스 목록 – 인스턴스에서 실행되는 각 프로세스의 세부 정보를 표시합니다.
- 성능 개선 도우미(Performance Insights) – DB 인스턴스에 대한 Amazon RDS 성능 개선 도우미 대시보드를 엽니다. 성능 개선 도우미에 대한 개요는 성능 개선 도우미를 통한 Amazon RDS 모니터링 섹션을 참조하세요. Lambda Insights 지표 목록은 Amazon RDS 성능 개선 도우미를 위한 Amazon CloudWatch 지표 섹션을 참조하세요.

Amazon RDS는 이제 성능 개선 도우미 대시보드에서 성능 개선 도우미 및 CloudWatch 지표에 대한 통합 보기를 제공합니다. 이 보기를 사용하려면 DB 인스턴스에서 성능 개선 도우미가 켜져 있어야 합니다. 모니터링 탭에서 새 모니터링 보기를 선택하거나 탐색 창에서 성능 개선 도우미를 선택할 수 있습니다. 이 뷰를 선택하는 방법에 대한 지침을 보려면 성능 개선 도우미 대시보드와 결합된 지표 보기 섹션을 참조하세요.

성능 개선 도우미 대시보드와 결합된 지표 보기

Important

AWS는 Performance Insights의 수명 종료일을 2025년 11월 30일로 발표했습니다. 이 날짜 이후에는 Amazon RDS가 더 이상 Performance Insights 콘솔 환경, 유연한 보존 기간(1~24개월) 및 연결 요금을 지원하지 않습니다. Performance Insights API는 요금 변경 없이 계속 제공됩니다. Performance Insights API 비용은 CloudWatch Database Insights 비용과 함께 AWS 청구서에 표시됩니다.

Performance Insights의 유료 티어를 사용하는 DB 인스턴스를 2025년 11월 30일 이전에 Database Insights의 고급 모드로 업그레이드하는 것이 좋습니다. Database Insights의 고급 모드로 업그레이드하는 방법에 대한 자세한 내용은 Amazon RDS에 Database Insights의 고급 모드 켜기 섹션을 참조하세요.

조치를 취하지 않으면 Performance Insights를 사용하는 DB 인스턴스가 기본적으로 Database Insights의 표준 모드를 사용하게 됩니다. Database Insights의 표준 모드를 사용하는 경우 7일이 지난 성능 데이터 기록에 액세스할 수 없으며 Amazon RDS 콘솔에서 실행 계획 및 온디맨드 분석 기능을 사용하지 못할 수 있습니다. 2025년 11월 30일 이후에는 Database Insights의 고급 모드만 실행 계획과 온디맨드 분석을 지원합니다.

CloudWatch Database Insights를 사용하면 데이터베이스 플릿의 데이터베이스 로드를 모니터링하며 대규모로 성능을 분석하고 문제를 해결할 수 있습니다. Database Insights에 대한 자세한 내용은 CloudWatch Database Insights를 사용하여 Amazon RDS 데이터베이스 모니터링 섹션을 참조하세요. 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

Amazon RDS는 성능 개선 도우미 대시보드에서 DB 인스턴스에 대한 성능 개선 도우미 및 CloudWatch 지표에 대한 통합 보기를 제공합니다. 사전 구성된 대시보드를 사용하거나 사용자 지정 대시보드를 만들 수 있습니다. 사전 구성된 대시보드는 데이터베이스 엔진의 성능 문제를 진단하는 데 도움이 되는 가장 일반적으로 사용되는 지표를 제공합니다. 또는 분석 요구 사항을 충족하는 데이터베이스 엔진 지표로 사용자 지정 대시보드를 만들 수 있습니다. 그런 다음 AWS 계정에 있는 해당 데이터베이스 엔진 유형의 모든 DB 인스턴스에 대해 이 대시보드를 사용하세요.

모니터링 탭에서 모니터링 보기를 선택하거나 탐색 창에서 성능 개선 도우미를 선택할 수 있습니다.

성능 개선 도우미 대시보드에서 결합 지표를 보려면 DB 인스턴스에 대해 성능 개선 도우미를 켜야 합니다. 성능 개선 도우미 켜기에 대한 자세한 내용은 Amazon RDS의 성능 개선 도우미 설정 및 해제 섹션을 참조하세요.

다음 섹션에서는 성능 개선 도우미 및 CloudWatch 지표를 표시하는 방법을 배워볼 수 있습니다.

주제

- 모니터링 탭에서 새 모니터링 보기 선택
- 성능 개선 도우미 페이지에서 새 모니터링 보기 선택
- 성능 개선 도우미를 사용하여 사용자 지정 대시보드 생성
- 성능 개선 도우미를 사용하여 사전 구성된 대시보드 선택

모니터링 탭에서 새 모니터링 보기 선택

Important

AWS는 Performance Insights의 수명 종료일을 2025년 11월 30일로 발표했습니다. 이 날짜 이후에는 Amazon RDS가 더 이상 Performance Insights 콘솔 환경, 유연한 보존 기간(1~24개월) 및 연결 요금을 지원하지 않습니다. Performance Insights API는 요금 변경 없이 계속 제공됩니다. Performance Insights API 비용은 CloudWatch Database Insights 비용과 함께 AWS 청구서에 표시됩니다.

Performance Insights의 유료 티어를 사용하는 DB 인스턴스를 2025년 11월 30일 이전에 Database Insights의 고급 모드로 업그레이드하는 것이 좋습니다. Database Insights의 고급 모드로 업그레이드하는 방법에 대한 자세한 내용은 Amazon RDS에 Database Insights의 고급 모드 켜기 섹션을 참조하세요.

조치를 취하지 않으면 Performance Insights를 사용하는 DB 인스턴스가 기본적으로 Database Insights의 표준 모드를 사용하게 됩니다. Database Insights의 표준 모드를 사용하는 경우 7일이 지난 성능 데이터 기록에 액세스할 수 없으며 Amazon RDS 콘솔에서 실행 계획 및 온디맨드 분석 기능을 사용하지 못할 수 있습니다. 2025년 11월 30일 이후에는 Database Insights의 고급 모드만 실행 계획과 온디맨드 분석을 지원합니다.

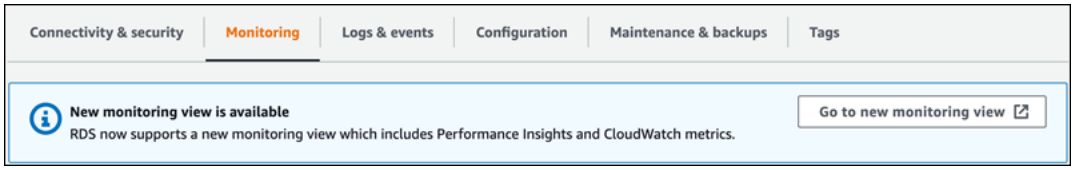
CloudWatch Database Insights를 사용하면 데이터베이스 플릿의 데이터베이스 로드를 모니터링하며 대규모로 성능을 분석하고 문제를 해결할 수 있습니다. Database Insights에 대한 자세한 내용은 CloudWatch Database Insights를 사용하여 Amazon RDS 데이터베이스 모니터링 섹션을 참조하세요. 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

Amazon RDS 콘솔에서 새 모니터링 보기를 선택하여 DB 인스턴스에 대한 성능 개선 도우미 및 CloudWatch 지표를 볼 수 있습니다.

모니터링 탭에서 새 모니터링 보기를 선택하려면

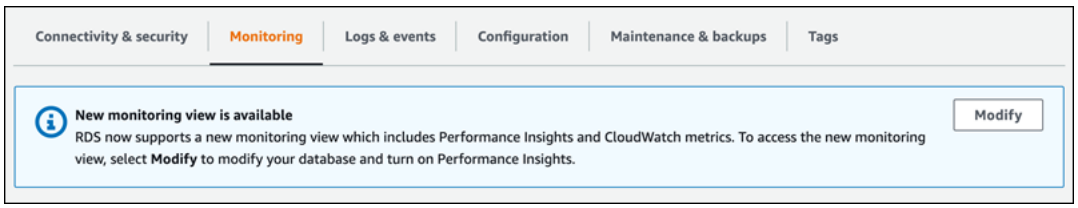
1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 데이터베이스를 선택합니다.
3. 모니터링하려는 DB 인스턴스를 선택합니다.
4. 아래로 스크롤하여 모니터링 탭을 선택합니다.

   새 모니터링 보기를 선택할 수 있는 옵션이 포함된 배너가 나타납니다. 다음 예제에서는 새 모니터링 보기를 선택하는 배너를 보여 줍니다.



5. 새 모니터링 보기로 이동을 선택하여 에 대한 성능 개선 도우미 및 CloudWatch 지표가 포함된 성능 개선 도우미 대시보드를 엽니다.
6. (선택 사항) DB 인스턴스에 대해 성능 개선 도우미가 꺼져 있는 경우 DB 클러스터를 수정하고 성능 개선 도우미를 켤 수 있는 옵션이 포함된 배너가 나타납니다.

   다음 예제는 모니터링 탭의 DB 클러스터를 수정하기 위한 배너를 보여 줍니다.



   수정을 선택하여 DB 클러스터를 수정하고 성능 개선 도우미를 켭니다. 성능 개선 도우미 켜기에 대한 자세한 내용은 Amazon RDS의 성능 개선 도우미 설정 및 해제 섹션을 참조하세요.

성능 개선 도우미 페이지에서 새 모니터링 보기 선택

Important

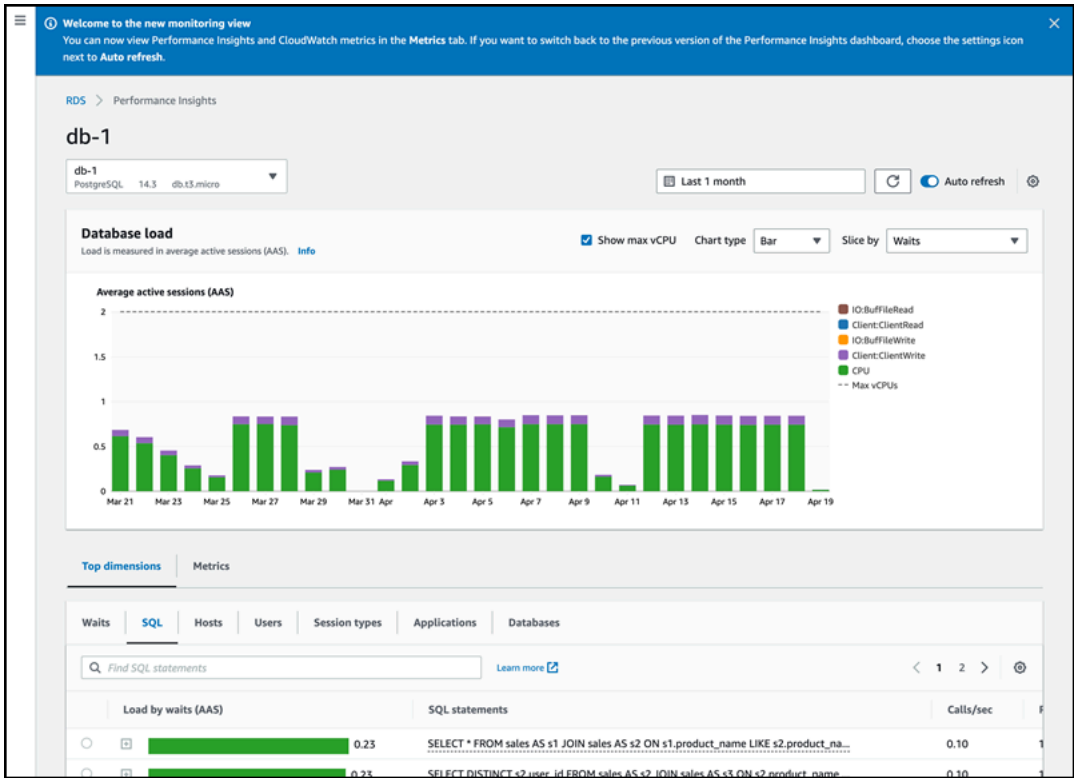
AWS는 Performance Insights의 수명 종료일을 2025년 11월 30일로 발표했습니다. 이 날짜 이후에는 Amazon RDS가 더 이상 Performance Insights 콘솔 환경, 유연한 보존 기간(1~24개월) 및 연결 요금을 지원하지 않습니다. Performance Insights API는 요금 변경 없이 계속 제공됩니

다. Performance Insights API 비용은 CloudWatch Database Insights 비용과 함께 AWS 청구서에 표시됩니다.
Performance Insights의 유료 티어를 사용하는 DB 인스턴스를 2025년 11월 30일 이전에 Database Insights의 고급 모드로 업그레이드하는 것이 좋습니다. Database Insights의 고급 모드로 업그레이드하는 방법에 대한 자세한 내용은 Amazon RDS에 Database Insights의 고급 모드 켜기 섹션을 참조하세요.
조치를 취하지 않으면 Performance Insights를 사용하는 DB 인스턴스가 기본적으로 Database Insights의 표준 모드를 사용하게 됩니다. Database Insights의 표준 모드를 사용하는 경우 7일이 지난 성능 데이터 기록에 액세스할 수 없으며 Amazon RDS 콘솔에서 실행 계획 및 온디맨드 분석 기능을 사용하지 못할 수 있습니다. 2025년 11월 30일 이후에는 Database Insights의 고급 모드만 실행 계획과 온디맨드 분석을 지원합니다.
CloudWatch Database Insights를 사용하면 데이터베이스 플릿의 데이터베이스 로드를 모니터링하며 대규모로 성능을 분석하고 문제를 해결할 수 있습니다. Database Insights에 대한 자세한 내용은 CloudWatch Database Insights를 사용하여 Amazon RDS 데이터베이스 모니터링 섹션을 참조하세요. 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

Amazon RDS 콘솔에서 새 모니터링 보기를 선택하여 DB 인스턴스에 대한 성능 개선 도우미 및 CloudWatch 지표를 볼 수 있습니다.

탐색 창의 성능 개선 도우미를 사용하여 새 모니터링 보기를 선택하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 성능 개선 도우미(Performance Insights)를 선택합니다.
3. DB 인스턴스를 선택하면 DB 인스턴스에 대한 성능 개선 도우미 및 CloudWatch 지표가 모두 표시되는 성능 개선 도우미 대시보드를 볼 수 있습니다.



성능 개선 도우미를 사용하여 사용자 지정 대시보드 생성

Important

AWS는 Performance Insights의 수명 종료일을 2025년 11월 30일로 발표했습니다. 이 날짜 이후에는 Amazon RDS가 더 이상 Performance Insights 콘솔 환경, 유연한 보존 기간(1~24개월) 및 연결 요금을 지원하지 않습니다. Performance Insights API는 요금 변경 없이 계속 제공됩니다. Performance Insights API 비용은 CloudWatch Database Insights 비용과 함께 AWS 청구서에 표시됩니다.

Performance Insights의 유료 티어를 사용하는 DB 인스턴스를 2025년 11월 30일 이전에 Database Insights의 고급 모드로 업그레이드하는 것이 좋습니다. Database Insights의 고급 모드로 업그레이드하는 방법에 대한 자세한 내용은 Amazon RDS에 Database Insights의 고급 모드 켜기 섹션을 참조하세요.

조치를 취하지 않으면 Performance Insights를 사용하는 DB 인스턴스가 기본적으로 Database Insights의 표준 모드를 사용하게 됩니다. Database Insights의 표준 모드를 사용하는 경우 7일이 지난 성능 데이터 기록에 액세스할 수 없으며 Amazon RDS 콘솔에서 실행 계획 및 온디맨드 분석 기능을 사용하지 못할 수 있습니다. 2025년 11월 30일 이후에는 Database Insights의 고급 모드만 실행 계획과 온디맨드 분석을 지원합니다.

CloudWatch Database Insights를 사용하면 데이터베이스 플릿의 데이터베이스 로드를 모니터링하며 대규모로 성능을 분석하고 문제를 해결할 수 있습니다. Database Insights에 대한 자세한 내용은 CloudWatch Database Insights를 사용하여 Amazon RDS 데이터베이스 모니터링 섹션을 참조하세요. 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

새로운 모니터링 보기에서 분석 요구 사항을 충족하는 데 필요한 지표가 포함된 사용자 지정 대시보드를 만들 수 있습니다.

DB 인스턴스에 대한 성능 개선 도우미 및 CloudWatch 지표를 선택하여 사용자 지정 대시보드를 만들 수 있습니다. AWS 계정에 있는 동일한 데이터베이스 엔진 유형의 다른 DB 인스턴스에 대해 이 대시보드를 사용하세요.

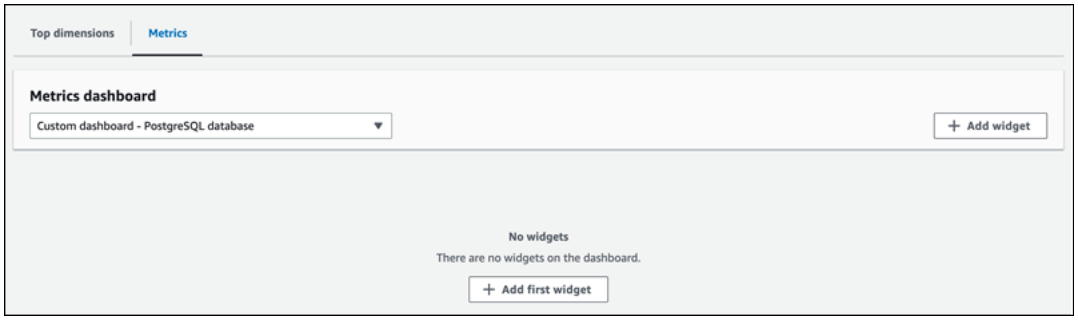
Note

사용자 지정 대시보드는 최대 50개의 지표를 지원합니다.

위젯 설정 메뉴를 사용하여 대시보드를 편집 또는 삭제하고 위젯 창을 이동하거나 규모를 조정합니다.

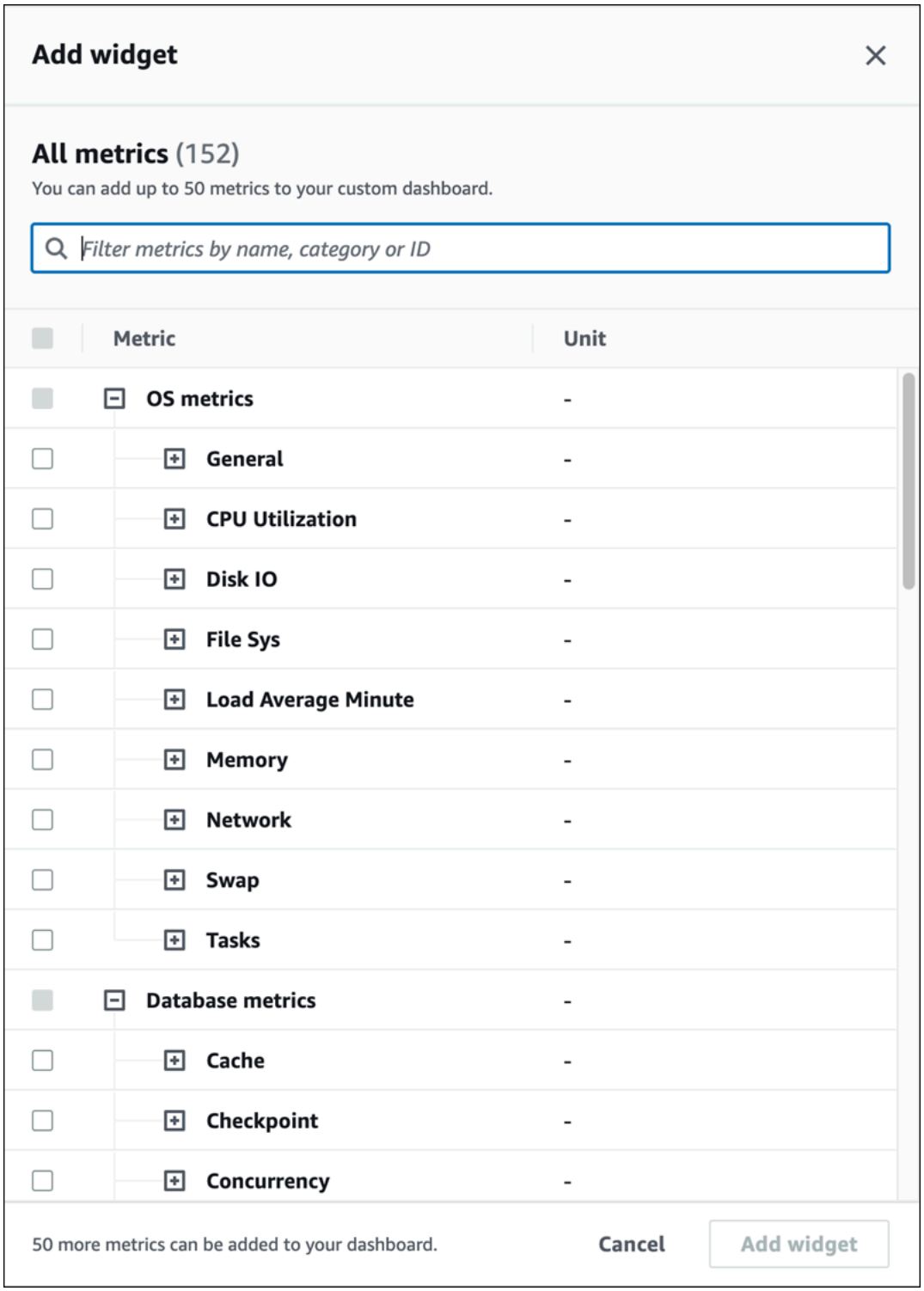
탐색 창의 성능 개선 도우미를 사용하여 사용자 지정 대시보드를 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 성능 개선 도우미(Performance Insights)를 선택합니다.
3. DB 인스턴스를 선택합니다.
4. 창의 지표 탭으로 스크롤을 내립니다.
5. 드롭다운 목록에서 사용자 지정 대시보드를 선택합니다. 다음 예제에서는 사용자 지정 대시보드 생성을 보여 줍니다.



6. 위젯 추가를 선택하여 위젯 추가 창을 엽니다. 창에서 사용 가능한 운영 체제(OS) 지표, 데이터베이스 지표 및 CloudWatch 지표를 열고 볼 수 있습니다.

   다음 예제는 지표가 포함된 위젯 추가 창을 보여 줍니다.



7. 대시보드에서 보려는 지표를 선택하고 위젯 추가를 선택합니다. 검색 필드를 사용하여 특정 지표를 찾을 수 있습니다.

   선택한 지표가 대시보드에 표시됩니다.
8. (선택 사항) 대시보드를 수정하거나 삭제하려면 위젯의 오른쪽 상단에 있는 설정 아이콘을 선택한 다음 메뉴에서 다음 작업 중 하나를 선택합니다.

   - 편집 – 창에서 지표 목록을 수정합니다. 대시보드의 지표를 선택한 후 위젯 업데이트를 선택합니다.
   - 삭제 – 위젯을 삭제합니다. 확인 창에서 삭제를 선택합니다.

성능 개선 도우미를 사용하여 사전 구성된 대시보드 선택

Important

AWS는 Performance Insights의 수명 종료일을 2025년 11월 30일로 발표했습니다. 이 날짜 이후에는 Amazon RDS가 더 이상 Performance Insights 콘솔 환경, 유연한 보존 기간(1~24개월) 및 연결 요금을 지원하지 않습니다. Performance Insights API는 요금 변경 없이 계속 제공됩니다. Performance Insights API 비용은 CloudWatch Database Insights 비용과 함께 AWS 청구서에 표시됩니다.
Performance Insights의 유료 티어를 사용하는 DB 인스턴스를 2025년 11월 30일 이전에 Database Insights의 고급 모드로 업그레이드하는 것이 좋습니다. Database Insights의 고급 모드로 업그레이드하는 방법에 대한 자세한 내용은 Amazon RDS에 Database Insights의 고급 모드 켜기 섹션을 참조하세요.
조치를 취하지 않으면 Performance Insights를 사용하는 DB 인스턴스가 기본적으로 Database Insights의 표준 모드를 사용하게 됩니다. Database Insights의 표준 모드를 사용하는 경우 7일이 지난 성능 데이터 기록에 액세스할 수 없으며 Amazon RDS 콘솔에서 실행 계획 및 온디맨드 분석 기능을 사용하지 못할 수 있습니다. 2025년 11월 30일 이후에는 Database Insights의 고급 모드만 실행 계획과 온디맨드 분석을 지원합니다.
CloudWatch Database Insights를 사용하면 데이터베이스 플릿의 데이터베이스 로드를 모니터링하며 대규모로 성능을 분석하고 문제를 해결할 수 있습니다. Database Insights에 대한 자세한 내용은 CloudWatch Database Insights를 사용하여 Amazon RDS 데이터베이스 모니터링 섹션을 참조하세요. 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

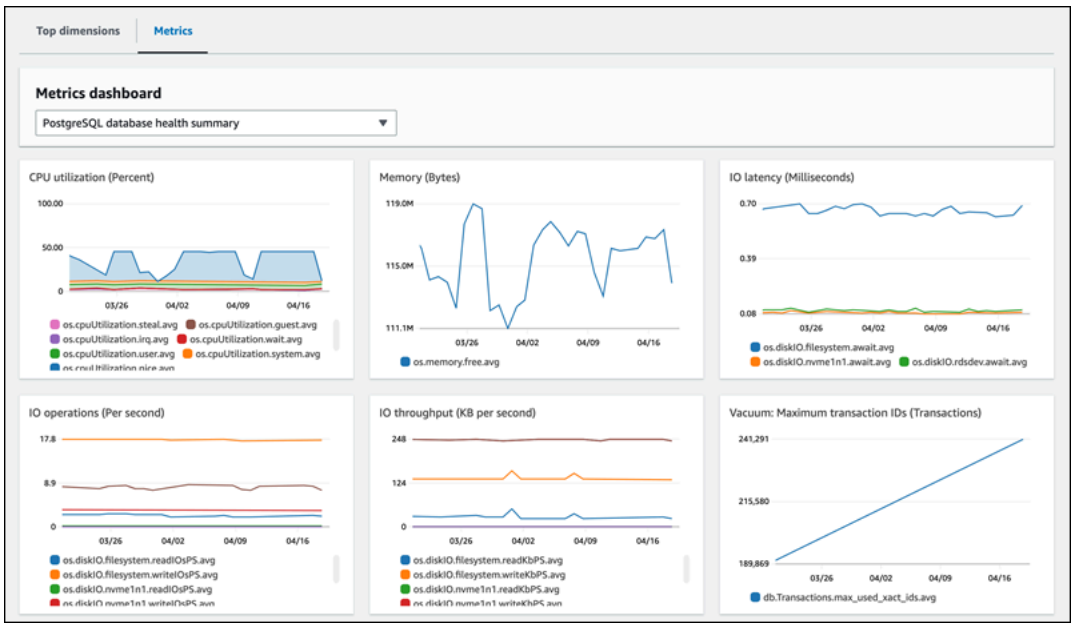
사전 구성된 대시보드에서 자주 사용하는 지표를 볼 수 있습니다. 이 대시보드는 데이터베이스 엔진의 성능 문제를 진단하고 평균 복구 시간을 몇 시간에서 몇 분으로 줄이는 데 도움이 됩니다.

> **Note**
>
> 이 대시보드는 편집할 수 없습니다.

탐색 창의 성능 개선 도우미를 사용하여 사전 구성된 대시보드를 선택하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 성능 개선 도우미(Performance Insights)를 선택합니다.
3. DB 인스턴스를 선택합니다.
4. 창의 지표 탭으로 스크롤을 내립니다.
5. 드롭다운 목록에서 사전 구성된 대시보드를 선택합니다.

   대시보드에서 DB 인스턴스에 대한 지표를 볼 수 있습니다. 다음 예제에서는 사전 구성된 지표 대시보드를 보여 줍니다.



Amazon CloudWatch로 Amazon RDS 지표 모니터링

Amazon CloudWatch는 지표 리포지토리입니다. 리포지토리는 Amazon RDS에서 원시 데이터를 수집한 후 판독이 가능한 지표로 실시간에 가깝게 처리합니다. CloudWatch로 전송되는 Amazon RDS 지표의 전체 목록은 Amazon RDS용 지표 참조를 참조하세요.

데이터베이스의 성능을 대규모로 분석하고 문제를 해결하려면 CloudWatch Database Insights를 사용합니다.

주제

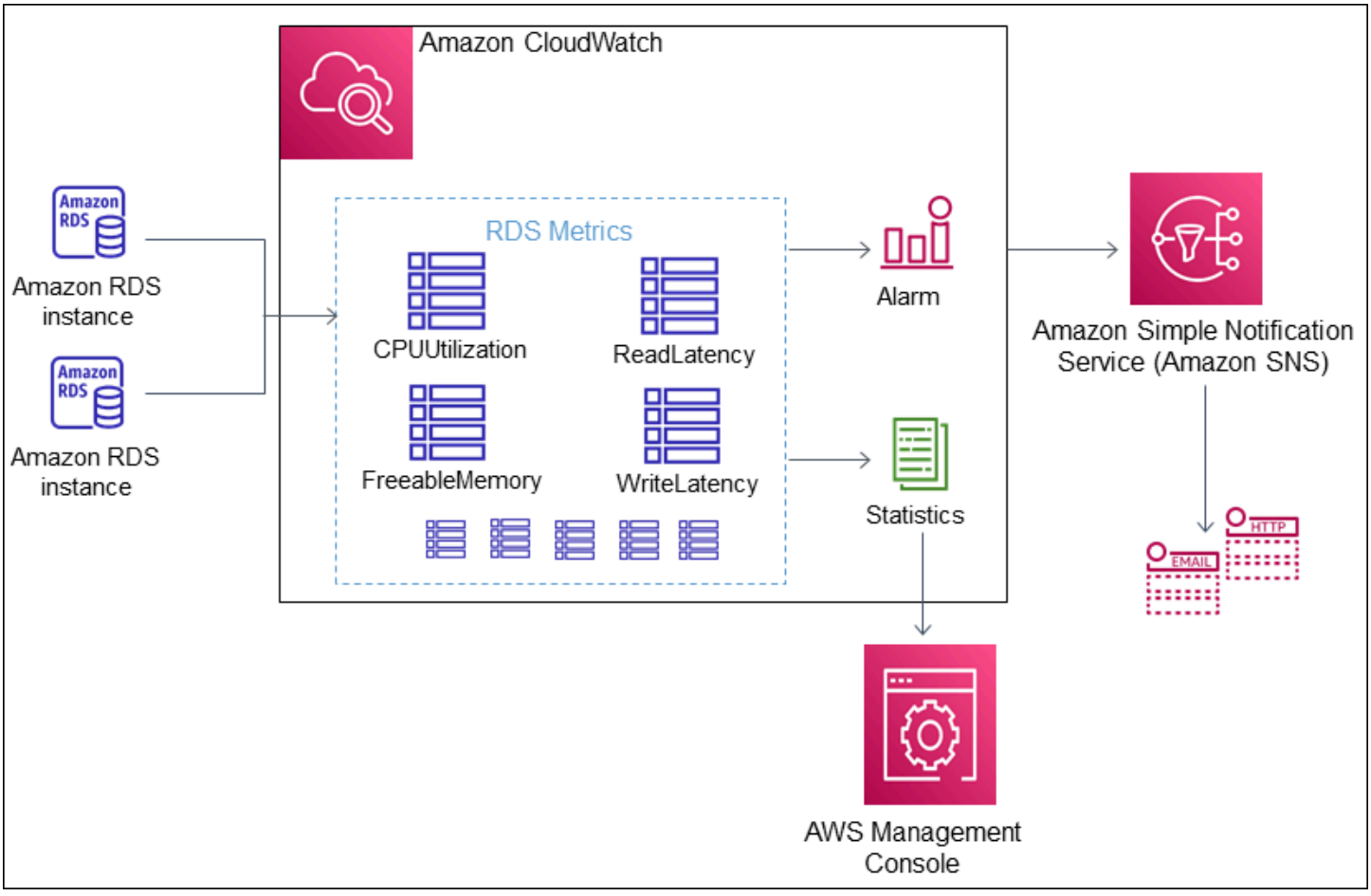
- Amazon RDS 및 Amazon CloudWatch 개요
- CloudWatch 콘솔 및 AWS CLI에서 DB 인스턴스 지표 보기
- CloudWatch에 성능 개선 도우미 지표 내보내기
- Amazon RDS을 모니터링하는 CloudWatch 경보 생성
- 자습서: Amazon RDS에 대한 다중 AZ DB 클러스터 복제본 지연에 대한 Amazon CloudWatch 경보 생성

Amazon RDS 및 Amazon CloudWatch 개요

Amazon RDS는 기본적으로 1분 단위로 CloudWatch에 지표 데이터를 자동 전송합니다. 예를 들어 `CPUUtilization` 지표는 시간 경과에 따른 DB 인스턴스의 CPU 사용률을 기록합니다. 기간이 60초(1분)로 설정된 데이터 요소들은 15일 동안 사용할 수 있습니다. 즉, 기록 정보에 액세스하고 웹 애플리케이션 또는 서비스가 어떻게 실행되고 있는지 살펴볼 수 있습니다.

이제 성능 개선 도우미 지표 대시보드를 Amazon RDS에서 Amazon CloudWatch로 내보낼 수 있습니다. 사전 구성된 지표 대시보드 또는 사용자 지정 지표 대시보드를 새 대시보드로 내보내거나 기존 CloudWatch 대시보드에 추가할 수 있습니다. 내보낸 대시보드를 CloudWatch 콘솔에서 볼 수 있습니다. 성능 개선 도우미 지표 대시보드를 CloudWatch로 내보내는 방법에 대한 자세한 내용은 CloudWatch에 성능 개선 도우미 지표 내보내기 섹션을 참조하세요.

다음 다이어그램에서 볼 수 있듯이 CloudWatch 지표에 대해 경보를 설정할 수 있습니다. 예를 들어 인스턴스의 CPU 사용률이 70% 이상일 때 알리는 경보를 생성할 수 있습니다. 임계값을 초과하면 이메일을 보내도록 Amazon Simple Notification Service를 구성할 수 있습니다.



Amazon RDS는 다음과 같은 유형의 지표를 Amazon CloudWatch에 게시합니다.

- RDS DB 인스턴스에 대한 지표

  이들 지표에 대한 표는 Amazon RDS에 대한 Amazon CloudWatch 지표 단원을 참조하세요.
- 성능 개선 도우미 지표

  이들 지표에 대한 표는 Amazon RDS 성능 개선 도우미를 위한 Amazon CloudWatch 지표 및 성능 개선 도우미 카운터 단원을 참조하세요.
- 향상된 모니터링 지표(Amazon CloudWatch Logs에 게시됨)

  이들 지표에 대한 표는 향상된 모니터링의 OS 지표 단원을 참조하세요.
- 사용자 AWS 계정의 Amazon RDS 서비스 할당량에 대한 사용량 지표

  이들 지표에 대한 표는 Amazon RDS에 대한 Amazon CloudWatch 사용량 지표 단원을 참조하세요. Amazon RDS 할당량에 대한 자세한 내용은 Amazon RDS에 대한 할당량 및 제약 조건 단원을 참조하세요.

CloudWatch에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서에서 Amazon CloudWatch란 무엇입니까?를 참조하십시오. CloudWatch 지표 보존 기간에 대한 자세한 내용은 지표 보존 기간을 참조하세요.

CloudWatch 콘솔 및 AWS CLI에서 DB 인스턴스 지표 보기

다음은 CloudWatch를 사용하여 DB 인스턴스에 대한 지표를 보는 방법에 대한 세부 정보입니다. CloudWatch Logs을 사용하여 DB 인스턴스의 운영 체제에 대한 지표를 실시간으로 모니터링하는 방법에 대한 자세한 내용은 Enhanced Monitoring을 사용하여 OS 지표 모니터링 단원을 참조하십시오.

Amazon RDS 리소스를 사용할 때 Amazon RDS는 1분마다 지표와 측정기준을 Amazon CloudWatch로 전송합니다.

이제 Amazon RDS에서 Amazon CloudWatch로 성능 개선 도우미 지표 대시보드를 내보내고 CloudWatch 콘솔에서 해당 지표를 볼 수 있습니다. 성능 개선 도우미 지표 대시보드를 CloudWatch로 내보내는 방법에 대한 자세한 내용은 CloudWatch에 성능 개선 도우미 지표 내보내기 섹션을 참조하세요.

다음 절차에 따라 CloudWatch 콘솔 및 CLI에서 Amazon RDS에 대한 지표를 확인합니다.

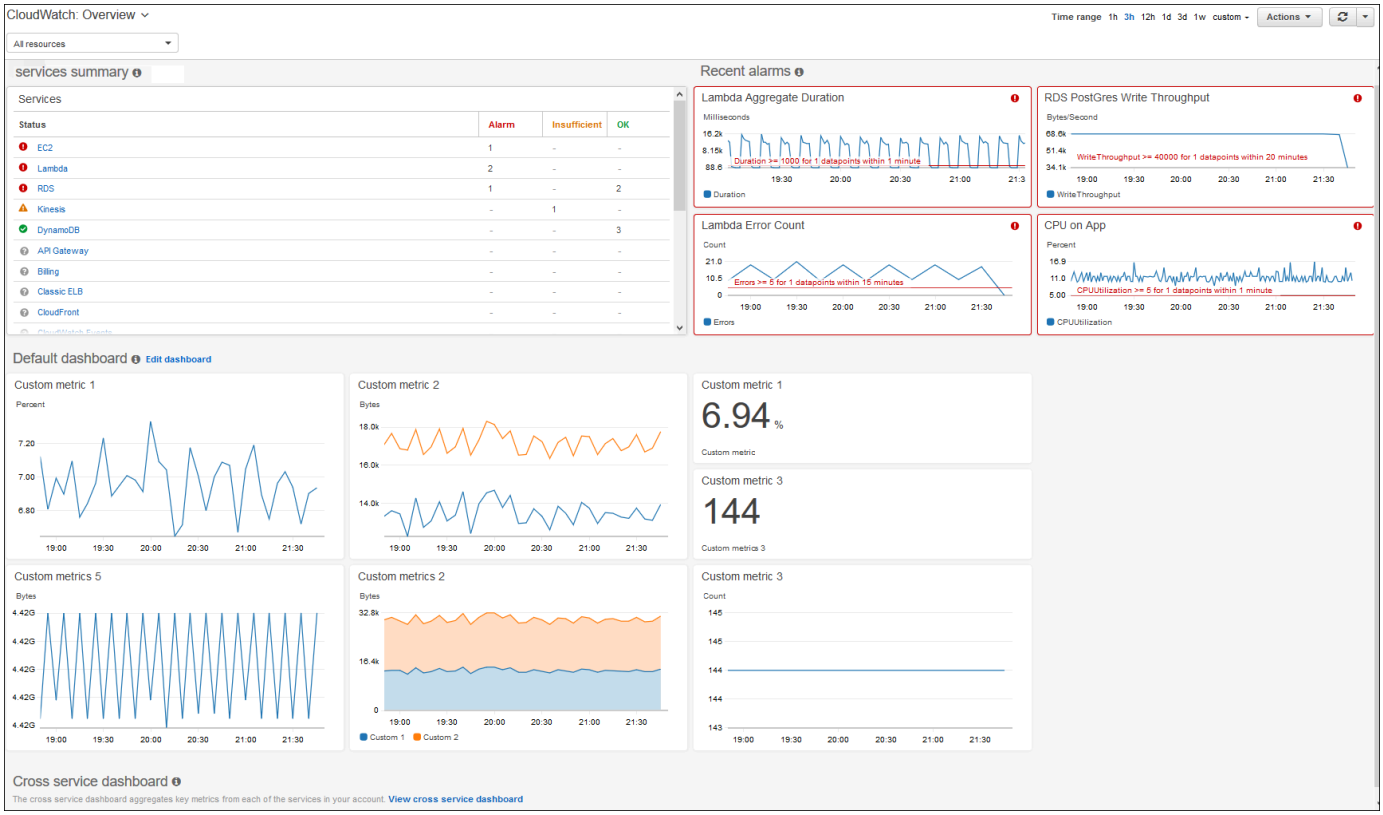
콘솔

Amazon CloudWatch 콘솔을 사용한 메트릭 확인

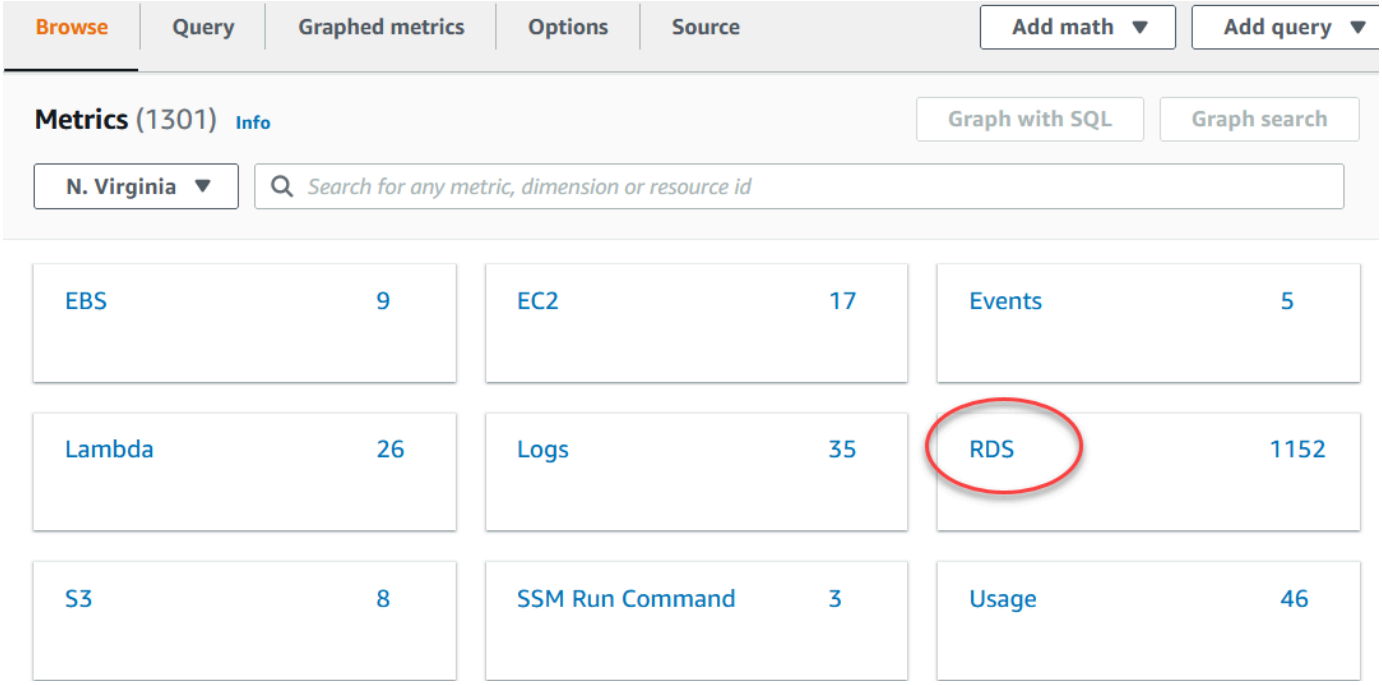
측정치는 먼저 서비스 네임스페이스별로 그룹화된 다음, 각 네임스페이스 내에서 다양한 차원 조합별로 그룹화됩니다.

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.

   CloudWatch 개요 홈페이지가 표시됩니다.

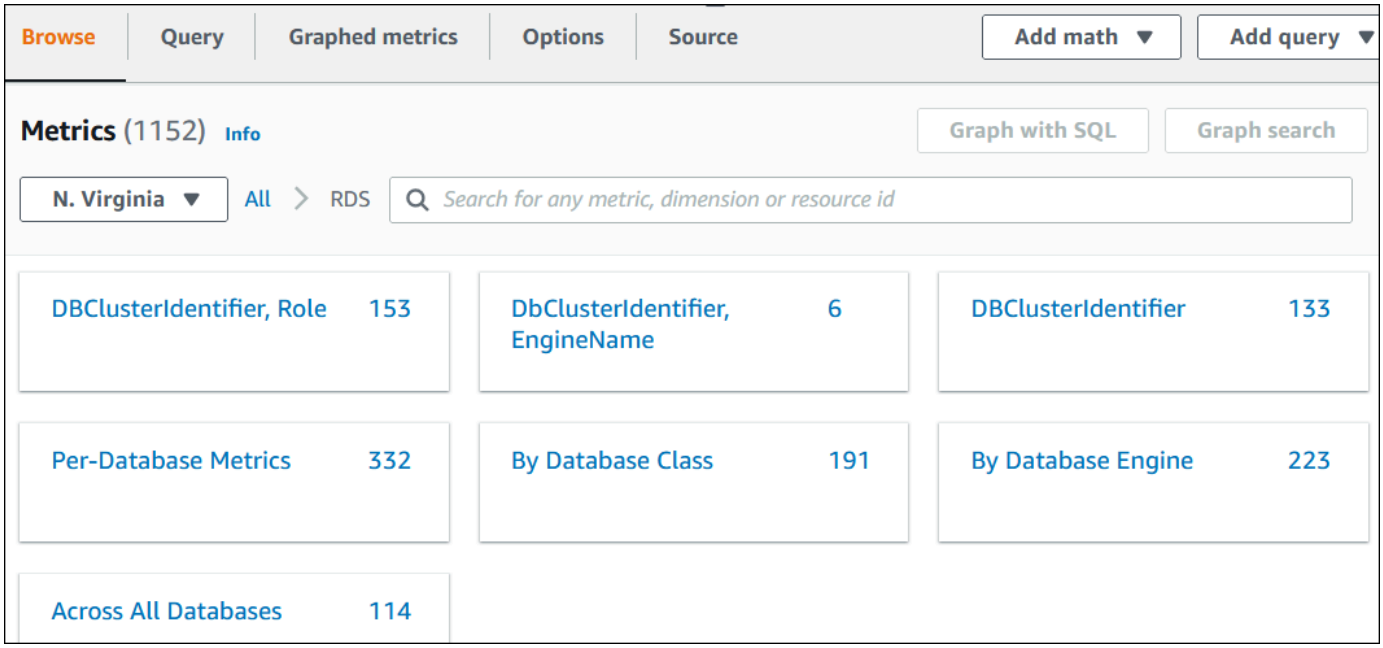


2. 필요한 경우 AWS 리전을 변경합니다. 탐색 모음에서 AWS 리전 리소스가 있는 AWS을 선택합니다. 자세한 내용은 리전 및 엔드포인트 섹션을 참조하십시오.
3. 탐색 창에서 지표(Metrics)를 선택한 다음 모든 지표(All metrics)를 선택합니다.

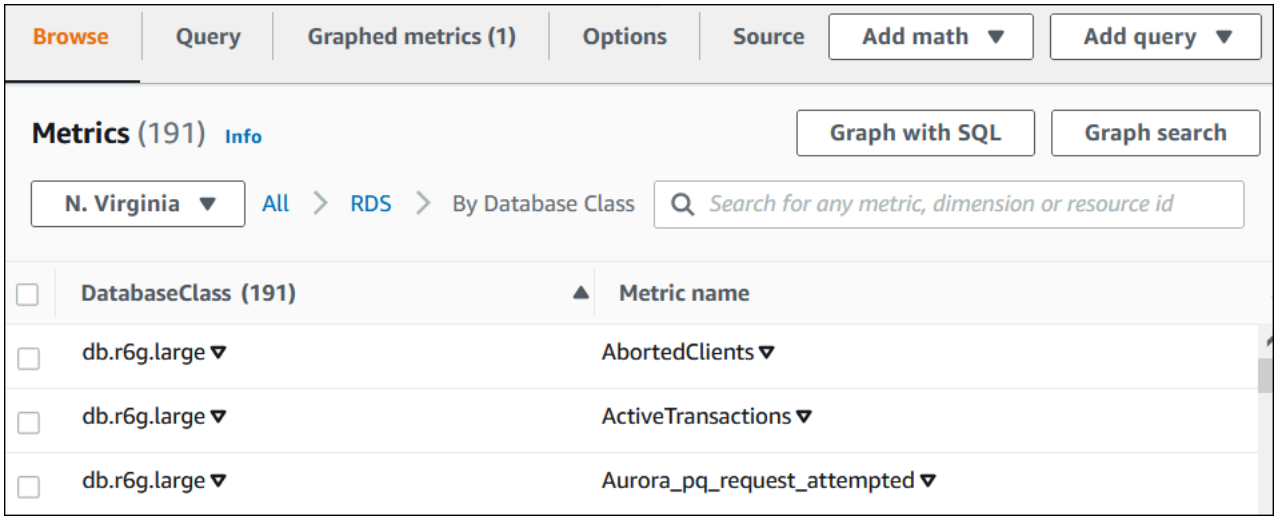


4. 아래로 스크롤하여 RDS 지표 네임스페이스를 선택합니다.

   페이지에 Amazon RDS 측정기준이 표시됩니다. 이러한 측정기준에 대한 설명은 Amazon RDS에 대한 Amazon CloudWatch 측정기준 목록 섹션을 참조하세요.



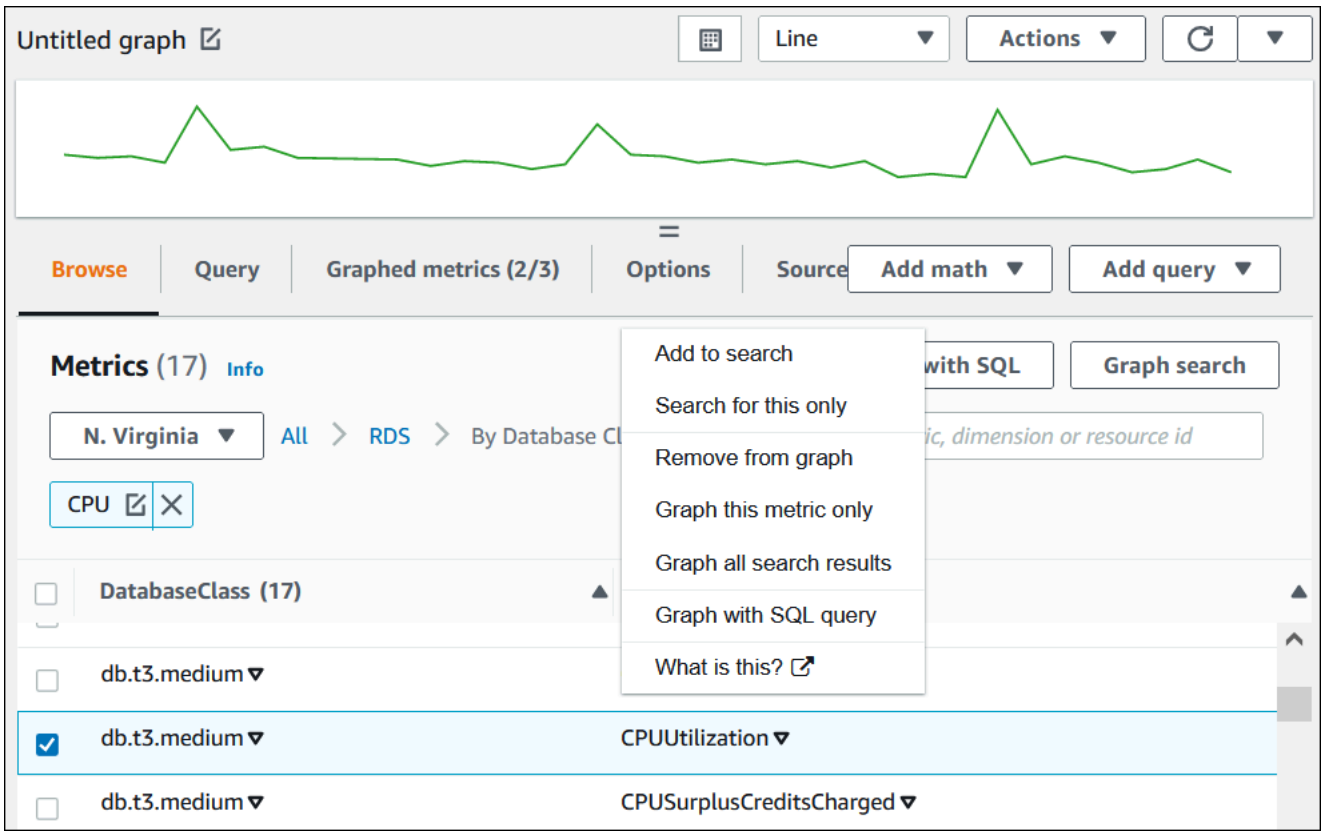
5. 측정치 차원(예: 데이터베이스 클래스별)을 선택합니다.



6. 다음 작업을 수행합니다.

   - 지표를 정렬하려면 열 머리글을 사용합니다.
   - 측정치를 그래프로 표시하려면 측정치 옆에 있는 확인란을 선택합니다.
   - 리소스로 필터링하려면 리소스 ID를 선택한 후 검색에 추가를 선택합니다.
   - 지표로 필터링하려면 지표 이름을 선택한 후 검색에 추가를 선택합니다.

   다음 예제는 db.t3.medium 클래스를 필터링하고 CPUUtilization 지표를 그래프로 나타냅니다.



AWS CLI

AWS CLI를 사용하여 지표 정보를 얻으려면 CloudWatch 명령 `list-metrics`를 사용합니다. 다음 예에서는 AWS/RDS 네임스페이스의 모든 지표를 나열합니다.

```
aws cloudwatch list-metrics --namespace AWS/RDS
```

지표 데이터를 얻으려면 `get-metric-data` 명령을 사용합니다.

다음 예제는 5분 단위로 특정 24시간 동안의 my-instance 인스턴스에 대한 CPUUtilization 통계를 가져옵니다.

다음 콘텐츠를 포함하는 CPU_metric.json JSON 파일을 생성합니다.

```
{
   "StartTime" : "2023-12-25T00:00:00Z",
   "EndTime" : "2023-12-26T00:00:00Z",
   "MetricDataQueries" : [{
     "Id" : "cpu",
     "MetricStat" : {
```

```
    "Metric" : {
        "Namespace" : "AWS/RDS",
        "MetricName" : "CPUUtilization",
        "Dimensions" : [{ "Name" : "DBInstanceIdentifier" , "Value" : my-instance}]
    },
       "Period" : 360,
       "Stat" : "Minimum"
     }
   }]
}
```

Example

대상 LinuxmacOS, 또는Unix:

```
aws cloudwatch get-metric-data \
    --cli-input-json file://CPU_metric.json
```

Windows의 경우:

```
aws cloudwatch get-metric-data ^
     --cli-input-json file://CPU_metric.json
```

샘플 출력은 다음과 같이 나타납니다.

```
{
    "MetricDataResults": [
        {
            "Id": "cpu",
            "Label": "CPUUtilization",
            "Timestamps": [
                "2023-12-15T23:48:00+00:00",
                "2023-12-15T23:42:00+00:00",
                "2023-12-15T23:30:00+00:00",
                "2023-12-15T23:24:00+00:00",
                ...
            ],
            "Values": [
                13.299778337027714,
                13.677507543049558,
                14.24976250395827,
                13.02521708695145,
```

```
                ...
            ],
            "StatusCode": "Complete"
        }
    ],
    "Messages": []
}
```

자세한 내용은 Amazon CloudWatch 사용 설명서의 지표에 대한 통계 얻기를 참조하세요.

CloudWatch에 성능 개선 도우미 지표 내보내기

성능 개선 도우미를 사용하면 DB 인스턴스에 대한 사전 구성된 지표 또는 사용자 지정 지표 대시보드를 Amazon CloudWatch로 내보낼 수 있습니다. 지표 대시보드를 새 대시보드로 내보내거나 기존 CloudWatch 대시보드에 추가할 수 있습니다. 기존 CloudWatch 대시보드에 대시보드를 추가하는 경우 지표가 CloudWatch 대시보드의 별도 섹션에 표시되도록 헤더 레이블을 만들 수 있습니다.

내보낸 지표 대시보드를 CloudWatch 콘솔에서 볼 수 있습니다. 성능 개선 도우미 지표 대시보드를 내보낸 후 새 지표를 추가하는 경우 이 대시보드를 다시 내보내야 CloudWatch 콘솔에서 새 지표를 볼 수 있습니다.

성능 개선 도우미 대시보드에서 지표 위젯을 선택하고 CloudWatch 콘솔에서 지표 데이터를 볼 수도 있습니다.

CloudWatch 콘솔에서 지표를 보는 방법에 대한 자세한 내용은 CloudWatch 콘솔 및 AWS CLI에서 DB 인스턴스 지표 보기 섹션을 참조하세요.

다음 섹션에서는 성능 개선 도우미 지표를 CloudWatch에 신규 또는 기존 대시보드로 내보내고 CloudWatch에서 성능 개선 도우미 지표를 확인합니다.

주제

- 성능 개선 도우미 지표를 CloudWatch에 새 대시보드로 내보내기
- 기존 CloudWatch 대시보드에 성능 개선 도우미 지표 추가
- CloudWatch에서 성능 개선 도우미 지표 위젯 보기

성능 개선 도우미 지표를 CloudWatch에 새 대시보드로 내보내기

성능 개선 도우미 대시보드에서 사전 구성된 지표 대시보드 또는 사용자 지정 지표 대시보드를 선택하고 CloudWatch에 새 대시보드로 내보냅니다. 내보낸 대시보드를 CloudWatch 콘솔에서 볼 수 있습니다.

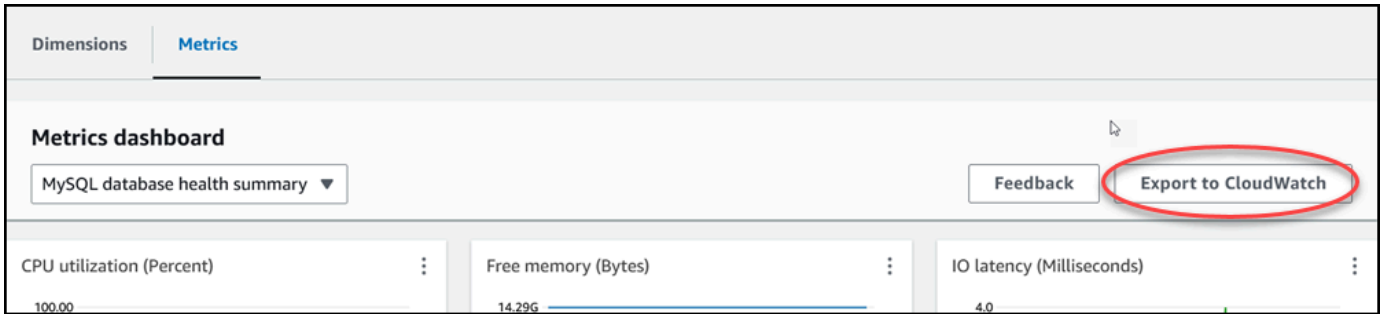
성능 개선 도우미 지표 대시보드를 CloudWatch에 새 대시보드로 내보내는 방법

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 성능 개선 도우미(Performance Insights)를 선택합니다.
3. DB 인스턴스를 선택합니다.

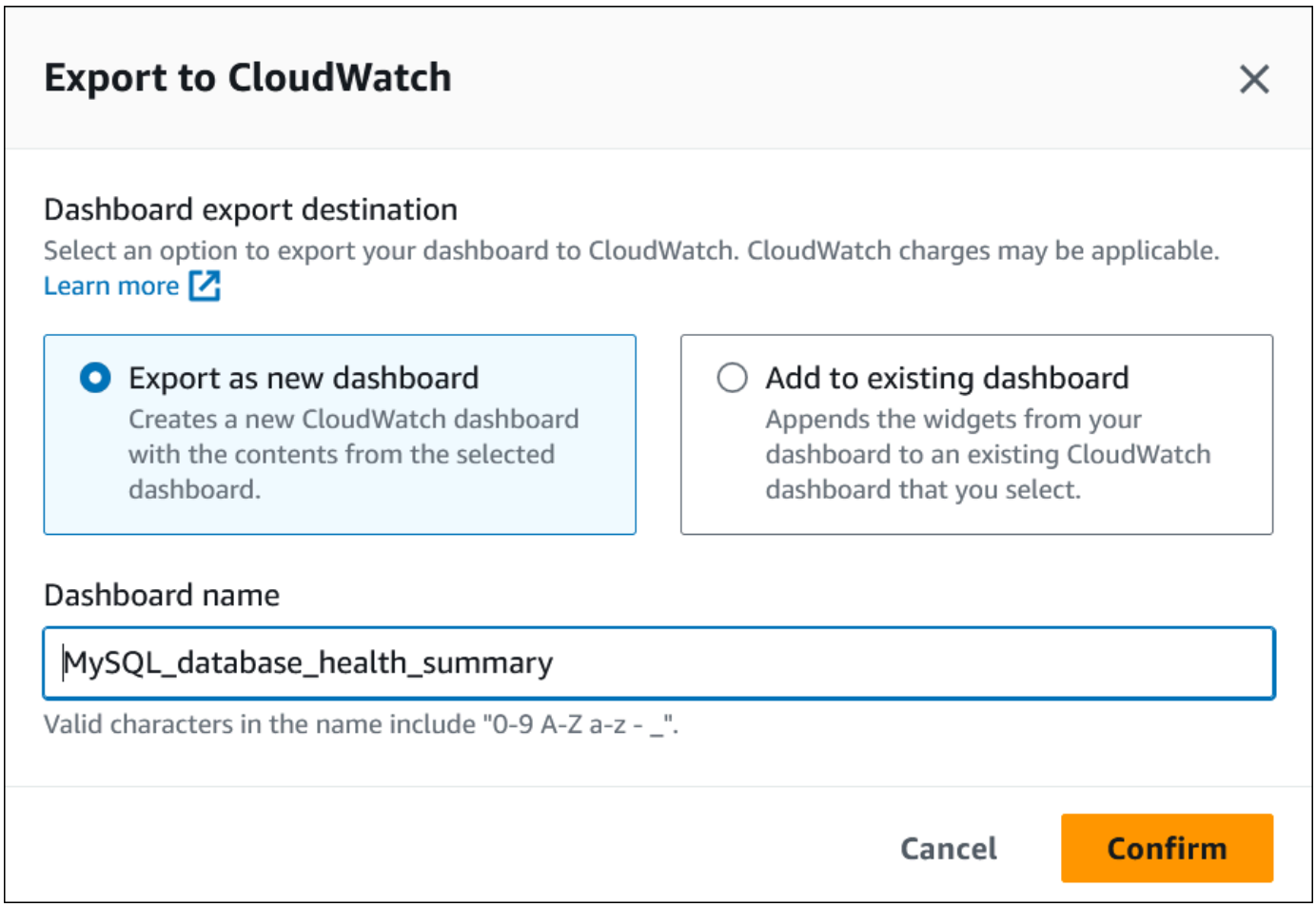
   해당 DB 인스턴스에 대해 성능 개선 도우미 대시보드가 표시됩니다.
4. 아래로 스크롤하여 지표를 선택합니다.

   기본적으로 성능 개선 도우미 지표가 포함된 사전 구성된 대시보드가 표시됩니다.
5. 사전 구성된 대시보드 또는 사용자 지정 대시보드를 선택한 다음 CloudWatch로 내보내기를 선택합니다.

   CloudWatch로 내보내기 창이 나타납니다.

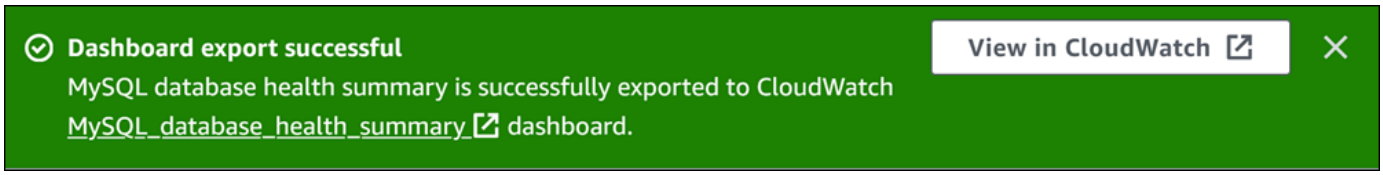


6. 새 대시보드로 내보내기를 선택합니다.



7. 대시보드 이름 필드에 새 대시보드의 이름을 입력하고 확인을 선택합니다.

   대시보드 내보내기에 성공하면 배너에 메시지가 표시됩니다.



8. CloudWatch 콘솔에서 지표 대시보드를 보려면 배너에서 링크 또는 CloudWatch에서 보기를 선택합니다.

기존 CloudWatch 대시보드에 성능 개선 도우미 지표 추가

사전 구성된 지표 대시보드 또는 사용자 지정 지표 대시보드를 기존 CloudWatch 대시보드에 추가합니다. CloudWatch 대시보드의 별도 섹션에 표시되도록 지표 대시보드에 레이블을 추가할 수 있습니다.

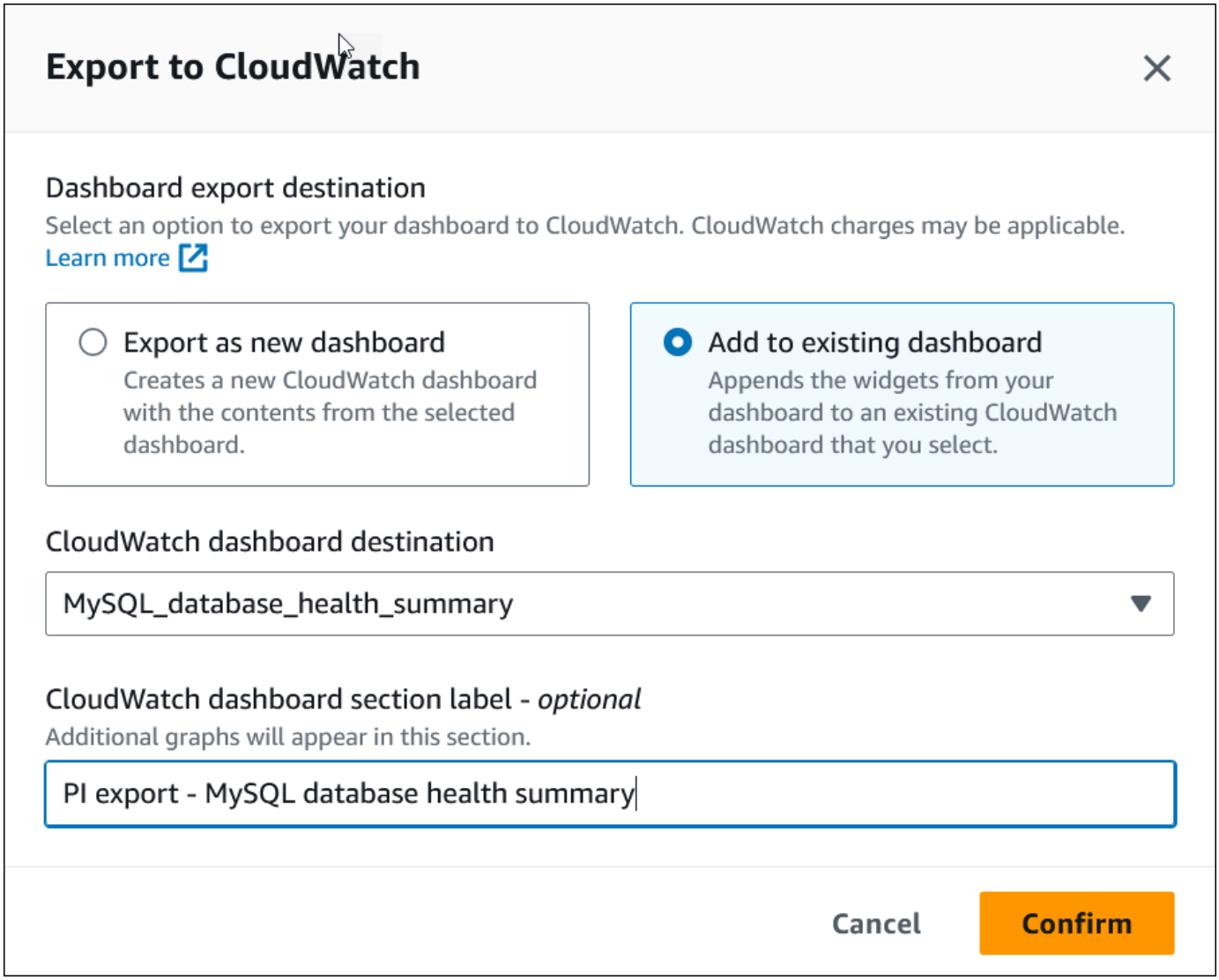
지표를 기존 CloudWatch 대시보드로 내보내는 방법

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 성능 개선 도우미(Performance Insights)를 선택합니다.
3. DB 인스턴스를 선택합니다.

   해당 DB 인스턴스에 대해 성능 개선 도우미 대시보드가 표시됩니다.
4. 아래로 스크롤하여 지표를 선택합니다.

   기본적으로 성능 개선 도우미 지표가 포함된 사전 구성된 대시보드가 표시됩니다.
5. 사전 구성된 대시보드 또는 사용자 지정 대시보드를 선택한 다음 CloudWatch로 내보내기를 선택합니다.

   CloudWatch로 내보내기 창이 나타납니다.
6. 기존 대시보드에 추가를 선택합니다.



7. 대시보드 대상 및 레이블을 지정한 다음 확인을 선택합니다.

   - CloudWatch 대시보드 대상 - 기존 클라우드워치 대시보드를 선택합니다.
   - CloudWatch 대시보드 섹션 레이블 - 선택 사항 - CloudWatch 대시보드의 이 섹션에 표시할 성능 개선 도우미 지표의 이름을 입력합니다.

   대시보드 내보내기에 성공하면 배너에 메시지가 표시됩니다.

8. CloudWatch 콘솔에서 지표 대시보드를 보려면 배너에서 링크 또는 CloudWatch에서 보기를 선택합니다.

CloudWatch에서 성능 개선 도우미 지표 위젯 보기

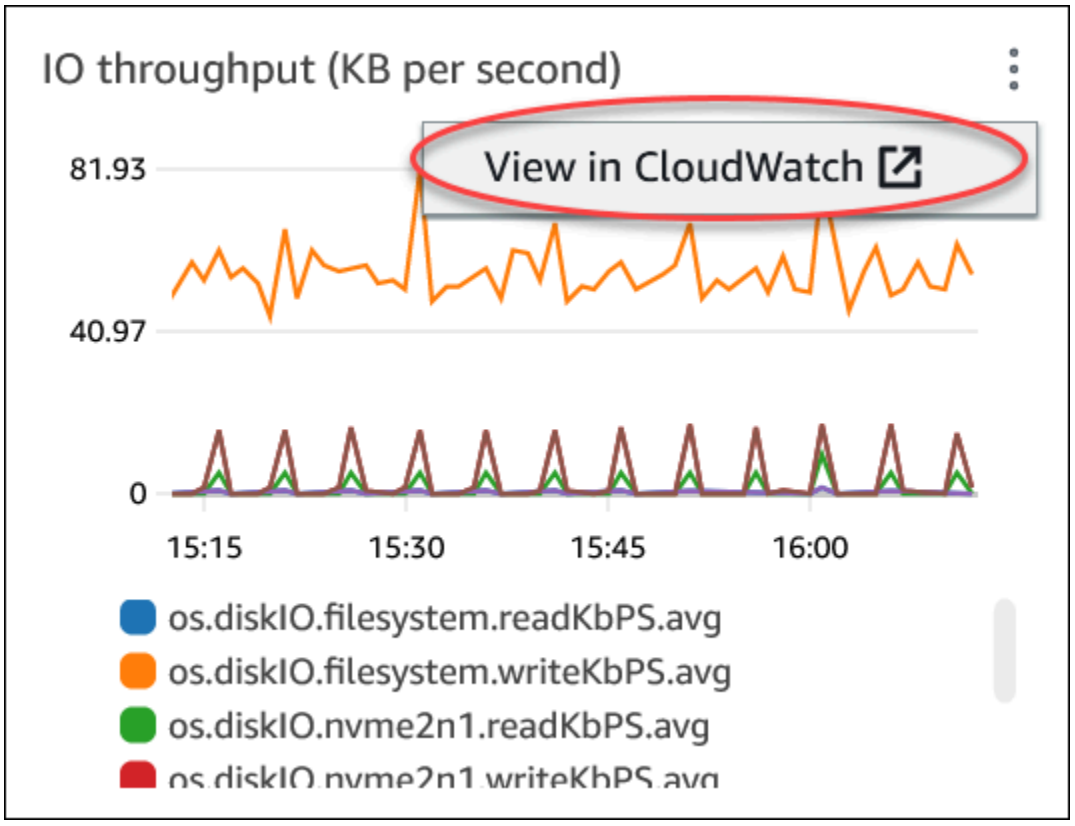
Amazon RDS 성능 개선 도우미 대시보드에서 성능 개선 도우미 지표 위젯을 선택하고 CloudWatch 콘솔에서 지표 데이터를 확인합니다.

지표 위젯을 내보내고 CloudWatch 콘솔에서 지표 데이터를 확인하는 방법

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 성능 개선 도우미(Performance Insights)를 선택합니다.
3. DB 인스턴스를 선택합니다.

   해당 DB 인스턴스에 대해 성능 개선 도우미 대시보드가 표시됩니다.
4. 아래로 스크롤하여 지표로 이동합니다.

   기본적으로 성능 개선 도우미 지표가 포함된 사전 구성된 대시보드가 표시됩니다.
5. 지표 위젯을 선택한 다음 메뉴에서 CloudWatch에서 보기를 선택합니다.



   지표 데이터가 CloudWatch 콘솔에 표시됩니다.

Amazon RDS을 모니터링하는 CloudWatch 경보 생성

경보로 인해 상태가 변경되면 Amazon SNS 메시지를 보내는 CloudWatch 경보를 생성할 수 있습니다. 경보는 지정한 기간 동안 단일 지표를 감시합니다. 또한 경보는 여러 기간에 대해 주어진 임계값과 지표 값을 비교하여 하나 이상의 작업을 수행할 수 있습니다. 이 작업은 Amazon SNS 주제나 Amazon EC2 Auto Scaling 정책으로 전송되는 알림입니다.

경보는 지속적인 상태 변경에 대해서만 작업을 호출합니다. CloudWatch 경보는 특정 상태에 있다고 해서 작업을 호출하지는 않습니다. 상태가 변경되어 지정된 기간 동안 유지되어야 합니다.

CloudWatch 콘솔의 DB_PERF_INSIGHTS 지표 수학 함수를 사용하여 Amazon RDS에서 성능 개선 도우미 카운터 지표를 쿼리할 수 있습니다. DB_PERF_INSIGHTS 함수에는 1분 미만의 간격으로 DBLoad 지표도 포함됩니다. 이러한 지표에 대해 CloudWatch 경보를 설정할 수 있습니다.

경보를 만드는 방법에 대한 자세한 내용은 AWS 데이터베이스의 성능 개선 도우미 카운터 지표에 대한 경보 생성을 참조하세요.

AWS CLI를 사용하여 경보 설정

- `put-metric-alarm`을 호출합니다. 자세한 내용은 AWS CLI 명령 참조를 참조하세요.

CloudWatch API를 사용하여 경보 설정

- 을 호출합니다..`PutMetricAlarm` 자세한 내용은 Amazon CloudWatch API 참조를 참조하세요.

Amazon SNS 주제 및 경보 설정에 대한 자세한 내용은 Amazon CloudWatch 경보 사용을 참조하세요.

자습서: Amazon RDS에 대한 다중 AZ DB 클러스터 복제본 지연에 대한 Amazon CloudWatch 경보 생성

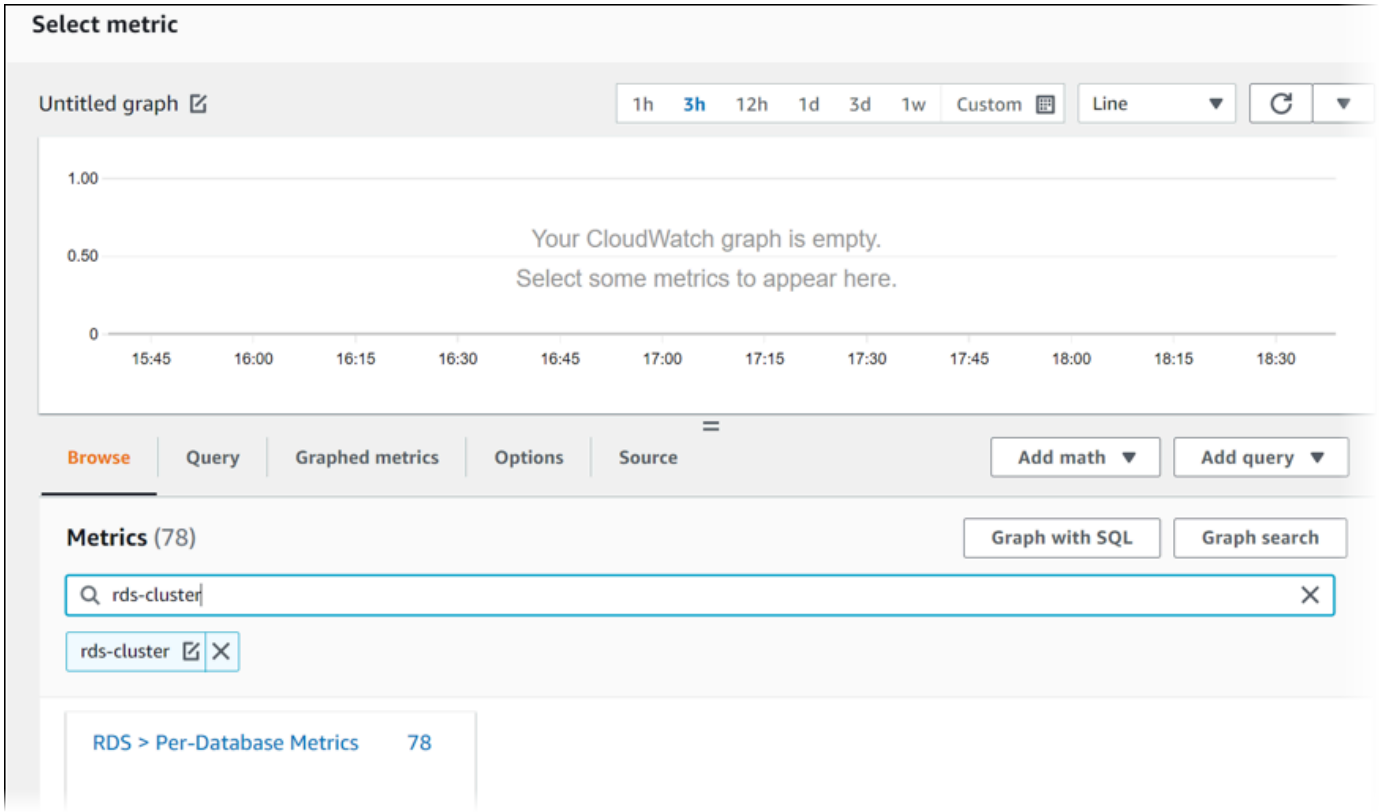
다중 AZ DB 클러스터의 복제본 지연이 임계값을 초과하면 Amazon SNS 메시지를 보내는 Amazon CloudWatch 경보를 생성할 수 있습니다. 경보는 지정한 기간 동안 `ReplicaLag` 지표를 감시합니다. 이 작업은 Amazon SNS 주제나 Amazon EC2 Auto Scaling 정책으로 전송되는 알림입니다.

다중 AZ DB 클러스터 복제본 지연에 대한 CloudWatch 경보 설정

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Alarms, All alarms를 선택합니다.

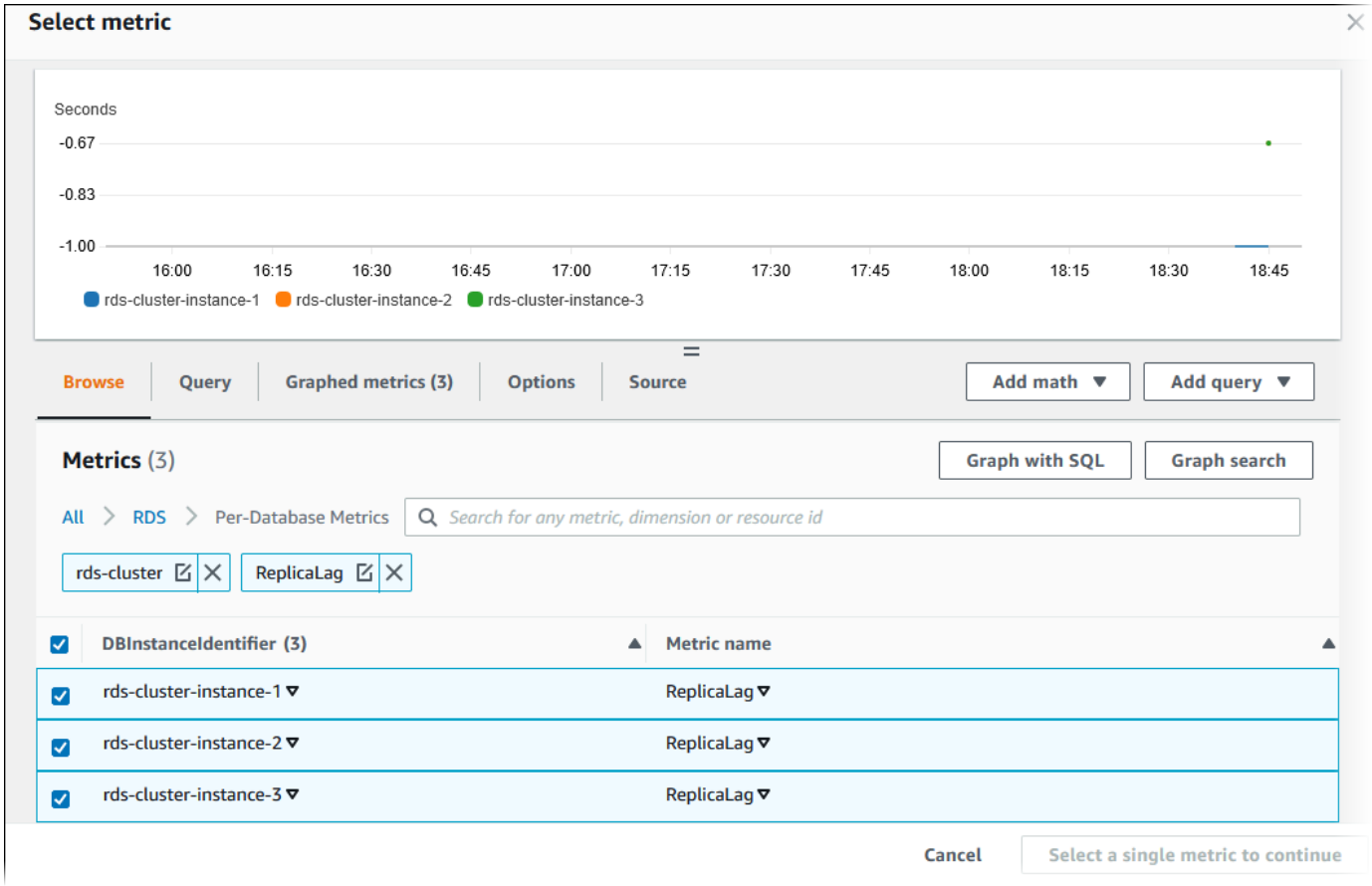
3. Create alarm(경보 생성)을 선택하세요.
4. 지표 및 조건 지정(Specify metric and conditions) 페이지에서 지표 선택(Select metric)을 선택합니다.
5. 검색 상자에 다중 AZ DB 클러스터의 이름을 입력하고 Enter를 누릅니다.

   다음 그림은 `rds-cluster`로 입력된 다중 AZ DB 클러스터를 사용한 지표 선택(Select metric) 페이지를 보여줍니다.



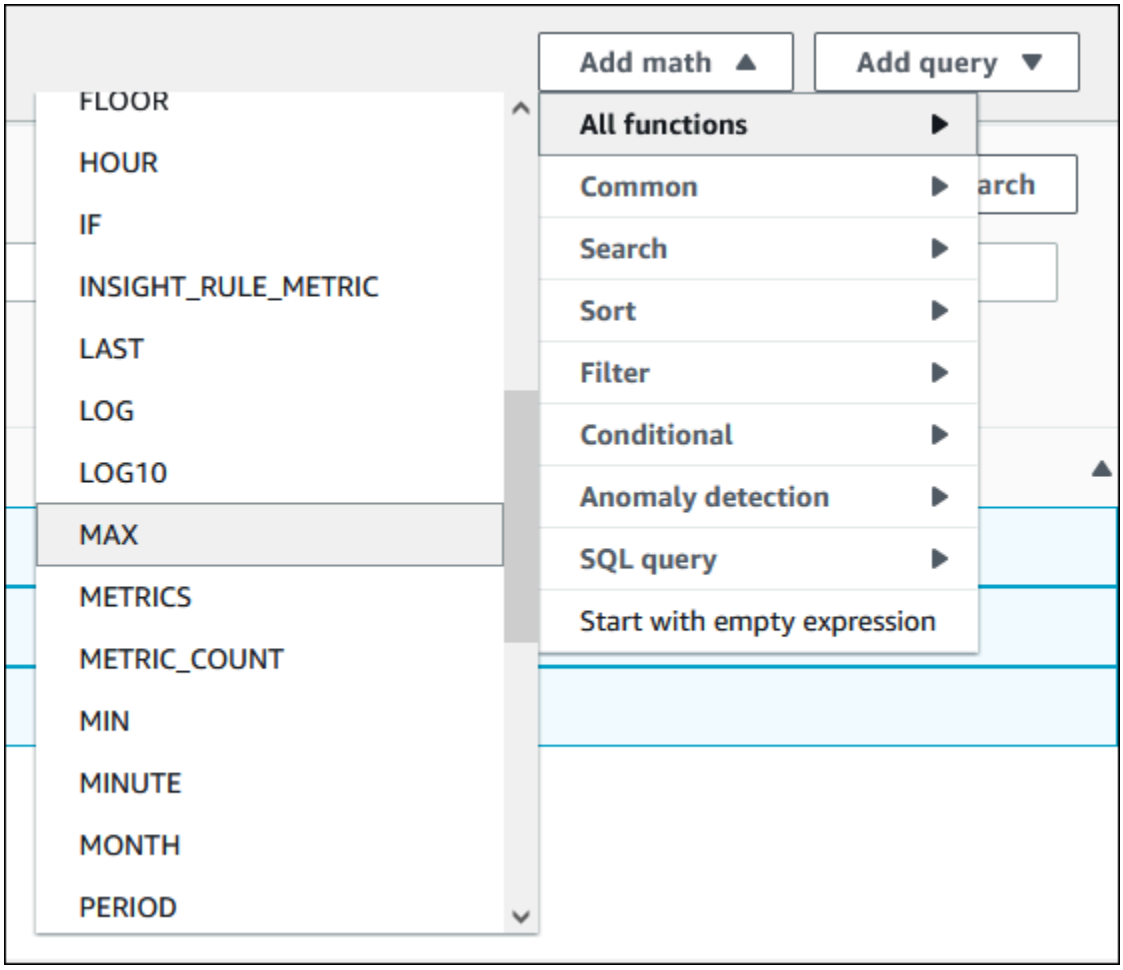
6. RDS의 데이터베이스별 지표(Per-Database Metrics)를 선택합니다.
7. 검색 상자에 **`ReplicaLag`**를 입력하고 Enter를 누른 다음 DB 클러스터에서 각 DB 인스턴스를 선택합니다.

   다음 그림은 ReplicaLag 지표에 대한 DB 인스턴스가 선택되어 있는 지표 선택(Select metric) 페이지를 보여줍니다.



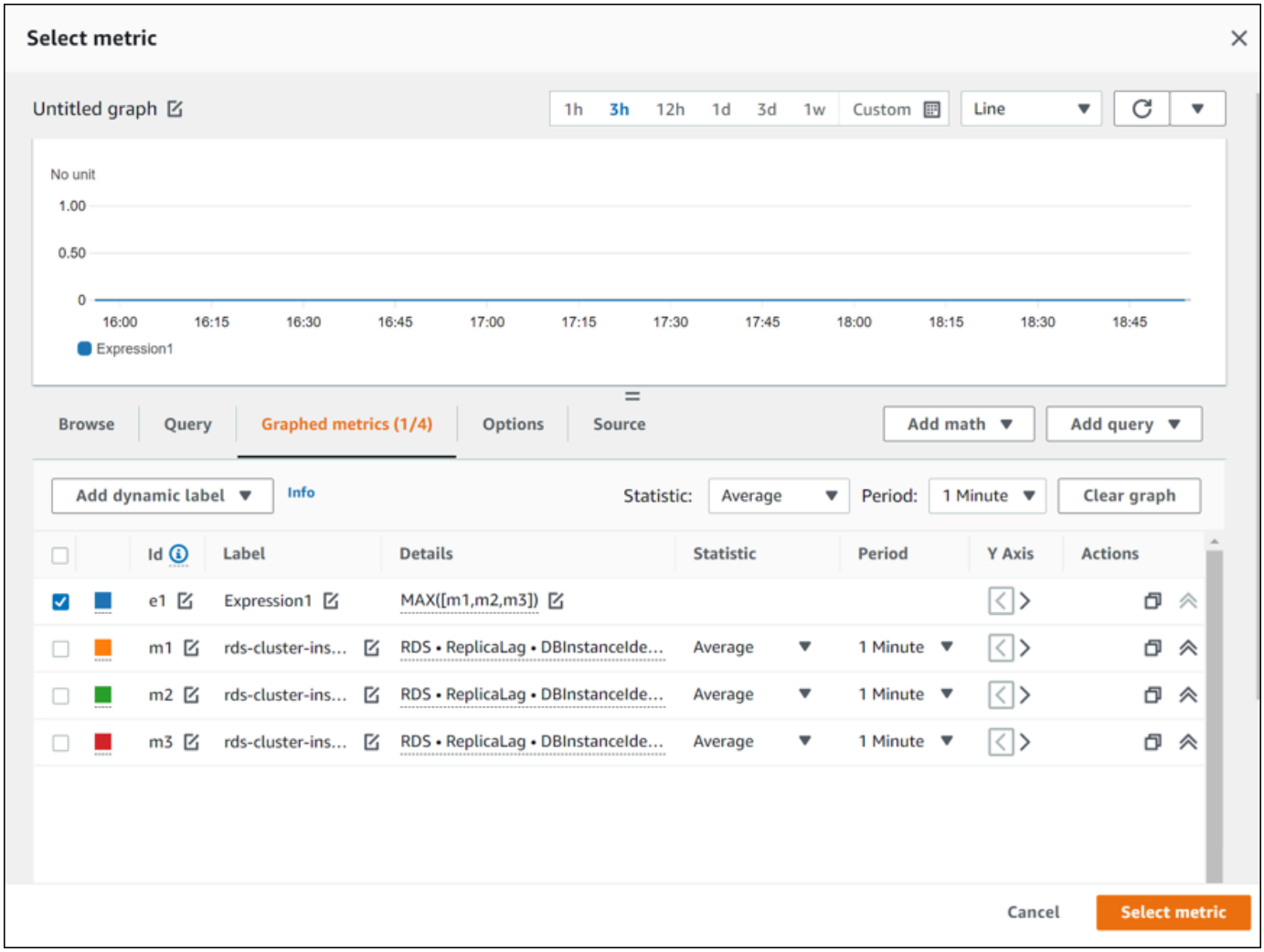
이 경보에서는 다중 AZ DB 클러스터의 DB 인스턴스 3개 모두에 대한 복제본 지연을 고려합니다. DB 인스턴스가 임계값을 초과하면 경보가 응답합니다. 지표 3개의 최대값을 반환하는 수학 표현식을 사용합니다. 지표 이름별로 정렬하여 시작한 후 3가지 ReplicaLag 지표를 모두 선택합니다.

8. 수식 추가(Add math)에서 모든 함수(All functions)의 최대(MAX)를 선택합니다.



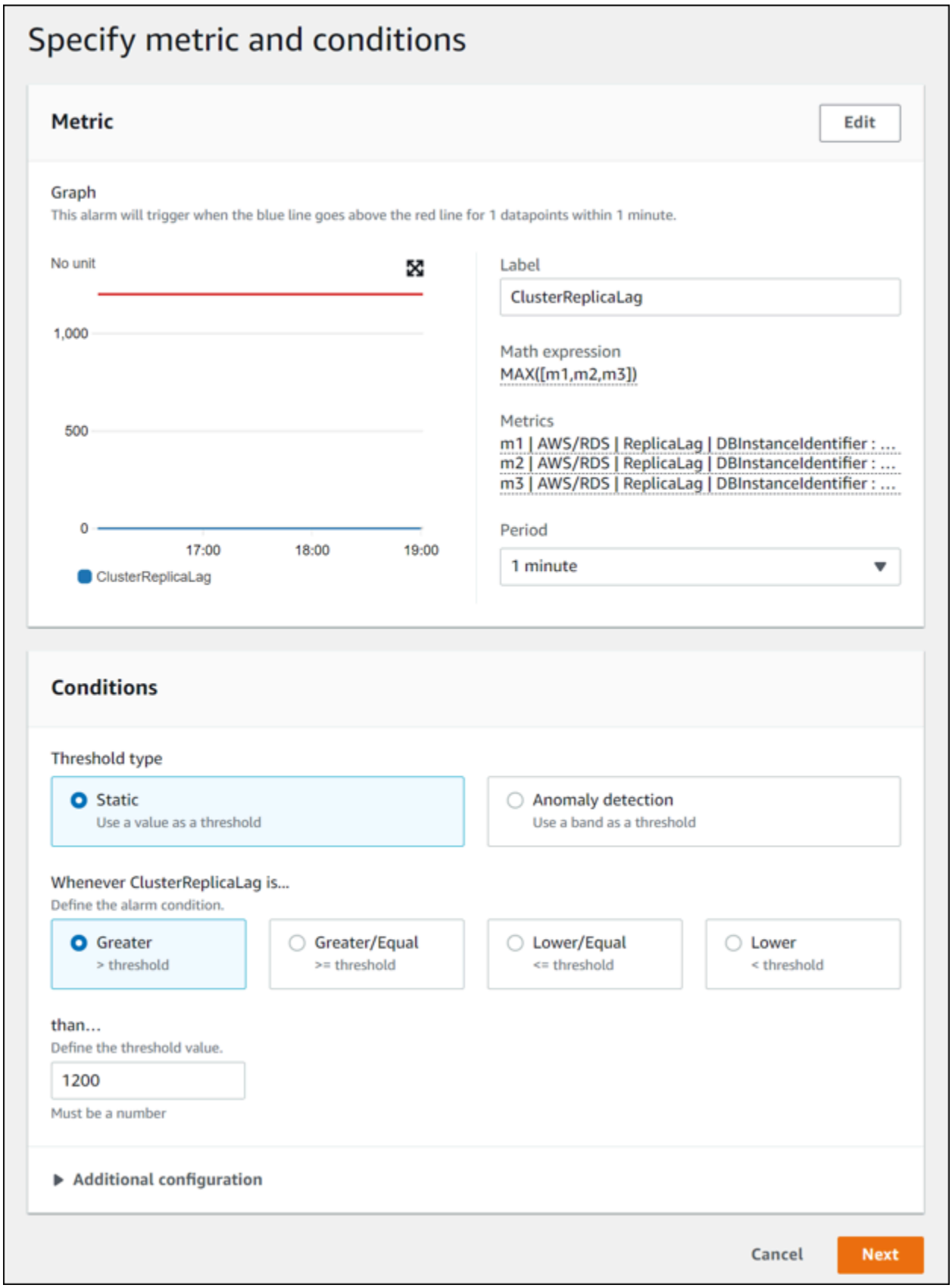
9. 그래프 지표(Graphed metrics) 탭을 선택한 다음 Expression1에 대한 세부 정보를 **`MAX([m1,m2,m3])`**로 편집합니다.
10. 세 가지 ReplicaLag 지표 모두는 기간(Period)을 1분(1 minute)으로 변경합니다.
11. Expression1을 제외한 모든 지표에서 선택을 지웁니다.

    지표 선택 페이지는 다음 이미지와 비슷해야 합니다.



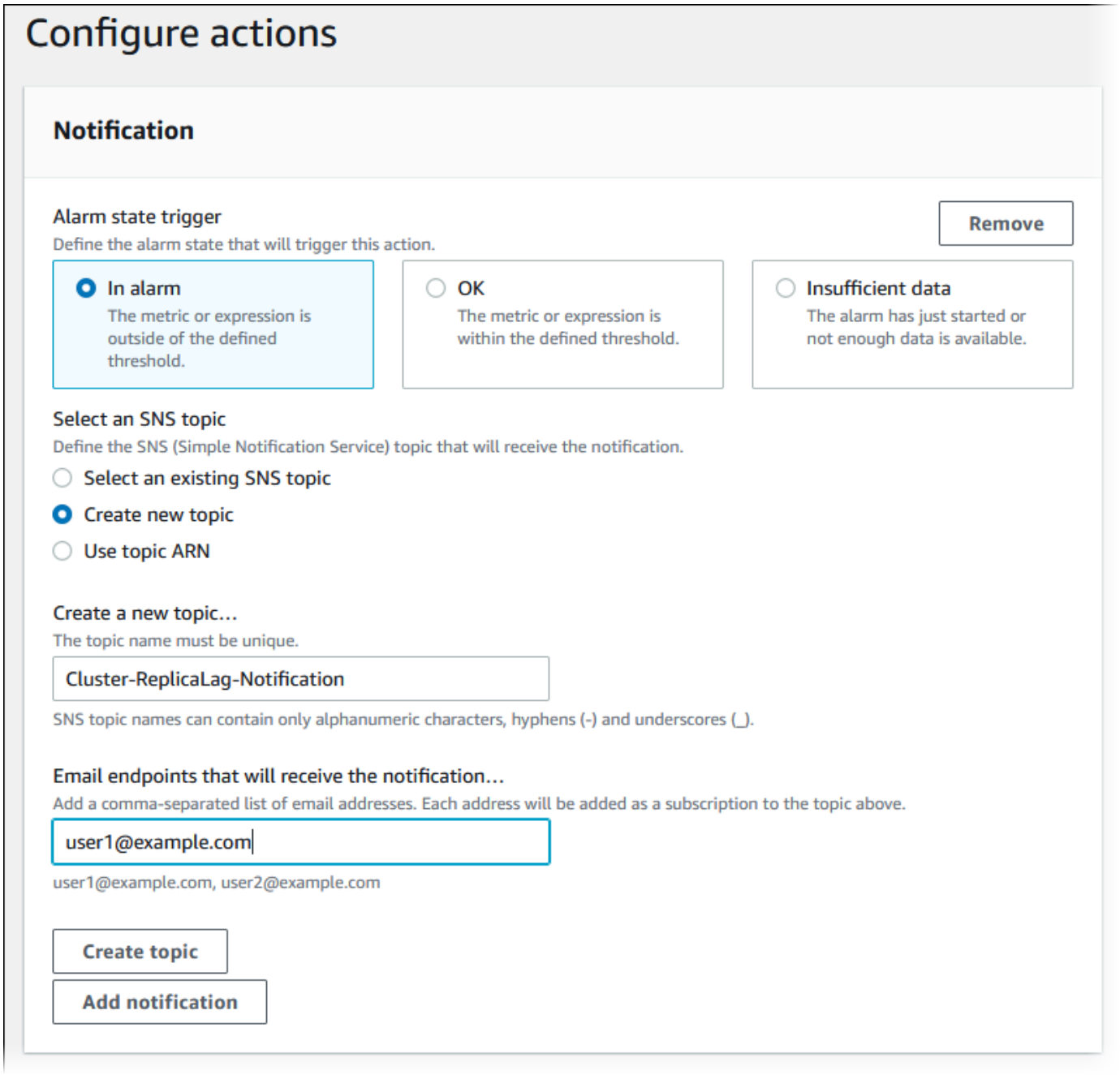
12. 지표 선택을 선택하세요.
13. 지표 및 조건 지정(Specify metric and conditions) 페이지에서 레이블을 **`ClusterReplicaLag`**와 같이 의미 있는 이름으로 변경하고, 임계값 정의(Define the threshold value)에서 시간(초)을 입력합니다. 이 자습서에서는 **`1200`**초(20분)를 입력합니다. 워크로드 요구 사항에 맞게 이 값을 조정할 수 있습니다.

    지표 및 조건 지정(Specify metric and conditions) 페이지는 다음 이미지와 비슷해야 합니다.

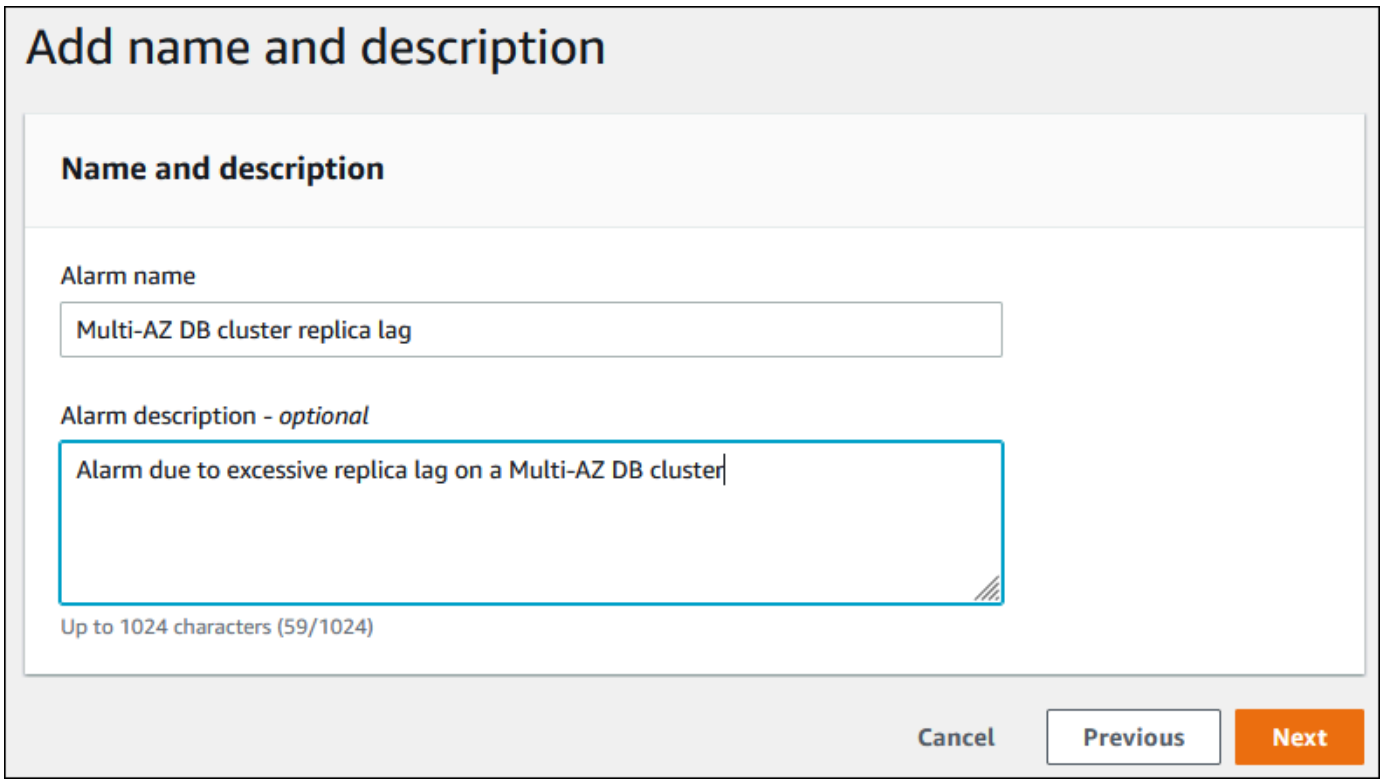


14. 다음(Next)을 선택하면 작업 구성(Configure actions) 페이지가 표시됩니다.

15. 선택된 경보(In alarm)를 유지하고, 새 주제 생성(Create new topic)을 선택하여, 주제 이름과 유효한 이메일 주소를 입력합니다.



16. 주제 생성(Create topic)성을 선택하고 다음(Next)을 선택합니다.
17. 이름 및 설명 추가 페이지에서 경보 이름(Alarm name) 및 경보 설명(Alarm description)을 입력하고 다음(Next)을 선택합니다.



18. 미리 보기 및 생성 페이지에서 생성하려는 경보를 미리 본 다음 경보 생성(Create alarm)을 선택합니다.

CloudWatch Database Insights를 사용하여 Amazon RDS 데이터베이스 모니터링

Database Insights를 사용하여 Amazon RDS DB 인스턴스 플릿의 데이터베이스 로드(DB 로드)를 모니터링합니다. DB 로드는 데이터베이스의 세션 활동 수준을 측정합니다. Database Insights를 사용하여 Amazon RDS 데이터베이스의 성능을 대규모로 분석하고 문제를 해결할 수 있습니다.

Database Insights를 사용하면 데이터베이스 플릿의 DB 로드를 시각화하고 대기 시간, SQL 문, 호스트 또는 사용자를 기준으로 로드를 필터링할 수 있습니다.

기본적으로 RDS는 Amazon RDS 데이터베이스에 대해 Database Insights의 표준 모드를 사용합니다.

Amazon CloudWatch 콘솔에서 Database Insights를 사용하는 방법에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 CloudWatch Database Insights를 참조하세요.

요금

요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

주제

- Amazon RDS DB 엔진, 리전 및 인스턴스 클래스에서 Database Insights 지원
- Amazon RDS에 Database Insights의 고급 모드 켜기
- Amazon RDS에 Database Insights의 표준 모드 켜기
- Amazon RDS용 Database Insights를 사용하여 느린 SQL 쿼리를 모니터링하도록 데이터베이스 구성
- Amazon RDS용 Database Insights의 고려 사항

Amazon RDS DB 엔진, 리전 및 인스턴스 클래스에서 Database Insights 지원

다음 테이블에는 Database Insights를 지원하는 Amazon RDS DB 엔진이 나와 있습니다.

Amazon RDS DB 엔진	지원되는 엔진 버전 및 리전	인스턴스 클래스 제한 사항
Amazon RDS for MariaDB	RDS for MariaDB를 사용한 Database Insights의 버전 및 리	Database Insights는 다음 인스턴스 클래스에서 지원되지 않습니다.

Amazon RDS DB 엔진	지원되는 엔진 버전 및 리전	인스턴스 클래스 제한 사항
	전 가용성에 대한 자세한 내용은 Amazon RDS에서 성능 개선 도우미를 지원하는 리전 및 DB 엔진 섹션을 참조하시기 바랍니다.	• db.t2.micro • db.t2.small • db.t3.micro • db.t3.small • db.t4g.micro • db.t4g.small
RDS for MySQL	RDS for MySQL에서 사용하는 Database Insights의 버전 및 리전 가용성에 대한 자세한 내용은 Amazon RDS에서 성능 개선 도우미를 지원하는 리전 및 DB 엔진 섹션을 참조하시기 바랍니다.	Database Insights는 다음 인스턴스 클래스에서 지원되지 않습니다. • db.t2.micro • db.t2.small • db.t3.micro • db.t3.small • db.t4g.micro • db.t4g.small
Amazon RDS for Microsoft SQL Server	RDS for SQL Server에서 사용하는 Database Insights의 버전 및 리전 가용성에 대한 자세한 내용은 Amazon RDS에서 성능 개선 도우미를 지원하는 리전 및 DB 엔진 섹션을 참조하시기 바랍니다.	N/A

Amazon RDS DB 엔진	지원되는 엔진 버전 및 리전	인스턴스 클래스 제한 사항
Amazon RDS for PostgreSQL	RDS for PostgreSQL에서 사용하는 Database Insights의 버전 및 리전 가용성에 대한 자세한 내용은 Amazon RDS에서 성능 개선 도우미를 지원하는 리전 및 DB 엔진 섹션을 참조하시기 바랍니다.	N/A
Amazon RDS for Oracle	RDS for Oracle에서 사용하는 Database Insights의 버전 및 리전 가용성에 대한 자세한 내용은 Amazon RDS에서 성능 개선 도우미를 지원하는 리전 및 DB 엔진 섹션을 참조하시기 바랍니다.	N/A

Amazon RDS DB 엔진, 리전 및 인스턴스 클래스에서 Database Insights 기능 지원

다음 표에는 Database Insights 기능을 지원하는 Amazon RDS DB 엔진이 나와 있습니다.

Feature	요금 티어	지원되는 리전	지원되는 DB 엔진	지원되는 인스턴스 클래스
성능 개선 도우미에 대한 SQL 통계	모두	모두	모두	모두
Amazon RDS용 성능 개선 도우미 대시보드를 사용한 Oracle 실행 계획 분석	모두	모두	RDS for Oracle	모두
일정 기간 동안의 데이터베이스 성능 분석	유료 티어만	모두	• RDS for MariaDB	모두

Feature	요금 티어	지원되는 리전	지원되는 DB 엔진	지원되는 인스턴스 클래스
			• RDS for MySQL • RDS for PostgreSQL	

Feature	요금 티어	지원되는 리전	지원되는 DB 엔진	지원되는 인스턴스 클래스
성능 개선 도우미 사전 권장 사항 보기	유료 티어만	• 미국 동부(오하이오) • 미국 동부(버지니아 북부) • 미국 서부(캘리포니아 북부) • 미국 서부(오리건) • 아시아 태평양(뭄바이) • 아시아 태평양(서울) • 아시아 태평양(싱가포르) • 아시아 태평양(시드니) • 아시아 태평양(도쿄) • 캐나다(중부) • 유럽(프랑크푸르트) • 유럽(아일랜드) • 유럽(런던) • 유럽(파리) • 유럽(스톡홀름) • 남아메리카(상파울루)	모두	모두

Database Insights를 지원하는 Amazon RDS 리전

Amazon RDS가 다음 AWS 리전에서 Database Insights를 지원합니다.

- 미국 동부(버지니아 북부)
- 미국 동부(오하이오)
- 미국 서부(캘리포니아 북부)
- 미국 서부(오리건)
- 아프리카(케이프타운)
- 아시아 태평양(홍콩)
- 아시아 태평양(하이데라바드)
- 아시아 태평양(자카르타)
- 아시아 태평양(말레이시아)
- 아시아 태평양(멜버른)
- 아시아 태평양(뭄바이)
- 아시아 태평양(오사카)
- 아시아 태평양(서울)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(도쿄)
- 캐나다(중부)
- 캐나다 서부(캘거리)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(밀라노)
- 유럽(파리)
- 유럽(스페인)
- 유럽(스톡홀름)
- 유럽(취리히)

- 이스라엘(텔아비브)
- 중동(바레인)
- 중동(UAE)
- 남아메리카(상파울루)
- AWS GovCloud(미국 동부)
- AWS GovCloud(미국 서부)

Amazon RDS에 Database Insights의 고급 모드 켜기

Amazon RDS에 Database Insights의 고급 모드를 켜려면 다음 프로시저를 따르세요.

DB 인스턴스 또는 다중 AZ DB 클러스터를 만들 때 Database Insights의 고급 모드 켜기

Amazon RDS에 대한 데이터베이스를 만들 때 Database Insights의 고급 모드를 켭니다.

Console

콘솔에서 DB 인스턴스나 다중 AZ DB 클러스터를 만들 때 Database Insights의 고급 모드를 켤 수 있습니다.

콘솔을 사용하여 DB 인스턴스 또는 다중 AZ DB 클러스터를 만들 때 Database Insights의 고급 모드를 켜는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 데이터베이스를 선택합니다.
3. 데이터베이스 생성를 선택합니다.
4. Database Insights 섹션에서 고급 모드를 선택합니다. 그런 다음, 다음 옵션 중 하나를 선택합니다.
   - 보존 – 성능 개선 도우미 데이터를 보존할 시간입니다. Database Insights의 고급 모드에서 보존 기간은 15개월이어야 합니다.
   - AWS KMS key - KMS 키를 지정합니다. 성능 개선 도우미는 KMS 키를 사용하여 잠재적으로 민감한 데이터를 모두 암호화합니다. 데이터는 암호화된 상태로 전송 및 저장됩니다. 자세한 내용은 Amazon RDS 리소스 암호화 섹션을 참조하세요.
5. 데이터베이스 생성를 선택합니다.

AWS CLI

DB 인스턴스나 다중 AZ DB 클러스터를 만들 때 Database Insights의 고급 모드를 켜려면 create-db-instance 또는 create-db-cluster AWS CLI 명령을 호출하고 다음 값을 제공합니다.

- `--database-insights-mode advanced`를 사용하여 Database Insights의 고급 모드를 켭니다.
- `--engine` - DB 인스턴스에 대한 데이터베이스 엔진입니다.
- `--db-instance-identifier` - DB 인스턴스의 식별자 또는 `--db-cluster-identifier` - 다중 AZ DB 클러스터의 식별자입니다.
- `--enable-performance-insights`를 사용하여 Database Insights에 대한 성능 개선 도우미를 켭니다.
- `--performance-insights-retention-period` - DB 인스턴스 또는 다중 AZ DB 클러스터의 데이터 보존 기간입니다. Database Insights를 켜려면 보존 기간이 최소 465일이어야 합니다.

다음 예시에서는 DB 인스턴스를 만들 때 Database Insights의 고급 모드를 사용 설정합니다.

Linux, macOS, Unix의 경우:

```
aws rds create-db-instance \
    --database-insights-mode advanced \
    --engine postgresql \
    --db-instance-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 465
```

Windows의 경우:

```
aws rds create-db-instance ^
    --database-insights-mode advanced ^
    --engine postgresql ^
    --db-instance-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 465
```

RDS API

DB 인스턴스 또는 다중 AZ DB 클러스터를 만들 때 Database Insights의 고급 모드를 켜려면 CreateDBInstance 또는 CreateDBCluster Amazon RDS API 작업에 대해 다음 파라미터를 지정합니다.

- `DatabaseInsightsMode~advanced`
- `EnablePerformanceInsights~True`
- `PerformanceInsightsRetentionPeriod` 최소 465일로 설정

DB 인스턴스 또는 다중 AZ DB 클러스터를 수정할 때 Database Insights의 고급 모드 켜기

Amazon RDS에 대한 데이터베이스를 수정할 때 Database Insights를 켭니다. Database Insights의 고급 모드를 활성화하도록 DB 인스턴스를 수정해도 가동 중지가 발생하지 않습니다.

> **Note**
>
> Database Insights를 사용 설정하려면 다중 AZ DB 클러스터의 각 DB 인스턴스에 동일한 성능 개선 도우미 및 향상된 모니터링 설정이 있어야 합니다.

Console

콘솔에서 DB 인스턴스나 다중 AZ DB 클러스터를 수정할 때 Database Insights의 고급 모드를 켤 수 있습니다.

콘솔을 사용하여 DB 인스턴스 또는 다중 AZ DB 클러스터를 수정할 때 Database Insights의 고급 모드를 켜는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 데이터베이스를 선택합니다.
3. DB 인스턴스 또는 다중 AZ DB 클러스터를 선택하고 수정(Modify)을 선택합니다.
4. Database Insights 섹션에서 고급 모드를 선택합니다. 그런 다음, 다음 옵션 중 하나를 선택합니다.

- 보존 – 성능 개선 도우미 데이터를 보존할 시간입니다. Database Insights의 고급 모드에서 보존 기간은 15개월이어야 합니다.
- AWS KMS key - KMS 키를 지정합니다. 성능 개선 도우미는 KMS 키를 사용하여 잠재적으로 민감한 데이터를 모두 암호화합니다. 데이터는 암호화된 상태로 전송 및 저장됩니다. 자세한 내용은 Amazon RDS 리소스 암호화 섹션을 참조하세요.

5. Continue(계속)를 선택합니다.
6. 수정 스케줄링에 대해 즉시 적용을 선택합니다. 예약된 다음 유지 관리 기간에 적용을 선택하면 데이터베이스에서 이 설정을 무시하고 Database Insights의 고급 모드를 즉시 켭니다.
7. 인스턴스 수정을 선택합니다.

AWS CLI

DB 인스턴스나 다중 AZ DB 클러스터를 수정할 때 Database Insights의 고급 모드를 켜려면 modify-db-instance 또는 modify-db-cluster AWS CLI 명령을 호출하고 다음 값을 제공합니다.

- `--database-insights-mode advanced`를 사용하여 Database Insights의 고급 모드를 켭니다.
- `--db-instance-identifier` - DB 인스턴스의 식별자 또는 `--db-cluster-identifier` - 다중 AZ DB 클러스터의 식별자입니다.
- `--enable-performance-insights`를 사용하여 Database Insights에 대한 성능 개선 도우미를 켭니다.
- `--performance-insights-retention-period` - DB 인스턴스의 데이터 보존 기간입니다. Database Insights의 고급 모드를 켜려면 보존 기간이 최소 465일이어야 합니다.

다음 예시에서는 DB 인스턴스를 수정할 때 Database Insights의 고급 모드를 사용 설정합니다.

Linux, macOS, Unix의 경우:

```
aws rds modify-db-instance \
    --database-insights-mode advanced \
    --db-instance-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 465
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --database-insights-mode advanced ^
    --db-instance-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 465
```

RDS API

DB 인스턴스 또는 다중 AZ DB 클러스터를 수정 때 Database Insights의 고급 모드를 켜려면 ModifyDBInstance 또는 ModifyDBCluster Amazon RDS API 작업에 대해 다음 파라미터를 지정합니다.

- `DatabaseInsightsMode~advanced`
- `EnablePerformanceInsights~True`
- `PerformanceInsightsRetentionPeriod` 최소 465일로 설정

Amazon RDS에 Database Insights의 표준 모드 켜기

Amazon RDS에 Database Insights의 표준 모드를 켜려면 다음 프로시저를 사용합니다.

DB 인스턴스 또는 다중 AZ DB 클러스터를 만들 때 Database Insights의 표준 모드 켜기

Amazon RDS에 대한 데이터베이스를 만들 때 Database Insights의 표준 모드를 켭니다.

Console

콘솔에서 DB 인스턴스 또는 다중 AZ DB 클러스터를 만들 때 Database Insights의 표준 모드를 켤 수 있습니다.

콘솔을 사용하여 DB 인스턴스 또는 다중 AZ DB 클러스터를 만들 때 Database Insights의 표준 모드를 켜는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 데이터베이스를 선택합니다.
3. 데이터베이스 생성를 선택합니다.
4. Database Insights 섹션에서 표준 모드를 선택합니다. 그런 다음, 다음 옵션 중에서 선택하여 성능 개선 도우미를 켜거나 끕니다.

- 성능 개선 도우미를 끄려면 성능 개선 도우미 활성화를 선택 취소합니다.
- 성능 개선 도우미를 켜려면 성능 개선 도우미 활성화를 선택합니다. 성능 개선 도우미를 구성하려면 다음 옵션을 지정합니다.
  - 보존 – 성능 개선 도우미 데이터를 보존할 시간입니다. 보존 기간은 최소 7일이어야 합니다.
  - AWS KMS key - KMS 키를 지정합니다. 성능 개선 도우미는 KMS 키를 사용하여 잠재적으로 민감한 데이터를 모두 암호화합니다. 데이터는 암호화된 상태로 전송 및 저장됩니다. 자세한 내용은 Amazon RDS 리소스 암호화 섹션을 참조하세요.

5. 데이터베이스 생성를 선택합니다.

AWS CLI

DB 인스턴스나 다중 AZ DB 클러스터를 만들 때 Database Insights의 표준 모드를 켜려면 create-db-instance 또는 create-db-cluster AWS CLI 명령을 호출하고 다음 값을 제공합니다.

- `--database-insights-mode standard`를 사용하여 Database Insights의 표준 모드를 켭니다.
- `--engine` - DB 인스턴스에 대한 데이터베이스 엔진입니다.
- `--db-instance-identifier` - DB 인스턴스의 식별자 또는 `--db-cluster-identifier` - 다중 AZ DB 클러스터의 식별자입니다.
- `--enable-performance-insights` 또는 `--no-enable-performance-insights`를 사용하여 성능 개선 도우미를 켜거나 끕니다. `--enable-performance-insights`를 지정하는 경우 `--performance-insights-retention-period`도 지정해야 함 - DB 인스턴스의 데이터 보존 기간입니다. 보존 기간은 최소 7일이어야 합니다.

다음 예시에서는 DB 인스턴스를 만들 때 Database Insights의 표준 모드 및 성능 개선 도우미를 사용 설정합니다.

Linux, macOS, Unix의 경우:

```
aws rds create-db-instance \
    --database-insights-mode standard \
    --engine postgresql \
    --db-instance-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 7
```

Windows의 경우:

```
aws rds create-db-instance ^
    --database-insights-mode standard ^
    --engine postgresql ^
    --db-instance-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 7
```

다음 예시에서는 DB 인스턴스를 만들 때 Database Insights의 표준 모드를 사용 설정하고 성능 개선 도우미를 사용 해제합니다.

Linux, macOS, Unix의 경우:

```
aws rds create-db-instance \
    --database-insights-mode standard \
    --engine postgresql \
    --db-instance-identifier sample-db-identifier \
    --no-enable-performance-insights
```

Windows의 경우:

```
aws rds create-db-instance ^
    --database-insights-mode standard ^
    --engine postgresql ^
    --db-instance-identifier sample-db-identifier ^
    --no-enable-performance-insights
```

RDS API

DB 인스턴스 또는 다중 AZ DB 클러스터를 만들 때 Database Insights의 표준 모드를 켜려면 CreateDBInstance 또는 CreateDBCluster Amazon RDS API 작업에 대해 다음 파라미터를 지정합니다.

- `DatabaseInsightsMode~standard`
- `EnablePerformanceInsights`를 `True` 또는 `False`로 지정합니다. `EnablePerformanceInsights`를 `True`로 설정한 경우 `PerformanceInsightsRetentionPeriod`를 7일 이상으로 설정해야 합니다.

DB 인스턴스 또는 다중 AZ DB 클러스터를 수정할 때 Database Insights의 표준 모드 켜기

Amazon RDS에 대한 데이터베이스를 수정할 때 Database Insights의 표준 모드를 켭니다. Database Insights의 표준 모드를 활성화하도록 DB 인스턴스를 수정해도 가동 중지가 발생하지 않습니다.

> **Note**
>
> Database Insights를 사용 설정하려면 다중 AZ DB 클러스터의 각 DB 인스턴스에 동일한 성능 개선 도우미 및 향상된 모니터링 설정이 있어야 합니다.

Console

콘솔에서 DB 인스턴스 또는 다중 AZ DB 클러스터를 수정할 때 Database Insights의 표준 모드를 켤 수 있습니다.

콘솔을 사용하여 DB 인스턴스 또는 다중 AZ DB 클러스터를 수정할 때 Database Insights의 표준 모드를 켜는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 데이터베이스를 선택합니다.
3. DB 인스턴스 또는 다중 AZ DB 클러스터를 선택하고 수정(Modify)을 선택합니다.
4. Database Insights 섹션에서 표준 모드를 선택합니다. 그런 다음, 다음 옵션 중 하나를 선택합니다.
   - 성능 개선 도우미를 끄려면 성능 개선 도우미 활성화를 선택 취소합니다.
   - 성능 개선 도우미를 켜려면 성능 개선 도우미 활성화를 선택합니다. 성능 개선 도우미를 구성하려면 다음 옵션을 지정합니다.
     - 보존 – 성능 개선 도우미 데이터를 보존할 시간입니다. 보존 기간은 최소 7일이어야 합니다.
     - AWS KMS key - KMS 키를 지정합니다. 성능 개선 도우미는 KMS 키를 사용하여 잠재적으로 민감한 데이터를 모두 암호화합니다. 데이터는 암호화된 상태로 전송 및 저장됩니다. 자세한 내용은 Amazon RDS 리소스 암호화 섹션을 참조하세요.
5. Continue(계속)를 선택합니다.

6. 수정 스케줄링에 대해 즉시 적용을 선택합니다. 예약된 다음 유지 관리 기간에 적용을 선택하면 데이터베이스에서 이 설정을 무시하고 Database Insights의 표준 모드를 즉시 켭니다.
7. 인스턴스 수정을 선택합니다.

AWS CLI

DB 인스턴스나 다중 AZ DB 클러스터를 수정할 때 Database Insights의 표준 모드를 켜려면 modify-db-instance 또는 modify-db-cluster AWS CLI 명령을 호출하고 다음 값을 제공합니다.

- `--database-insights-mode standard`를 사용하여 Database Insights의 표준 모드를 켭니다.
- `--db-instance-identifier` - DB 인스턴스의 식별자 또는 `--db-cluster-identifier` - 다중 AZ DB 클러스터의 식별자입니다.
- `--enable-performance-insights` 또는 `--no-enable-performance-insights`를 사용하여 성능 개선 도우미를 켜거나 끕니다. `--enable-performance-insights`를 지정하는 경우 `--performance-insights-retention-period`도 지정해야 함 - DB 인스턴스 또는 다중 AZ DB 클러스터의 데이터 보존 기간입니다. 보존 기간은 최소 7일이어야 합니다.

다음 예시에서는 DB 인스턴스를 수정할 때 Database Insights의 표준 모드 및 성능 개선 도우미를 사용 설정합니다.

Linux, macOS, Unix의 경우:

```
aws rds modify-db-instance \
    --database-insights-mode standard \
    --db-instance-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 7
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --database-insights-mode standard ^
    --db-instance-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 7
```

다음 예시에서는 DB 인스턴스를 수정할 때 Database Insights의 표준 모드를 사용 설정하고 성능 개선 도우미를 사용 해제합니다.

Linux, macOS, Unix의 경우:

```
aws rds modify-db-instance \
    --database-insights-mode standard \
    --db-instance-identifier sample-db-identifier \
    --no-enable-performance-insights
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --database-insights-mode standard ^
    --db-instance-identifier sample-db-identifier ^
    --no-enable-performance-insights
```

RDS API

DB 인스턴스 또는 다중 AZ DB 클러스터를 수정 때 Database Insights의 표준 모드를 켜려면 ModifyDBInstance 또는 ModifyDBCluster Amazon RDS API 작업에 대해 다음 파라미터를 지정합니다.

- `DatabaseInsightsMode~standard`
- `EnablePerformanceInsights`를 `True` 또는 `False`로 지정합니다. `EnablePerformanceInsights`를 `True`로 설정한 경우 `PerformanceInsightsRetentionPeriod`를 7일 이상으로 설정해야 합니다.

Amazon RDS용 Database Insights를 사용하여 느린 SQL 쿼리를 모니터링하도록 데이터베이스 구성

데이터베이스의 느린 SQL 쿼리를 모니터링하기 위해 Database Insights 대시보드의 느린 SQL 쿼리 섹션을 사용할 수 있습니다. 느린 SQL 쿼리를 모니터링하도록 데이터베이스를 구성하기 전에는 느린 SQL 쿼리 섹션이 비어 있습니다.

Database Insights 대시보드에서 느린 SQL 쿼리를 모니터링하는 방법에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 CloudWatch Database Insights용 데이터베이스 인스턴스 대시보드 보기를 참조하시기 바랍니다.

Database Insights를 사용하여 느린 SQL 쿼리를 모니터링하도록 데이터베이스를 구성하려면 다음 단계를 완료합니다.

1. CloudWatch Logs로 로그 내보내기를 활성화합니다.
2. DB 인스턴스의 DB 파라미터 그룹을 생성하거나 수정합니다.

로그 내보내기 구성에 대한 자세한 내용은 Amazon RDS 사용 설명서의 Amazon CloudWatch Logs에 데이터베이스 로그 게시 및 를 참조하세요.

DB 파라미터 그룹을 만들거나 수정하려면 다음 주제를 참조하세요.

- Amazon RDS에서 DB 파라미터 그룹 생성
- Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정

RDS for MariaDB

느린 SQL 쿼리를 모니터링하도록 RDS for MariaDB DB 인스턴스를 구성하려면 다음 파라미터를 설정합니다.

- `log_slow_query` - `1`로 설정됨
- `log_slow_query_time` - `1.0`으로 설정됨
- `log_output` - `FILE`로 설정됨

RDS for MySQL

느린 SQL 쿼리를 모니터링하도록 RDS for MySQL DB 인스턴스를 구성하려면 다음 파라미터를 설정합니다.

- `slow_query_log` - `1`로 설정됨
- `long_query_time` - `1.0`으로 설정됨
- `log_output` - `FILE`로 설정됨

RDS for PostgreSQL

느린 SQL 쿼리를 모니터링하도록 RDS for PostgreSQL DB 인스턴스를 구성하려면 다음 파라미터를 설정합니다. 이러한 파라미터를 설정하면 DB 인스턴스의 성능이 저하될 수 있음을 알려 드립니다.

- `log_min_duration_statement` - `1000`로 설정됨
- `log_statement` - `none`으로 설정됨
- `log_destination` - `stderr`로 설정됨

> **Note**
>
> RDS for MySQL의 경우 `long_query_time` 파라미터(1마이크로초 단위)를 구성할 수 있습니다. 예를 들어 이 파라미터를 `0.000001`로 설정할 수 있습니다. DB 인스턴스의 쿼리 양에 따라 `long_query_time` 파라미터 값이 성능을 저하시킬 수 있습니다. `1.0` 값으로 시작하고 워크로드에 따라 조정합니다. 이 파라미터를 `0`으로 설정하면 Database Insights는 모든 쿼리를 로깅합니다.

RDS for MariaDB, RDS for MySQL 및 RDS for PostgreSQL 로그에 대한 자세한 내용은 다음을 참조하시기 바랍니다.

- MariaDB 데이터베이스 로그 파일
- MySQL 데이터베이스 로그 파일
- RDS for PostgreSQL 데이터베이스 로그 파일

Amazon RDS용 Database Insights의 고려 사항

다음은 Amazon RDS용 Database Insights의 고려 사항입니다.

- 다중 AZ DB 클러스터에서 DB 인스턴스에 대한 Database Insights를 관리할 수 없습니다.
- Database Insights의 고급 모드를 활성화하려면 성능 개선 도우미를 활성화하고 성능 개선 도우미 보존 기간을 최소 465일(15개월)로 설정해야 합니다. Database Insights 비용 외에 성능 개선 도우미 보존 기간을 15개월로 설정하는 데 드는 추가 비용은 없습니다. Database Insights 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.
- Database Insights를 사용 설정하려면 다중 AZ DB 클러스터의 각 DB 인스턴스에 동일한 성능 개선 도우미 및 향상된 모니터링 설정이 있어야 합니다.
- Database Insights 모드 중 하나를 활성화하도록 DB 인스턴스를 수정해도 가동 중지가 발생하지 않습니다.

성능 개선 도우미를 통한 Amazon RDS 모니터링

Important

AWS는 Performance Insights의 수명 종료일을 2025년 11월 30일로 발표했습니다. 이 날짜 이후에는 Amazon RDS가 더 이상 Performance Insights 콘솔 환경, 유연한 보존 기간(1~24개월) 및 연결 요금을 지원하지 않습니다. Performance Insights API는 요금 변경 없이 계속 제공됩니다. Performance Insights API 비용은 CloudWatch Database Insights 비용과 함께 AWS 청구서에 표시됩니다.

Performance Insights의 유료 티어를 사용하는 DB 인스턴스를 2025년 11월 30일 이전에 Database Insights의 고급 모드로 업그레이드하는 것이 좋습니다. Database Insights의 고급 모드로 업그레이드하는 방법에 대한 자세한 내용은 Amazon RDS에 Database Insights의 고급 모드 켜기 섹션을 참조하세요.

조치를 취하지 않으면 Performance Insights를 사용하는 DB 인스턴스가 기본적으로 Database Insights의 표준 모드를 사용하게 됩니다. Database Insights의 표준 모드를 사용하는 경우 7일이 지난 성능 데이터 기록에 액세스할 수 없으며 Amazon RDS 콘솔에서 실행 계획 및 온디맨드 분석 기능을 사용하지 못할 수 있습니다. 2025년 11월 30일 이후에는 Database Insights의 고급 모드만 실행 계획과 온디맨드 분석을 지원합니다.

CloudWatch Database Insights를 사용하면 데이터베이스 플릿의 데이터베이스 로드를 모니터링하며 대규모로 성능을 분석하고 문제를 해결할 수 있습니다. Database Insights에 대한 자세한 내용은 CloudWatch Database Insights를 사용하여 Amazon RDS 데이터베이스 모니터링 섹션을 참조하세요. 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

성능 개선 도우미는 기존 Amazon RDS 모니터링 기능을 확장한 것으로서 데이터베이스 성능을 표시하여 분석하는 데 효과적입니다. 성능 개선 도우미 대시보드를 사용하면 Amazon RDS DB 인스턴스 로드를 시각화하고 대기 시간, SQL 문, 호스트 또는 사용자를 기준으로 로드를 필터링할 수 있습니다. Amazon DocumentDB에서 성능 개선 도우미를 사용하는 방법에 대한 자세한 내용은 Amazon DocumentDB 개발자 가이드를 참조하세요.

주제

- Amazon RDS의 성능 개선 도우미 개요
- Amazon RDS의 성능 개선 도우미 설정 및 해제
- Amazon RDS for MariaDB 또는 MySQL에서 성능 개선 도우미의 성능 스키마 개요
- Performance Insights에 대한 액세스 정책 구성
- 성능 개선 도우미 대시보드를 사용한 지표 분석

- 성능 개선 도우미 사전 권장 사항 보기
- Amazon RDS용 성능 개선 도우미 API를 사용하여 지표 검색
- AWS CloudTrail을 사용하여 Performance Insights 호출 로깅
- 성능 개선 도우미 API 및 인터페이스 VPC 엔드포인트(AWS PrivateLink)

Amazon RDS의 성능 개선 도우미 개요

Important

AWS는 Performance Insights의 수명 종료일을 2025년 11월 30일로 발표했습니다. 이 날짜 이후에는 Amazon RDS가 더 이상 Performance Insights 콘솔 환경, 유연한 보존 기간(1~24개월) 및 연결 요금을 지원하지 않습니다. Performance Insights API는 요금 변경 없이 계속 제공됩니다. Performance Insights API 비용은 CloudWatch Database Insights 비용과 함께 AWS 청구서에 표시됩니다.

Performance Insights의 유료 티어를 사용하는 DB 인스턴스를 2025년 11월 30일 이전에 Database Insights의 고급 모드로 업그레이드하는 것이 좋습니다. Database Insights의 고급 모드로 업그레이드하는 방법에 대한 자세한 내용은 Amazon RDS에 Database Insights의 고급 모드 켜기 섹션을 참조하세요.

조치를 취하지 않으면 Performance Insights를 사용하는 DB 인스턴스가 기본적으로 Database Insights의 표준 모드를 사용하게 됩니다. Database Insights의 표준 모드를 사용하는 경우 7일이 지난 성능 데이터 기록에 액세스할 수 없으며 Amazon RDS 콘솔에서 실행 계획 및 온디맨드 분석 기능을 사용하지 못할 수 있습니다. 2025년 11월 30일 이후에는 Database Insights의 고급 모드만 실행 계획과 온디맨드 분석을 지원합니다.

CloudWatch Database Insights를 사용하면 데이터베이스 플릿의 데이터베이스 로드를 모니터링하며 대규모로 성능을 분석하고 문제를 해결할 수 있습니다. Database Insights에 대한 자세한 내용은 CloudWatch Database Insights를 사용하여 Amazon RDS 데이터베이스 모니터링 섹션을 참조하세요. 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

기본적으로 RDS는 콘솔 생성 마법사에서 모든 Amazon RDS 엔진에 대해 성능 개선 도우미를 활성화합니다. DB 인스턴스에 둘 이상의 데이터베이스가 있는 경우 성능 개선 도우미는 성능 데이터를 집계합니다.

다음 비디오에서 Amazon RDS 성능 개선 도우미의 개요를 볼 수 있습니다.

성능 개선 도우미를 사용하여 Amazon Aurora PostgreSQL의 성능 분석

> **Important**
>
> 다음 주제는 비 Aurora DB 엔진에서 Amazon RDS 성능 개선 도우미를 사용하는 경우에 대해 설명합니다. Amazon RDS 성능 개선 도우미를 Amazon Aurora와 함께 사용하는 방법에 대한 자세한 내용은 Amazon Aurora 사용 설명서의 Amazon RDS 성능 개선 도우미 사용을 참조하세요.

주제

- 데이터베이스 부하
- 최대 CPU
- Amazon RDS DB 엔진, 리전 및 인스턴스 클래스 Performance Insights 지원
- 성능 개선 도우미의 요금 및 데이터 보존

데이터베이스 부하

데이터베이스 로드(DB 로드)는 데이터베이스의 세션 활동 수준을 측정합니다. DBLoad는 성능 개선 도우미의 주요 지표이며, 성능 개선 도우미는 1초마다 DB 로드를 수집합니다.

주제

- 활성 세션
- 평균 활성 세션
- 평균 활성 실행
- 차원

활성 세션

데이터베이스 세션은 관계형 데이터베이스와 애플리케이션의 대화를 나타냅니다. 활성 세션이란 DB 엔진에 작업을 제출하여 현재 응답 대기 중인 연결 세션을 말합니다.

세션은 CPU에서 실행 중이거나 리소스가 계속 진행될 수 있도록 대기 중일 때 활성화됩니다. 예를 들어 활성 세션은 페이지(또는 블록)가 메모리로 읽힐 때까지 기다린 다음 페이지에서 데이터를 읽는 동안 CPU를 사용할 수 있습니다.

평균 활성 세션

평균 활성 세션(AAS)은 성능 개선 도우미의 DBLoad 지표에 대한 단위입니다. 데이터베이스에서 동시에 활성화된 세션 수를 측정합니다.

성능 개선 도우미는 매초 쿼리를 동시에 실행하는 세션 수를 샘플링합니다. 각 활성 세션에 대해 성능 개선 도우미는 다음 데이터를 수집합니다.

- SQL 문
- 세션 상태(CPU에서 실행 중이거나 대기 중)
- Host
- SQL을 실행하는 사용자

성능 개선 도우미는 특정 기간 동안의 총 세션 수를 총 샘플 수로 나눠 AAS를 계산합니다. 예를 들어 다음 표는 1초 간격으로 채취된 실행 중인 쿼리의 연속된 5개 샘플을 보여 줍니다.

샘플	쿼리를 실행 중인 세션 수	AAS	계산
1	2	2	총 세션 2회/샘플 1개
2	0	1	총 세션 2회/샘플 2개
3	4	2	총 세션 6회/샘플 3개
4	0	1.5	총 세션 6회/샘플 4개
5	4	2	총 세션 10회/샘플 5개

이전 예시에서 시간 간격에 대한 DB 로드는 2 AAS입니다. 이 측정은 5개의 샘플을 채취한 간격 동안 평균적으로 2회의 세션이 동시에 활성화 상태였음을 의미합니다.

평균 활성 실행

초당 평균 활성 실행(AAE)은 AAS와 관련이 있습니다. Performance Insights는 AAE를 계산하기 위해 쿼리의 총 실행 시간을 시간 간격으로 나눕니다. 다음 표는 앞의 표와 동일한 쿼리의 AAE 계산을 보여 줍니다.

경과 시간(초)	총 실행 시간(초)	AAE	계산
60	120	2	120초 실행/60초 경과
120	120	1	120초 실행/120초 경과
180	380	2.11	380초 실행/180초 경과
240	380	1.58	380초 실행/240초 경과
300	600	2	600초 실행/300초 경과

대부분의 경우 쿼리의 AAS와 AAE는 거의 동일합니다. 하지만 계산에 입력되는 데이터 원본이 다르기 때문에 계산 결과가 약간 다른 경우가 많습니다.

차원

db.load 지표는 차원이라는 하위 구성 요소로 구분할 수 있다는 점에서 다른 시계열 지표와 다릅니다. 차원을 DBLoad 지표의 다양한 특성에 대한 "분할 기준" 범주로 생각할 수 있습니다.

성능 문제를 진단할 때 다음과 같은 차원이 가장 유용한 경우가 많습니다.

주제

- 대기 이벤트
- 상위 SQL
- 계획

Amazon RDS 엔진의 전체 차원 목록은 차원을 기준으로 분할된 DB 로드 섹션을 참조하세요.

대기 이벤트

대기 이벤트란 SQL 문이 계속 실행되려면 특정 이벤트가 발생할 때까지 기다려야 합니다. 대기 이벤트는 작업이 방해되는 위치를 나타내므로 DB 로드에 대해 중요한 측정기준이나 범주입니다.

모든 활성 세션이 CPU에서 실행 중이거나 대기 중입니다. 예를 들어 세션은 메모리에서 버퍼를 검색하거나 계산을 수행하거나 프로시저 코드를 실행할 때 CPU를 사용합니다. 세션에서 CPU를 사용하지 않는 경우 메모리 버퍼가 비어 있거나 읽을 데이터 파일 또는 기록할 로그가 나올 때까지 대기할 수 있습니다. 세션이 리소스를 기다리는 시간이 길수록 CPU에서 실행되는 시간이 줄어듭니다.

데이터베이스를 튜닝할 때 세션이 기다리는 리소스를 찾으려고 하는 경우가 많습니다. 예를 들어 두 개 또는 세 개의 대기 이벤트가 DB 로드의 90%를 차지할 수 있습니다. 이 측정은 평균적으로 활성 세션이 소수의 리소스를 기다리는 데 대부분의 시간을 소비한다는 것을 의미합니다. 이러한 대기의 원인을 찾을 수 있는 경우 해결 방법을 시도할 수 있습니다.

대기 이벤트는 DB 엔진마다 다릅니다.

- 모든 MariaDB 및 MySQL 대기 이벤트에 대한 자세한 내용은 MySQL 설명서의 대기 이벤트 요약 테이블을 참조하세요.
- 모든 PostgreSQL 대기 이벤트에 대한 자세한 내용은 PostgreSQL 설명서의 통계 수집기 > 대기 이벤트 테이블을 참조하세요.
- 모든 Oracle 대기 이벤트에 대한 자세한 내용은 Oracle 설명서의 대기 이벤트 설명을 참조하세요.
- 모든 SQL Server 대기 이벤트에 대한 자세한 내용은 SQL Server 설명서의 대기 유형을 참조하십시오.

> **Note**
>
> Oracle의 경우 연결된 SQL 없이 백그라운드 프로세스가 때때로 수행됩니다. 이 경우 성능 개선 도우미는 콜론으로 연결된 백그라운드 프로세스의 유형과 이 백그라운드 프로세스에 연결된 대기 클래스를 보고합니다. 백그라운드 프로세스의 유형으로는 `LGWR`, `ARC0`, `PMON` 등이 있습니다.
> 예를 들어 아카이버가 I/O를 수행할 경우 성능 개선 도우미는 `ARC1:System I/O`와 같이 보고합니다. 경우에 따라 백그라운드 프로세스 유형이 누락되고 성능 개선 도우미가 대기 클래스만 보고할 때도 있습니다(예: `:System I/O`).

상위 SQL

대기 이벤트는 병목 현상을 표시하는 반면, 상위 SQL은 DB 로드에 가장 많이 기여하는 쿼리를 보여줍니다. 예를 들어 현재 데이터베이스에서 여러 쿼리가 실행 중이더라도 단일 쿼리가 DB 부하의 99%를 소비할 수 있습니다. 이 경우 부하가 높으면 쿼리에 문제가 있음을 나타낼 수 있습니다.

기본적으로 성능 개선 도우미 콘솔에는 데이터베이스 부하에 영향을 미치는 상위 SQL 쿼리가 표시됩니다. 콘솔에는 각 문에 관련된 통계도 표시됩니다. 특정 문의 성능 문제를 진단하기 위해 실행 계획을 검사할 수 있습니다.

계획

단순히 계획이라고도 하는 실행 계획은 데이터에 액세스하는 일련의 단계입니다. 예를 들어, 테이블 `t1`과 `t2`를 결합하기 위한 계획은 `t1`의 모든 행을 반복하고 각 행을 `t2`의 행과 비교할 수 있습니다. 관계형 데이터베이스에서 옵티마이저는 SQL 쿼리에 대한 가장 효율적인 계획을 결정하는 기본 제공 코드입니다.

DB 인스턴스의 경우 성능 개선 도우미는 실행 계획을 자동으로 수집합니다. SQL 성능 문제를 진단하려면 리소스 사용량이 많은 SQL 쿼리에 대해 캡처된 계획을 검토합니다. 계획은 데이터베이스가 쿼리를 구문 분석하고 실행하는 방법을 보여줍니다.

계획을 사용하여 DB 로드를 분석하는 방법을 알아보려면 다음 내용을 참조하세요.

- Oracle: Amazon RDS용 성능 개선 도우미 대시보드를 사용한 Oracle 실행 계획 분석
- SQL Server: Amazon RDS용 성능 개선 도우미 대시보드를 사용한 SQL Server 실행 계획 분석

계획 캡처

성능 개선 도우미는 5분마다 리소스 사용량이 가장 많은 쿼리를 식별하고 해당 계획을 캡처합니다. 따라서 수많은 계획을 수동으로 수집하고 관리할 필요가 없습니다. 대신 상위 SQL(Top SQL) 탭을 사용하여 가장 문제가 많은 쿼리에 대한 계획에 집중할 수 있습니다.

> **Note**
>
> 성능 개선 도우미는 텍스트가 수집 가능한 쿼리 텍스트의 최대 한도를 초과하는 쿼리에 대한 계획을 캡처하지 않습니다. 자세한 내용은 성능 개선 도우미 대시보드에서 더 많은 SQL 텍스트에 액세스 섹션을 참조하세요.

실행 계획의 보존 기간은 성능 개선 도우미 데이터와 동일합니다. 프리 티어의 보존 설정은 기본값(7일)입니다. 성능 데이터를 더 오래 보존하려면 1~24개월을 지정하십시오. 보존 기간에 대한 자세한 내용은 성능 개선 도우미의 요금 및 데이터 보존 섹션을 참조하세요.

다이제스트 쿼리

상위 SQL(Top SQL) 탭에는 기본적으로 다이제스트 쿼리가 표시됩니다. 다이제스트 쿼리 자체에는 계획이 없지만 리터럴 값을 사용하는 모든 쿼리에는 계획이 있습니다. 예를 들어 다이제스트 쿼리에는 `WHERE `email`=?` 텍스트가 포함될 수 있습니다. 다이제스트에는 `WHERE email=user1@example.com` 텍스트가 포함된 쿼리와 `WHERE email=user2@example.com` 텍스

트가 포함된 쿼리, 이렇게 2개의 쿼리가 포함될 수 있습니다. 이러한 리터럴 쿼리 각각에는 여러 계획이 포함될 수 있습니다.

다이제스트 쿼리를 선택할 때 해당 다이제스트의 하위 문에 대한 모든 계획이 콘솔에 표시됩니다. 따라서 계획을 찾기 위해 모든 하위 문을 살펴볼 필요가 없습니다. 상위 10개 하위 문의 표시 목록에 없는 계획을 볼 수 있습니다. 쿼리가 상위 10개에 속하는지 여부에 관계없이 계획이 수집된 모든 하위 쿼리에 대한 계획이 콘솔에 표시됩니다.

최대 CPU

대시보드에서 데이터베이스 로드 차트는 세션 정보를 수집, 집계 및 표시합니다. 활성 세션이 최대 CPU를 초과하는지 확인하려면 최대 vCPU 선과의 관계를 확인합니다. 성능 개선 도우미는 최대 vCPU 값을 DB 인스턴스에서 vCPU(가상 CPU) 코어의 수로 결정합니다.

한 번에 하나의 프로세스를 vCPU에서 실행할 수 있습니다. 프로세스 수가 vCPUs 수를 초과하면 프로세스가 대기열에 추가되기 시작합니다. 대기열이 늘어나면 데이터베이스 성능이 저하됩니다. DB 로드가 최대 vCPU 선을 상회하는 경우가 잦아지고 CPU가 기본 대기 상태라면 CPU에서 과부하가 발생한 것입니다. 이 경우 연결 수를 인스턴스에 맞게 조절하거나, CPU 부하가 높은 SQL 쿼리를 모두 조정하거나, 인스턴스 클래스의 크기를 늘리는 것이 좋습니다. 대기 상태의 인스턴스가 높고 일관적이라는 것은 해결해야 할 병목 현상이나 리소스 경합 문제가 있을 수 있음을 나타냅니다. DB 로드가 최대 vCPU 선을 넘지 않는다 하더라도 이러한 문제가 나타날 수 있습니다.

Amazon RDS DB 엔진, 리전 및 인스턴스 클래스 Performance Insights 지원

Important

AWS는 Performance Insights의 수명 종료일을 2025년 11월 30일로 발표했습니다. 이 날짜 이후에는 Amazon RDS가 더 이상 Performance Insights 콘솔 환경, 유연한 보존 기간(1~24개월) 및 연결 요금을 지원하지 않습니다. Performance Insights API는 요금 변경 없이 계속 제공됩니다. Performance Insights API 비용은 CloudWatch Database Insights 비용과 함께 AWS 청구서에 표시됩니다.

Performance Insights의 유료 티어를 사용하는 DB 인스턴스를 2025년 11월 30일 이전에 Database Insights의 고급 모드로 업그레이드하는 것이 좋습니다. Database Insights의 고급 모드로 업그레이드하는 방법에 대한 자세한 내용은 Amazon RDS에 Database Insights의 고급 모드 켜기 섹션을 참조하세요.

조치를 취하지 않으면 Performance Insights를 사용하는 DB 인스턴스가 기본적으로 Database Insights의 표준 모드를 사용하게 됩니다. Database Insights의 표준 모드를 사용하는 경우 7일이 지난 성능 데이터 기록에 액세스할 수 없으며 Amazon RDS 콘솔에서 실행 계획 및 온디맨

드 분석 기능을 사용하지 못할 수 있습니다. 2025년 11월 30일 이후에는 Database Insights의 고급 모드만 실행 계획과 온디맨드 분석을 지원합니다.
CloudWatch Database Insights를 사용하면 데이터베이스 플릿의 데이터베이스 로드를 모니터링하며 대규모로 성능을 분석하고 문제를 해결할 수 있습니다. Database Insights에 대한 자세한 내용은 CloudWatch Database Insights를 사용하여 Amazon RDS 데이터베이스 모니터링 섹션을 참조하세요. 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

다음 테이블에는 성능 개선 도우미를 지원하는 Amazon RDS DB 엔진이 나와 있습니다.

> **Note**
>
> Amazon Aurora에 대한 자세한 내용은 Amazon Aurora 사용 설명서에서 Amazon Aurora DB 엔진의 성능 개선 도우미 지원을 참조하세요.

Amazon RDS DB 엔진	지원되는 엔진 버전 및 리전	인스턴스 클래스 제한 사항
Amazon RDS for MariaDB	RDS for MariaDB를 사용한 성능 개선 도우미 인사이트의 버전 및 리전 가용성에 대한 자세한 내용은 Amazon RDS에서 성능 개선 도우미를 지원하는 리전 및 DB 엔진 단원을 참조하세요.	성능 개선 도우미는 다음 인스턴스 클래스에서 지원되지 않습니다. • db.t2.micro • db.t2.small • db.t3.micro • db.t3.small • db.t4g.micro • db.t4g.small
RDS for MySQL	MySQL용 RDS를 사용한 성능 개선 도우미 인사이트의 버전 및 리전 가용성에 대한 자세한 내용은	성능 개선 도우미는 다음 인스턴스 클래스에서 지원되지 않습니다. •

Amazon RDS DB 엔진	지원되는 엔진 버전 및 리전	인스턴스 클래스 제한 사항
	Amazon RDS에서 성능 개선 도우미를 지원하는 리전 및 DB 엔진 단원을 참조하세요.	• db.t2.micro • db.t2.small • db.t3.micro • db.t3.small • db.t4g.micro • db.t4g.small
Amazon RDS for Microsoft SQL Server	RDS for SQL Server를 사용한 성능 개선 도우미 인사이트의 버전 및 지역 가용성에 대한 자세한 내용은 Amazon RDS에서 성능 개선 도우미를 지원하는 리전 및 DB 엔진 단원을 참조하세요.	N/A
Amazon RDS for PostgreSQL	RDS for PostgreSQL 기반 성능 개선 도우미 인사이트의 버전 및 지역 가용성에 대한 자세한 내용은 Amazon RDS에서 성능 개선 도우미를 지원하는 리전 및 DB 엔진 단원을 참조하세요.	N/A
Amazon RDS for Oracle	RDS for Oracle을 사용한 성능 개선 도우미 버전 및 지역 가용성에 대한 자세한 내용은 을 참조하십시오.Amazon RDS에서 성능 개선 도우미를 지원하는 리전 및 DB 엔진.	N/A

성능 개선 도우미 기능에 대한 Amazon RDS DB 엔진, 리전 및 인스턴스 클래스 지원

다음 테이블에는 성능 개선 도우미 기능을 지원하는 Amazon RDS DB 엔진이 나와 있습니다.

Feature	요금 티어	지원되는 리전	지원되는 DB 엔진	지원되는 인스턴스 클래스
성능 개선 도우미에 대한 SQL 통계	모두	모두	모두	모두
Amazon RDS용 성능 개선 도우미 대시보드를 사용한 Oracle 실행 계획 분석	모두	모두	RDS for Oracle	모두
일정 기간 동안의 데이터베이스 성능 분석	유료 티어만	모두	RDS for PostgreSQL	모두
성능 개선 도우미 사전 권장 사항 보기	유료 티어만	• 미국 동부(오하이오) • 미국 동부(버지니아 북부) • 미국 서부(캘리포니아 북부) • 미국 서부(오리건) • 아시아 태평양(뭄바이) • 아시아 태평양(서울) • 아시아 태평양(싱가포르) • 아시아 태평양(시드니)	모두	모두

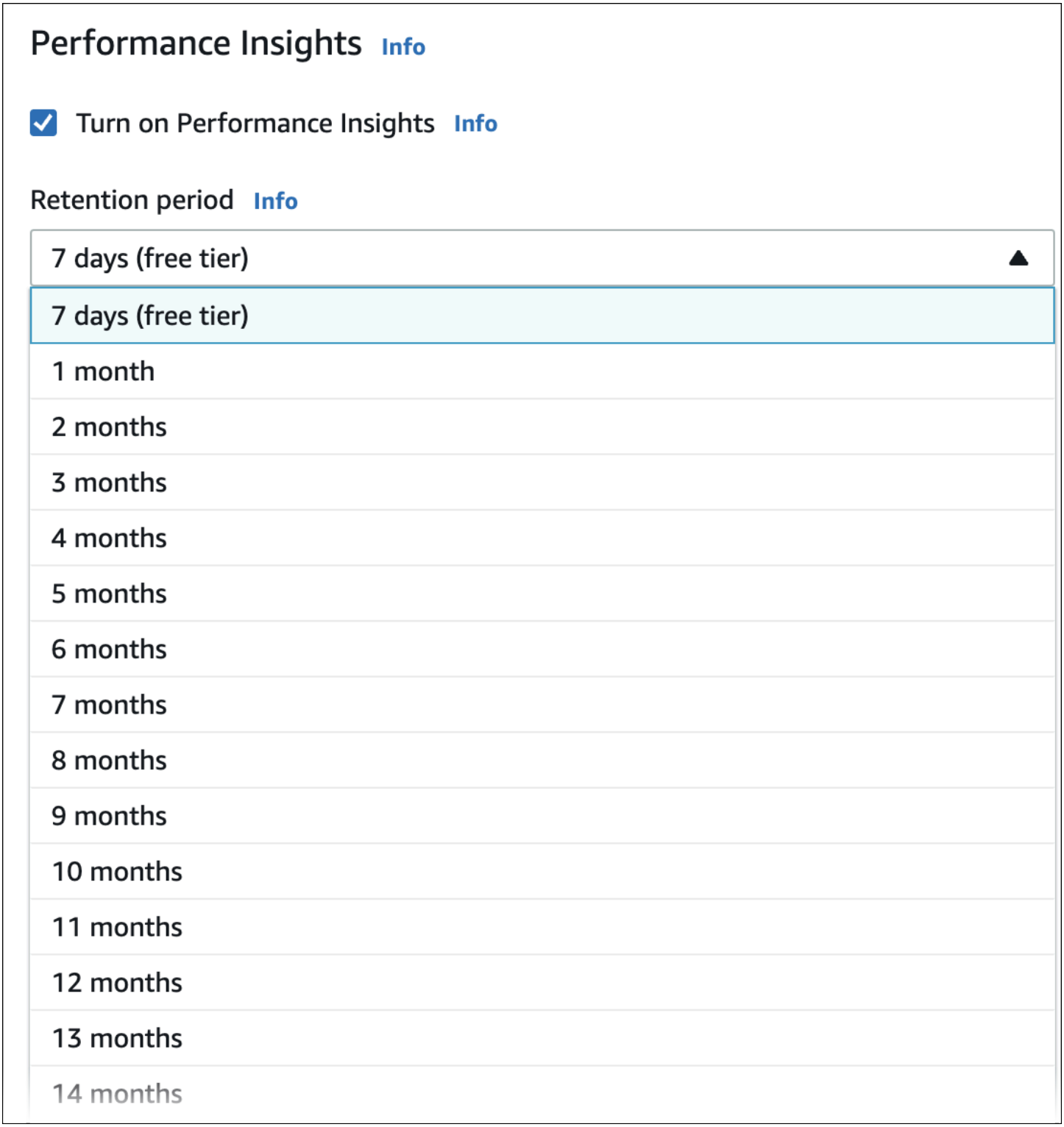
Feature	요금 티어	지원되는 리전	지원되는 DB 엔진	지원되는 인스턴스 클래스
		• 아시아 태평양(도쿄) • 캐나다(중부) • 유럽(프랑크푸르트) • 유럽(아일랜드) • 유럽(런던) • 유럽(파리) • 유럽(스톡홀름) • 남아메리카(상파울루)		

성능 개선 도우미의 요금 및 데이터 보존

기본적으로 성능 개선 도우미는 7일간의 성능 데이터 기록과 매달 1백만 건의 API 요청이 포함된 프리 티어를 제공합니다. 더 긴 보존 기간을 구매할 수도 있습니다. 전체 요금 정보는 성능 개선 도우미 요금을 참조하세요.

RDS 콘솔에서는 성능 개선 도우미 데이터에 대해 다음과 같은 보존 기간을 선택할 수 있습니다.

- 기본값(7일)
- ***n***개월(***n***은 1~24 사이의 숫자)



AWS CLI를 사용하여 보존 기간을 설정하는 방법은 ??? 섹션에서 알아보세요.

Amazon RDS의 성능 개선 도우미 설정 및 해제

Important

AWS는 Performance Insights의 수명 종료일을 2025년 11월 30일로 발표했습니다. 이 날짜 이후에는 Amazon RDS가 더 이상 Performance Insights 콘솔 환경, 유연한 보존 기간(1~24개월) 및 연결 요금을 지원하지 않습니다. Performance Insights API는 요금 변경 없이 계속 제공됩니다. Performance Insights API 비용은 CloudWatch Database Insights 비용과 함께 AWS 청구서에 표시됩니다.

Performance Insights의 유료 티어를 사용하는 DB 인스턴스를 2025년 11월 30일 이전에 Database Insights의 고급 모드로 업그레이드하는 것이 좋습니다. Database Insights의 고급 모드로 업그레이드하는 방법에 대한 자세한 내용은 Amazon RDS에 Database Insights의 고급 모드 켜기 섹션을 참조하세요.

조치를 취하지 않으면 Performance Insights를 사용하는 DB 인스턴스가 기본적으로 Database Insights의 표준 모드를 사용하게 됩니다. Database Insights의 표준 모드를 사용하는 경우 7일이 지난 성능 데이터 기록에 액세스할 수 없으며 Amazon RDS 콘솔에서 실행 계획 및 온디맨드 분석 기능을 사용하지 못할 수 있습니다. 2025년 11월 30일 이후에는 Database Insights의 고급 모드만 실행 계획과 온디맨드 분석을 지원합니다.

CloudWatch Database Insights를 사용하면 데이터베이스 플릿의 데이터베이스 로드를 모니터링하며 대규모로 성능을 분석하고 문제를 해결할 수 있습니다. Database Insights에 대한 자세한 내용은 CloudWatch Database Insights를 사용하여 Amazon RDS 데이터베이스 모니터링 섹션을 참조하세요. 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

DB 인스턴스 또는 다중 AZ DB 클러스터를 생성할 때 Performance Insights를 활성화할 수 있습니다. 필요한 경우 콘솔에서 DB 인스턴스를 수정하여 나중에 비활성화할 수 있습니다. 성능 개선 도우미를 활성화하거나 비활성화해도 가동 중지, 재부팅 또는 장애 조치가 발생하지 않습니다.

Note

성능 스키마는 Amazon RDS for MariaDB 또는 Amazon RDS for MySQL에서 사용하는 선택적 성능 도구입니다. 성능 스키마를 켜거나 끄면 재부팅해야 합니다. 그러나 성능 개선 도우미를 켜거나 끌 경우에는 재부팅할 필요가 없습니다. 자세한 내용은 Amazon RDS for MariaDB 또는 MySQL에서 성능 개선 도우미의 성능 스키마 개요 섹션을 참조하세요.

성능 개선 도우미 에이전트는 DB 호스트에서 제한된 CPU 및 메모리를 사용합니다. DB 로드가 높을 경우 에이전트는 데이터 수집 빈도를 줄여 성능에 미치는 영향을 제한합니다.

Console

콘솔에서 DB 인스턴스 또는 다중 AZ DB 클러스터를 생성하거나 수정할 때 성능 개선 도우미를 설정하거나 해제할 수 있습니다.

DB 인스턴스 또는 DB 클러스터를 만들 때 성능 개선 도우미 활성화 및 비활성화

새 DB 인스턴스 또는 다중 AZ DB 클러스터를 만든 후 Amazon RDS는 기본적으로 성능 개선 도우미를 활성화합니다. 성능 개선 도우미를 끄려면 Database Insights – 표준 옵션을 선택하고 성능 개선 도우미 활성화 옵션을 선택 취소합니다.

자세한 내용은 다음 항목을 참조하세요.

- DB 인스턴스를 생성하려면 Amazon RDS DB 인스턴스 생성의 DB 엔진에 대한 지침을 따르십시오.
- DB 클러스터를 만들려면 Amazon RDS용 다중 AZ DB 클러스터 생성 섹션의 지침을 따르세요.

다음 이미지는 성능 개선 도우미 섹션의 스크린샷입니다.



성능 개선 도우미 활성화를 선택하면 다음 옵션이 있습니다.

- 보존(Database Insights의 표준 모드에만 해당) - 성능 개선 도우미 데이터를 보존하는 시간입니다. 프리 티어의 보존 설정은 기본값(7일)입니다. 성능 데이터를 더 오래 보존하려면 1~24개월을

지정하십시오. 보존 기간에 대한 자세한 내용은 성능 개선 도우미의 요금 및 데이터 보존 섹션을 참조하세요.

- AWS KMS key – AWS KMS key을(를) 지정합니다. 성능 개선 도우미는 KMS 키를 사용하여 잠재적으로 민감한 데이터를 모두 암호화합니다. 데이터는 암호화된 상태로 전송 및 저장됩니다. 자세한 내용은 성능 개선 도우미를 위한 AWS KMS 정책 변경 섹션을 참조하세요.

DB 인스턴스 또는 다중 AZ DB 클러스터를 수정할 때 Performance Insights 켜기 또는 끄기

콘솔에서 DB 인스턴스 또는 다중 AZ DB 클러스터를 수정하여 Performance Insights를 관리할 수 있습니다.

콘솔을 사용하여 DB 인스턴스 또는 다중 AZ DB 클러스터에 대해 Performance Insights를 관리하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 데이터베이스를 선택합니다.
3. DB 인스턴스 또는 다중 AZ DB 클러스터를 선택하고 수정을 선택합니다.
4. 성능 개선 도우미를 켜려면 성능 개선 도우미 활성화를 선택합니다. 성능 개선 도우미를 끄려면 Database Insights – 표준 옵션을 선택하고 성능 개선 도우미 활성화 옵션을 선택 취소합니다.

   성능 개선 도우미 활성화를 선택하면 다음 옵션이 있습니다.

   - 보존(Database Insights의 표준 모드에만 해당) - 성능 개선 도우미 데이터를 보존하는 시간입니다. 프리 티어의 보존 설정은 기본값(7일)입니다. 성능 데이터를 더 오래 보존하려면 1~24개월을 지정하십시오. 보존 기간에 대한 자세한 내용은 성능 개선 도우미의 요금 및 데이터 보존 섹션을 참조하세요.
   - AWS KMS key - KMS 키를 지정합니다. 성능 개선 도우미는 KMS 키를 사용하여 잠재적으로 민감한 데이터를 모두 암호화합니다. 데이터는 암호화된 상태로 전송 및 저장됩니다. 자세한 내용은 Amazon RDS 리소스 암호화 섹션을 참조하세요.
5. Continue(계속)를 선택합니다.
6. Scheduling of Modifications(수정 사항 예약)에 대해 Apply immediately(즉시 적용)를 선택합니다. 예약된 다음 유지 관리 윈도우에 적용을 선택하면 인스턴스에서 이 설정을 무시하고 성능 개선 도우미를 즉시 활성화합니다.
7. Modify Instance(인스턴스 수정)를 선택합니다.

AWS CLI

create-db-instance AWS CLI 명령을 사용하는 경우, `--enable-performance-insights`를 지정하고 `--database-insights-mode`를 `advanced` 또는 `standard`로 설정하여 성능 개선 도우미를 활성화합니다. 성능 개선 도우미를 비활성화하려면 `--no-enable-performance-insights`를 지정하고 `database-insights-mode`를 `standard`로 설정합니다.

다음 AWS CLI 명령을 사용해 다음 값을 지정할 수도 있습니다.

- create-db-cluster
- modify-db-cluster
- create-db-instance-read-replica
- modify-db-instance
- restore-db-instance-from-s3

CLI에서 성능 개선 도우미를 활성화할 때 필요에 따라 `--performance-insights-retention-period` 옵션을 사용하여 성능 개선 도우미 데이터를 보존할 시간을 일 단위로 지정할 수 있습니다. 7, # * 31(#은 1~23 사이의 숫자여야 함) 또는 731이라고 지정할 수 있습니다. 예를 들어 성능 데이터를 3개월 동안 보존하려면 3에 31을 곱한 93을 지정하면 됩니다. 기본값은 7일입니다. 보존 기간에 대한 자세한 내용은 성능 개선 도우미의 요금 및 데이터 보존 섹션을 참조하세요.

다음 예에서는 `sample-db-cluster`의 성능 개선 도우미를 활성화하고 성능 개선 도우미 데이터를 93일(3개월) 동안 보존하도록 지정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-cluster \
 --database-insights-mode standard \
    --db-cluster-identifier sample-db-instance \
    --enable-performance-insights \
    --performance-insights-retention-period 93
```

Windows의 경우:

```
aws rds modify-db-cluster ^
 --database-insights-mode standard ^
    --db-cluster-identifier sample-db-instance ^
    --enable-performance-insights ^
```

```
    --performance-insights-retention-period 93
```

보존 기간을 94일처럼 유효하지 않은 값으로 지정하면 RDS에서 오류가 발생합니다.

```
An error occurred (InvalidParameterValue) when calling the CreateDBInstance
 operation:
Invalid Performance Insights retention period. Valid values are: [7, 31, 62, 93,
 124, 155, 186, 217,
248, 279, 310, 341, 372, 403, 434, 465, 496, 527, 558, 589, 620, 651, 682, 713, 731]
```

> **Note**
>
> 성능 개선 도우미가 클러스터 수준에서 관리되지 않는 DB 클러스터의 인스턴스에 대해서만 성능 개선 도우미를 전환할 수 있습니다.

RDS API

CreateDBInstance 작업 Amazon RDS API 작업을 사용하여 새 DB 인스턴스를 생성할 때 `EnablePerformanceInsights`를 `True`로 설정하면 성능 개선 도우미가 활성화됩니다. 성능 개선 도우미를 끄려면 `EnablePerformanceInsights`를 `False`로, `DatabaseInsightsMode`를 `standard`로 설정합니다.

다음 API 작업으로 `EnablePerformanceInsights` 값을 지정할 수도 있습니다.

- CreateDBCluster(다중 AZ DB 클러스터)
- ModifyDBCluster(다중 AZ DB 클러스터)
- ModifyDBInstance
- CreateDBInstanceReadReplica
- RestoreDBInstanceFromS3

성능 개선 도우미를 활성화할 때 선택적으로 `PerformanceInsightsRetentionPeriod` 파라미터를 사용하여 성능 개선 도우미 데이터를 보존할 시간을 일 단위로 지정할 수 있습니다. 7, # * 31(#은 1~23 사이의 숫자여야 함) 또는 731이라고 지정할 수 있습니다. 예를 들어 성능 데이터를 3개월 동안 보존하려면 3에 31을 곱한 93을 지정하면 됩니다. 기본값은 7일입니다. 보존 기간에 대한 자세한 내용은 성능 개선 도우미의 요금 및 데이터 보존 섹션을 참조하세요.

Amazon RDS for MariaDB 또는 MySQL에서 성능 개선 도우미의 성능 스키마 개요

성능 스키마는 Amazon RDS for MariaDB 또는 Amazon RDS for MySQL 런타임 성능을 낮은 세부 수준에서 모니터링하기 위한 선택적 기능입니다. 성능 스키마는 데이터베이스 성능에 미치는 영향을 최소화하도록 설계되었습니다. 성능 개선 도우미는 성능 스키마 사용 여부와 상관없이 사용할 수 있는 별도의 기능입니다.

주제

- 성능 스키마 개요
- 성능 개선 도우미 및 성능 스키마
- 성능 개선 도우미의 성능 스키마 자동 관리
- 성능 스키마에서 리부팅할 때의 효과
- 성능 개선 도우미의 성능 스키마 관리 여부 확인
- Amazon RDS for MariaDB 또는 MySQL에서 성능 스키마 활성화

성능 스키마 개요

성능 스키마가 MariaDB 및 MySQL 데이터베이스의 이벤트를 모니터링합니다. 이벤트는 시간을 소비하고 타이밍 정보를 수집할 수 있도록 계측된 데이터베이스 서버 작업입니다. 이벤트의 예는 다음과 같습니다.

- 함수 호출
- 운영 체제 대기
- SQL 실행 단계
- SQL 문 그룹

PERFORMANCE_SCHEMA 스토리지 엔진은 성능 스키마 기능을 구현하기 위한 메커니즘입니다. 이 엔진은 데이터베이스 소스 코드의 계측을 사용하여 이벤트 데이터를 수집합니다. 엔진은 수집된 이벤트를 `performance_schema` 데이터베이스의 메모리 전용 테이블에 저장합니다. 다른 테이블과 마찬가지로 `performance_schema`를 쿼리할 수 있습니다. 자세한 내용은 MySQL 참조 설명서에서 MySQL 성능 스키마를 참조하세요.

성능 개선 도우미 및 성능 스키마

성능 개선 도우미와 성능 스키마는 별개의 기능이지만 연결되어 있습니다. Amazon RDS for MariaDB 또는 MySQL의 성능 개선 도우미의 동작은 성능 스키마가 켜져 있는지 여부와 켜져 있으면 성능 개선 도우미가 성능 스키마를 자동으로 관리하는지 여부에 따라 다릅니다. 다음 표는 동작에 대한 설명입니다.

성능 스키마 켜짐	성능 개선 도우미 관리 모드	성능 개선 도우미 행동
예	자동	• 세부적인 저수준 모니터링 정보 수집 • 1초마다 활성 세션 지표 수집 • 병목 현상을 식별하는 데 사용할 수 있는 세부 대기 이벤트별로 분류된 DB 로드 표시
예	수동	• 대기 이벤트 및 SQL별 지표 수집 • 1초마다 활성 세션 지표 수집 • 삽입 및 전송과 같은 사용자 상태를 보고하므로 병목 현상을 식별하는 데 도움이 되지 않음
No	N/A	• 대기 이벤트, SQL별 지표 또는 기타 세부적인 저수준 모니터링 정보를 수집하지 않음 • 1초가 아닌 5초마다 활성 세션 지표 수집 • 삽입 및 전송과 같은 사용자 상태를 보고하므로 병목 현상을 식별하는 데 도움이 되지 않음

성능 개선 도우미의 성능 스키마 자동 관리

성능 개선 도우미를 활성화한 상태에서 Amazon RDS MariaDB 또는 Amazon RDS for MySQL DB 인스턴스를 생성하면 성능 스키마도 활성화됩니다. 이 경우 성능 개선 도우미는 성능 스키마 파라미터를 자동으로 관리합니다. 이는 권장되는 구성입니다.

성능 개선 도우미에서 성능 스키마를 자동으로 관리하는 경우 `performance_schema`의 소스는 `System default`입니다.

> **Note**
>
> t4g.medium 인스턴스 클래스에는 성능 스키마의 자동 관리가 지원되지 않습니다.

`performance_schema` 파라미터 값을 수동으로 변경한 후 나중에 자동 관리로 되돌리려면 Amazon RDS for MariaDB 또는 MySQL에서 성능 스키마 활성화 섹션을 참조하세요.

> **Important**
>
> 성능 개선 도우미가 성능 스키마를 활성화하더라도 파라미터 그룹 값은 변경되지 않습니다. 그러나 실행 중인 DB 인스턴스에 대한 값이 변경됩니다. 변경된 값을 볼 수 있는 유일한 방법은 `SHOW GLOBAL VARIABLES` 명령을 실행하는 것입니다.

성능 스키마에서 리부팅할 때의 효과

성능 개선 도우미와 성능 스키마는 DB 인스턴스 재부팅에 대한 요구 사항이 다릅니다.

성능 스키마

이 기능을 활성화하거나 비활성화하기 위해 DB 인스턴스를 재부팅해야 합니다.

성능 개선 도우미

이 기능을 활성화하거나 비활성화하기 위해 DB 인스턴스를 재부팅하지 않아도 됩니다.

현재 성능 스키마가 활성화되어 있지 않고 DB 인스턴스를 재부팅하지 않고 성능 개선 도우미를 활성화하면 성능 스키마가 활성화되지 않습니다.

성능 개선 도우미의 성능 스키마 관리 여부 확인

성능 개선 도우미가 현재 지원되는 모든 메이저 엔진 버전에 대한 성능 스키마를 관리하고 있는지 확인하려면 다음 표를 검토하세요.

performance_schema 파라미터 설정	소스 열 설정	성능 개선 도우미가 성능 스키마를 관리하는가?
`0`	`System default`	예
`0` 또는 `1`	`Modified`	No

다음 절차에서는 성능 개선 도우미가 성능 스키마를 자동으로 관리하는지 확인해 봅니다.

성능 개선 도우미가 성능 스키마를 자동으로 관리하는지 확인하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 파라미터 그룹을 선택합니다.
3. DB 인스턴스에 대한 파라미터 그룹 이름을 선택합니다.
4. 검색줄에 **`performance_schema`**를 입력합니다.
5. 소스가 시스템 기본값이고 값이 0인지 확인합니다. 그렇다면 성능 개선 도우미가 성능 스키마를 자동으로 관리하고 있습니다.

   여기에 표시된 예제에서는 성능 개선 도우미가 성능 스키마를 자동으로 관리하지 않습니다.



Amazon RDS for MariaDB 또는 MySQL에서 성능 스키마 활성화

DB 인스턴스 또는 다중 AZ DB 클러스터에 대해 성능 개선 도우미가 활성화되어 있지만 현재 성능 스키마를 관리하고 있지 않다고 가정합시다. 성능 개선 도우미가 성능 스키마를 자동으로 관리하도록 허용하려면 다음 단계를 완료하세요.

자동 관리를 위한 성능 스키마 구성 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 파라미터 그룹을 선택합니다.
3. DB 인스턴스 또는 다중 AZ DB 클러스터에 대한 파라미터 그룹 이름을 선택합니다.
4. 편집을 선택합니다.
5. 검색줄에 **performance_schema**를 입력합니다.
6. performance_schema 파라미터를 선택합니다.
7. 기본값으로 설정을 선택합니다.
8. 기본값으로 설정을 선택하여 확인합니다.
9. 변경 사항 저장(Save Changes)을 선택합니다.
10. DB 인스턴스 또는 다중 AZ DB 클러스터를 재부팅합니다.

> **Important**
>
> 성능 스키마를 활성화하거나 비활성화할 때마다 DB 인스턴스 또는 다중 AZ DB 클러스터를 재부팅해야 합니다.

인스턴스 파라미터 수정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 단원을 참조하십시오. 대시보드에 대한 자세한 내용은 성능 개선 도우미 대시보드를 사용한 지표 분석 단원을 참조하십시오. MySQL 성능 스키마에 대한 자세한 내용은 MySQL 설명서에서 MySQL Performance Schema(8.0의 경우) 및 MySQL Performance Schema(8.4의 경우)를 참조하세요.

Performance Insights에 대한 액세스 정책 구성

성능 개선 도우미에 액세스하려면 보안 주체에게 AWS Identity and Access Management(IAM)의 적절한 권한이 있어야 합니다. 다음과 같은 방법으로 액세스를 부여할 수 있습니다.

- AmazonRDSPerformanceInsightsReadOnly 관리형 정책을 권한 세트 또는 역할에 연결하여 성능 개선 도우미 API의 모든 읽기 전용 작업에 액세스할 수 있습니다. 다음의 CloudWatch 권한(GetMetricStatistics, ListMetrics 및 GetMetricData)을 연결합니다. CloudWatch 권한에 대한 자세한 내용은 Amazon CloudWatch 권한 참조를 참조하시기 바랍니다.
- AmazonRDSPerformanceInsightsFullAccess 관리형 정책을 권한 세트 또는 역할에 연결하여 성능 개선 도우미 API의 모든 작업에 액세스할 수 있습니다. 다음의 CloudWatch 권한

(GetMetricStatistics, ListMetrics 및 GetMetricData)을 연결합니다. CloudWatch 권한에 대한 자세한 내용은 Amazon CloudWatch 권한 참조를 참조하시기 바랍니다.

- 사용자 지정 IAM 정책을 생성하고 권한 세트 또는 역할에 연결합니다.

성능 개선 도우미를 활성화할 때 고객 관리형 키를 지정한 경우 계정의 사용자에게 AWS KMS key에서 kms:Decrypt 및 kms:GenerateDataKey 권한이 있는지 확인합니다.

다음 섹션에서는 IAM 보안 주체에 AWS 관리형 정책을 연결하고, 사용자 지정 IAM 정책을 생성하고, AWS KMS 정책을 변경하고, 성능 개선 도우미에 대한 세분화된 액세스 권한을 부여합니다.

주제

- IAM 보안 주체에 AmazonRDSPerformanceInsightsReadOnly 정책 연결
- IAM 보안 주체에 AmazonRDSPerformanceInsightsFullAccess 정책 연결
- 성능 개선 도우미를 위한 사용자 지정 IAM 정책 만들기
- 성능 개선 도우미를 위한 AWS KMS 정책 변경
- 성능 개선 도우미에 대한 세분화된 액세스 권한 부여

IAM 보안 주체에 AmazonRDSPerformanceInsightsReadOnly 정책 연결

AmazonRDSPerformanceInsightsReadOnly는 Amazon RDS 성능 개선 도우미 API의 모든 읽기 전용 작업에 대한 액세스 권한을 부여하는 AWS 관리형 정책입니다.

AmazonRDSPerformanceInsightsReadOnly를 권한 세트 또는 역할에 연결하면 수신자는 성능 개선 도우미를 다른 콘솔 기능과 함께 사용할 수 있습니다.

자세한 내용은 AWS 관리형 정책: AmazonRDSPerformanceInsightsReadOnly 단원을 참조하십시오.

IAM 보안 주체에 AmazonRDSPerformanceInsightsFullAccess 정책 연결

AmazonRDSPerformanceInsightsFullAccess는 Amazon RDS 성능 개선 도우미 API의 모든 작업에 대한 액세스 권한을 부여하는 AWS 관리형 정책입니다.

AmazonRDSPerformanceInsightsFullAccess를 권한 세트 또는 역할에 연결하면 수신자는 성능 개선 도우미를 다른 콘솔 기능과 함께 사용할 수 있습니다.

자세한 내용은 AWS 관리형 정책: AmazonRDSPerformanceInsightsFullAccess 단원을 참조하십시오.

성능 개선 도우미를 위한 사용자 지정 IAM 정책 만들기

`AmazonRDSPerformanceInsightsReadOnly` 또는 `AmazonRDSPerformanceInsightsFullAccess` 정책이 없는 사용자의 경우, 사용자 관리형 IAM 정책을 생성 또는 수정하여 성능 개선 도우미에 대한 액세스 권한을 부여할 수 있습니다. IAM 권한 세트 또는 역할에 이 정책을 연결하면 수신자가 성능 개선 도우미를 사용할 수 있습니다.

사용자 지정 정책을 생성하는 방법

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 Policies를 선택합니다.
3. 정책 생성을 선택합니다.
4. 정책 생성 페이지에서 JSON 옵션을 선택합니다.
5. AmazonRDSPerformanceInsightsReadOnly 또는 AmazonRDSPerformanceInsightsFullAccess 정책에 대한 AWS 관리형 정책 참조 안내서의 JSON 정책 문서 섹션에 제공된 텍스트를 복사하여 붙여넣습니다.
6. 정책 검토를 선택합니다.
7. 정책의 이름과 설명(선택 사항)을 지정한 다음 정책 검토를 선택합니다.

이제 정책을 권한 세트 또는 역할에 연결할 수 있습니다. 다음 절차에서는 이 목적으로 사용할 수 있는 사용자가 이미 있다고 가정합니다.

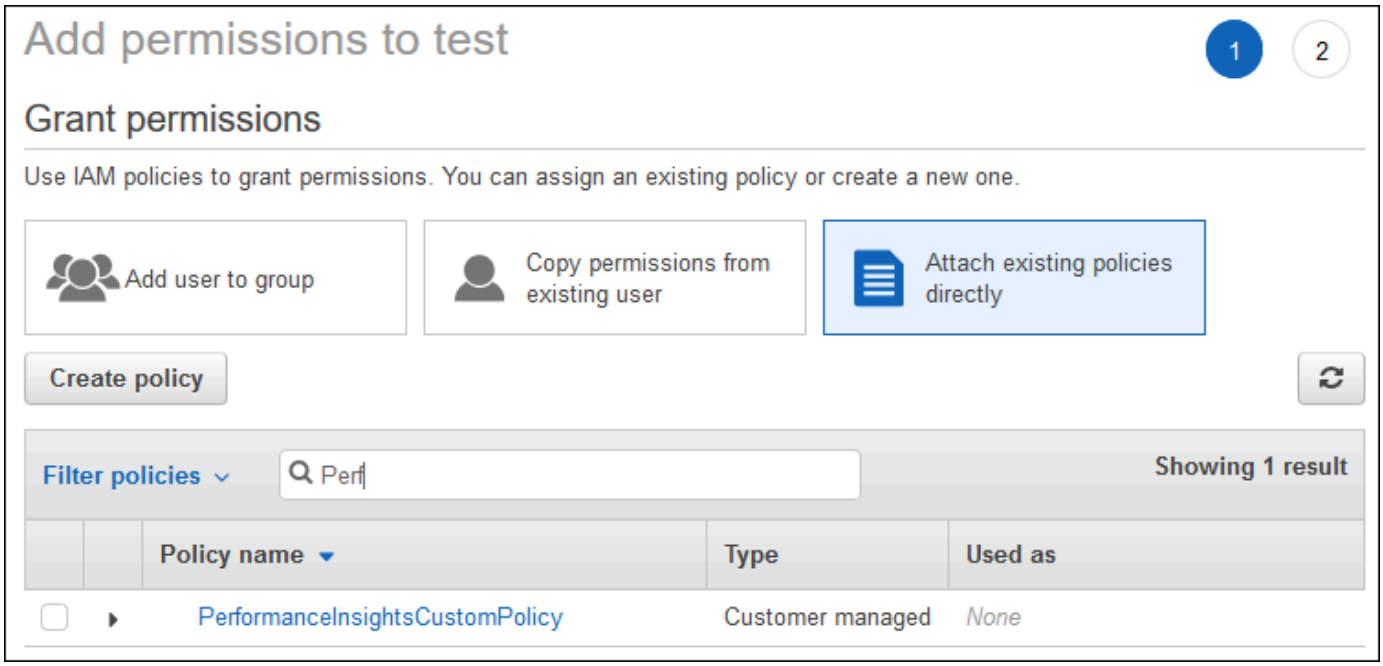
사용자에게 정책을 연결

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 목록에서 기존 사용자를 선택합니다.

> **Important**
>
> 성능 개선 도우미를 사용하려면 사용자 지정 정책 외에 Amazon RDS에 대한 액세스 권한이 있어야 합니다. 예를 들어 `AmazonRDSPerformanceInsightsReadOnly` 사전 정의 정책은 Amazon RDS에 대한 읽기 전용 액세스를 제공합니다. 자세한 내용은 정책을 사용하여 액세스 관리 섹션을 참조하세요.

4. [Summary] 페이지에서 [Add permissions]를 선택합니다.

5. 기존 정책 직접 첨부를 선택합니다. 검색에 다음 이미지와 같이 정책 이름의 첫 문자 몇 개를 입력합니다.



6. 정책을 선택하고 다음: 검토를 선택합니다.
7. 권한 추가를 선택합니다.

성능 개선 도우미를 위한 AWS KMS 정책 변경

성능 개선 도우미는 AWS KMS key을(를) 사용하여 민감한 데이터를 암호화합니다. API 또는 콘솔을 통해 성능 개선 도우미를 활성화하면 다음 중 한 가지를 수행할 수 있습니다.

- 기본 AWS 관리형 키를 선택합니다.

  Amazon RDS는 새 DB 인스턴스에 대해 AWS 관리형 키을(를) 사용합니다. Amazon RDS는 AWS 계정에 대해 AWS 관리형 키를 생성합니다. AWS 계정에 각 AWS 리전의 Amazon RDS에 대해 각기 다른 AWS 관리형 키가 있습니다.

- 고객 관리형 키를 선택합니다.

  고객 관리형 키를 지정하는 경우 성능 개선 도우미 API를 호출하는 계정의 사용자는 KMS 키에 대한 `kms:Decrypt` 및 `kms:GenerateDataKey` 권한이 필요합니다. IAM 정책을 통해 이러한 권한을 구성할 수 있습니다. 그러나 KMS 키 정책을 통해 이러한 권한을 관리하는 것이 좋습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 AWS KMS의 키 정책을 참조하세요.

Example

다음 예는 KMS 키 정책에 문을 추가하는 방법을 보여줍니다. 이러한 문을 통해 Performance Insights에 액세스할 수 있습니다. KMS 키를 사용하는 방법에 따라 일부 제한 사항을 변경할 수 있습니다. 정책에 문을 추가하기 전에 모든 문을 제거하세요.

```
{
"Version" : "2012-10-17",
 "Id" : "your-policy",
 "Statement" : [ {
    //This represents a statement that currently exists in your policy.
 }
 ....,
 //Starting here, add new statement to your policy for Performance Insights.
 //We recommend that you add one new statement for every RDS instance
{
    "Sid" : "Allow viewing RDS Performance Insights",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            //One or more principals allowed to access Performance Insights
            "arn:aws:iam::444455556666:role/Role1"
        ]
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition" : {
        "StringEquals" : {
            //Restrict access to only RDS APIs (including Performance Insights).
            //Replace region with your AWS Region.
            //For example, specify us-west-2.
            "kms:ViaService" : "rds.region.amazonaws.com"
        },
        "ForAnyValue:StringEquals": {
            //Restrict access to only data encrypted by Performance Insights.
            "kms:EncryptionContext:aws:pi:service": "rds",
            "kms:EncryptionContext:service": "pi",

            //Restrict access to a specific RDS instance.
            //The value is a DbiResourceId.
```

```
            "kms:EncryptionContext:aws:rds:db-id": "db-AAAAABBBBBCCCCDDDDDEEEEE"
        }
    }
}
```

성능 개선 도우미에서 AWS KMS 고객 관리형 키를 사용하는 방법

성능 개선 도우미는 고객 관리형 키를 사용하여 민감한 데이터를 암호화합니다. 성능 개선 도우미를 켜면 API를 통해 AWS KMS 키를 입력할 수 있습니다. 성능 개선 도우미는 이 키에 대한 KMS 권한을 생성합니다. 여기서는 키를 사용하고 민감한 데이터를 처리하는 데 필요한 작업을 수행합니다. 민감한 데이터에는 사용자, 데이터베이스, 애플리케이션, SQL 쿼리 텍스트 등의 필드가 포함됩니다. 성능 개선 도우미는 저장된 데이터와 전송 중인 데이터 모두에서 데이터가 암호화된 상태로 유지되도록 합니다.

성능 개선 도우미 IAM가 AWS KMS와 작동하는 방식

IAM은 특정 API에 권한을 부여합니다. 성능 개선 도우미에는 IAM 정책을 사용하여 제한할 수 있는 다음과 같은 공개 API가 포함됩니다.

- `DescribeDimensionKeys`
- `GetDimensionKeyDetails`
- `GetResourceMetadata`
- `GetResourceMetrics`
- `ListAvailableResourceDimensions`
- `ListAvailableResourceMetrics`

다음 API 요청을 사용하여 민감한 데이터를 가져올 수 있습니다.

- `DescribeDimensionKeys`
- `GetDimensionKeyDetails`
- `GetResourceMetrics`

API를 사용하여 민감한 데이터를 가져오는 경우 성능 개선 도우미는 호출자의 보안 인증 정보를 활용합니다. 이 확인을 통해 민감한 데이터에 대한 액세스가 KMS 키에 액세스할 수 있는 사용자로 제한됩니다.

이러한 API를 호출하려면 IAM 정책을 통해 API를 호출할 수 있는 권한과 AWS KMS 키 정책을 통해 `kms:decrypt` 작업을 직접적으로 호출할 수 있는 권한이 있어야 합니다.

GetResourceMetrics API는 민감한 데이터와 그렇지 않은 데이터를 모두 반환할 수 있습니다. 요청 파라미터로는 응답에 민감한 데이터를 포함해야 하는지 여부를 결정합니다. 요청 시 필터 또는 그룹별 파라미터에 민감한 차원이 포함된 경우 API는 민감한 데이터를 반환합니다.

GetResourceMetrics API와 함께 사용할 수 있는 차원에 관한 자세한 내용은 DimensionGroup을 참조하세요.

Example 예시

다음 예시에서는 다음 db.user 그룹에서 민감한 데이터를 요청합니다.

```
POST / HTTP/1.1
Host: <Hostname>
Accept-Encoding: identity
X-Amz-Target: PerformanceInsightsv20180227.GetResourceMetrics
Content-Type: application/x-amz-json-1.1
User-Agent: <UserAgentString>
X-Amz-Date: <Date>
Authorization: AWS4-HMAC-SHA256 Credential=<Credential>, SignedHeaders=<Headers>,
 Signature=<Signature>
Content-Length: <PayloadSizeBytes>
{
  "ServiceType": "RDS",
  "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
  "MetricQueries": [
    {
      "Metric": "db.load.avg",
      "GroupBy": {
        "Group": "db.user",
        "Limit": 2
      }
    }
  ],
  "StartTime": 1693872000,
  "EndTime": 1694044800,
  "PeriodInSeconds": 86400
}
```

Example

다음 예시에서는 다음 db.load.avg 지표에서 민감하지 않은 데이터를 요청합니다.

```
POST / HTTP/1.1
Host: <Hostname>
Accept-Encoding: identity
X-Amz-Target: PerformanceInsightsv20180227.GetResourceMetrics
Content-Type: application/x-amz-json-1.1
User-Agent: <UserAgentString>
X-Amz-Date: <Date>
Authorization: AWS4-HMAC-SHA256 Credential=<Credential>, SignedHeaders=<Headers>,
 Signature=<Signature>
Content-Length: <PayloadSizeBytes>
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "MetricQueries": [
        {
            "Metric": "db.load.avg"
        }
    ],
    "StartTime": 1693872000,
    "EndTime": 1694044800,
    "PeriodInSeconds": 86400
}
```

성능 개선 도우미에 대한 세분화된 액세스 권한 부여

세분화된 액세스 제어를 사용하면 추가적인 방법으로 성능 개선 도우미에 대한 액세스를 제어할 수 있습니다. 이 액세스 제어는 `GetResourceMetrics`, `DescribeDimensionKeys` 및 `GetDimensionKeyDetails` 성능 개선 도우미 작업의 개별 차원에 대한 액세스를 허용하거나 거부할 수 있습니다. 세분화된 액세스를 사용하려면 조건 키를 사용하여 IAM 정책에서 차원을 지정하세요. 액세스 평가는 IAM 정책 평가 로직을 따릅니다. 자세한 내용은 IAM 사용 설명서의 정책 평가 로직을 참조하세요. IAM 정책 문에 차원이 지정되지 않은 경우 해당 문은 지정된 작업의 모든 차원에 대한 액세스를 제어합니다. 사용 가능한 차원 목록은 DimensionGroup 섹션을 참조하세요.

자격 증명에 액세스 권한이 부여된 차원을 확인하려면 `ListAvailableResourceDimensions`에서 `AuthorizedActions` 파라미터를 사용하고 작업을 지정하세요. `AuthorizedActions`에 허용되는 값은 다음과 같습니다.

- `GetResourceMetrics`
- `DescribeDimensionKeys`

- GetDimensionKeyDetails

예를 들어 AuthorizedActions 파라미터에 GetResourceMetrics를 지정하면 ListAvailableResourceDimensions는 GetResourceMetrics 작업에 액세스 권한이 부여된 차원 목록이 반환됩니다. AuthorizedActions 파라미터에 여러 작업을 지정하는 경우 ListAvailableResourceDimensions는 해당 작업에 액세스 권한이 부여된 차원의 교차 항목을 반환합니다.

Example

다음 예시에서는 GetResourceMetrics 및 DescribeDimensionKeys 작업에 지정된 차원에 대한 액세스를 제공합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowToDiscoverDimensions",
            "Effect": "Allow",
            "Action": [
                "pi:ListAvailableResourceDimensions"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ]
        },
        {
            "Sid": "SingleAllow",
            "Effect": "Allow",
            "Action": [
                "pi:GetResourceMetrics",
                "pi:DescribeDimensionKeys"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    // only these dimensions are allowed. Dimensions not included in
                    // a policy with "Allow" effect will be denied
```

```
                    "pi:Dimensions": [
                        "db.sql_tokenized.id",
                        "db.sql_tokenized.statement"
                    ]
                }
            }
        }


    ]
}
```

요청된 차원에 대한 응답은 다음과 같습니다.

```
 // ListAvailableResourceDimensions API
// Request
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "Metrics": [ "db.load" ],
    "AuthorizedActions": ["DescribeDimensionKeys"]
}

// Response
{
    "MetricDimensions": [ {
        "Metric": "db.load",
        "Groups": [
            {
                "Group": "db.sql_tokenized",
                "Dimensions": [
                    { "Identifier": "db.sql_tokenized.id" },
                  //  { "Identifier": "db.sql_tokenized.db_id" }, // not included
 because not allows in the IAM Policy
                    { "Identifier": "db.sql_tokenized.statement" }
                ]
            }

        ] }
    ]
}
```

다음 예시에서는 차원에 대해 액세스 허용 1개와 거부 액세스 거부 2개를 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
          {
            "Sid": "AllowToDiscoverDimensions",
            "Effect": "Allow",
            "Action": [
                "pi:ListAvailableResourceDimensions"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ]
          },

          {
            "Sid": "O01AllowAllWithoutSpecifyingDimensions",
            "Effect": "Allow",
            "Action": [
                "pi:GetResourceMetrics",
                "pi:DescribeDimensionKeys"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ]
        },

        {
            "Sid": "O01DenyAppDimensionForAll",
            "Effect": "Deny",
            "Action": [
                "pi:GetResourceMetrics",
                "pi:DescribeDimensionKeys"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "pi:Dimensions": [
```

```
                        "db.application.name"
                    ]
                }
            }
        },

        {
            "Sid": "O01DenySQLForGetResourceMetrics",
            "Effect": "Deny",
            "Action": [
                "pi:GetResourceMetrics"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "pi:Dimensions": [
                        "db.sql_tokenized.statement"
                    ]
                }
            }
        }
    ]
}
```

요청된 차원에 대한 응답은 다음과 같습니다.

```
   // ListAvailableResourceDimensions API
// Request
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "Metrics": [ "db.load" ],
    "AuthorizedActions": ["GetResourceMetrics"]
}

// Response
{
    "MetricDimensions": [ {
```

```
        "Metric": "db.load",
        "Groups": [
            {
                "Group": "db.application",
                "Dimensions": [

                  // removed from response because denied by the IAM Policy
                  //  { "Identifier": "db.application.name" }
                ]
            },
            {
                "Group": "db.sql_tokenized",
                "Dimensions": [
                    { "Identifier": "db.sql_tokenized.id" },
                    { "Identifier": "db.sql_tokenized.db_id" },

                  // removed from response because denied by the IAM Policy
                  //  { "Identifier": "db.sql_tokenized.statement" }
                ]
            },
            ...
        ] }
    ]
}
```

```
// ListAvailableResourceDimensions API
// Request
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "Metrics": [ "db.load" ],
    "AuthorizedActions": ["DescribeDimensionKeys"]
}

// Response
{
    "MetricDimensions": [ {
        "Metric": "db.load",
        "Groups": [
            {
                "Group": "db.application",
                "Dimensions": [
```

```
                // removed from response because denied by the IAM Policy
                //  { "Identifier": "db.application.name" }
              ]
          },
          {
              "Group": "db.sql_tokenized",
              "Dimensions": [
                  { "Identifier": "db.sql_tokenized.id" },
                  { "Identifier": "db.sql_tokenized.db_id" },

                // allowed for DescribeDimensionKeys because our IAM Policy
                // denies it only for GetResourceMetrics
                  { "Identifier": "db.sql_tokenized.statement" }
              ]
          },
          ...
      ] }
  ]
}
```

성능 개선 도우미 대시보드를 사용한 지표 분석

Important

AWS는 Performance Insights의 수명 종료일을 2025년 11월 30일로 발표했습니다. 이 날짜 이후에는 Amazon RDS가 더 이상 Performance Insights 콘솔 환경, 유연한 보존 기간(1~24개월) 및 연결 요금을 지원하지 않습니다. Performance Insights API는 요금 변경 없이 계속 제공됩니다. Performance Insights API 비용은 CloudWatch Database Insights 비용과 함께 AWS 청구서에 표시됩니다.

Performance Insights의 유료 티어를 사용하는 DB 인스턴스를 2025년 11월 30일 이전에 Database Insights의 고급 모드로 업그레이드하는 것이 좋습니다. Database Insights의 고급 모드로 업그레이드하는 방법에 대한 자세한 내용은 Amazon RDS에 Database Insights의 고급 모드 켜기 섹션을 참조하세요.

조치를 취하지 않으면 Performance Insights를 사용하는 DB 인스턴스가 기본적으로 Database Insights의 표준 모드를 사용하게 됩니다. Database Insights의 표준 모드를 사용하는 경우 7일이 지난 성능 데이터 기록에 액세스할 수 없으며 Amazon RDS 콘솔에서 실행 계획 및 온디맨드 분석 기능을 사용하지 못할 수 있습니다. 2025년 11월 30일 이후에는 Database Insights의 고급 모드만 실행 계획과 온디맨드 분석을 지원합니다.

CloudWatch Database Insights를 사용하면 데이터베이스 플릿의 데이터베이스 로드를 모니터링하며 대규모로 성능을 분석하고 문제를 해결할 수 있습니다. Database Insights에 대한 자세

한 내용은 CloudWatch Database Insights를 사용하여 Amazon RDS 데이터베이스 모니터링 섹션을 참조하세요. 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

성능 개선 도우미 대시보드에는 성능 문제를 분석하여 해결할 수 있는 데이터베이스 성능 정보가 포함됩니다. 메인 대시보드 페이지에서 데이터베이스 부하에 대한 정보를 확인할 수 있습니다. 대기 이벤트 또는 SQL과 같은 차원을 기준으로 DB 로드를 "분할"할 수 있습니다.

성능 개선 도우미 대시보드

- 성능 개선 도우미 대시보드 개요
- 성능 개선 도우미 대시보드 액세스
- 대기 이벤트별 DB 로드 분석
- 일정 기간 동안의 데이터베이스 성능 분석
- 성능 개선 도우미의 상위 SQL 탭을 사용하여 쿼리 분석
- 상위 Oracle PDB 로드 분석
- Amazon RDS용 성능 개선 도우미 대시보드를 사용한 실행 계획 분석

성능 개선 도우미 대시보드 개요

대시보드는 성능 개선 도우미와 상호 작용하는 가장 간편한 방법입니다. 다음 예제는 PostgreSQL DB 인스턴스의 대시보드를 보여줍니다.

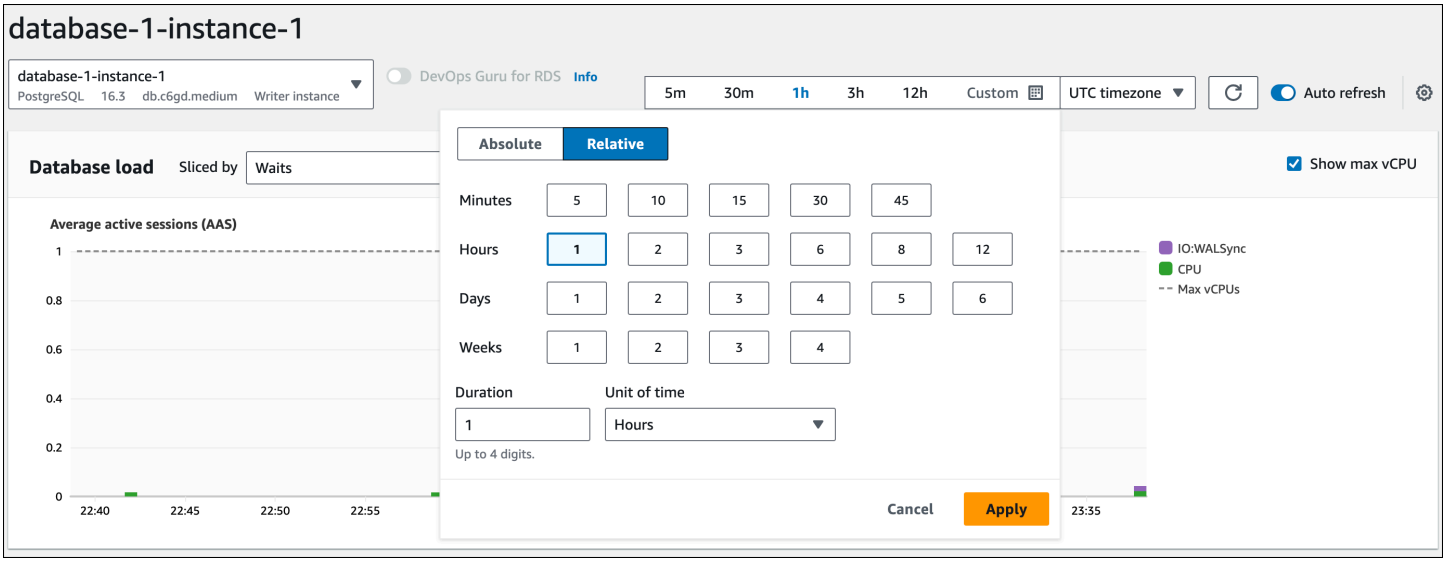


주제

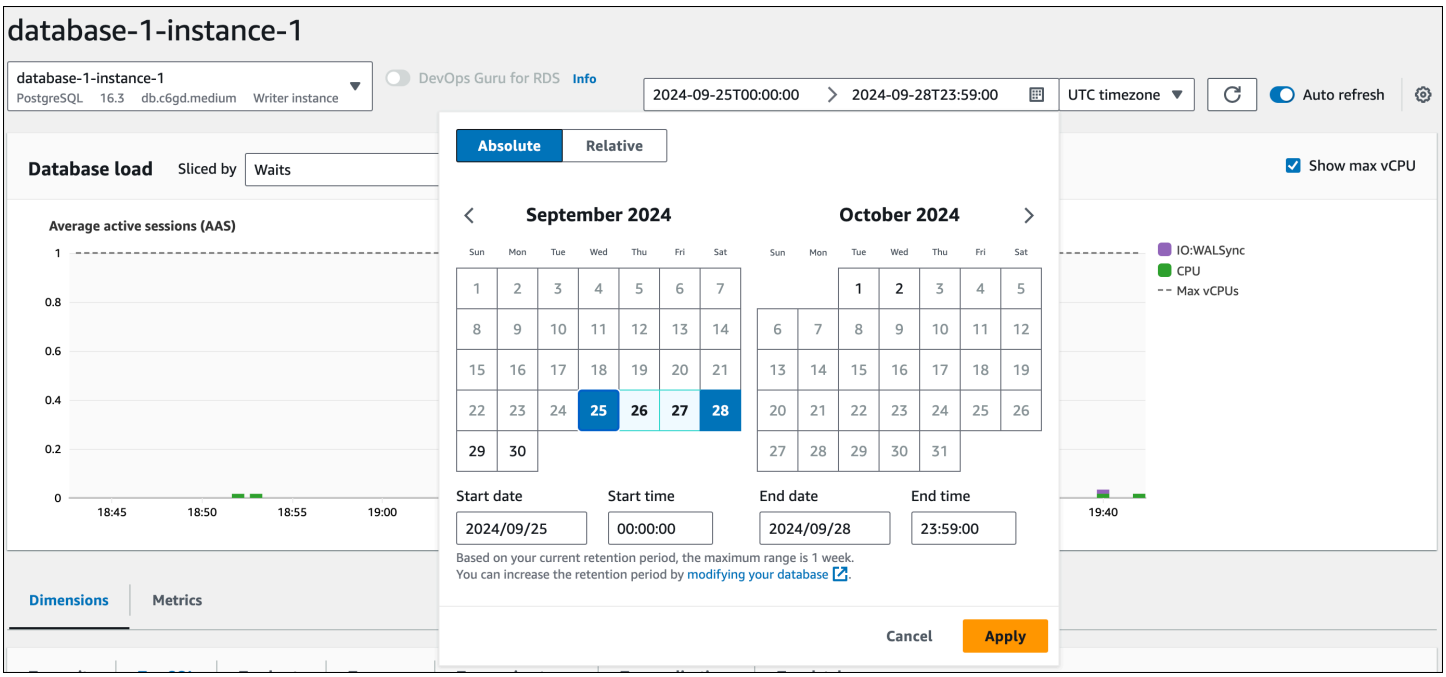
- 시간 범위 필터
- 카운터 지표 차트
- 데이터베이스 로드 차트
- 상위 측정기준 테이블

시간 범위 필터

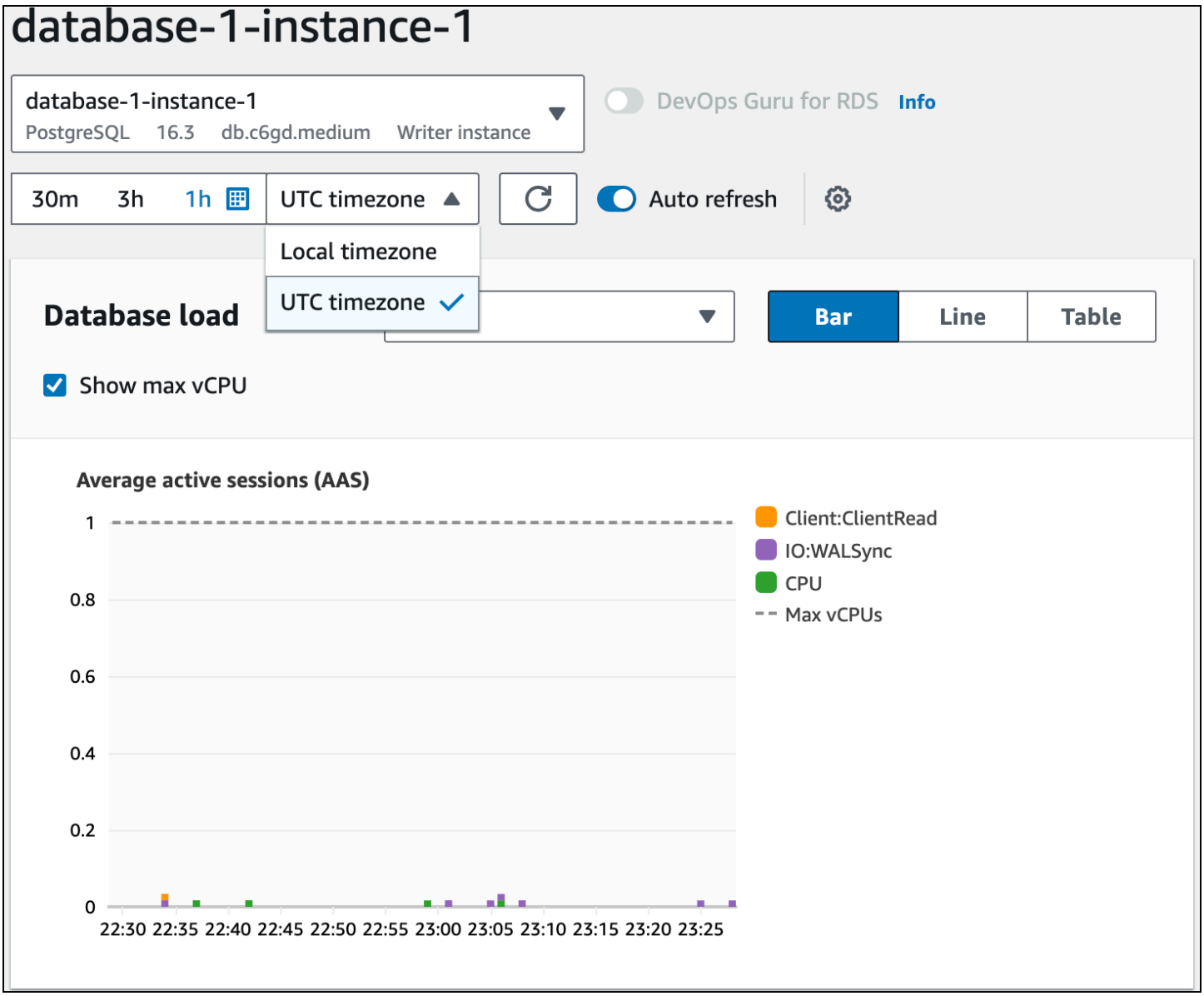
성능 개선 도우미 대시보드는 기본적으로 마지막 1시간 동안 수집된 데이터를 표시합니다. 이 범위를 최소 5분 또는 최대 2년으로 조정할 수 있습니다. 상대적인 범위를 직접 선택할 수도 있습니다.



시작 및 종료 날짜와 시간이 있는 절대 범위를 선택할 수 있습니다. 다음 예에서는 2024년 9월 25일 자정에 시작해서 2024년 9월 28일 오후 11:59에 끝나는 시간 범위를 보여줍니다.



기본적으로 성능 개선 도우미 대시보드의 시간대는 협정 세계시(UTC)입니다. 현지 시간대를 선택할 수도 있습니다.



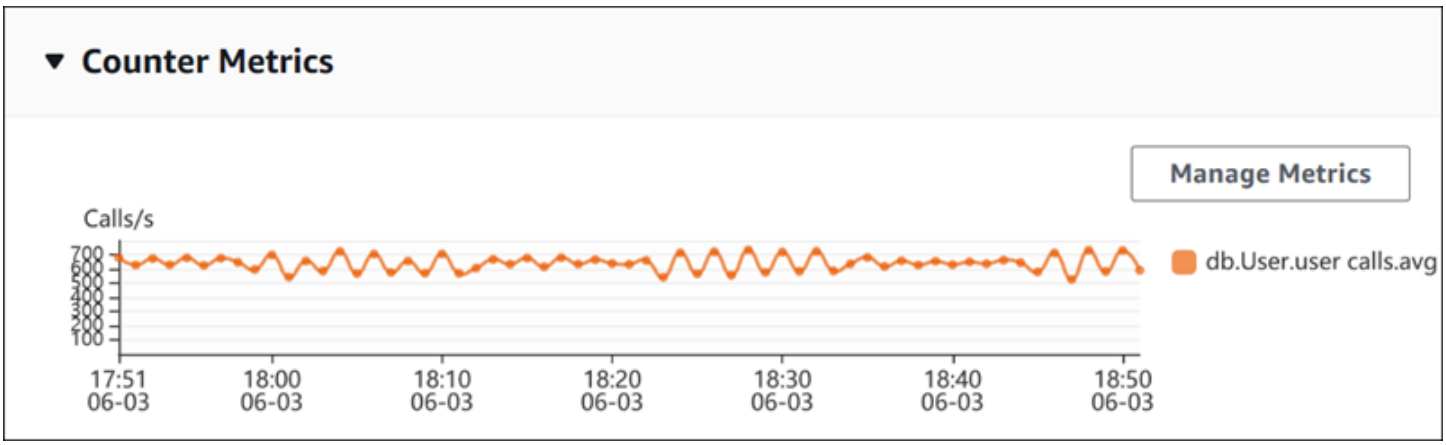
카운터 지표 차트

계수기 지표를 통해 성능 개선 도우미 대시보드에 최대 10개의 추가 그래프가 포함되도록 사용자 지정할 수 있습니다. 이 그래프에는 수십 건의 운영 체제 및 데이터베이스 성능 지표 모음이 표시됩니다. 이 정보와 데이터베이스 로드를 연관지으면 성능 문제를 식별하고 분석하는 데 도움이 됩니다.

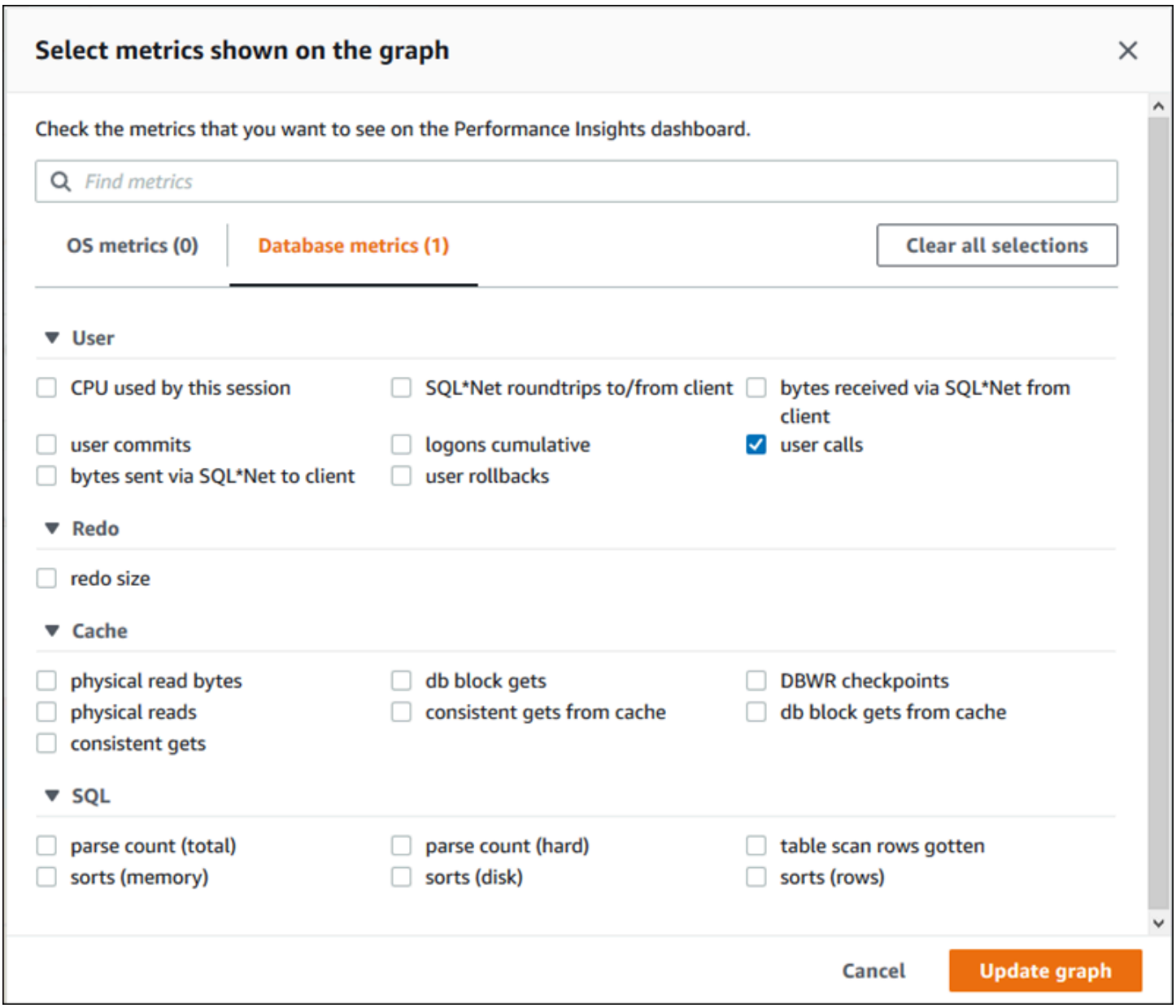
카운터 지표 차트에는 성능 카운터의 데이터가 표시됩니다. 기본 지표는 DB 엔진에 따라 다릅니다.

- MySQL 및 MariaDB - `db.SQL.Innodb_rows_read.avg`
- Oracle – `db.User.user calls.avg`
- Microsoft SQL Server - `db.Databases.Active Transactions(_Total).avg`

- PostgreSQL – `db.Transactions.xact_commit.avg`



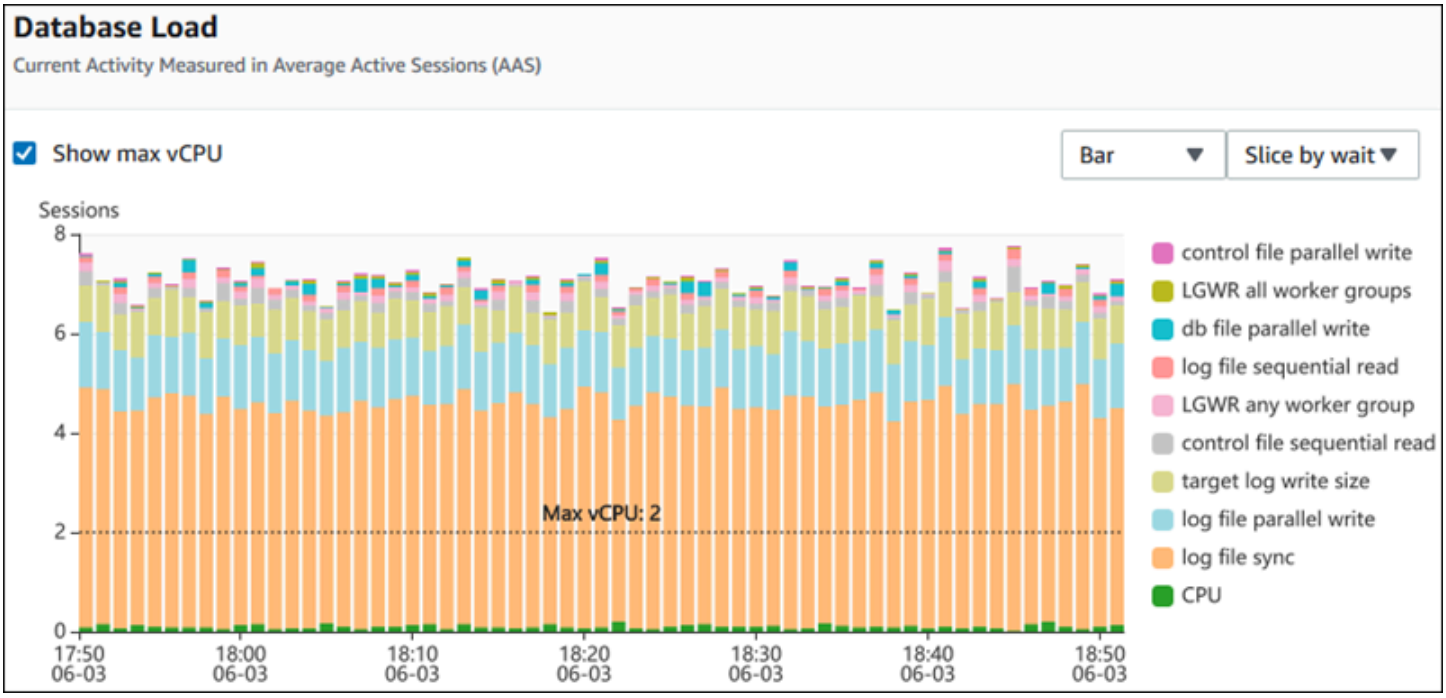
지표 관리(Manage Metrics)를 선택하여 성능 카운터를 변경합니다. 다음 스크린샷과 같이 여러 OS 지표 또는 데이터베이스 지표를 선택할 수 있습니다. 지표에 대한 세부 정보를 보려면 지표 이름 위에 마우스 포인터를 놓습니다.



각 DB 엔진에 추가할 수 있는 카운터 지표에 대한 설명은 성능 개선 도우미 카운터 섹션을 참조하세요.

데이터베이스 로드 차트

데이터베이스 로드 차트는 데이터베이스 로드와 DB 인스턴스 용량을 비교하여 최대 vCPU 선으로 표시합니다. 기본적으로 누적 꺾은선형 차트는 단위 시간당 평균 활성 세션으로 DB 로드를 나타냅니다. DB 로드는 대기 상태에 따라 슬라이스(그룹화) 됩니다.

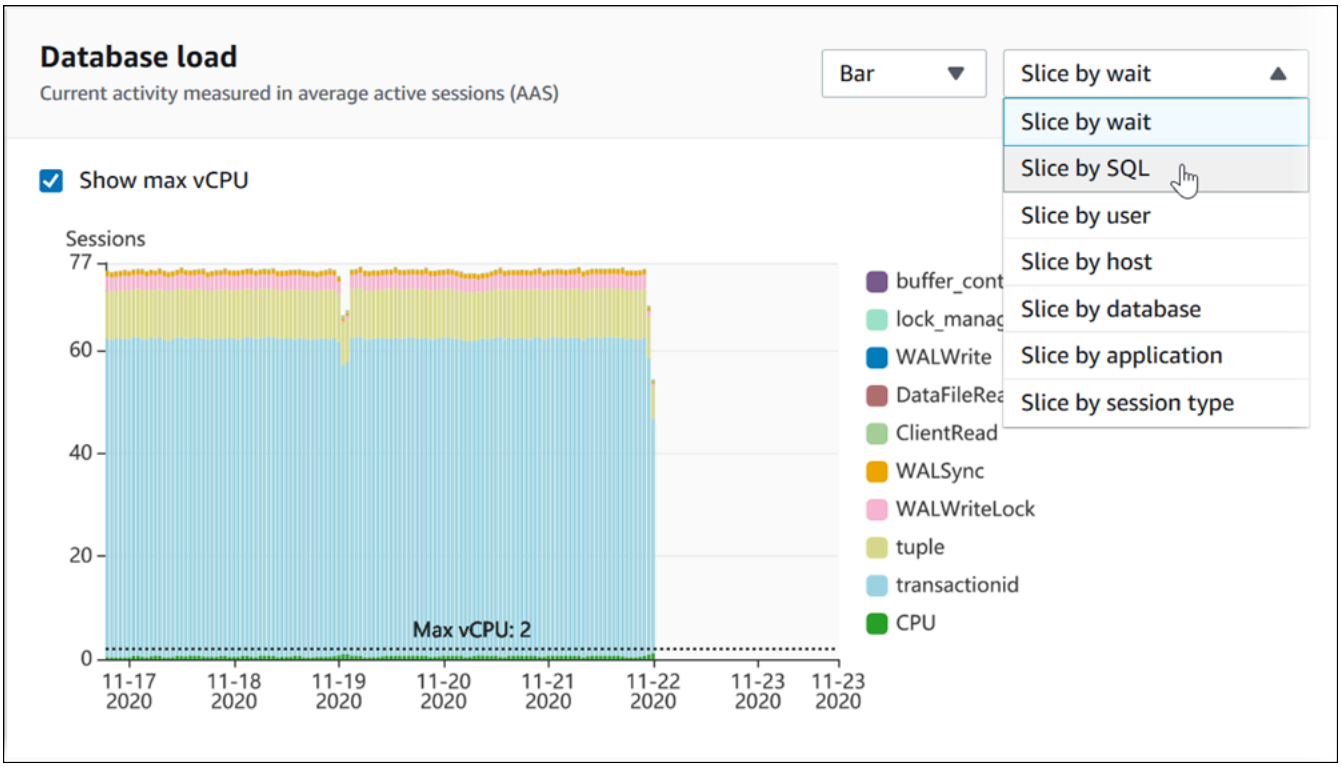


차원을 기준으로 분할된 DB 로드

지원되는 차원별로 그룹화된 활성 세션으로 로드를 표시하도록 선택할 수 있습니다. 다음 표에서는 다양한 엔진에 지원되는 차원을 보여줍니다.

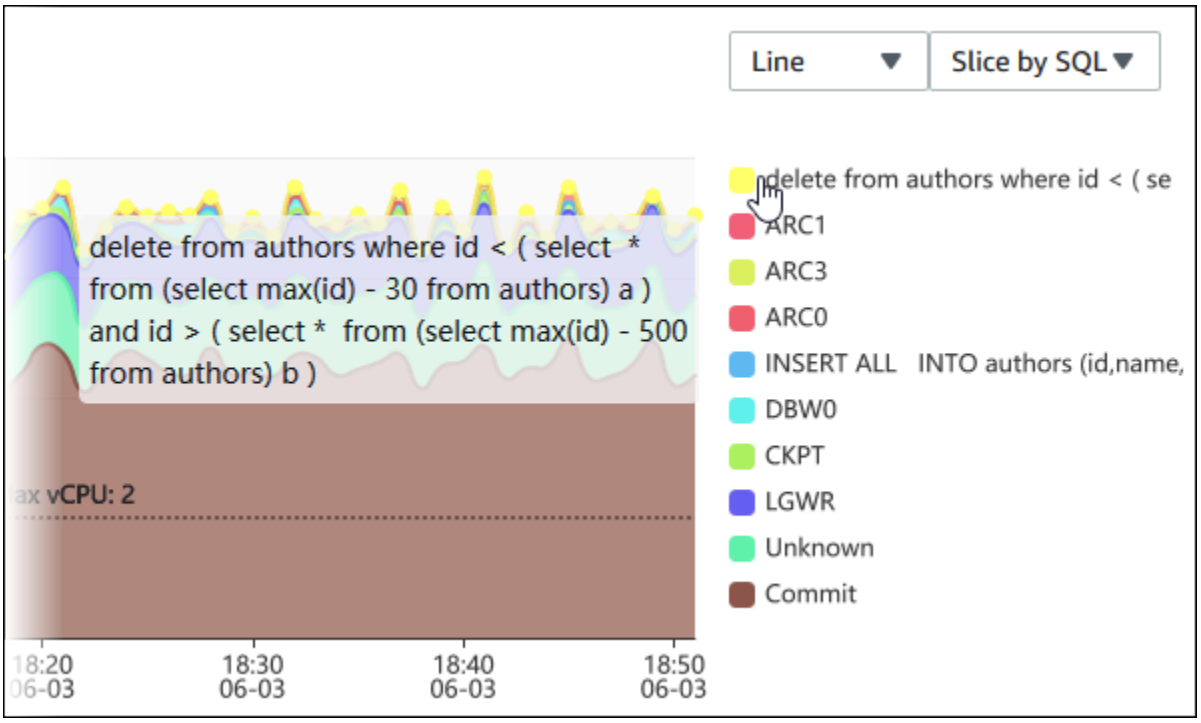
차원	Oracle	SQL Server	PostgreSQL	MySQL
Host	예	예	예	예
SQL	예	예	예	예
User	예	예	예	예
대기	예	예	예	예
계획	예	아니요	아니요	No
Application	No	아니요	예	No
데이터베이스	No	아니요	예	예
세션 유형	No	아니요	예	No

다음 이미지에서는 PostgreSQL DB 인스턴스의 차원을 보여줍니다.

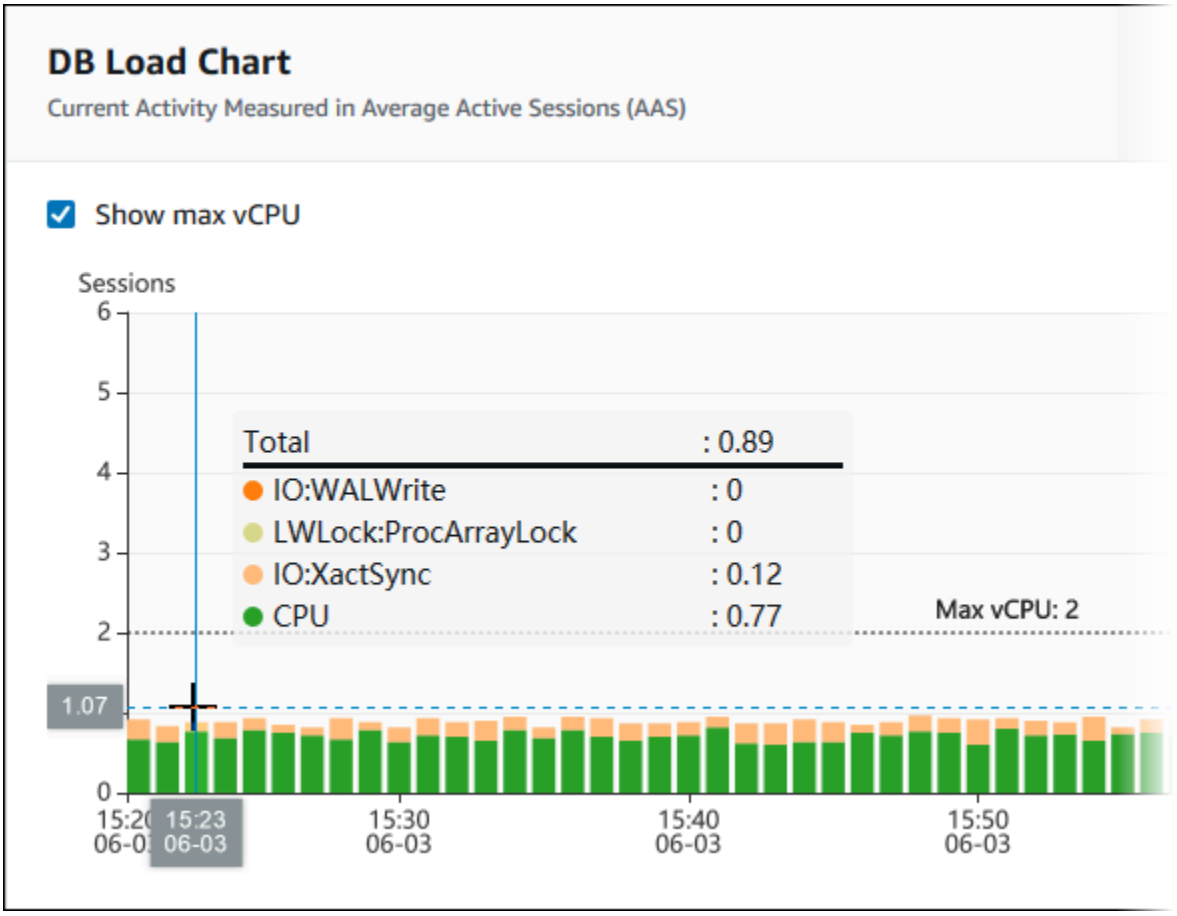


차원 항목에 대한 DB 로드 세부 정보

차원 내의 DB 로드 항목에 대한 세부 정보를 보려면 항목 이름 위로 마우스를 가져갑니다. 다음 이미지에서는 SQL 문의 세부 정보를 보여줍니다.

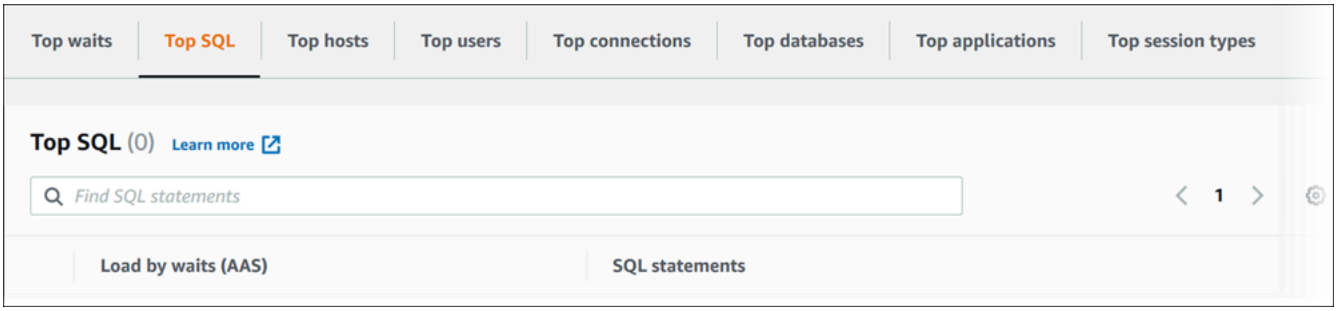


범례에서 선택한 기간의 항목에 대한 세부 정보를 보려면 해당 항목 위에 마우스 포인터를 놓습니다.



상위 측정기준 테이블

상위 측정기준 테이블은 DB 로드를 다른 차원으로 슬라이스합니다. 차원은 DB 로드의 다양한 특성에 대한 범주 또는 "분할 기준"입니다. 측정기준이 SQL인 경우 상위 SQL(Top SQL)에서는 DB 로드에 가장 많이 기여하는 SQL 문을 보여줍니다.



다음 차원 탭 중 하나를 선택합니다.

탭	설명	지원되는 엔진
상위 SQL	현재 실행 중인 SQL 문	모두
상위 대기(Top waits)	데이터베이스 백엔드가 대기 중인 이벤트	모두
상위 호스트(Top hosts)	연결된 클라이언트의 호스트 이름	모두
상위 사용자(Top users)	데이터베이스에 로그인한 사용자	모두
상위 데이터베이스(Top databases)	클라이언트가 연결된 데이터베이스의 이름	PostgreSQL, MySQL, MariaDB, SQL Server만 해당
상위 애플리케이션	데이터베이스에 연결된 애플리케이션의 이름	PostgreSQL 및 SQL Server만 해당
상위 세션 유형(Top session types)	현재 세션의 유형	PostgreSQL만

상위 SQL(Top SQL) 탭을 사용하여 쿼리를 분석하는 방법을 알아보려면 상위 SQL(Top SQL) 탭 개요 섹션을 참조하세요.

성능 개선 도우미 대시보드 액세스

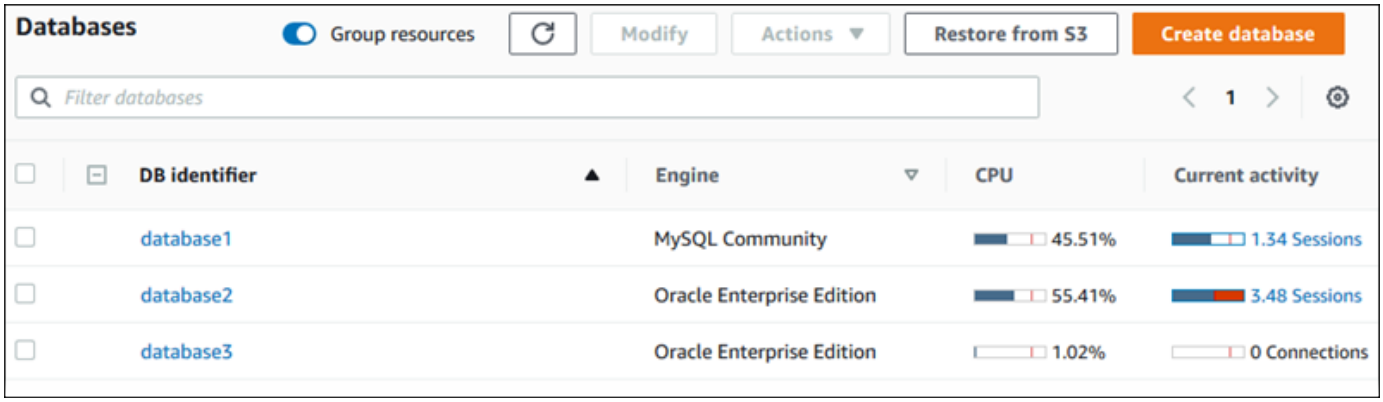
Amazon RDS는 성능 개선 도우미 대시보드에서 성능 개선 도우미 및 CloudWatch 지표에 대한 통합 보기를 제공합니다.

성능 개선 도우미 대시보드에 액세스하려면 다음 절차를 따르세요.

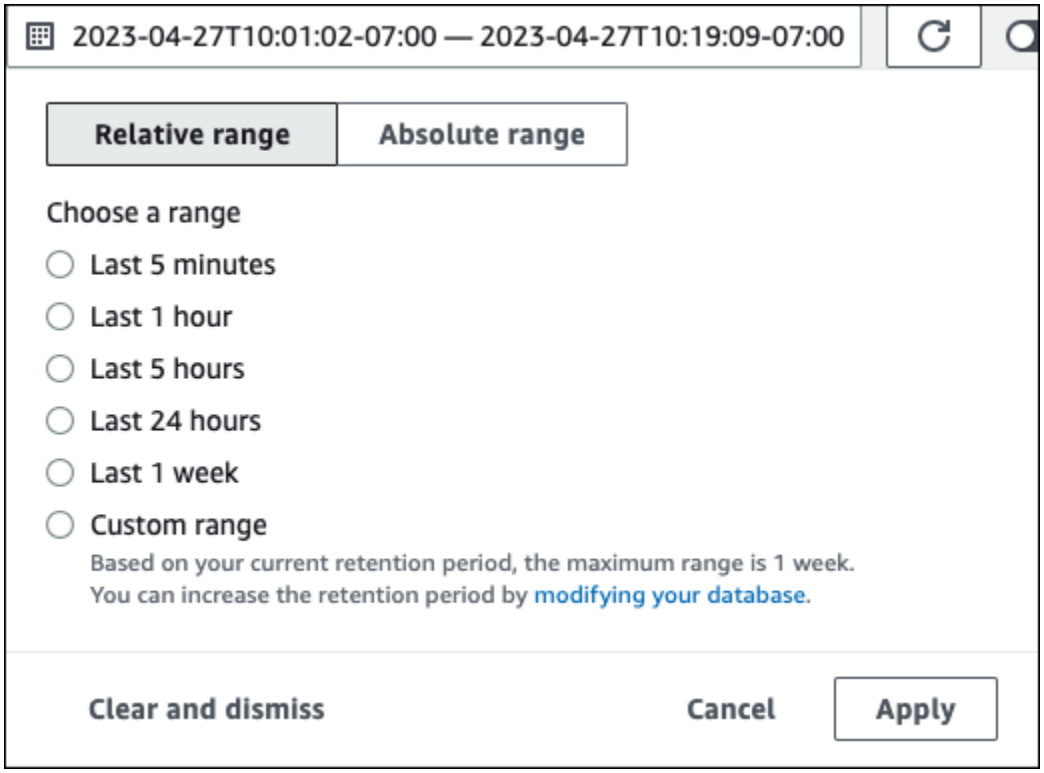
AWS 관리 콘솔에서 성능 개선 도우미 대시보드를 보려면

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 성능 개선 도우미(Performance Insights)를 선택합니다.
3. DB 인스턴스를 선택합니다.

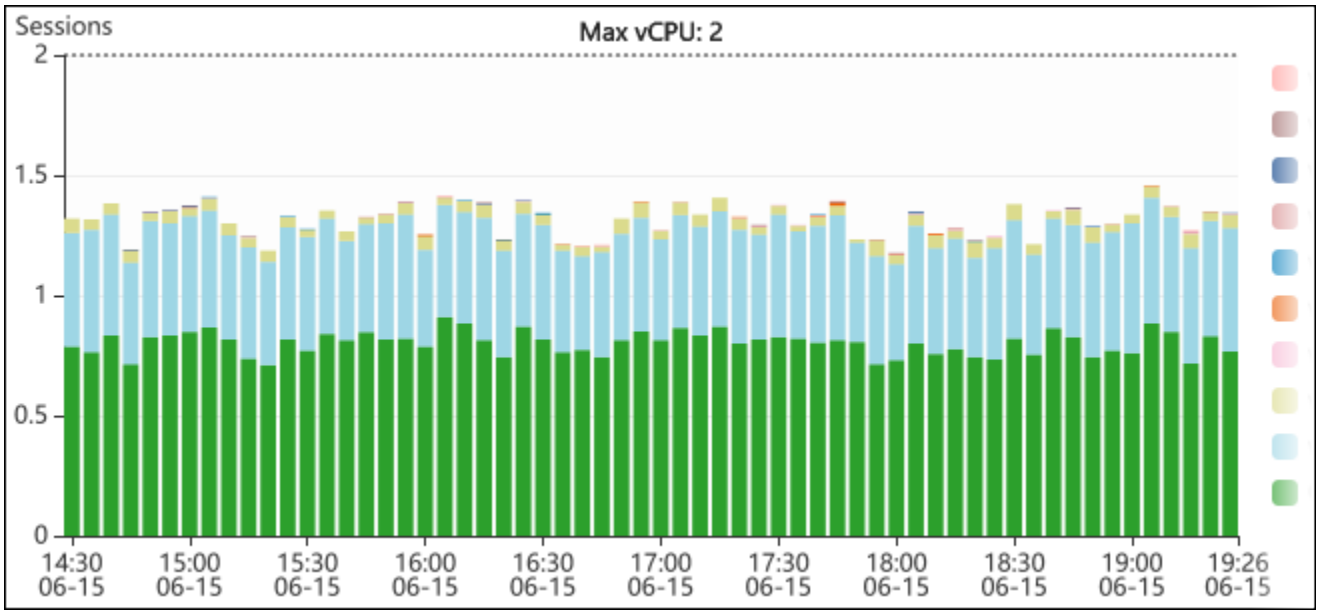
성능 개선 도우미가 켜진 DB 인스턴스의 경우, DB 인스턴스 목록에서 세션 항목을 선택하여 성능 개선 도우미 대시보드에 액세스할 수도 있습니다. 현재 활동에서 세션 항목은 지난 5분 동안 평균 활동 세션의 데이터베이스 로드를 보여 줍니다. 로드가 막대 모양으로 표시됩니다. 막대가 비어 있으면 DB 인스턴스가 유휴 상태입니다. 로드가 증가하면 막대가 파란색으로 채워집니다. 로드에서 DB 인스턴스 클래스의 가상 CPU(vCPU) 수를 전달하면 막대가 빨간색으로 바뀌고 병목 가능성을 나타냅니다.



4. (선택 사항) 오른쪽 상단의 날짜 또는 시간 범위를 선택하고 다른 상대 또는 절대 시간 간격을 지정합니다. 이제 기간을 지정하고 데이터베이스 성능 분석 보고서를 생성할 수 있습니다. 보고서는 식별된 인사이트와 권장 사항을 제공합니다. 자세한 내용은 성능 개선 도우미에서 성과 분석 보고서 생성 단원을 참조하십시오.

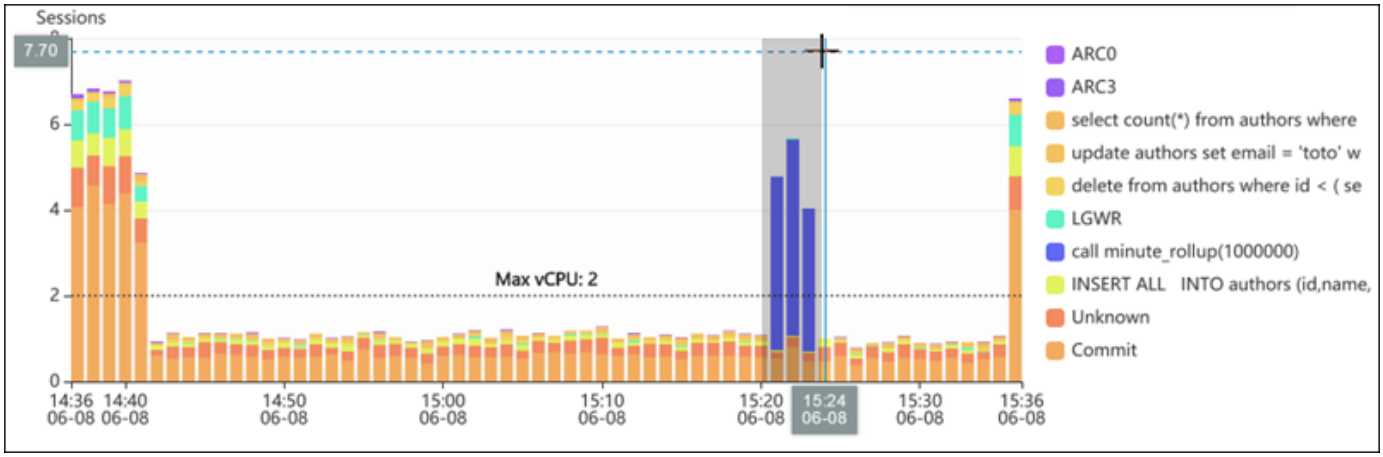


다음 스크린샷에서 DB 부하 간격은 5시간입니다.

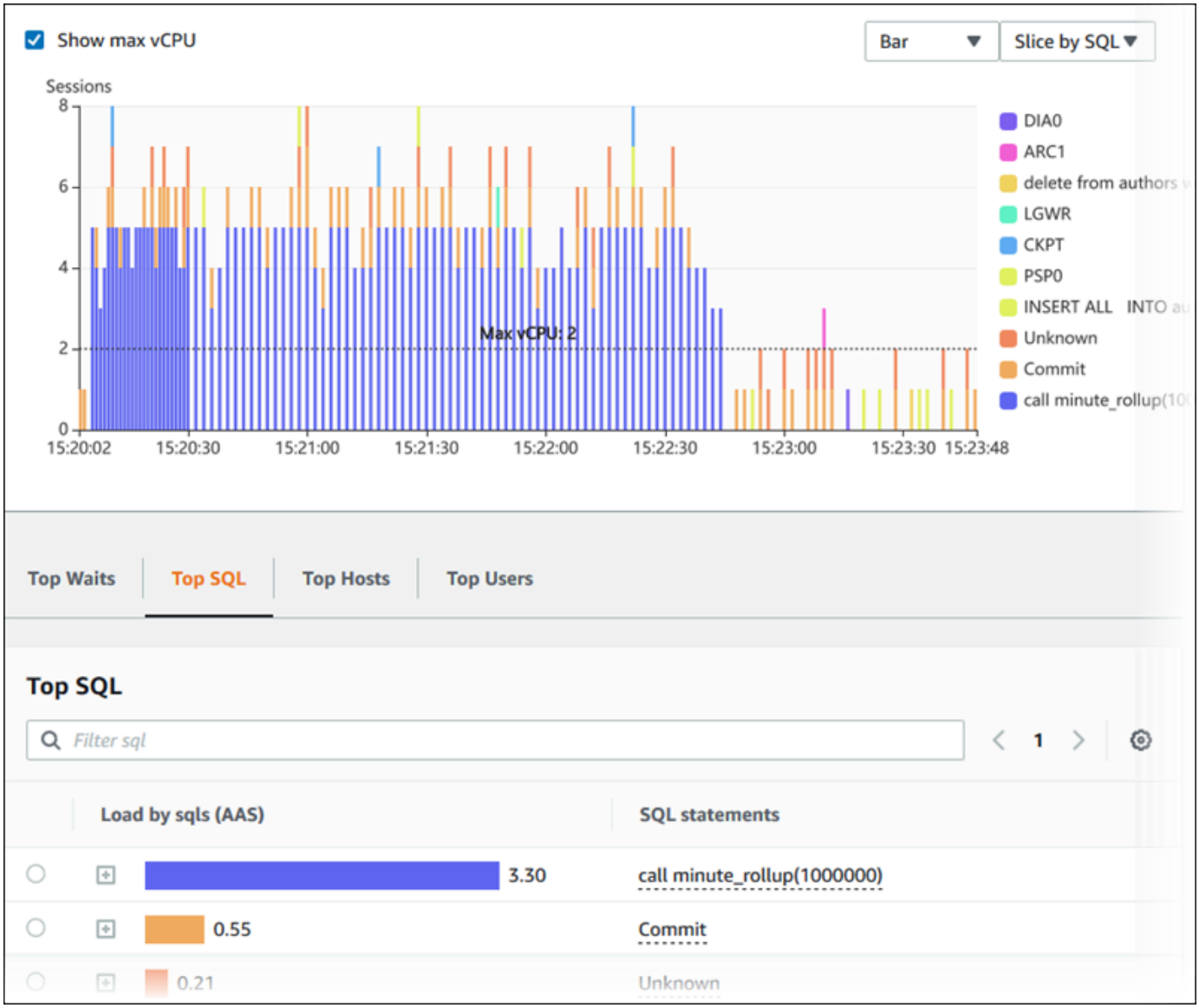


5. (선택 사항) DB 로드 차트의 한 부분을 확대하려면 시작 시간을 선택하고 원하는 기간의 끝까지 끌어옵니다.

   선택한 영역이 DB 로드 차트에서 강조 표시됩니다.



   마우스를 놓으면 DB 로드드 차트의 선택한 AWS 리전이 확대되고 상위 차원(Top dimensions) 테이블이 다시 계산됩니다.



6. (선택 사항) 데이터를 자동으로 새로 고치려면 자동 새로 고침을 선택합니다.



성능 개선 도우미 대시보드는 새 데이터로 자동으로 고쳐집니다. 새로 고침 속도는 표시되는 데이터의 양에 따라 다릅니다:

- 5분은 10초마다 새로 고침됩니다.
- 1시간은 5분마다 새로 고침됩니다.
- 5시간은 5분마다 새로 고침됩니다.
- 24시간은 30분마다 새로 고침됩니다.
- 1주는 매일 새로 고침됩니다.

- 1개월은 매일 새로 고침됩니다.

대기 이벤트별 DB 로드 분석

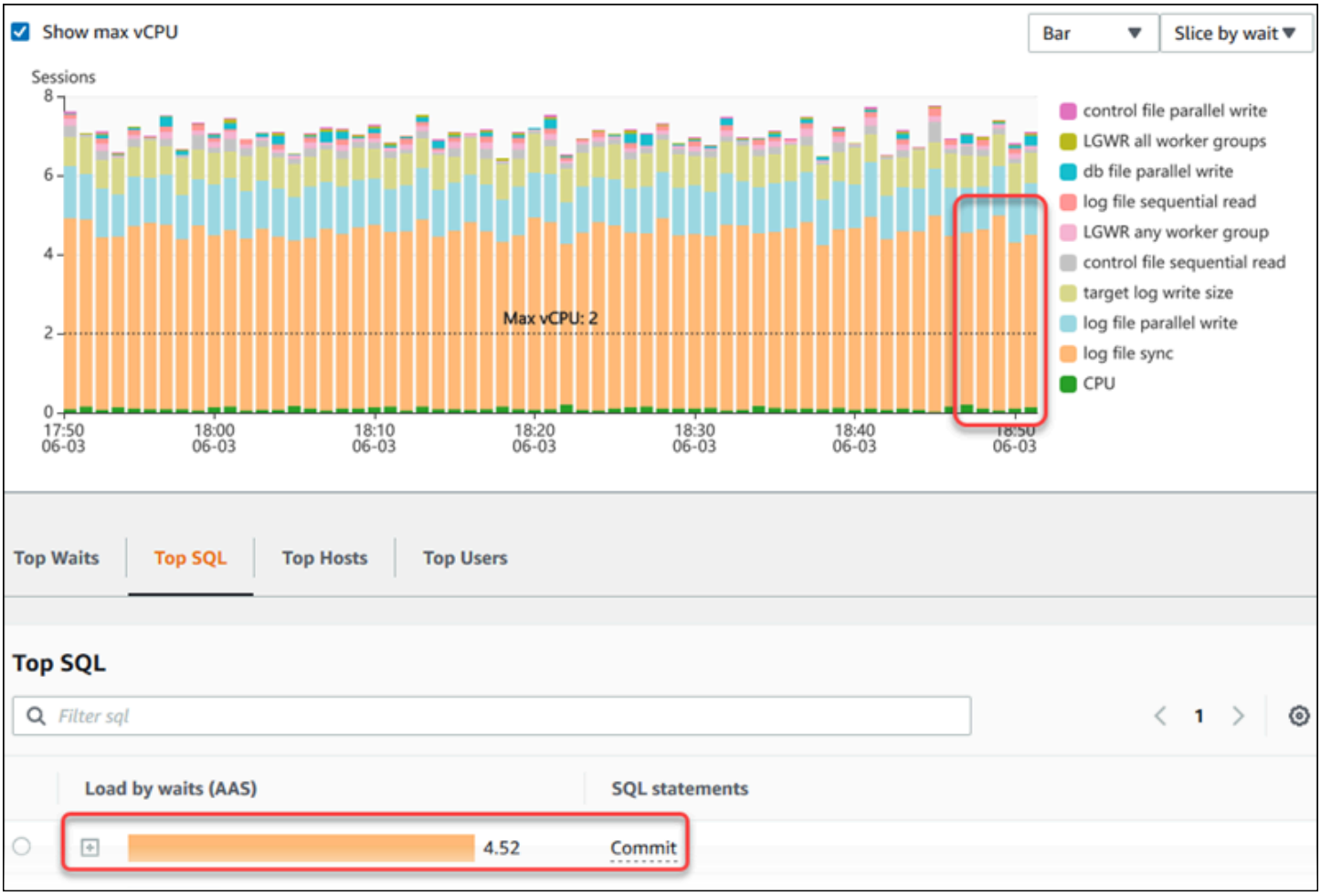
데이터베이스 로드(Database load) 차트가 병목 현상을 보일 때는 로드가 발생하는 위치를 찾아낼 수 있습니다. 이렇게 하려면 데이터베이스 로드(Database load) 차트 아래의 상위 로드 항목 테이블을 살펴보세요. SQL 쿼리나 사용자 같은 특정 항목을 선택하여 드릴다운을 통해 세부 정보까지 확인할 수 있습니다.

대기 상태와 상위 SQL 쿼리를 기준으로 구분된 DB 부하가 기본 Performance Insights 대시보드 보기입니다. 이 보기는 일반적으로 성능 문제를 가장 정확하게 파악할 수 있는 조합입니다. 대기 상태를 기준으로 구분된 DB 부하는 데이터베이스의 리소스 또는 동시성 병목 현상 유무를 표시합니다. 이 경우 상위 항목 테이블의 [SQL] 부하를 야기하는 쿼리를 표시합니다.

성능 문제를 진단하는 일반 워크플로우는 다음과 같습니다:

1. 데이터베이스 로드 차트를 보면서 데이터베이스 로드가 최대 CPU 선을 상회하는지 확인합니다.
2. 상회하는 경우가 있으면 데이터베이스 로드(Database load) 차트를 보면서 원인이 되는 대기 상태를 식별합니다.
3. 상위 부하 항목 테이블의 [SQL] 탭에서 어떤 쿼리가 대기 상태에 가장 큰 영향을 미치는지 모니터링하면서 부하를 야기하는 요약 쿼리를 식별합니다. DB Load by Wait(대기별 DB 로드) 열을 보면 이러한 요약 쿼리를 식별할 수 있습니다.
4. [SQL] 탭에서 요약 쿼리 중 하나를 선택하여 확장한 다음 구성하고 있는 하위 쿼리를 확인합니다.

예를 들어 다음 대시보드에서 로그 파일 동기화 대기 시간은 대부분의 DB 부하를 차지합니다. LGWR 모든 작업자 그룹 대기 시간도 높습니다. 상위 SQL(Top SQL) 차트는 로그 파일 동기화 대기의 원인인 자주 사용된 `COMMIT` 문을 보여줍니다. 이 경우 커밋 빈도를 줄이면 DB 부하가 줄어듭니다.



일정 기간 동안의 데이터베이스 성능 분석

일정 기간에 대한 성능 분석 보고서를 생성하여 온디맨드 분석을 통해 데이터베이스 성능을 분석합니다. 성능 분석 보고서를 확인하여 리소스 병목 현상 또는 DB 인스턴스의 쿼리 변경과 같은 성능 문제를 찾아낼 수 있습니다. 성능 개선 도우미 대시보드를 사용하면 기간을 선택하고 성능 분석 보고서를 생성할 수 있습니다. 보고서에 태그를 하나 이상 추가할 수도 있습니다.

이 기능을 사용하려면 유료 등급의 보존 기간을 사용해야 합니다. 자세한 내용은 성능 개선 도우미의 요금 및 데이터 보존 단원을 참조하세요.

보고서는 성능 분석 보고서 - 신규 탭에서 선택하여 볼 수 있습니다. 보고서에는 인사이트, 관련 지표, 성능 문제 해결을 위한 권장 사항이 포함되어 있습니다. 보고서는 성능 개선 도우미 보존 기간 동안 볼 수 있습니다.

보고서 분석 기간의 시작 시간이 보존 기간을 벗어나면 보고서가 삭제됩니다. 보존 기간이 끝나기 전에 보고서를 삭제할 수도 있습니다.

DB 인스턴스에 대한 성능 문제를 감지하고 분석 보고서를 생성하려면 성능 개선 도우미를 켜야 합니다. 성능 개선 도우미 켜기에 대한 자세한 내용은 Amazon RDS의 성능 개선 도우미 설정 및 해제 섹션을 참조하세요.

이 기능에 대한 리전, DB 엔진 및 인스턴스 클래스 지원 정보는 성능 개선 도우미 기능에 대한 Amazon RDS DB 엔진, 리전 및 인스턴스 클래스 지원 섹션을 참조하세요.

다음 섹션에서 성능 분석 보고서를 만들고, 보고, 태그를 추가하고, 삭제할 수 있습니다.

주제

- 성능 개선 도우미에서 성과 분석 보고서 생성
- 성능 개선 도우미에서 성과 분석 보고서 보기
- 성능 개선 도우미에서 성과 분석 보고서에 태그 추가
- 성능 개선 도우미에서 성과 분석 보고서 삭제

성능 개선 도우미에서 성과 분석 보고서 생성

성능 개선 도우미 대시보드에서 특정 기간에 대한 성과 분석 보고서를 만들 수 있습니다. 기간을 선택하고 하나 이상의 태그를 분석 보고서에 추가할 수 있습니다.

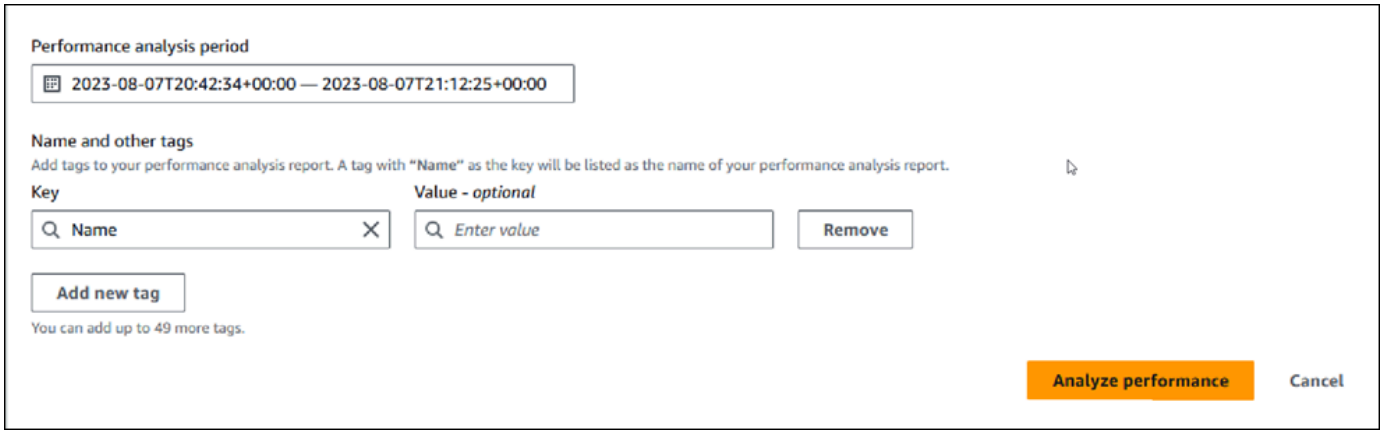
분석 기간은 5분에서 6일 사이로 선택할 수 있습니다. 분석 시작 시간 전에 최소 24시간의 성능 데이터가 있어야 합니다.

이 기능에 대한 리전, DB 엔진 및 인스턴스 클래스 지원 정보는 성능 개선 도우미 기능에 대한 Amazon RDS DB 엔진, 리전 및 인스턴스 클래스 지원 섹션을 참조하세요.

특정 기간에 대한 성과 분석 보고서를 생성하는 방법

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 성능 개선 도우미(Performance Insights)를 선택합니다.
3. DB 인스턴스를 선택합니다.
4. 성능 개선 도우미 대시보드의 데이터베이스 로드 섹션에서 성과 분석을 선택합니다.

   기간을 설정하고 성능 분석 보고서에 하나 이상의 태그를 추가하는 필드가 표시됩니다.

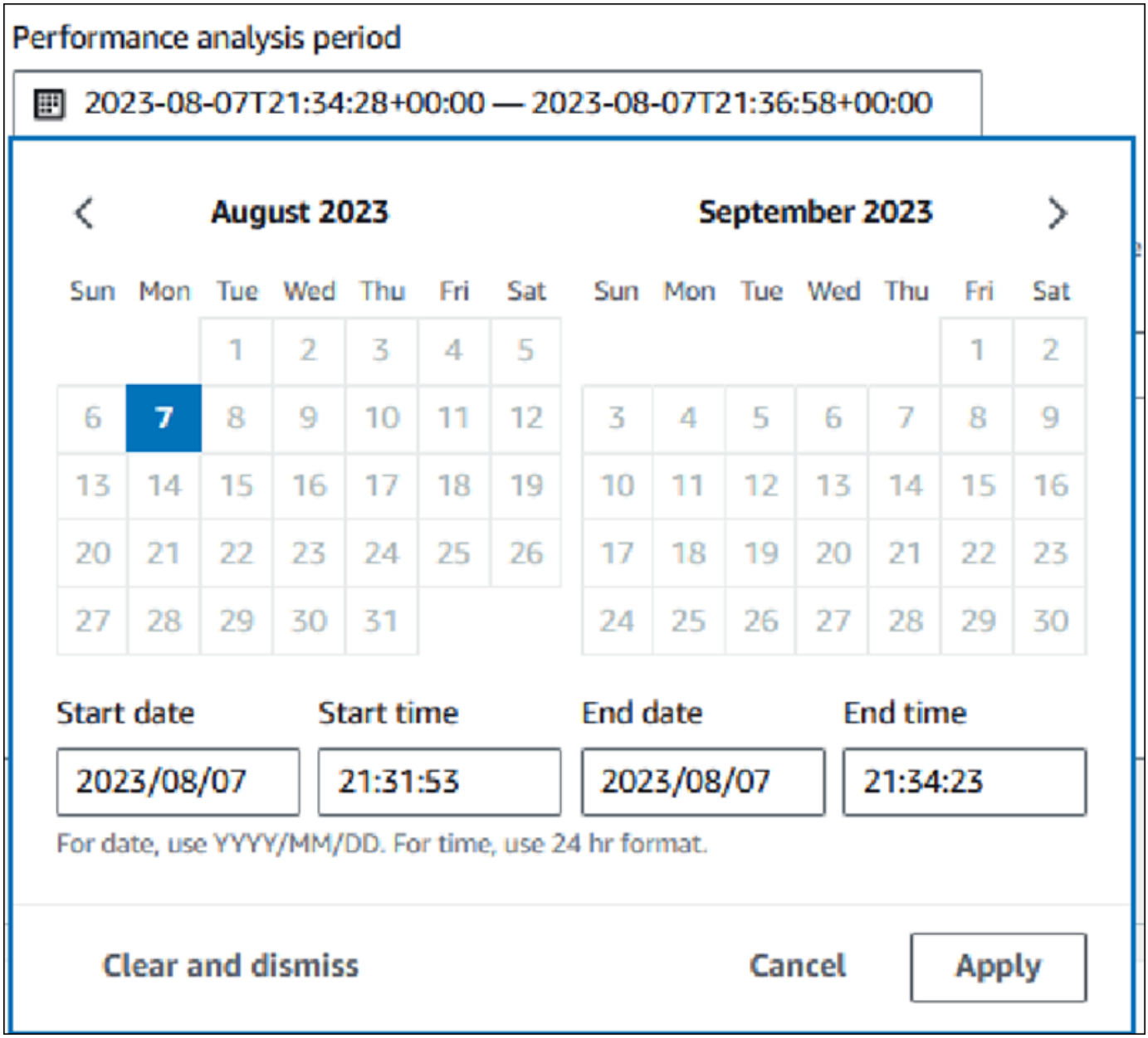


5. 기간을 선택합니다. 오른쪽 상단의 상대 범위 또는절대 범위에서 기간을 설정하면 이 기간 내의 분석 보고서 날짜 및 시간만 입력하거나 선택할 수 있습니다. 이 기간을 벗어나도록 분석 기간을 선택하면 오류 메시지가 표시됩니다.

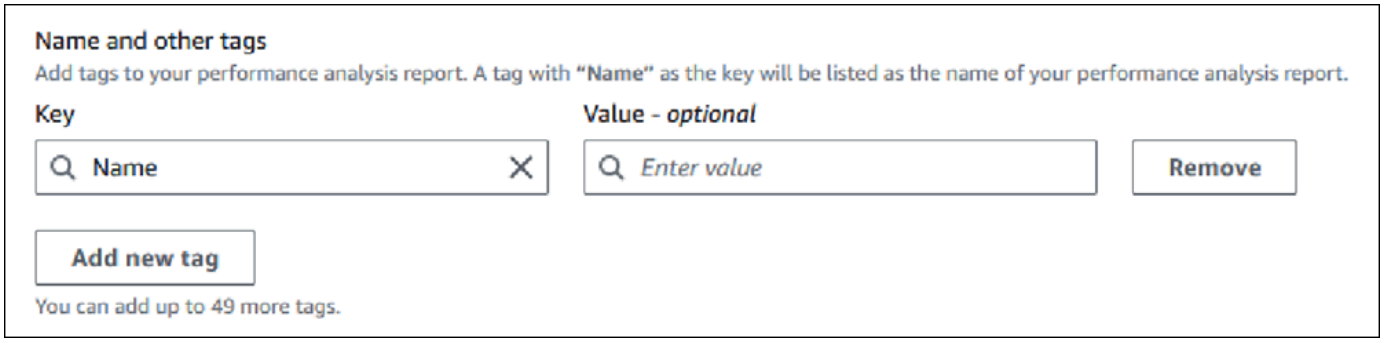
   기간을 설정하려면 다음 중 하나를 수행할 수 있습니다.

   - DB 로드 차트의 슬라이더 중 하나를 눌러 드래그합니다.

     성과 분석 기간 상자에 선택한 기간이 표시되고 DB 로드 차트에 선택한 기간이 강조 표시됩니다.
   - 성과 분석 기간 상자에서 시작 날짜, 시작 시간, 종료 날짜, 종료 시간을 선택합니다.



6. (선택 사항) 키와 값-선택 사항을 입력하여 보고서에 태그를 추가합니다.



7. 성과 분석을 선택합니다.

   배너에 보고서 생성에 성공했는지, 실패했는지 나타내는 메시지가 표시됩니다. 이 메시지에는 보고서를 볼 수 있는 링크도 제공됩니다.

   다음 예시에서는 보고서 생성 성공 메시지가 포함된 배너를 보여줍니다.



   보고서는 성능 분석 보고서 - 신규 탭에서 볼 수 있습니다.

AWS CLI를 사용하여 성과 분석 보고서를 생성할 수 있습니다. AWS CLI를 사용하여 보고서를 생성하는 방법에 대한 예는 특정 기간에 대한 성과 분석 보고서 생성 섹션을 참조하세요.

성능 개선 도우미에서 성과 분석 보고서 보기

성능 분석 보고서 - 신규 탭에 DB 인스턴스에 대해 생성된 모든 보고서가 나열됩니다. 각 보고서에는 다음 내용이 표시됩니다.

- ID: 보고서의 고유 식별자입니다.
- 이름: 보고서에 추가된 태그 키입니다.
- 보고서 생성 시간: 보고서를 생성한 시간입니다.
- 분석 시작 시간: 보고서의 분석 시작 시간입니다.
- 분석 종료 시간: 보고서의 분석 종료 시간입니다.

성능 분석 보고서를 보는 방법

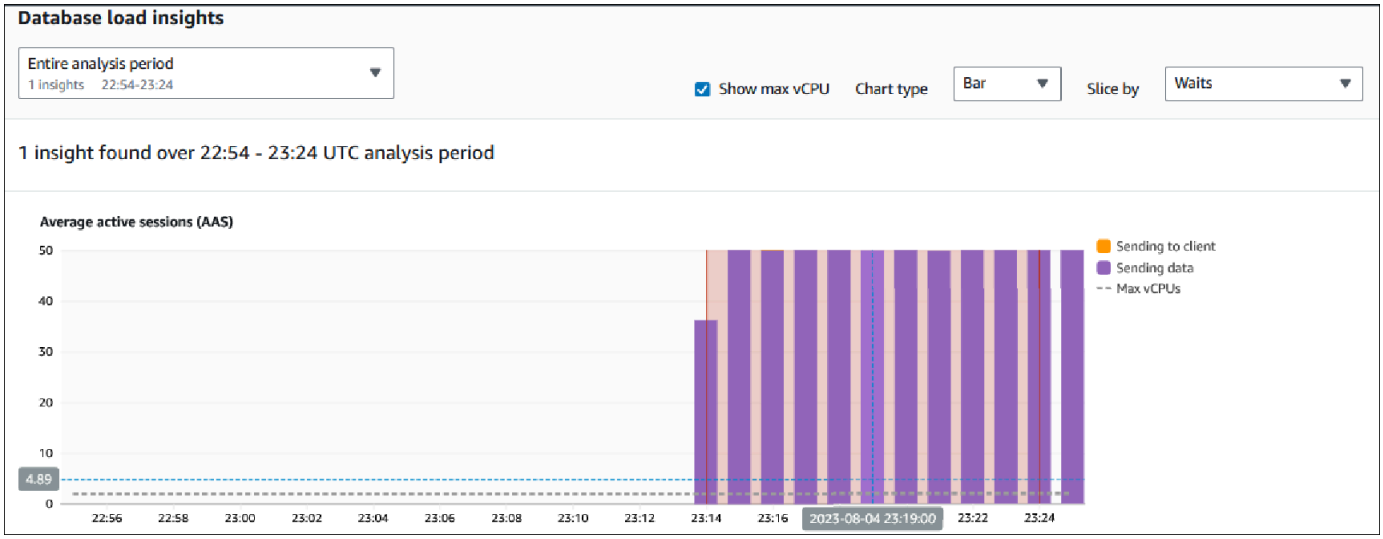
1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 성능 개선 도우미(Performance Insights)를 선택합니다.
3. 분석 보고서를 보려는 DB 인스턴스를 선택합니다.
4. 성능 개선 도우미 대시보드에서 아래로 스크롤하여 성능 분석 보고서 - 신규 탭을 선택합니다.

   서로 다른 기간에 대한 모든 분석 보고서가 표시됩니다.
5. 보려는 보고서의 ID를 선택합니다.

하나 이상의 인사이트가 식별된 경우 DB 로드 차트에는 기본적으로 전체 분석 기간이 표시됩니다. 보고서에서 하나의 인사이트를 식별한 경우 DB 로드 차트에는 기본적으로 해당 인사이트가 표시됩니다.

대시보드에는 태그 섹션에 보고서의 태그도 나열되어 있습니다.

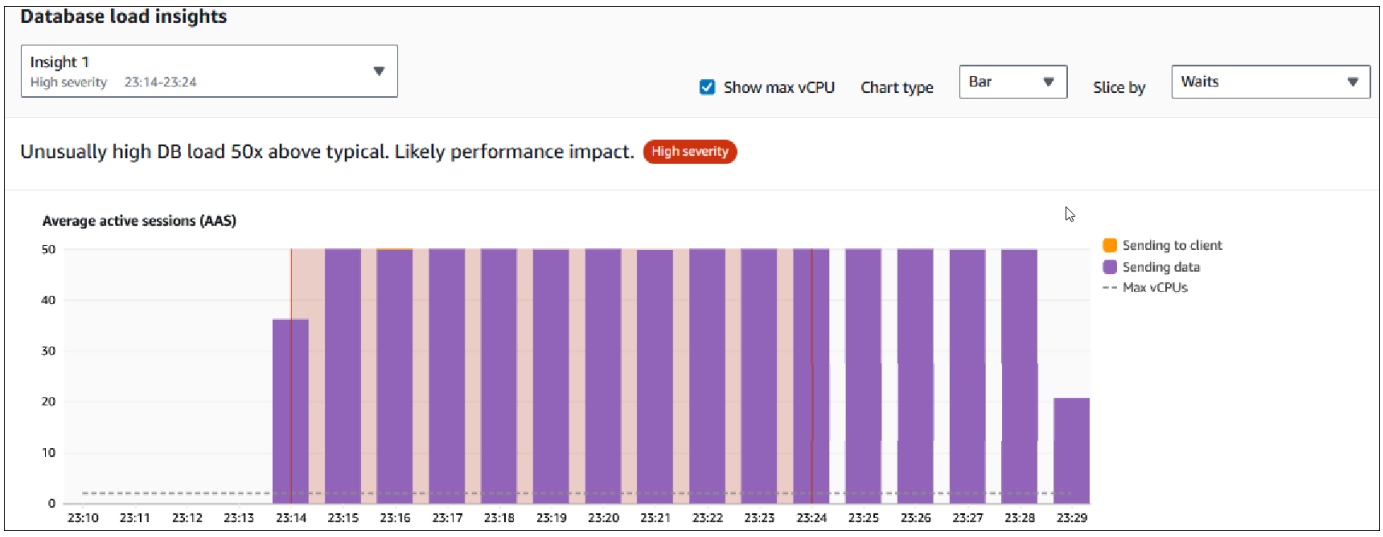
다음 예시는 보고서의 전체 분석 기간을 보여줍니다.

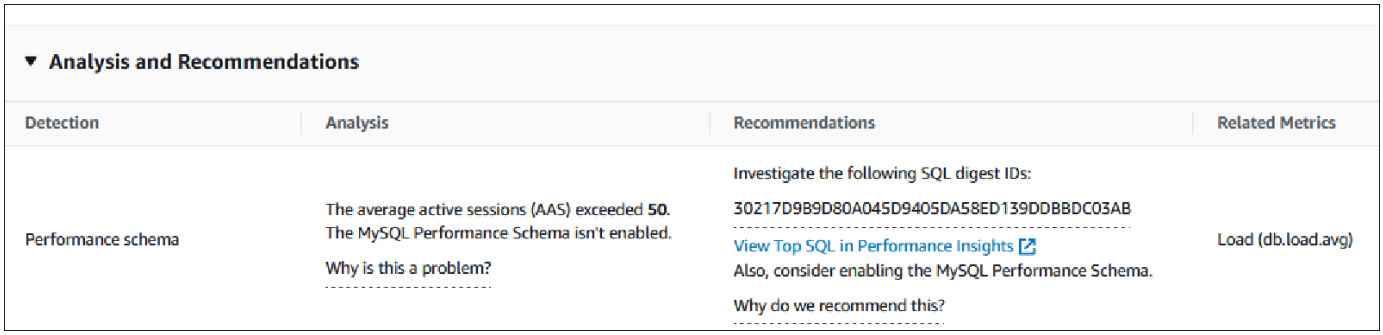


6. 보고서에 인사이트가 하나 이상 식별된 경우 데이터베이스 로드 인사이트 목록에서 보려는 인사이트를 선택하세요.

대시보드에는 인사이트 메시지, 인사이트 기간이 강조 표시된 DB 로드 차트, 분석 및 권장 사항, 보고서 태그 목록이 표시됩니다.

다음 예시는 보고서의 DB 로드 인사이트를 보여줍니다.





성능 개선 도우미에서 성과 분석 보고서에 태그 추가

보고서를 만들거나 볼 때 태그를 추가할 수 있습니다. 보고서에는 태그의 수는 최대 50개입니다.

태그를 추가하려면 권한이 있어야 합니다. 성능 개선 도우미의 액세스 정책에 대한 자세한 내용은 Performance Insights에 대한 액세스 정책 구성 섹션을 참조하세요.

보고서를 만드는 동안 하나 이상의 태그를 추가하려면 성능 개선 도우미에서 성과 분석 보고서 생성 프로시저의 6단계를 참조하세요.

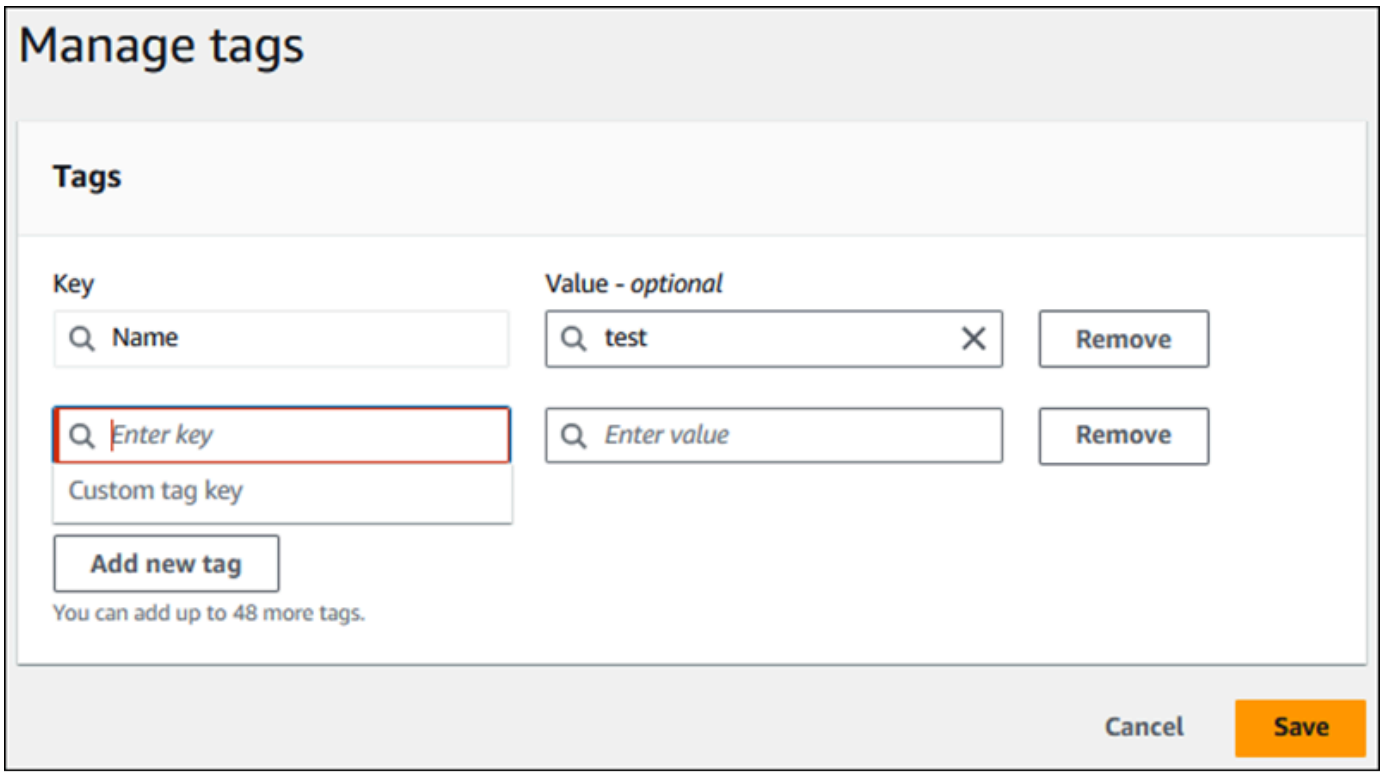
보고서를 볼 때 하나 이상의 태그를 추가하는 방법

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 성능 개선 도우미(Performance Insights)를 선택합니다.
3. DB 인스턴스를 선택합니다.

   해당 DB 인스턴스에 대해 성능 개선 도우미 대시보드가 표시됩니다.
4. 아래로 스크롤하여 성능 분석 보고서 - 신규 탭을 선택합니다.
5. 태그를 추가할 보고서를 선택합니다.

   대시보드에 보고서가 표시됩니다.
6. 태그로 스크롤하여 태그 관리를 선택합니다.
7. 새 태그 추가를 선택합니다.
8. 키와 값 - 선택 사항을 입력하고 새 태그 추가를 선택합니다.

   다음 예시에서는 선택한 보고서에 새 태그를 추가하는 옵션을 보여줍니다.



보고서에 새 태그가 생성됩니다.

보고서의 태그 목록이 대시보드의 태그 섹션에 표시됩니다. 보고서에서 태그를 제거하려면 태그 옆의 제거를 선택합니다.

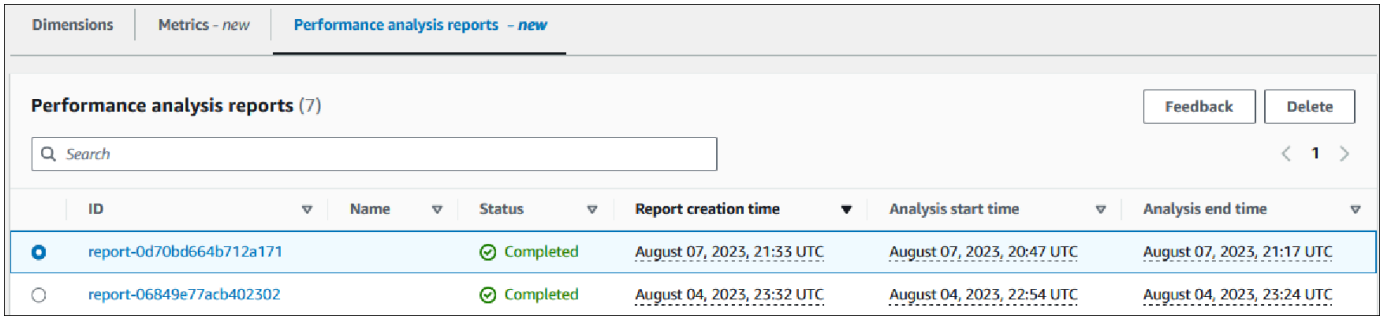
성능 개선 도우미에서 성과 분석 보고서 삭제

성능 분석 보고서 탭에 표시된 보고서 목록에서 보고서를 삭제하거나 보고서를 보는 동안 삭제할 수 있습니다.

보고서를 삭제하려면

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 성능 개선 도우미(Performance Insights)를 선택합니다.
3. DB 인스턴스를 선택합니다.

   해당 DB 인스턴스에 대해 성능 개선 도우미 대시보드가 표시됩니다.
4. 아래로 스크롤하여 성능 분석 보고서 - 신규 탭을 선택합니다.
5. 삭제하려는 보고서를 선택하고 오른쪽 상단의 삭제를 선택합니다.



확인 창이 표시됩니다. 확인을 선택하면 보고서가 삭제됩니다.

6. (선택 사항) 삭제하려는 보고서의 ID를 선택합니다.

   보고서 페이지의 오른쪽 상단에 있는 삭제를 선택합니다.

   확인 창이 표시됩니다. 확인을 선택하면 보고서가 삭제됩니다.

성능 개선 도우미의 상위 SQL 탭을 사용하여 쿼리 분석

Amazon RDS 성능 개선 도우미 대시보드에서 상위 차원(Top dimensions) 테이블의 상위 SQL(Top SQL) 탭에서 실행 중인 쿼리 및 최근 쿼리에 대한 정보를 확인할 수 있습니다. 이 정보를 사용하여 쿼리를 튜닝할 수 있습니다.

주제

- 상위 SQL(Top SQL) 탭 개요
- 성능 개선 도우미 대시보드에서 더 많은 SQL 텍스트에 액세스
- 성능 개선 도우미 대시보드에서 SQL 통계 보기

상위 SQL(Top SQL) 탭 개요

기본적으로 상위 SQL 탭은 DB 로드에 가장 많은 영향을 미치는 25개의 쿼리를 보여줍니다. 쿼리를 조정하는 데 도움이 되도록 쿼리 텍스트 및 SQL 통계와 같은 정보를 분석할 수 있습니다. 상위 SQL(Top SQL) 탭에 표시될 통계를 선택할 수도 있습니다.

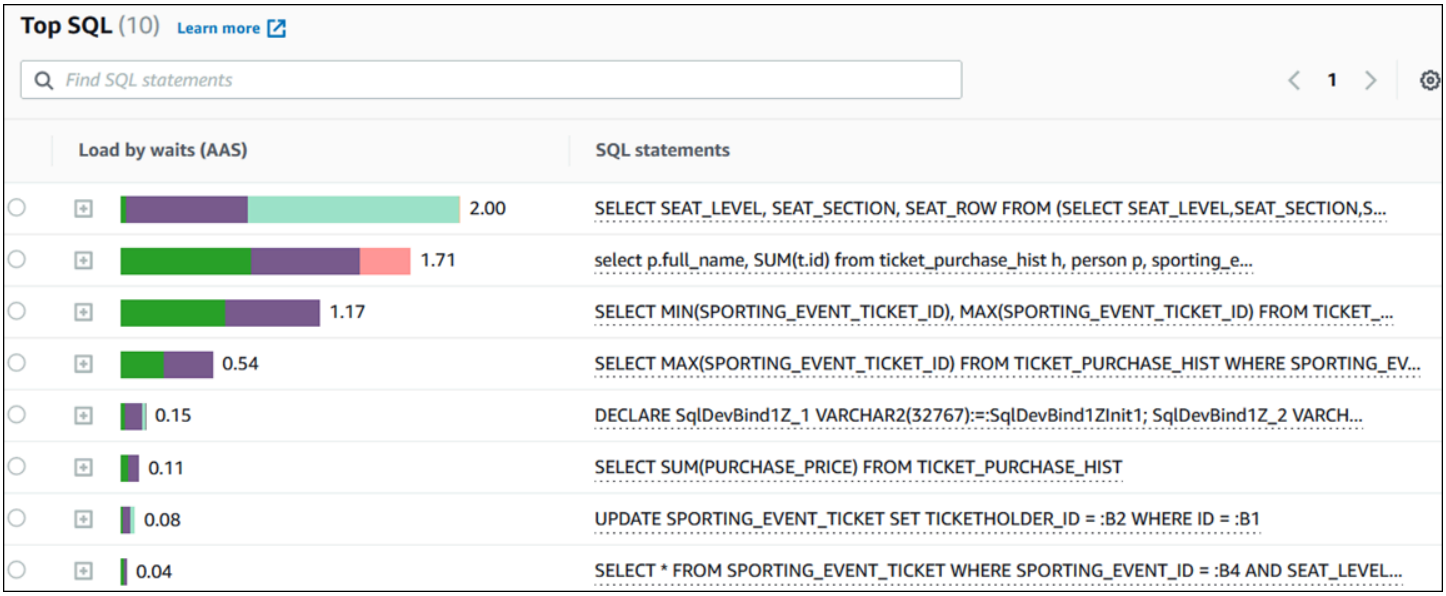
주제

- SQL 텍스트
- SQL 통계
- 대기별 로드(AAS)

- SQL 정보 보기
- 통계 기본 설정 선택

SQL 텍스트

기본적으로 상위 SQL 테이블의 각 행에는 각 문에 대해 500바이트의 텍스트가 표시됩니다.

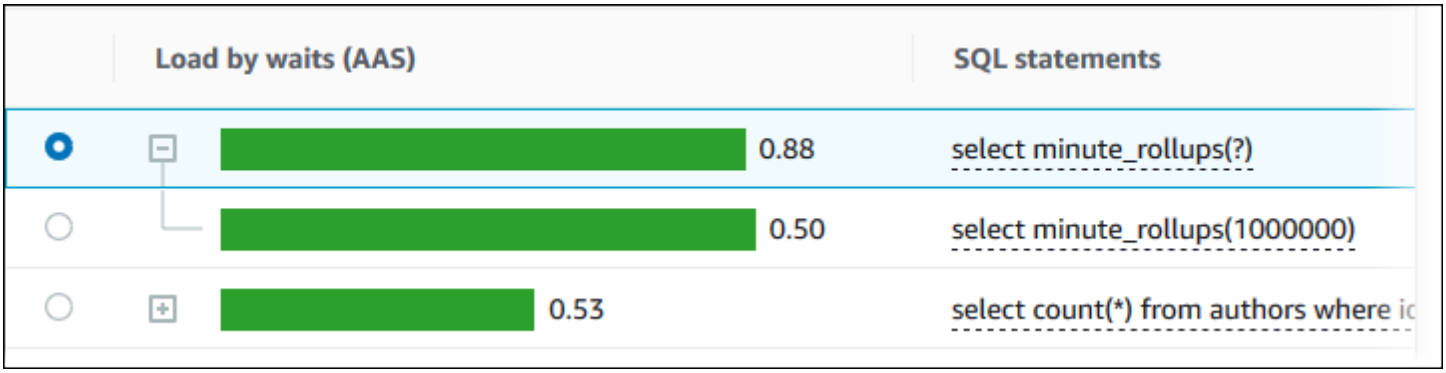


기본 500바이트 이상의 SQL 텍스트를 보는 방법에 대한 자세한 내용은 성능 개선 도우미 대시보드에서 더 많은 SQL 텍스트에 액세스 섹션을 참조하세요.

SQL 다이제스트(SQL digest)는 구조적으로 유사하지만 리터럴 값이 다를 수 있는 여러 실제 쿼리의 조합입니다. 다이제스트는 하드 코딩된 값을 물음표로 바꿉니다. 예를 들어, 다이제스트는 `SELECT * FROM emp WHERE lname= ?`가 될 수 있습니다. 이 다이제스트에는 다음 하위 쿼리가 포함될 수 있습니다.

```
SELECT * FROM emp WHERE lname = 'Sanchez'
SELECT * FROM emp WHERE lname = 'Olagappan'
SELECT * FROM emp WHERE lname = 'Wu'
```

다이제스트에서 리터럴 SQL 문을 보려면 쿼리를 선택한 다음 더하기 기호 (+) 를 선택합니다. 다음 예에서 선택한 쿼리는 다이제스트입니다.



> **Note**
>
> SQL 다이제스트는 유사한 SQL 문을 그룹화하지만, 민감한 정보는 삭제하지 않습니다.

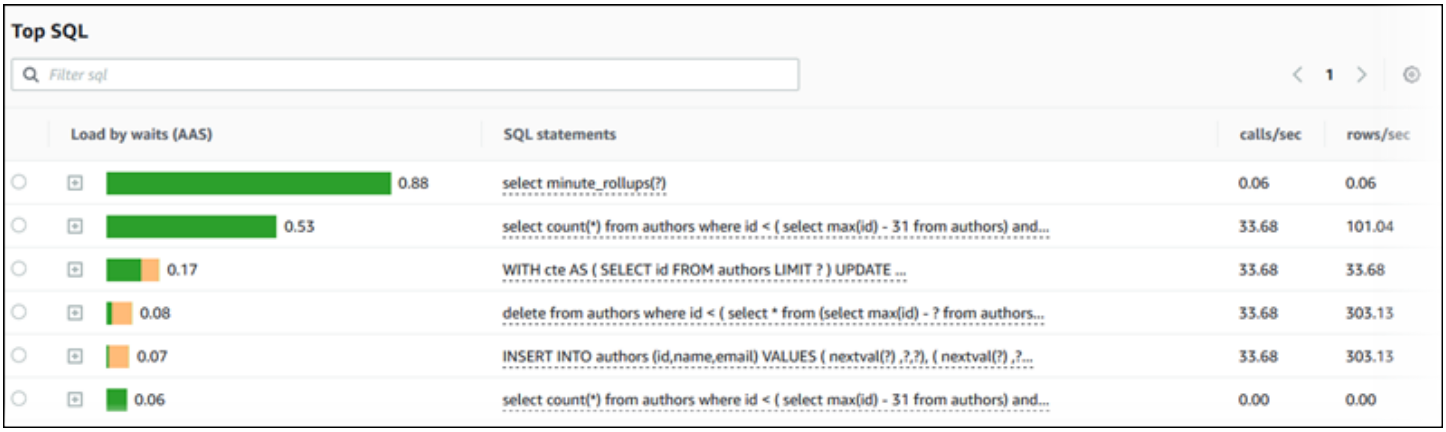
성능 개선 도우미는 Oracle SQL 텍스트를 Unknown으로 표시할 수 있습니다. 각 상황에서 텍스트의 상태는 다음과 같습니다.

- SYS 이외의 Oracle 데이터베이스 사용자가 활성 상태이지만 현재 SQL을 실행하고 있지 않은 경우입니다. 예를 들어 병렬 쿼리가 완료되면 쿼리 조정자는 도우미가 세션 통계를 전송하기 위해 처리하도록 기다립니다. 대기 기간 동안 쿼리 텍스트가 Unknown으로 표시됩니다.
- Standard Edition 2의 RDS for Oracle 인스턴스의 경우 Oracle Resource Manager가 병렬 스레드의 수를 제한합니다. 이 작업을 수행하는 백그라운드 프로세스로 인해 쿼리 텍스트가 Unknown으로 표시됩니다.

SQL 통계

SQL 통계(SQL statistics)는 SQL 쿼리에 대한 성능 관련 지표입니다. 예를 들어 성능 개선 도우미는 초당 실행 횟수 또는 초당 처리된 행을 표시할 수 있습니다. 성능 개선 도우미는 가장 일반적인 쿼리에 대한 통계만 수집합니다. 일반적으로 이러한 쿼리는 성능 개선 도우미 대시보드에 표시된 부하별로 상위 쿼리와 일치합니다.

상위 SQL(Top SQL) 테이블의 모든 라인은 다음 예에 나온 것처럼 SQL 문 또는 다이제스트에 대한 관련 통계를 보여줍니다.



성능 개선 도우미는 SQL 통계에 대해 `0.00` 및 -(알 수 없음)을 보고할 수 있습니다. 이 상황은 다음 조건에서 발생합니다.

- 샘플이 하나만 존재합니다. 예를 들어, 성능 개선 도우미는 `pg_stat_statements` 보기에서 여러 개의 샘플을 기반으로 RDS PostgreSQL 쿼리의 변경 비율을 계산합니다. 워크로드가 짧은 시간 동안 실행되면 성능 개선 도우미에서 샘플을 하나만 수집할 수 있으며 이런 경우 변경 비율을 계산할 수 없습니다. 값을 알 수 없으므로 대시(-)로 표시됩니다.
- 두 샘플의 값이 같은 경우입니다. 변경이 발생하지 않았기 때문에 성능 개선 도우미가 변경 비율을 계산할 수 없으므로 비율을 `0.00`으로 보고합니다.
- RDS PostgreSQL 문에 유효한 식별자가 없는 경우입니다. PostgreSQL은 구문 분석 및 분석 후에만 문에 대한 식별자를 만듭니다. 따라서 PostgreSQL 내부 인 메모리 구조에 식별자가 없는 상태로 문이 존재할 수 있습니다. 성능 개선 도우미는 초당 한 번 내부 인 메모리 구조를 샘플링하므로 하나의 샘플에 대해서만 대기 시간이 짧은 쿼리가 나타날 수 있습니다. 이 샘플에 대해 쿼리 식별자를 사용할 수 없는 경우 성능 개선 도우미는 이 문을 통계와 연결할 수 없습니다. 값을 알 수 없으므로 대시(-)로 표시됩니다.

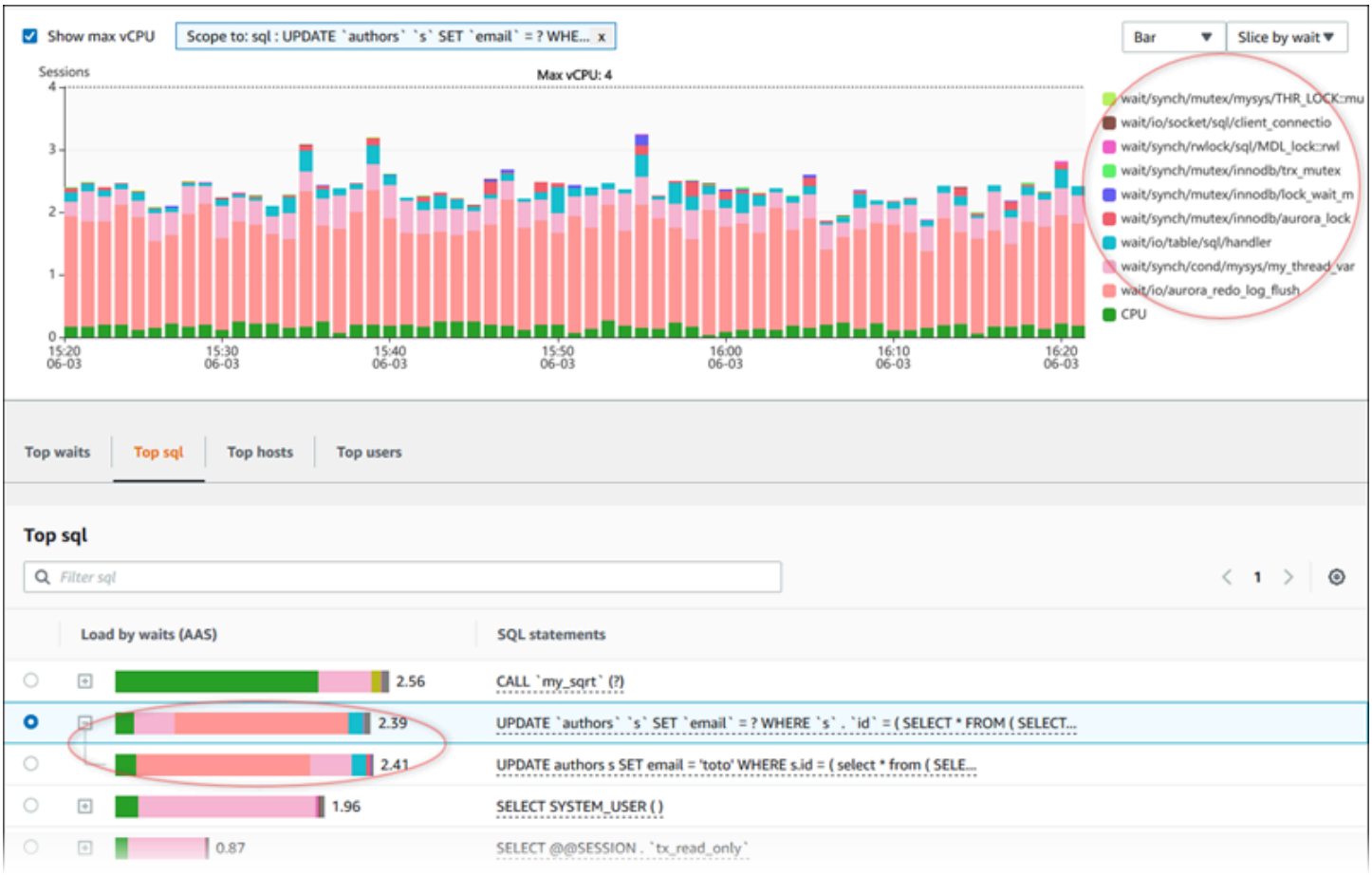
Amazon RDS엔진의 SQL 통계에 대한 설명을 보려면 성능 개선 도우미에 대한 SQL 통계 섹션을 참조하세요.

대기별 로드(AAS)

상위 SQL(Top SQL)에서 대기 시간별 로드(AAS)(Load by waits (AAS)) 열은 각 상위 로드 항목과 연결된 데이터베이스 로드의 비율을 나타냅니다. 이 열에는 현재 DB 부하 차트에서 어떤 그룹화 기준을 선택하든 그 기준에 따라 해당 항목의 부하가 반영됩니다. 평균 활성 세션(AAS)에 대한 자세한 내용은 평균 활성 세션 섹션을 참조하세요.

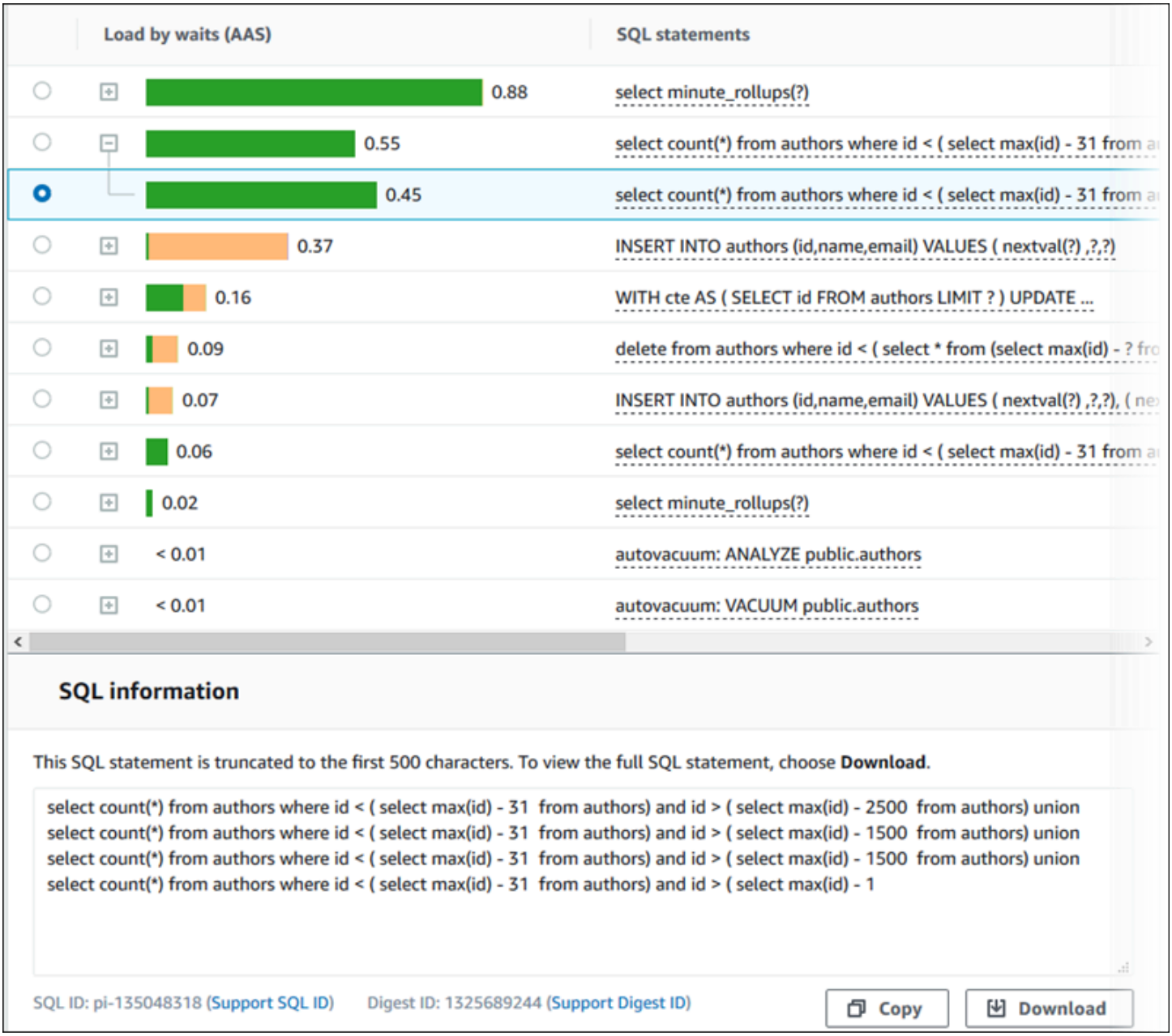
예를 들어 DB 로드(DB load) 차트를 대기 상태별로 그룹화할 수 있습니다. 상위 부하 항목 테이블에서 SQL 쿼리를 검사합니다. 이 경우 DB Load by Waits(대기별 DB 로드) 막대는 쿼리가 영향을 미치는 대

기 상태의 정도를 크기, 세그먼트 및 컬러 코드로 표시합니다. 또한 선택한 쿼리에 영향을 미치는 대기 상태를 표시합니다.



SQL 정보 보기

상위 SQL(Top SQL) 테이블에서 명령문을 열어 해당 정보를 볼 수 있습니다. 맨 아래 창에 정보가 나타납니다.

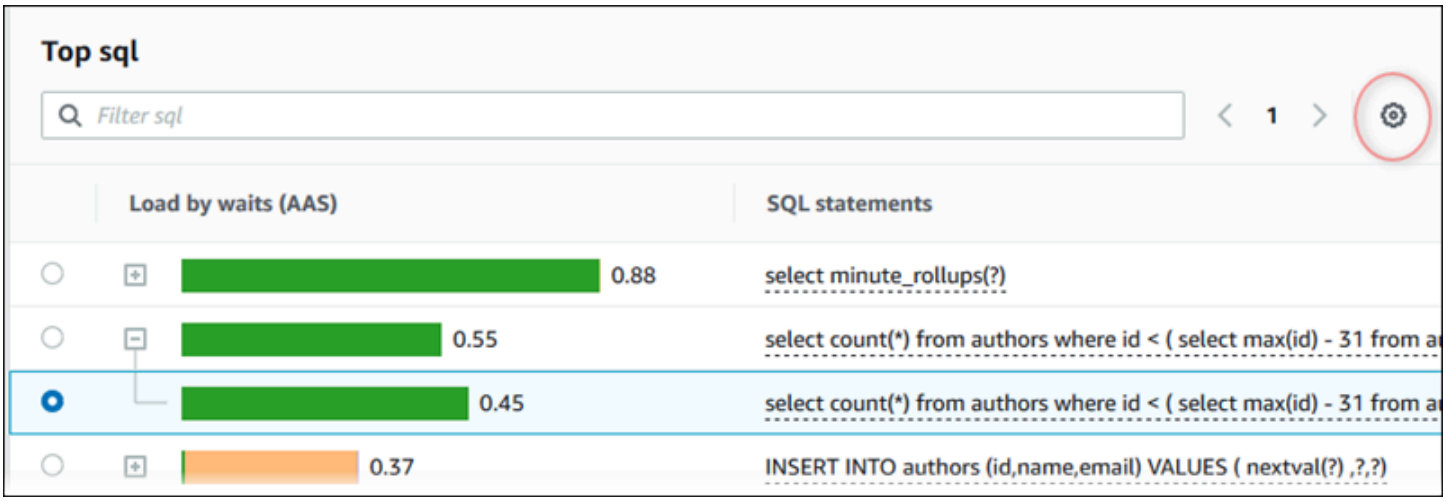


상위 SQL 탭에서 SQL 문과 연결된 다음과 같은 식별자(ID) 유형을 볼 수 있습니다.

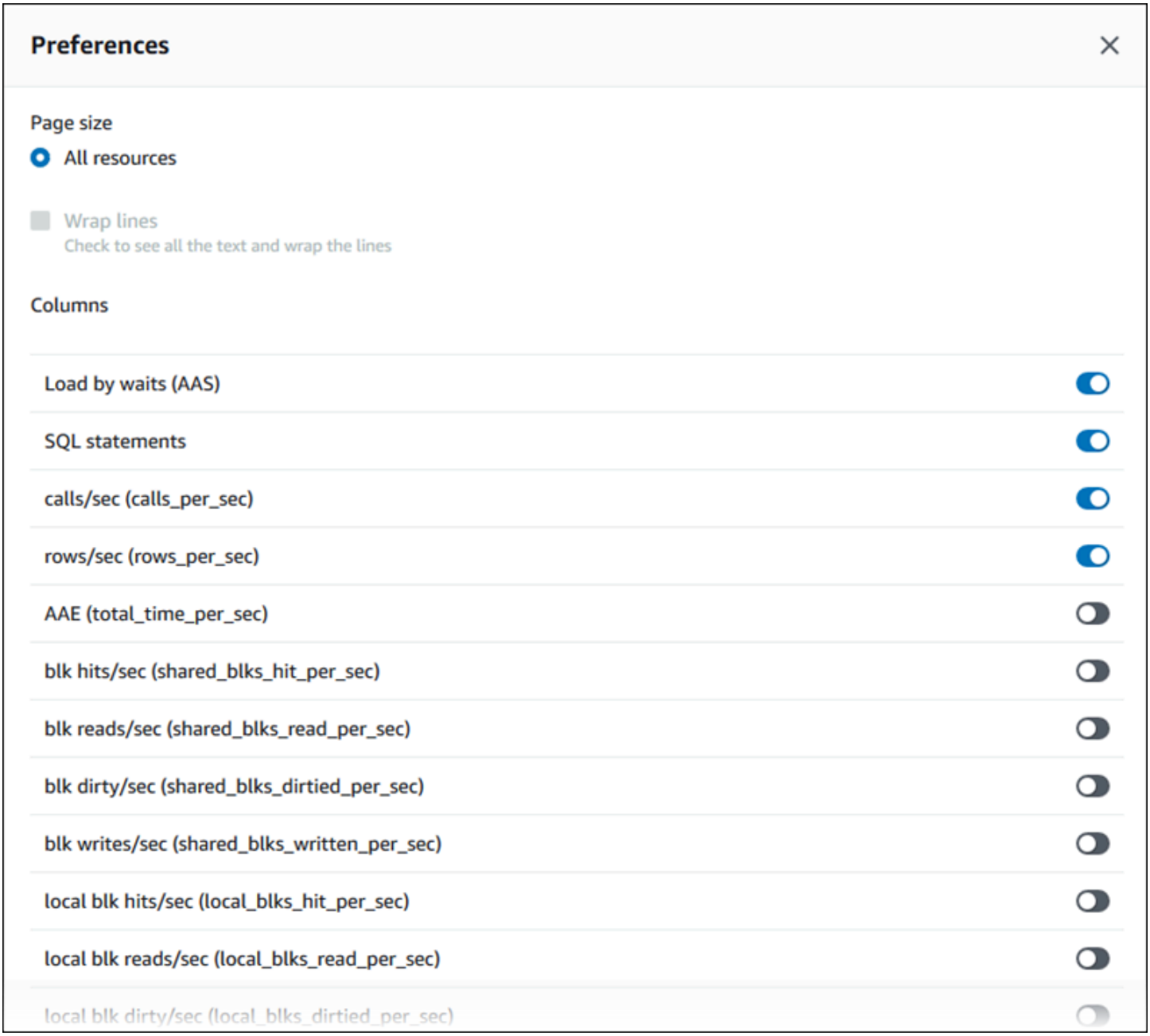
- SQL ID 지원 – SQL ID의 해시 값입니다. 이 값은 AWS Support를 이용할 때 SQL ID를 참조하는 용도로만 사용됩니다. AWS Support는 실제 SQL ID 및 SQL 텍스트에 액세스할 수 없습니다.
- Support Digest ID(다이제스트 ID 지원) – 다이제스트 ID의 해시 값입니다. 이 값은 AWS Support를 이용할 때 다이제스트 ID를 참조하는 용도로만 사용됩니다. AWS Support는 실제 다이제스트 ID 및 SQL 텍스트에 액세스할 수 있는 권한이 없습니다.

통계 기본 설정 선택

기본 설정(Preferences) 아이콘을 선택하여 상위 SQL(Top SQL) 탭에 표시되는 통계를 제어할 수 있습니다.



기본 설정 아이콘을 선택하면 기본 설정 창이 열립니다. 다음 스크린샷은 기본 설정 창의 예입니다.



상위 SQL(Top SQL) 탭에 표시하려는 통계를 활성화하고 마우스를 사용하여 창 아래쪽으로 스크롤한 다음 계속(Continue)을 선택합니다.

Amazon RDS 엔진의 초당 또는 호출당 통계에 대한 자세한 내용은 성능 개선 도우미에 대한 SQL 통계의 엔진별 SQL 통계 섹션을 참조하십시오.

성능 개선 도우미 대시보드에서 더 많은 SQL 텍스트에 액세스

기본적으로 상위 SQL(Top SQL) 테이블의 각 행에는 각 SQL 문에 대해 500바이트의 SQL 텍스트가 표시됩니다.



SQL 문이 500바이트를 초과하면 상위 SQL 테이블 아래의 SQL 텍스트 섹션에서 더 많은 텍스트를 볼 수 있습니다. 이 경우 SQL 텍스트에 표시된 쿼리의 최대 길이는 4KB입니다. 이 제한은 콘솔에 의해 도입되며 데이터베이스 엔진에서 설정한 제한에 따라 달라집니다. SQL 텍스트에 표시된 텍스트를 저장하려면 다운로드를 선택합니다.

주제

- Amazon RDS 엔진 텍스트 크기 제한
- Amazon RDS for PostgreSQL DB 인스턴스에 대한 SQL 텍스트 한도 설정
- 성능 개선 도우미 대시보드에서 SQL 텍스트 보기 및 다운로드

Amazon RDS 엔진 텍스트 크기 제한

SQL 텍스트를 다운로드할 때 데이터베이스 엔진은 최대 길이를 결정합니다. 다음의 엔진별 제한에 해당하는 SQL 텍스트를 다운로드할 수 있습니다.

DB 엔진	다운로드한 텍스트의 최대 길이
Amazon RDS for MySQL 및 MariaDB	성능 스키마가 활성화된 경우 길이는 4,096바이트로 고정됩니다. 성능 스키마가 활성화되지 않은 경우 길이는 65,535바이트로 고정됩니다.
Amazon RDS for Microsoft SQL Server	4,096자
Amazon RDS for Oracle	1,000바이트

성능 개선 도우미 콘솔의 SQL 텍스트 섹션은 엔진이 반환하는 최대값까지 표시합니다. 예들 들어, MySQL은 최대 1KB를 성능 개선 도우미에 반환하며, 원래 쿼리가 더 큰 경우에도 1KB만 수집하고 표시할 수 있습니다. 따라서 SQL 텍스트의 쿼리를 보거나 다운로드하면 성능 개선 도우미가 동일한 바이트 수를 반환합니다.

AWS CLI 또는 API, 성능 개선 도우미에는 콘솔이 적용하는 4KB 제한이 없으며, `DescribeDimensionKeys` 및 `GetResourceMetrics`로 최대 500바이트를 반환합니다.

> **Note**
>
> `GetDimensionKeyDetails`는 전체 쿼리를 반환하지만 크기에는 엔진 제한이 적용됩니다.

Amazon RDS for PostgreSQL DB 인스턴스에 대한 SQL 텍스트 한도 설정

Amazon RDS for PostgreSQL은 텍스트를 다르게 처리합니다. DB 인스턴스 파라미터 `track_activity_query_size`를 사용하여 텍스트 크기 제한을 설정할 수 있습니다. 이 파라미터에는 다음과 같은 특성이 있습니다.

기본 텍스트 크기

Amazon RDS for PostgreSQL 버전 9.6에서 `track_activity_query_size` 파라미터에 대한 기본 설정은 1,024바이트입니다. Amazon RDS for PostgreSQL 버전 10이상에서 기본 설정은 4,096바이트입니다.

최대 텍스트 크기

Amazon RDS for PostgreSQL 버전 12 이하에 대한 `track_activity_query_size` 제한은 102,400바이트입니다. 버전 13 이상에서는 최대 1MB입니다.

엔진이 성능 개선 도우미에 1MB를 반환하면 콘솔에는 처음 4KB만 표시됩니다. 쿼리를 다운로드하면 전체 1MB를 받을 수 있습니다. 이 경우 보기 및 다운로드하면 다른 바이트 수가 반환됩니다. `track_activity_query_size` DB 파라미터에 대한 자세한 내용은 PostgreSQL 설명서에서 런타임 통계를 참조하세요.

SQL 텍스트 크기를 늘리려면 `track_activity_query_size` 제한을 늘립니다. 파라미터를 수정하려면 Amazon RDS for PostgreSQL DB 인스턴스와 연결된 파라미터 그룹에서 파라미터 설정을 변경하세요.

인스턴스가 기본 파라미터 그룹을 사용할 때 설정 변경

1. 적절한 DB 엔진 및 DB 엔진 버전에 대해 새로운 DB 인스턴스 파라미터 그룹을 생성합니다.
2. 새 파라미터 그룹에 파라미터를 설정합니다.
3. 새 파라미터 그룹을 DB 인스턴스에 연결합니다.

DB 인스턴스 파라미터 설정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 단원을 참조하세요.

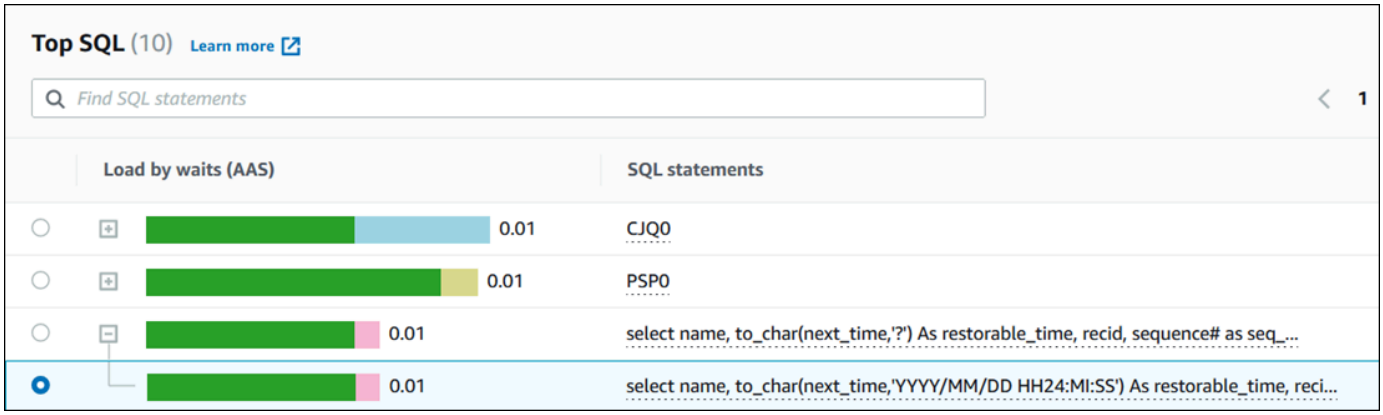
성능 개선 도우미 대시보드에서 SQL 텍스트 보기 및 다운로드

성능 개선 도우미 대시보드에서 SQL 텍스트를 보기 및 다운로드할 수 있습니다.

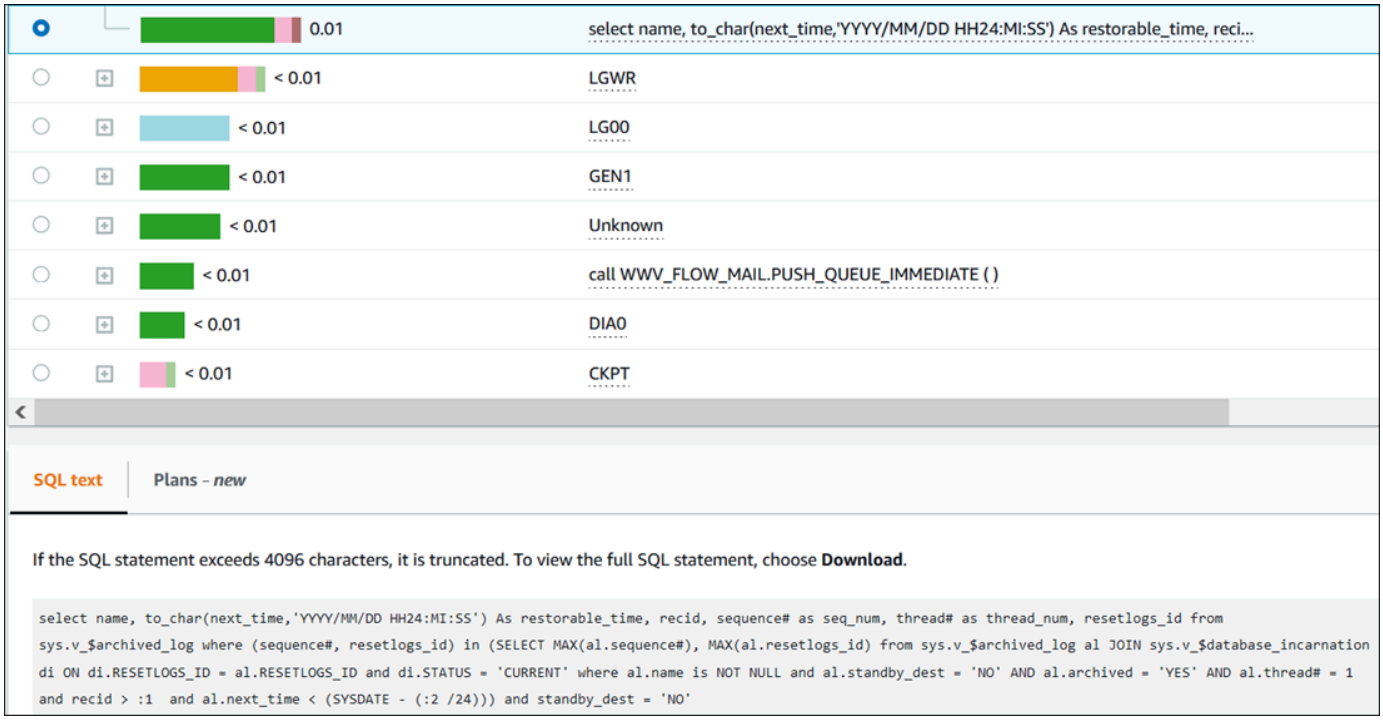
성능 개선 도우미 대시보드에서 더 많은 SQL 텍스트를 보려면

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 성능 개선 도우미을 선택합니다.
3. DB 인스턴스를 선택합니다.
4. 성능 개선 도우미 대시보드에서 아래로 스크롤하여 상위 SQL 탭으로 이동합니다.
5. 더하기(+) 기호를 선택하여 SQL 다이제스트를 펼치고 다이제스트의 하위 쿼리 중 하나를 선택합니다.

   500바이트 이상의 텍스트가 있는 SQL 문은 다음 이미지와 유사합니다.



6. SQL 텍스트 탭까지 아래로 스크롤합니다.



성능 개선 도우미 대시보드는 각 SQL 문에 최대 4,096바이트를 표시할 수 있습니다.

7. (선택 사항) 복사를 선택하여 표시된 SQL 문을 복사하거나 다운로드를 선택하여 최대 DB 엔진 한도까지 SQL 텍스트를 볼 수 있는 SQL 문을 다운로드합니다.

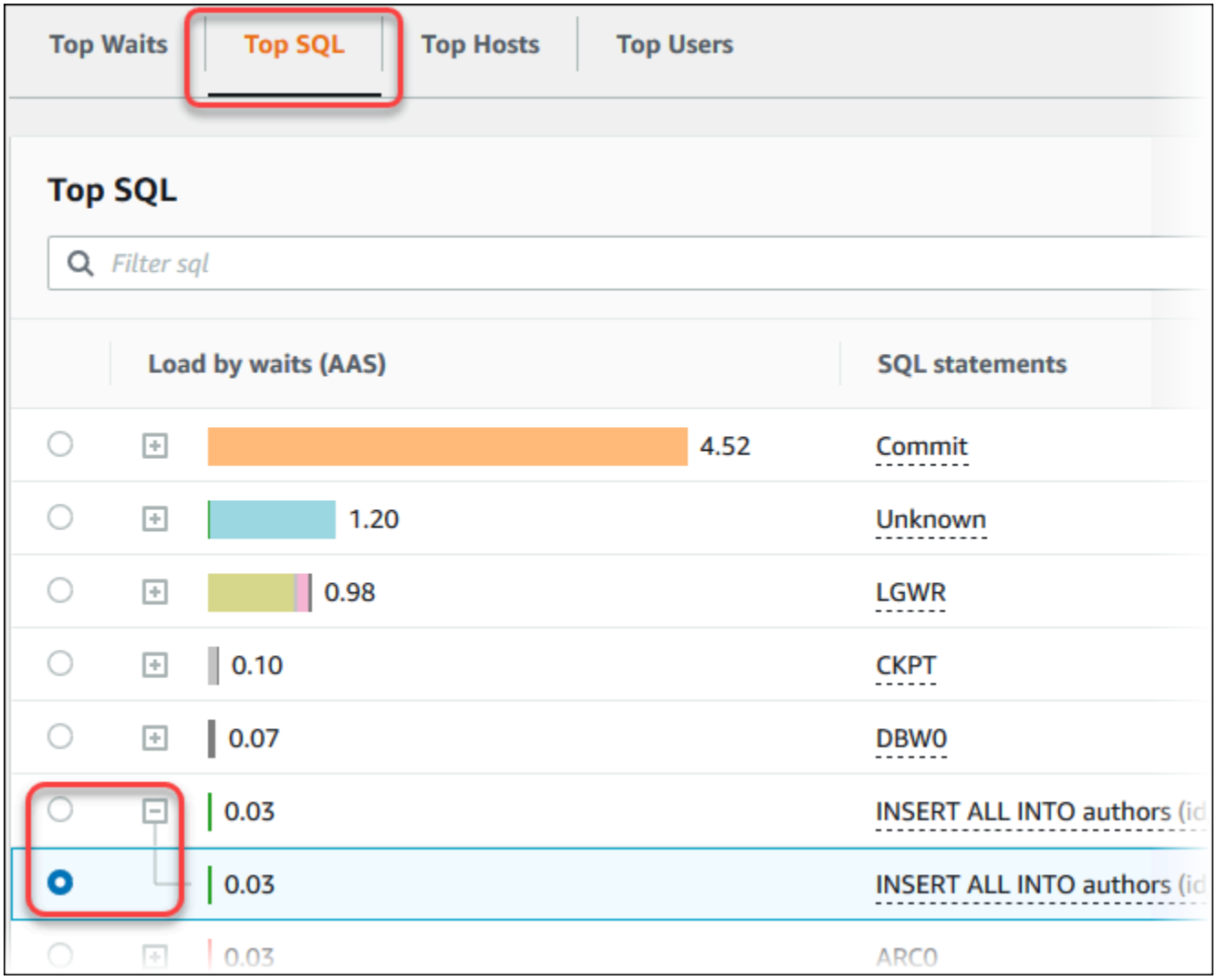
> **Note**
>
> SQL 문을 복사하거나 다운로드하려면 팝업 차단 기능을 비활성화하세요.

성능 개선 도우미 대시보드에서 SQL 통계 보기

성능 개선 도우미 대시보드에서 SQL 통계는 데이터베이스 로드(Database load) 차트의 상위 SQL(Top SQL) 탭에서 확인할 수 있습니다.

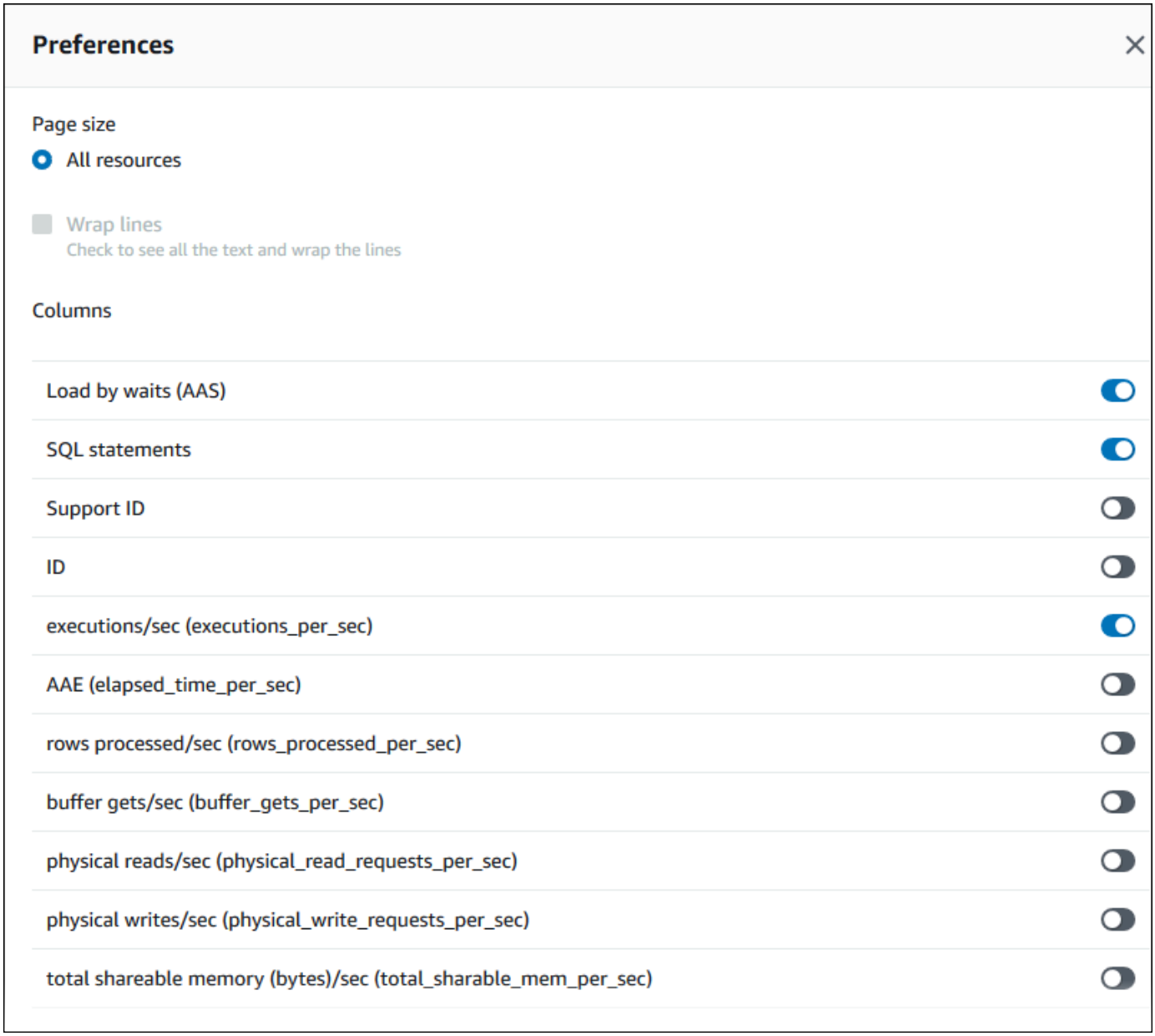
SQL 통계를 보는 방법

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 성능 개선 도우미(Performance Insights)를 선택합니다.
3. 페이지 상단에서 SQL 통계를 확인하려는 데이터베이스를 선택합니다.
4. 페이지의 하단으로 스크롤하고 상위 SQL(Top SQL)을 선택합니다.
5. 개별 문 또는 다이제스트 쿼리를 선택합니다.

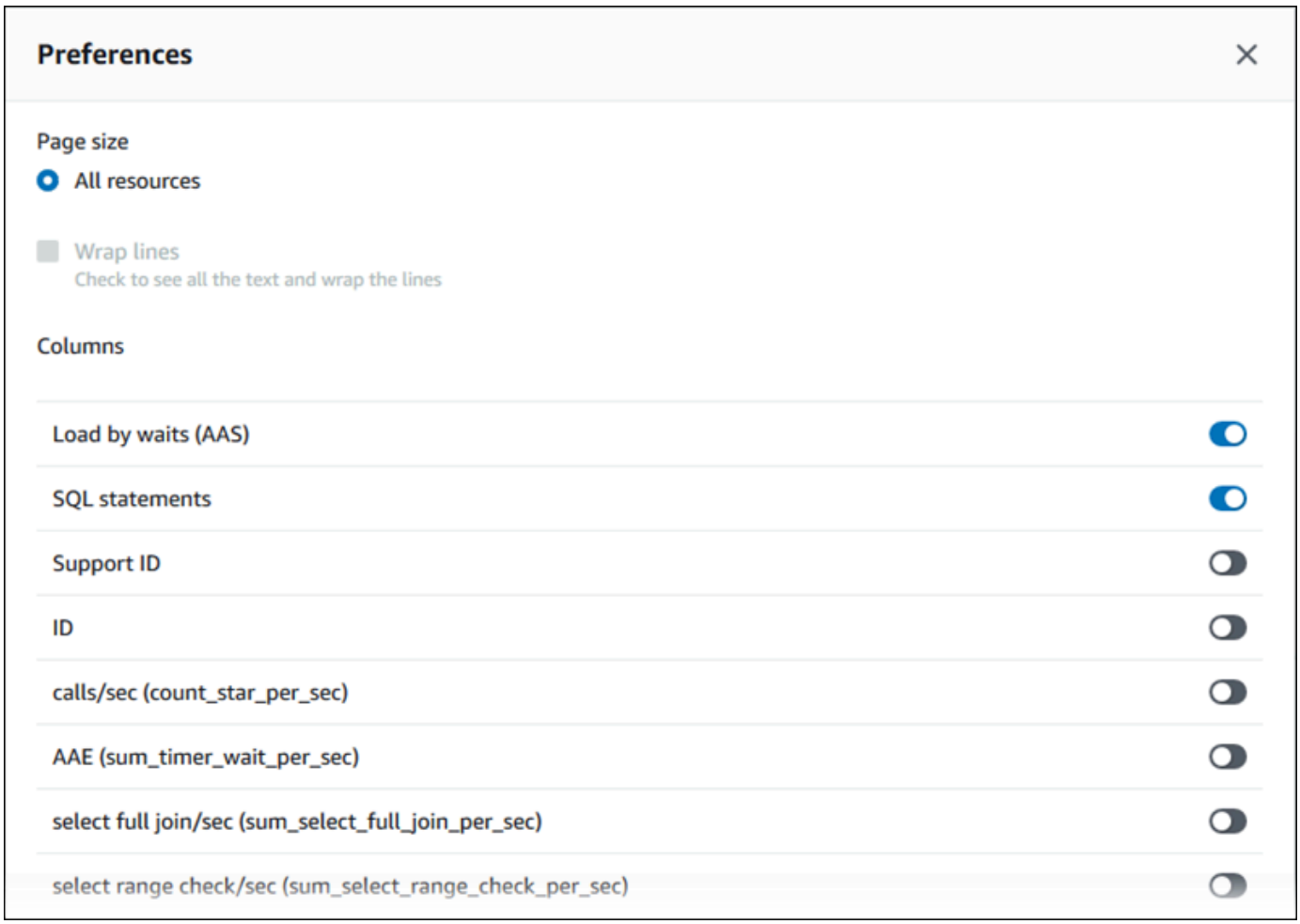


6. 차트 오른쪽 상단 모서리에 있는 기어 모양 아이콘을 선택하여 표시할 통계를 선택하십시오. Amazon RDS엔진의 SQL 통계에 대한 설명을 보려면 성능 개선 도우미에 대한 SQL 통계 섹션을 참조하세요.

   다음 예는 Oracle DB 인스턴스의 통계 기본 설정을 보여줍니다.



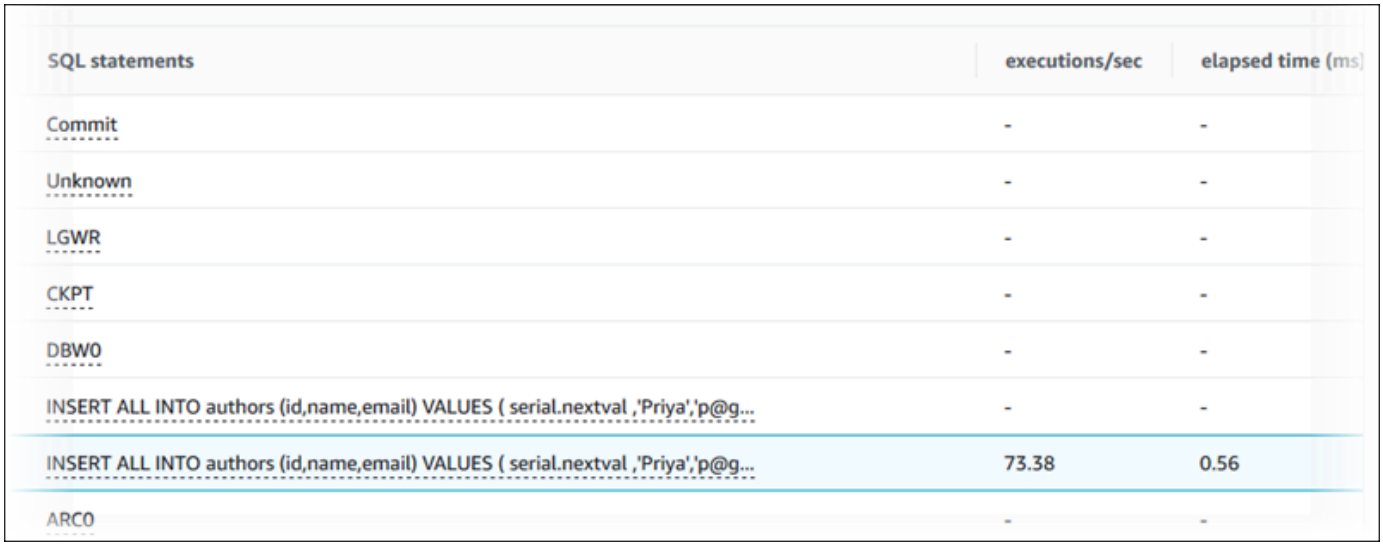
다음 예는 MariaDB 및 MySQL DB 인스턴스의 기본 설정을 보여줍니다.



7. 저장(Save)을 선택하여 기본 설정을 저장합니다.

   상위 SQL(Top SQL) 테이블이 새로 고쳐집니다.

   다음 예는 Oracle SQL 쿼리의 통계를 보여줍니다.



SQL statements	executions/sec	elapsed time (ms)
Commit	-	-
Unknown	-	-
LGWR	-	-
CKPT	-	-
DBW0	-	-
INSERT ALL INTO authors (id,name,email) VALUES (serial.nextval ,'Priya','p@g...	-	-
INSERT ALL INTO authors (id,name,email) VALUES (serial.nextval ,'Priya','p@g...	73.38	0.56
ARC0	-	-

상위 Oracle PDB 로드 분석

Oracle 컨테이너 DB(CDB) 로드를 분석할 때 DB 로드에 가장 많이 기여하는 플러그 가능 데이터베이스(PDB)를 식별하고 싶을 수 있습니다. 성능을 미세 조정하기 위해 유사한 쿼리를 실행하는 개별 PDB의 성능을 비교하고 싶을 수도 있습니다. Oracle CDB에 대한 자세한 내용은 RDS for Oracle 데이터베이스 아키텍처 섹션을 참조하세요.

Amazon RDS 성능 개선 도우미 대시보드에서 차원 탭의 상위 PDB 탭에서 플러그 가능 데이터베이스(PDB)에 대한 정보를 알아볼 수 있습니다.

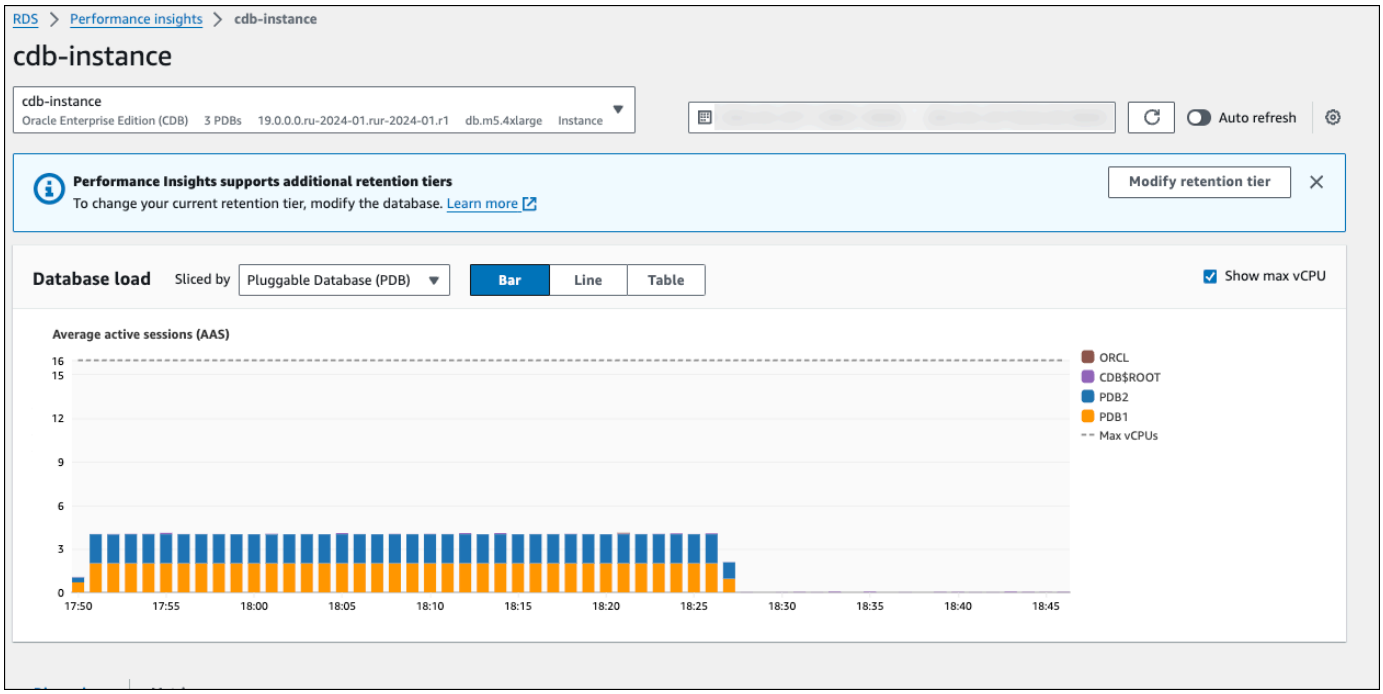
이 기능에 대한 리전, DB 엔진 및 인스턴스 클래스 지원 정보는 성능 개선 도우미 기능에 대한 Amazon RDS DB 엔진, 리전 및 인스턴스 클래스 지원 섹션을 참조하세요.

Oracle CDB의 상위 PDB 로드를 분석하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 성능 개선 도우미를 선택합니다.
3. Oracle CDB 인스턴스를 선택합니다.

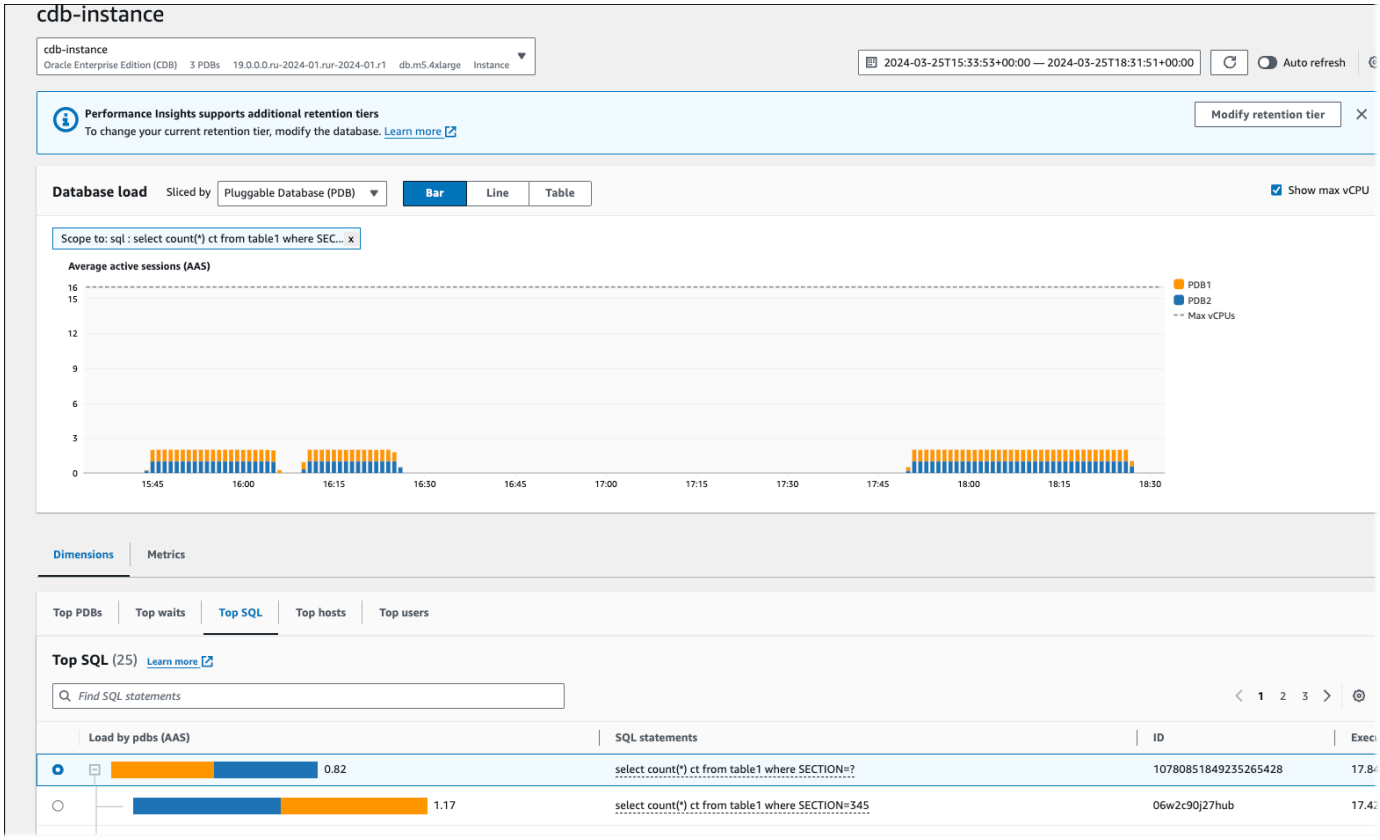
   해당 DB 인스턴스에 대해 성능 개선 도우미 대시보드가 표시됩니다.
4. 데이터베이스 로드(DB 로드) 섹션에서 슬라이스 기준 옆에 있는 플러그 가능 데이터베이스(PDB)를 선택합니다.

평균 활성 세션 차트는 로드가 가장 많은 PDB를 보여줍니다. PDB 식별자는 색상으로 구분된 사각형의 오른쪽에 나타납니다. 각 식별자는 PDB를 고유하게 식별합니다.

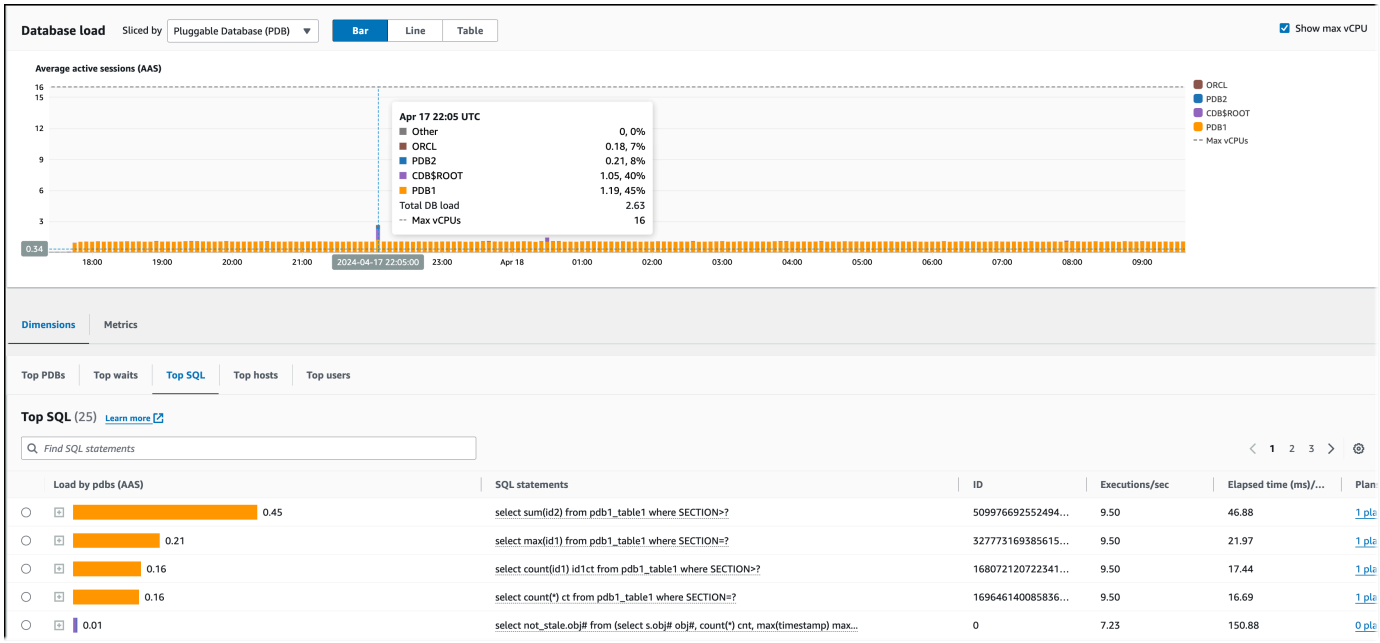


5. 상위 SQL 탭까지 아래로 스크롤합니다.

   다음 예시에서는 동일한 SQL 쿼리와 이로 인해 여러 PDB에 전달되는 로드를 확인할 수 있습니다.



다음 예시에서는 단일 PDB가 CDB의 다른 PDB보다 더 높은 로드를 처리합니다.



Oracle CDB에 대한 자세한 내용은 CDBs and PDBs를 참조하세요.

Amazon RDS용 성능 개선 도우미 대시보드를 사용한 실행 계획 분석

Amazon RDS 성능 개선 도우미 대시보드에서 Oracle 및 SQL Server DB 인스턴스의 실행 계획에 대한 정보를 찾을 수 있습니다. 이 정보를 사용하여 어떤 계획이 DB 로드에 가장 많이 기여하는지 알 수 있습니다.

Oracle 또는 SQL Server 실행 계획을 분석하려면 다음 주제를 참조하세요.

실행 계획 분석

- Amazon RDS용 성능 개선 도우미 대시보드를 사용한 Oracle 실행 계획 분석
- Amazon RDS용 성능 개선 도우미 대시보드를 사용한 SQL Server 실행 계획 분석

Amazon RDS에 대한 실행 계획 분석 개요

Amazon RDS 성능 개선 도우미 대시보드를 사용하여 Oracle 및 SQL Server DB 인스턴스의 DB 로드에 가장 많이 기여하는 계획을 파악할 수 있습니다.

예를 들어 지정된 시간의 상위 SQL 문이 다음 표에 표시된 계획을 사용하고 있을 수 있습니다.

상위 SQL	계획
SELECT SUM(amount_sold) FROM sales WHERE prod_id = 10	계획 A
SELECT SUM(amount_sold) FROM sales WHERE prod_id = 521	계획 B
SELECT SUM(s_total) FROM sales WHERE region = 10	계획 A
SELECT * FROM emp WHERE emp_id = 1000	계획 C
SELECT SUM(amount_sold) FROM sales WHERE prod_id = 72	계획 A

성능 개선 도우미의 계획 기능을 사용하여 다음을 수행할 수 있습니다.

- 상위 SQL 쿼리에서 사용하는 계획을 찾습니다.

예를 들어 대부분의 DB 로드가 계획 A와 계획 B를 사용하는 쿼리에 의해 생성되고 계획 C를 사용하는 비율은 적음을 알 수 있습니다.

- 동일한 쿼리에 대해 여러 계획을 비교합니다.

  앞의 예에서 3개의 쿼리는 제품 ID를 제외하고 동일합니다. 두 쿼리는 계획 A를 사용하지만 한 쿼리는 계획 B를 사용합니다. 두 계획의 차이를 보려면 성능 개선 도우미를 사용할 수 있습니다.

- 쿼리가 새 계획으로 전환된 시기를 확인합니다.

  쿼리가 계획 A를 사용한 다음 특정 시간에 계획 B로 전환했음을 확인할 수 있습니다. 이 시점에서 데이터베이스의 변경 사항이 있었나요? 예를 들어 테이블이 비어 있는 경우 옵티마이저는 전체 테이블 스캔을 선택할 수 있습니다. 테이블에 백만 개의 행이 로드되면 옵티마이저가 인덱스 범위 스캔으로 전환할 수 있습니다.

- 비용이 가장 많이 드는 계획의 특정 단계로 드릴다운합니다.

  예를 들어 장기 실행 쿼리의 경우 동등 조인에서 조인 조건 누락이 표시될 수 있습니다. 이 누락된 조건은 두 테이블의 모든 행을 조인하는 데카르트 조인을 강제 실행합니다.

성능 개선 도우미의 계획 캡처 기능을 사용하여 이전 태스크를 수행할 수 있습니다. 대기 이벤트 및 상위 SQL을 기준으로 쿼리를 분할할 수 있는 것처럼 계획 차원을 기준으로 쿼리를 분할할 수 있습니다.

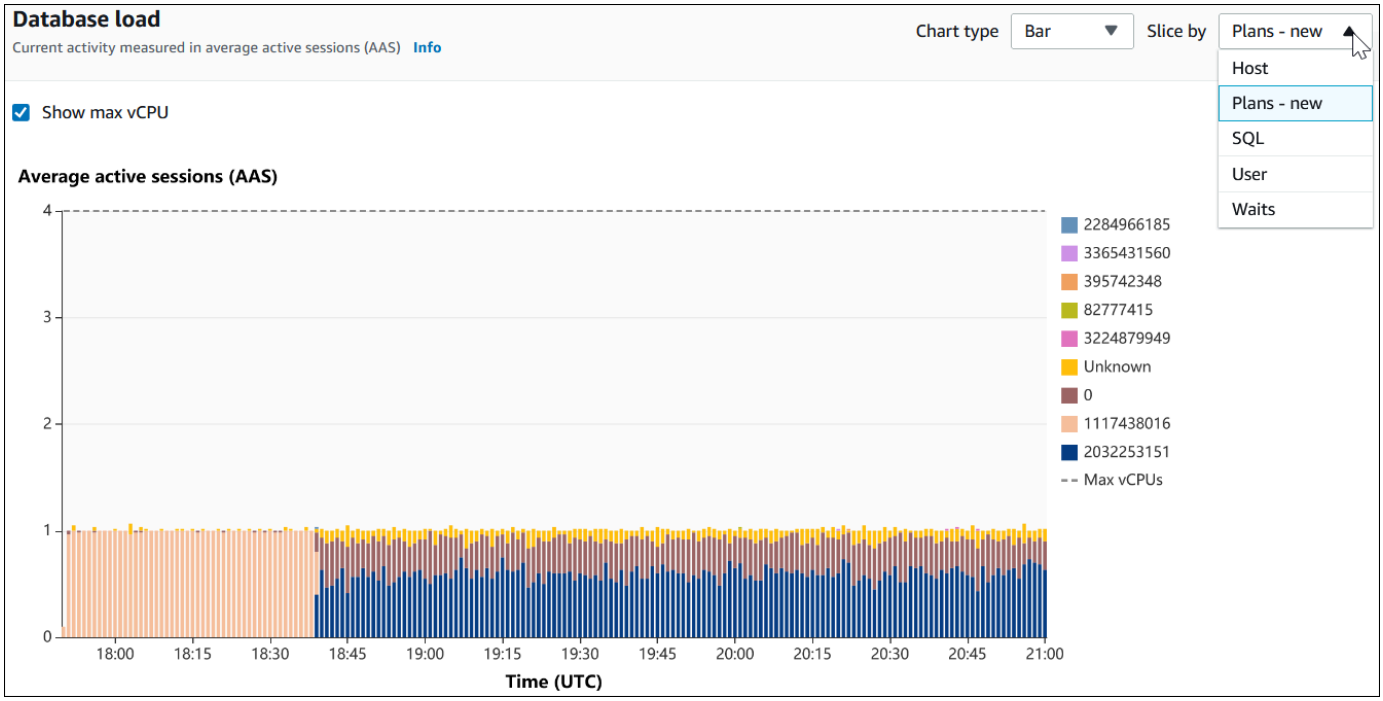
Amazon RDS용 성능 개선 도우미 대시보드를 사용한 Oracle 실행 계획 분석

Oracle Database에서 DB 로드를 분석할 때 DB 로드에 가장 많이 기여하는 계획을 알고 싶을 수 있습니다. 성능 개선 도우미의 계획 캡처 특성을 사용하여 DB 로드에 가장 많이 기여하는 계획을 확인할 수 있습니다.

콘솔을 사용하여 Oracle 실행 계획 분석

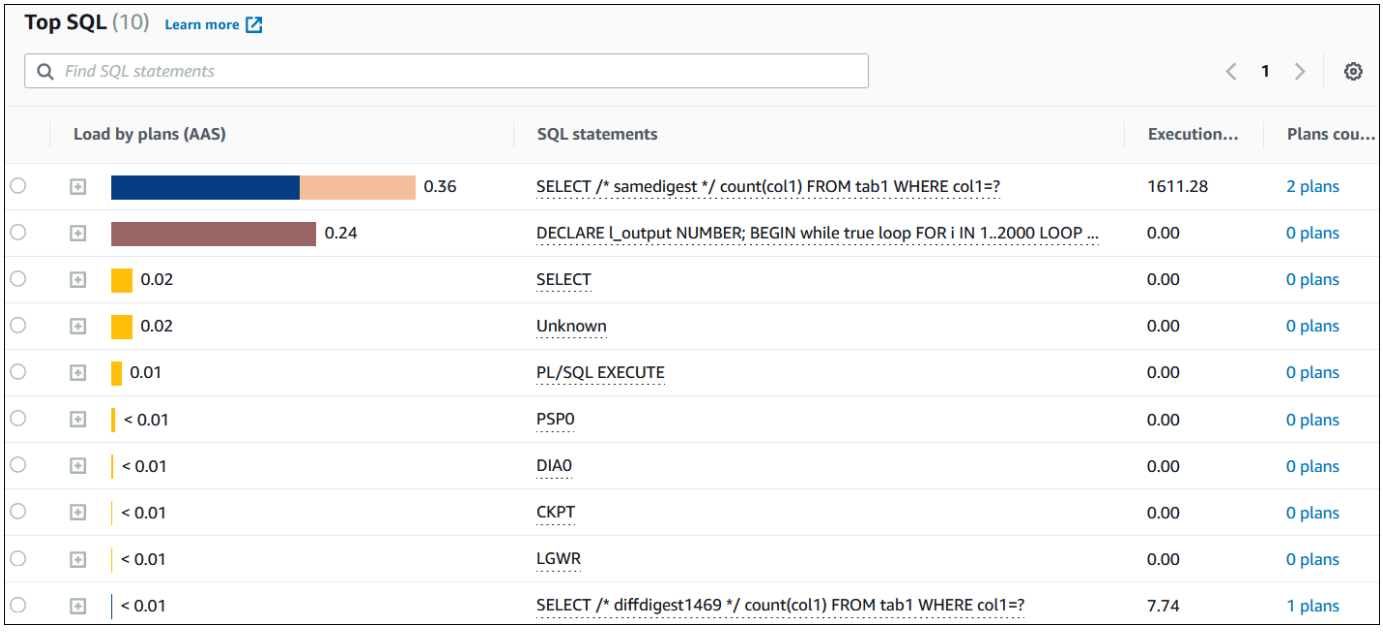
1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 성능 개선 도우미을 선택합니다.
3. Oracle DB 인스턴스를 선택합니다. 선택한 DB 인스턴스에 대한 성능 개선 도우미 대시보드가 표시됩니다.
4. 데이터베이스 로드(DB 로드)(Database load (DB load)) 섹션에서 분할 기준(Slice by) 옆에 있는 계획(Plans)을 선택합니다.

   평균 활성 세션 차트는 상위 SQL 문이 사용하는 계획을 보여줍니다. 계획 해시 값은 색상으로 구분된 사각형의 오른쪽에 나타납니다. 각 해시 값은 계획을 고유하게 식별합니다.



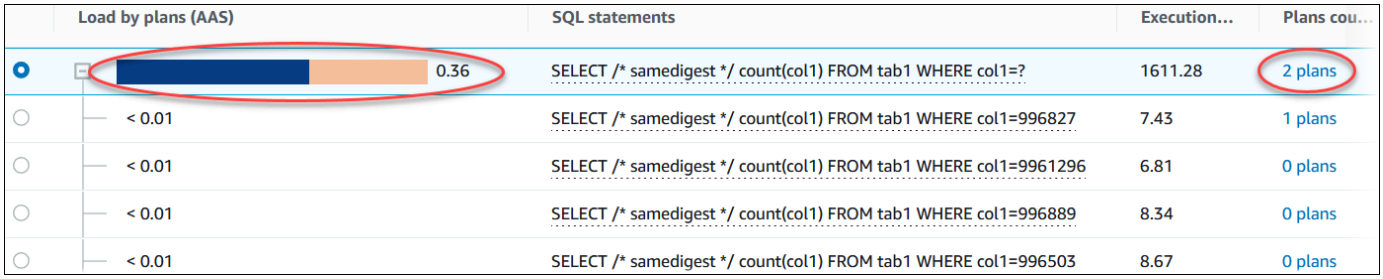
5. 상위 SQL(Top SQL) 탭까지 아래로 스크롤합니다.

   다음 예에서 상위 SQL 다이제스트에는 2개의 계획이 있습니다. 문장의 물음표를 보면 다이제스트임을 알 수 있습니다.



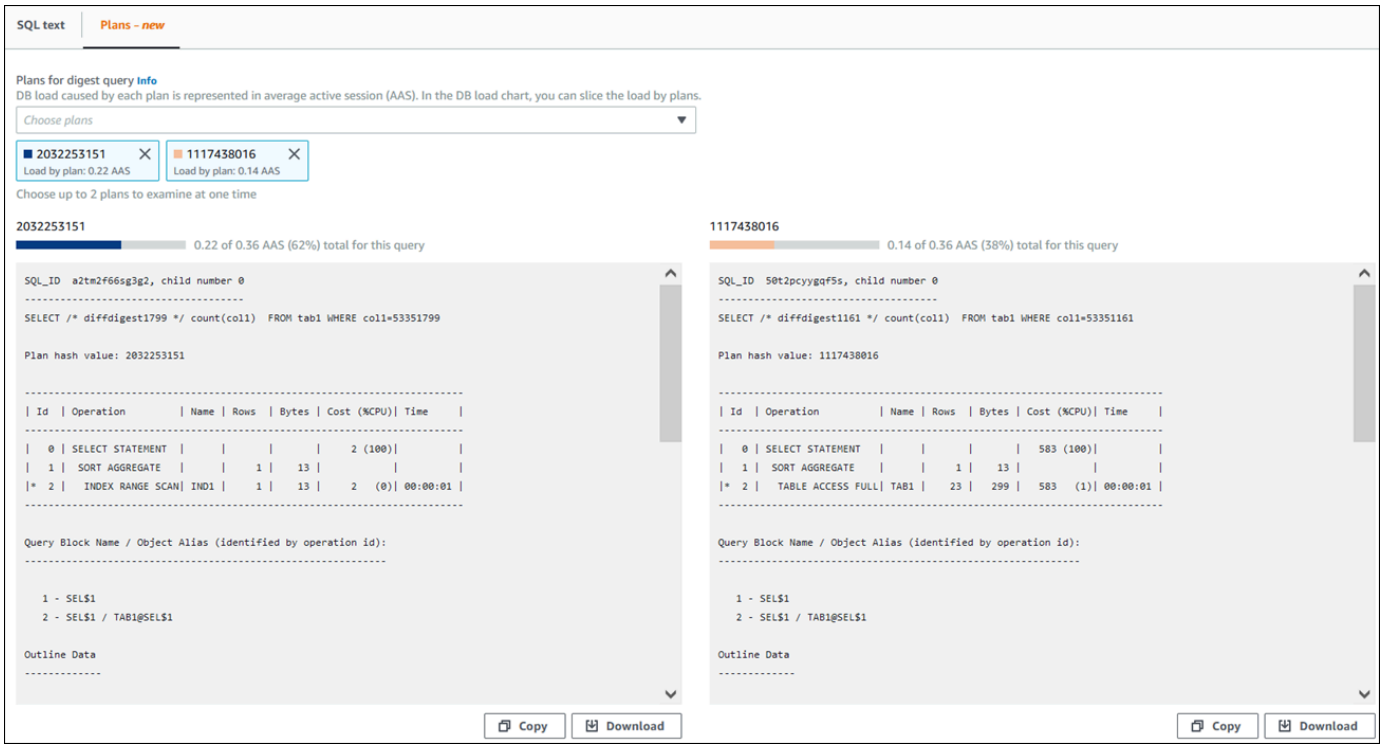
6. 다이제스트를 선택하여 구성 요소 문으로 확장합니다.

다음 예에서 SELECT 문은 다이제스트 쿼리입니다. 다이제스트의 구성 요소 쿼리는 2개의 다른 계획을 사용합니다. 계획의 색상은 데이터베이스 로드 차트에 해당합니다. 다이제스트의 총 계획 수는 두 번째 열에 표시됩니다.

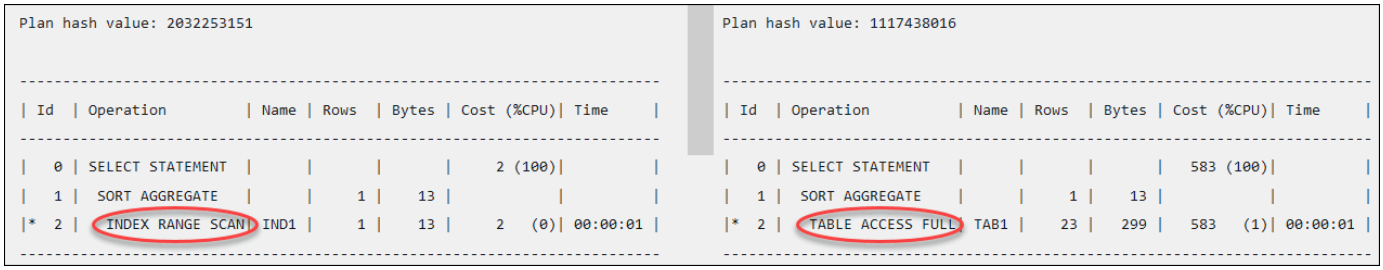


7. 아래로 스크롤하여 다이제스트 쿼리 계획(Plans for digest query) 목록에서 비교할 2개의 계획(Plans)을 선택합니다.

쿼리에 대해 한 번에 하나 또는 2개의 계획을 볼 수 있습니다. 다음 스크린샷에서는 해시 2032253151 및 해시 1117438016과 함께 다이제스트의 두 계획을 비교합니다. 다음 예에서 이 다이제스트 쿼리를 실행하는 평균 활성 세션의 62%는 왼쪽 계획을 사용하고 있는 반면 38%는 오른쪽 계획을 사용하고 있습니다.



이 예에서는 계획이 중요한 방식으로 다릅니다. 계획 2032253151의 2단계는 인덱스 스캔을 사용하는 반면 계획 1117438016은 전체 테이블 스캔을 사용합니다. 행 수가 많은 테이블의 경우 인덱스 스캔을 사용하면 단일 행의 쿼리가 거의 항상 더 빠릅니다.



8. (선택 사항) 복사(Copy)를 선택하여 클립보드에 계획을 복사하거나 다운로드(Download)를 선택하여 하드 드라이브에 계획을 저장합니다.

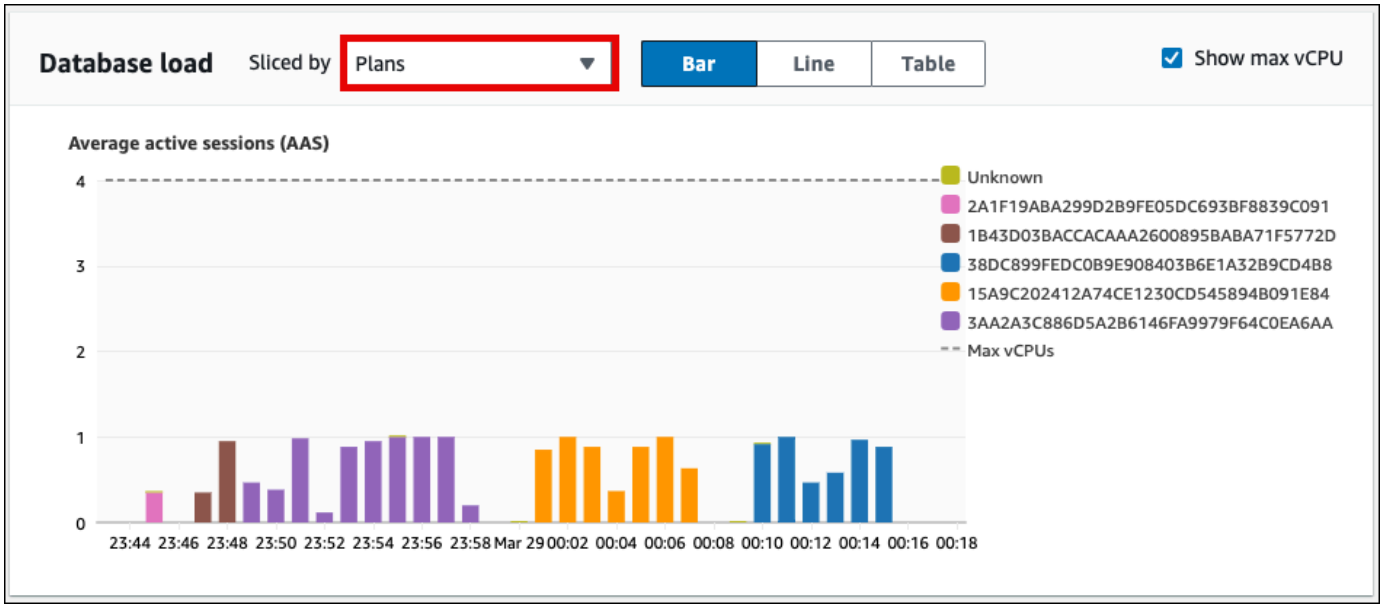
Amazon RDS용 성능 개선 도우미 대시보드를 사용한 SQL Server 실행 계획 분석

SQL Server 데이터베이스에서 DB 로드를 분석할 때 DB 로드에 가장 많이 기여하는 계획을 알고 싶을 수 있습니다. 성능 개선 도우미의 계획 캡처 특성을 사용하여 DB 로드에 가장 많이 기여하는 계획을 확인할 수 있습니다.

콘솔을 사용하여 SQL Server 실행 계획을 분석하려면 다음과 같이 하세요.

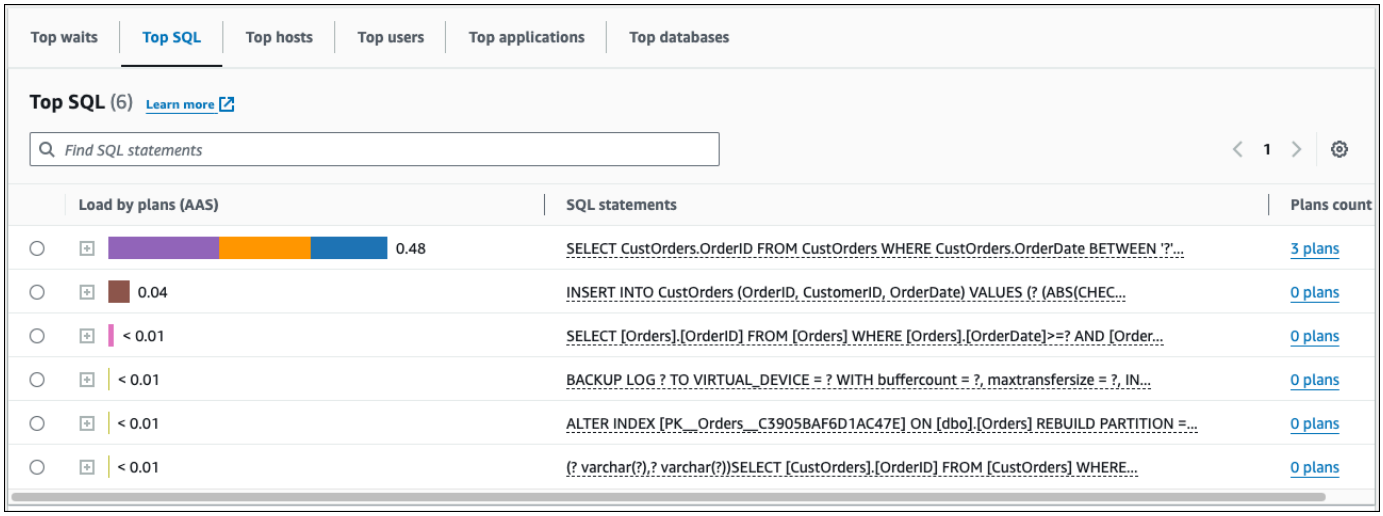
1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 성능 개선 도우미을 선택합니다.
3. SQL Server DB 인스턴스를 선택합니다. 선택한 DB 인스턴스에 대한 성능 개선 도우미 대시보드가 표시됩니다.
4. 데이터베이스 로드(DB 로드)(Database load (DB load)) 섹션에서 분할 기준(Slice by) 옆에 있는 계획(Plans)을 선택합니다.

   평균 활성 세션 차트는 상위 SQL 문이 사용하는 계획을 보여줍니다. 계획 해시 값은 색상으로 구분된 사각형의 오른쪽에 나타납니다. 각 해시 값은 계획을 고유하게 식별합니다.



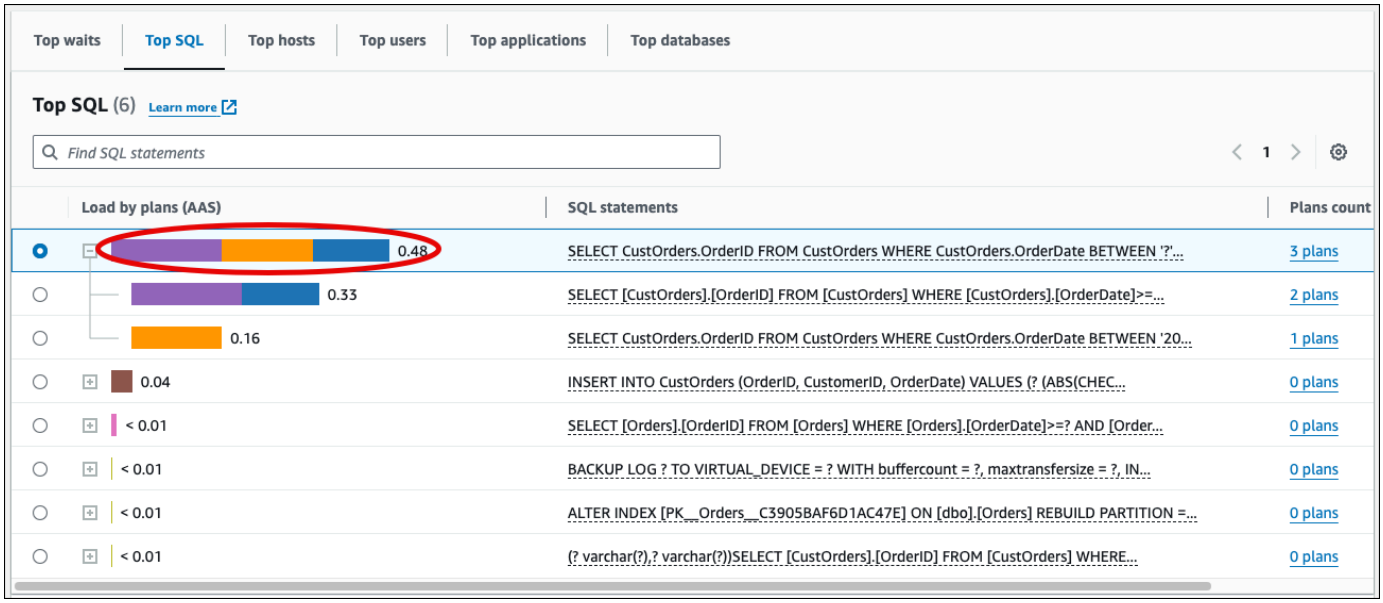
5. 상위 SQL(Top SQL) 탭까지 아래로 스크롤합니다.

   다음 예제의 경우 상위 SQL 다이제스트에는 3개의 계획이 있습니다. SQL 문에 물음표가 있으면 해당 문이 다이제스트라는 의미입니다. 전체 SQL 문을 보려면 SQL 문 열에서 값을 선택합니다.



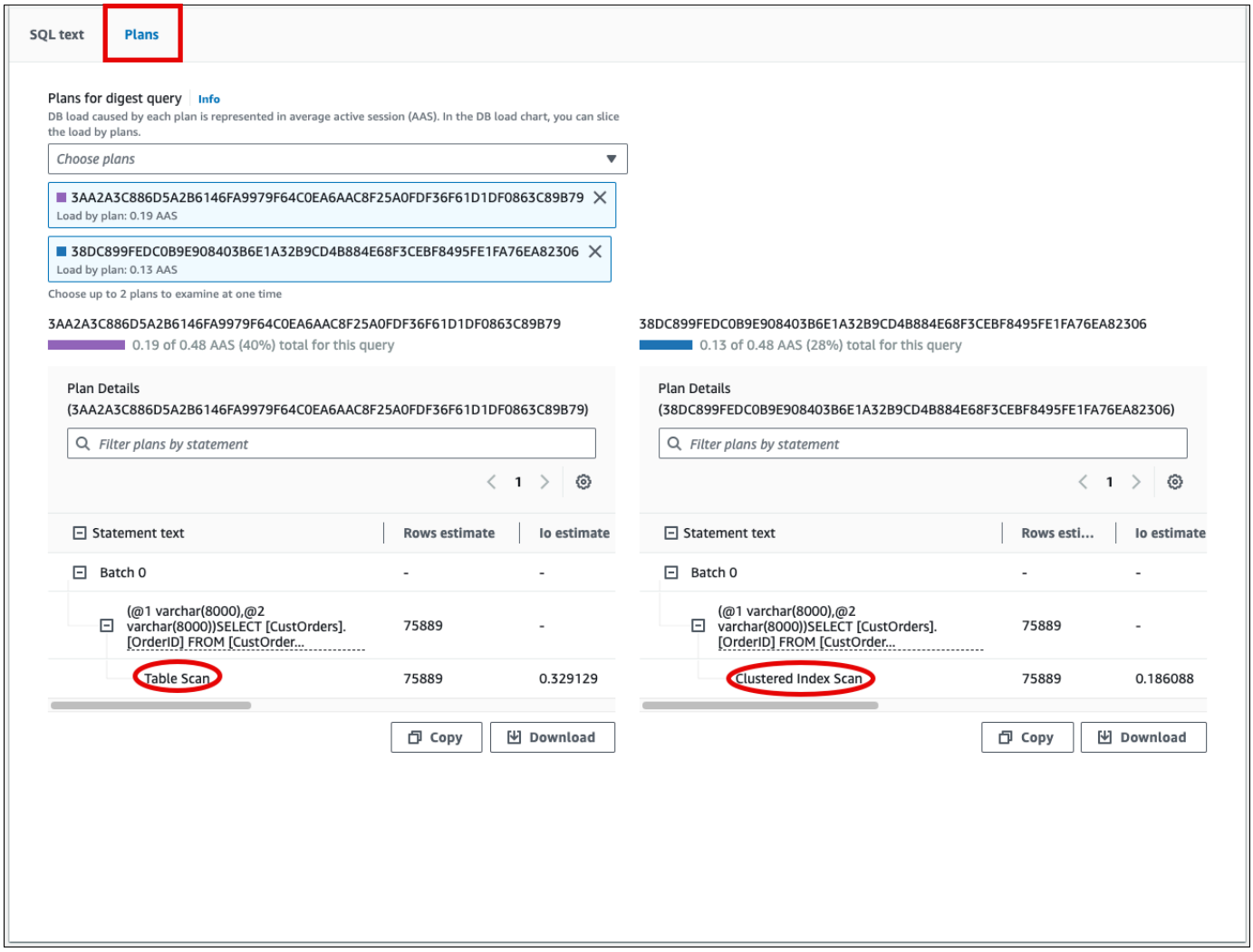
6. 다이제스트를 선택하여 구성 요소 문으로 확장합니다.

   다음 예에서 SELECT 문은 다이제스트 쿼리입니다. 다이제스트의 구성 요소 쿼리는 3개의 다른 실행 계획을 사용합니다. 계획에 할당된 색상은 데이터베이스 로드 차트에 해당합니다.



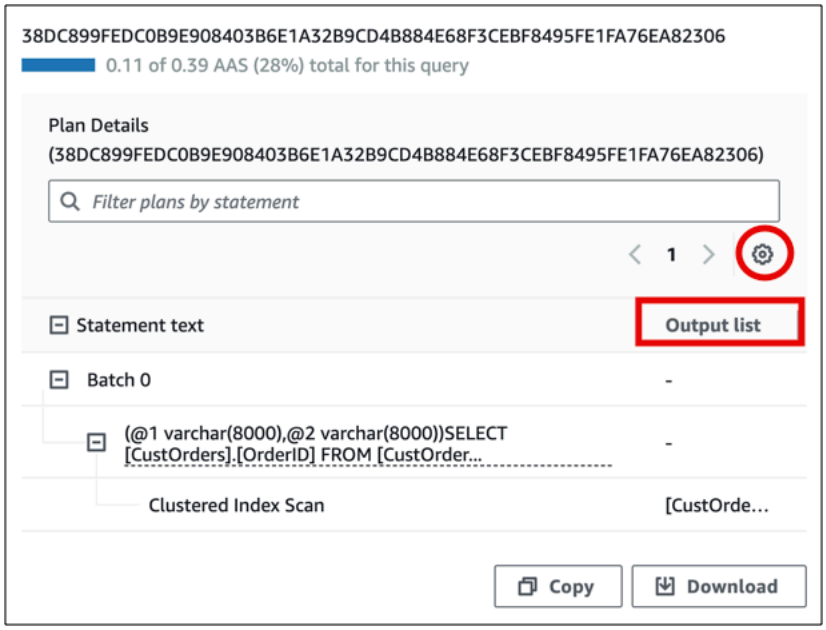
7. 아래로 스크롤하여 다이제스트 쿼리 계획(Plans for digest query) 목록에서 비교할 2개의 계획(Plans)을 선택합니다.

   쿼리에 대해 한 번에 하나 또는 2개의 계획을 볼 수 있습니다. 다음 스크린샷에서는 다이제스트의 두 계획을 비교합니다. 다음 예제에서 이 다이제스트 쿼리를 실행하는 평균 활성 세션의 40%는 왼쪽 계획을 사용하고 있는 반면 28%는 오른쪽 계획을 사용하고 있습니다.



이 예제에서 계획은 방식에서 중요한 차이가 있습니다. 왼쪽 계획의 2단계는 테이블 스캔을 사용하는 반면 오른쪽 계획은 클러스터형 인덱스 스캔을 사용합니다. 행 수가 많은 테이블의 경우 클러스터형 인덱스 스캔을 사용하면 단일 행의 쿼리 검색이 거의 항상 더 빠릅니다.

8. (선택 사항) 계획 세부 정보 테이블에서 설정 아이콘을 선택하여 열의 표시 여부 및 순서를 사용자 정의합니다. 다음 스크린샷은 결과 목록 열이 두 번째 열로 있는 계획 세부 정보 테이블을 보여줍니다.



9. (선택 사항) 복사(Copy)를 선택하여 클립보드에 계획을 복사하거나 다운로드(Download)를 선택하여 하드 드라이브에 계획을 저장합니다.

> **Note**
>
> 성능 개선 도우미는 계층적 트리 테이블을 사용하여 예상 실행 계획을 표시합니다. 테이블에는 각 문의 부분 실행 정보가 포함되어 있습니다. 계획 세부 정보 테이블의 열에 대한 자세한 내용은 SQL Server 설명서의 SET SHOWPLAN_ALL을 참조하시기 바랍니다. 예상 실행 계획의 전체 실행 정보를 표시하려면 다운로드를 선택하여 계획을 다운로드한 다음 SQL Server Management Studio에 계획을 업로드합니다. SQL Server Management Studio를 사용하여 예상 실행 계획을 표시하는 방법에 대한 자세한 내용은 SQL Server 설명서의 예상 실행 계획 표시를 참조하시기 바랍니다.

성능 개선 도우미 사전 권장 사항 보기

Amazon RDS 성능 개선 도우미는 특정 지표를 모니터링하고 특정 리소스에 대해 문제가 될 수 있다고 간주되는 수준을 분석하여 임계값을 자동으로 생성합니다. 지정된 기간 동안 새 지표 값이 사전 정의된 임계값을 초과하면 성능 개선 도우미는 사전 예방적 권장 사항을 생성합니다. 이 권장 사항은 향후 데이터베이스 성능에 미치는 영향을 방지하는 데 도움이 됩니다. 이러한 사전 예방적 권장 사항을 받으려면 유료 등급 보존 기간과 함께 성능 개선 도우미를 활성화해야 합니다.

성능 개선 도우미 켜기에 대한 자세한 내용은 Amazon RDS의 성능 개선 도우미 설정 및 해제 섹션을 참조하세요. 성능 개선 도우미의 요금 및 데이터 보존에 대한 자세한 내용은 성능 개선 도우미의 요금 및 데이터 보존 섹션을 참조하세요.

사전 예방적 권장 사항이 지원되는 리전, DB 엔진 및 인스턴스 클래스를 알아보려면 성능 개선 도우미 기능에 대한 Amazon RDS DB 엔진, 리전 및 인스턴스 클래스 지원 섹션을 참조하세요.

권장 사항 세부 정보 페이지에서 사전 권장 사항에 대한 자세한 분석 및 권장 조사를 볼 수 있습니다.

권장 사항에 대한 자세한 내용은 Amazon RDS의 권장 사항 단원을 참조하세요.

사전 예방적 권장 사항에 대한 자세한 분석을 보려면

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 다음 중 하나를 수행합니다.
   - 권장 사항을 선택합니다.

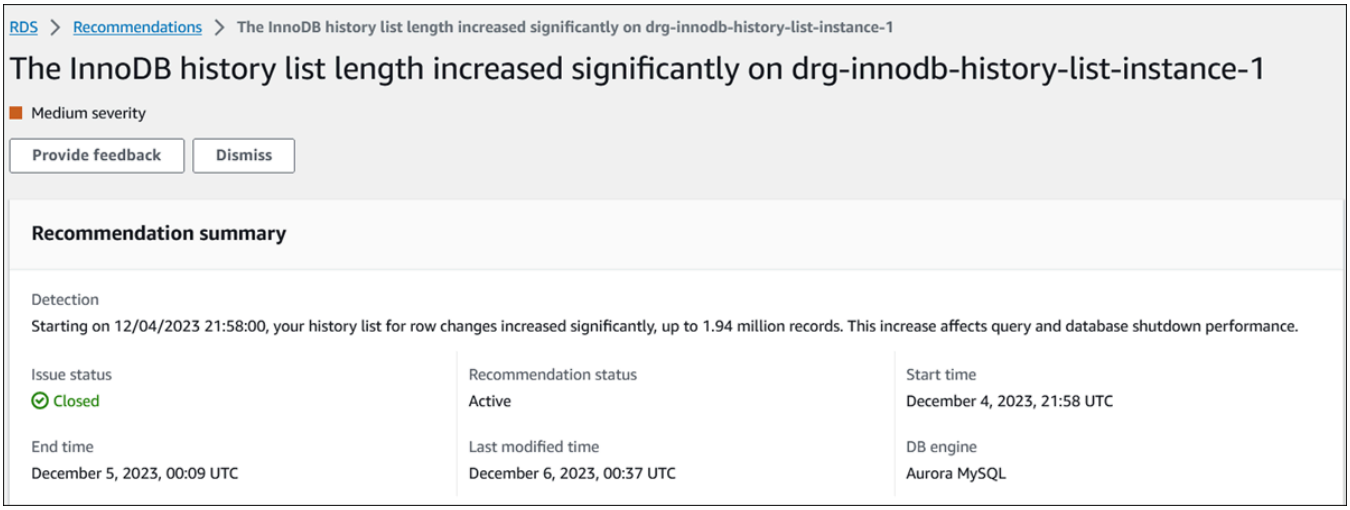
     권장 사항 페이지에는 계정 내 모든 리소스의 심각도별로 정렬된 권장 사항 목록이 표시됩니다.
   - 데이터베이스를 선택한 다음 데이터베이스 페이지에서 리소스에 대한 권장 사항을 선택합니다.

     권장 사항 탭에는 선택한 리소스에 대한 권장 사항 및 세부 정보가 표시됩니다.
3. 사전 예방적 권장 사항을 찾아 세부 정보 보기를 선택합니다.

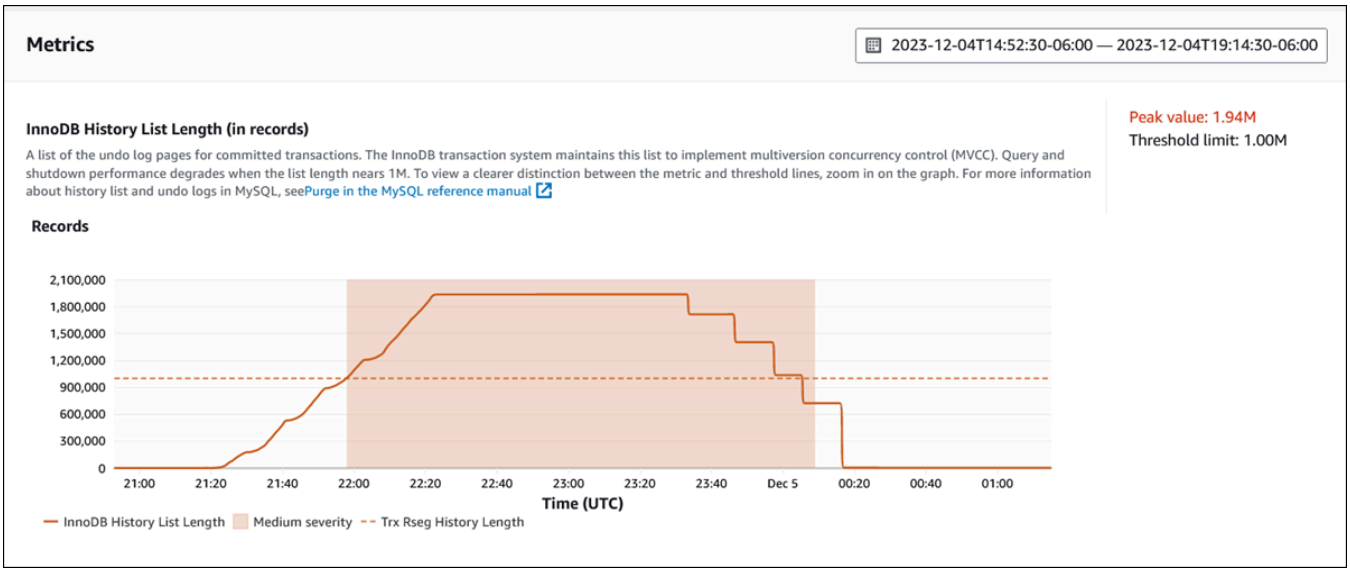
   권장 사항 세부 정보 페이지가 표시됩니다. 제목은 발견된 문제가 있는 리소스의 이름과 심각도를 제공합니다.

   권장 사항 세부 정보 페이지의 구성 요소는 다음과 같습니다.

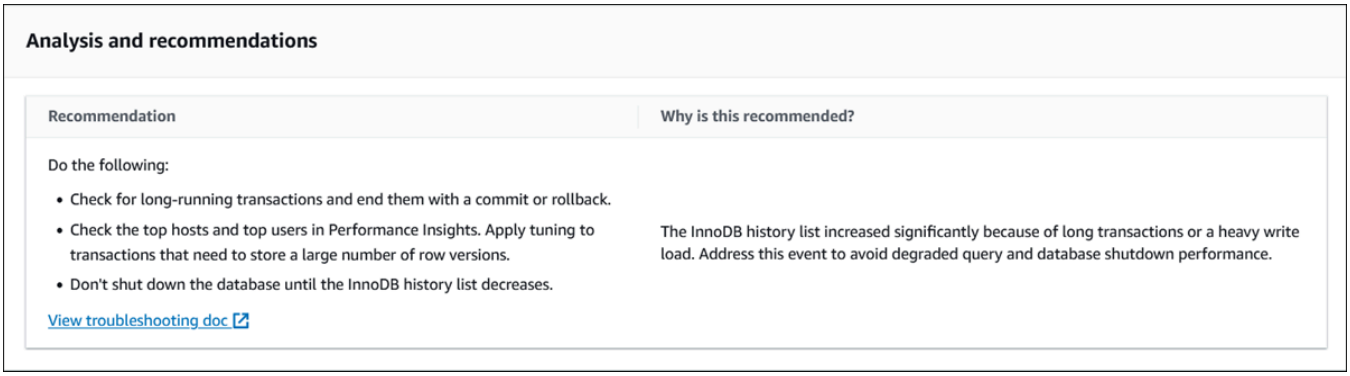
   - 권장 사항 요약 - 감지된 문제, 권장 사항 및 문제 상태, 문제 시작 및 종료 시간, 권장 사항 수정 시간, 엔진 유형입니다.



- 지표 - 감지된 문제의 그래프입니다. 각 그래프에는 리소스의 기준 동작에 따라 결정된 임계값과 이상 발생 시점부터 보고된 지표 데이터가 표시됩니다.



- 분석 및 권장 사항 - 권장 사항 및 해당 권장 사항을 제안한 이유입니다.



문제의 원인을 검토한 다음 제안된 권장 조치를 수행하여 문제를 해결하거나 오른쪽 상단의 삭제를 선택하여 권장 사항을 무시할 수 있습니다.

Amazon RDS용 성능 개선 도우미 API를 사용하여 지표 검색

성능 개선 도우미를 켜면 API에서 인스턴스 성능에 대한 가시성을 제공합니다. Amazon CloudWatch Logs는 AWS 서비스의 벤딩 모니터링 지표에 대해 신뢰할 수 있는 소스를 제공합니다.

성능 개선 도우미는 평균 활성 세션(AAS) 수로 측정되는 데이터베이스 로드에 대한 도메인 별 보기를 제공합니다. 이 지표는 API 소비자에게 2차원 시계열 데이터 세트로 표시됩니다. 데이터의 시간 차원은 쿼리된 시간 범위 내 각 시점에 대한 DB 로드 데이터를 제공합니다. 각 시점에서는 요청된 차원에 관해 해당 시점에서 측정되는 전체 부하를 분해합니다(예: `SQL`, `Wait-event`, `User` 또는 `Host`).

Amazon RDS Performance Insights는 데이터베이스 성능을 분석하고 관련 문제를 해결할 수 있도록 Amazon RDS DB 인스턴스를 모니터링합니다. 성능 개선 도우미 데이터를 볼 수 있는 한 가지 방법은 AWS Management Console에서 보는 것입니다. 또한 성능 개선 도우미는 사용자가 자신의 데이터를 쿼리할 수 있도록 퍼블릭 API도 제공합니다. API를 사용하여 다음을 수행할 수 있습니다:

- 데이터를 데이터베이스로 오프로드
- 기존 모니터링 대시보드에 성능 개선 도우미 데이터 추가
- 모니터링 도구 구축

성능 개선 도우미 API를 사용하려면 Amazon RDS DB 인스턴스 중 하나에서 성능 개선 도우미를 활성화하십시오. 성능 개선 도우미 활성화에 대한 자세한 내용은 Amazon RDS의 성능 개선 도우미 설정 및 해제 단원을 참조하십시오. 성능 개선 도우미 API에 대한 자세한 내용은 Amazon RDS 성능 개선 도우미 API 참조를 참조하십시오.

성능 개선 도우미 API에서는 다음과 같은 작업을 제공합니다.

성능 개선 도우미 작업	AWS CLI 명령	설명
`CreatePerformanceAnalysisReport`	`aws pi create-performance-analysis-report`	DB 인스턴스의 특정 기간에 대한 성능 분석 보고서를 생성합니다. 결과는 보고서의 고유 식별자인 `AnalysisReportId` 입니다.

성능 개선 도우미 작업	AWS CLI 명령	설명
DeletePerformanceAnalysisReport	aws pi delete-performance-analysis-report	성능 분석 보고서를 삭제합니다.
DescribeDimensionKeys	aws pi describe-dimension-keys	특정 기간에 대해 지표의 상위 N개 차원 키를 검색합니다.
GetDimensionKeyDetails	aws pi get-dimension-key-details	DB 인스턴스 또는 데이터 소스에 지정된 차원 그룹의 속성을 검색합니다. 예를 들어 SQL ID를 지정하고 차원 세부 정보를 사용할 수 있는 경우 GetDimensionKeyDetails 는 이 ID에 연결된 차원 db.sql.statement 의 전체 텍스트를 검색합니다. GetResourceMetrics 및 DescribeDimensionKeys 는 대용량 SQL 문 텍스트 검색을 지원하지 않으므로 이 작업을 유용하게 사용할 수 있습니다.
GetPerformanceAnalysisReport	aws pi get-performance-analysis-report	보고서에 대한 인사이트가 포함된 보고서를 검색합니다. 결과에는 보고서 상태, 보고서 ID, 보고서 시간 세부 정보, 인사이트 및 권장 사항이 포함됩니다.
GetResourceMetadata	aws pi get-resource-metadata	다양한 기능에 대한 특성을 검색합니다. 예를 들어 메타데이터는 특정 DB 인스턴스에서 특성이 켜지거나 꺼짐을 나타낼 수 있습니다.

성능 개선 도우미 작업	AWS CLI 명령	설명
GetResourceMetrics	aws pi get-resource-metrics	일정 기간의 데이터 소스 집합에 대한 성능 개선 도우미 지표를 검색합니다. 특정 차원 그룹 및 차원을 제공하고 각 그룹에 집계 및 필터링 기준을 제공할 수 있습니다.
ListAvailableResourceDimensions	aws pi list-available-resource-dimensions	지정된 인스턴스에서 지정된 각 지표 유형에 대해 쿼리할 수 있는 차원을 검색합니다.
ListAvailableResourceMetrics	aws pi list-available-resource-metrics	지정된 DB 인스턴스에 대해 쿼리할 수 있는 지정된 지표 유형의 사용 가능한 모든 지표를 검색합니다.
ListPerformanceAnalysisReports	aws pi list-performance-analysis-reports	DB 인스턴스에 대해 사용할 수 있는 모든 분석 보고서를 검색합니다. 각 보고서의 시작 시간을 기준으로 보고서가 나열됩니다.
ListTagsForResource	aws pi list-tags-for-resource	리소스에 추가된 모든 메타데이터 태그를 나열합니다. 목록에는 태그의 이름과 값이 포함됩니다.
TagResource	aws pi tag-resource	메타데이터 태그를 Amazon RDS 리소스에 추가합니다. 태그에는 이름과 값이 포함됩니다.
UntagResource	aws pi untag-resource	리소스에서 메타데이터 태그를 제거합니다.

성능 개선 도우미의 AWS CLI 예제 및 시계열 지표 검색에 대한 자세한 내용은 다음 주제를 참조하세요.

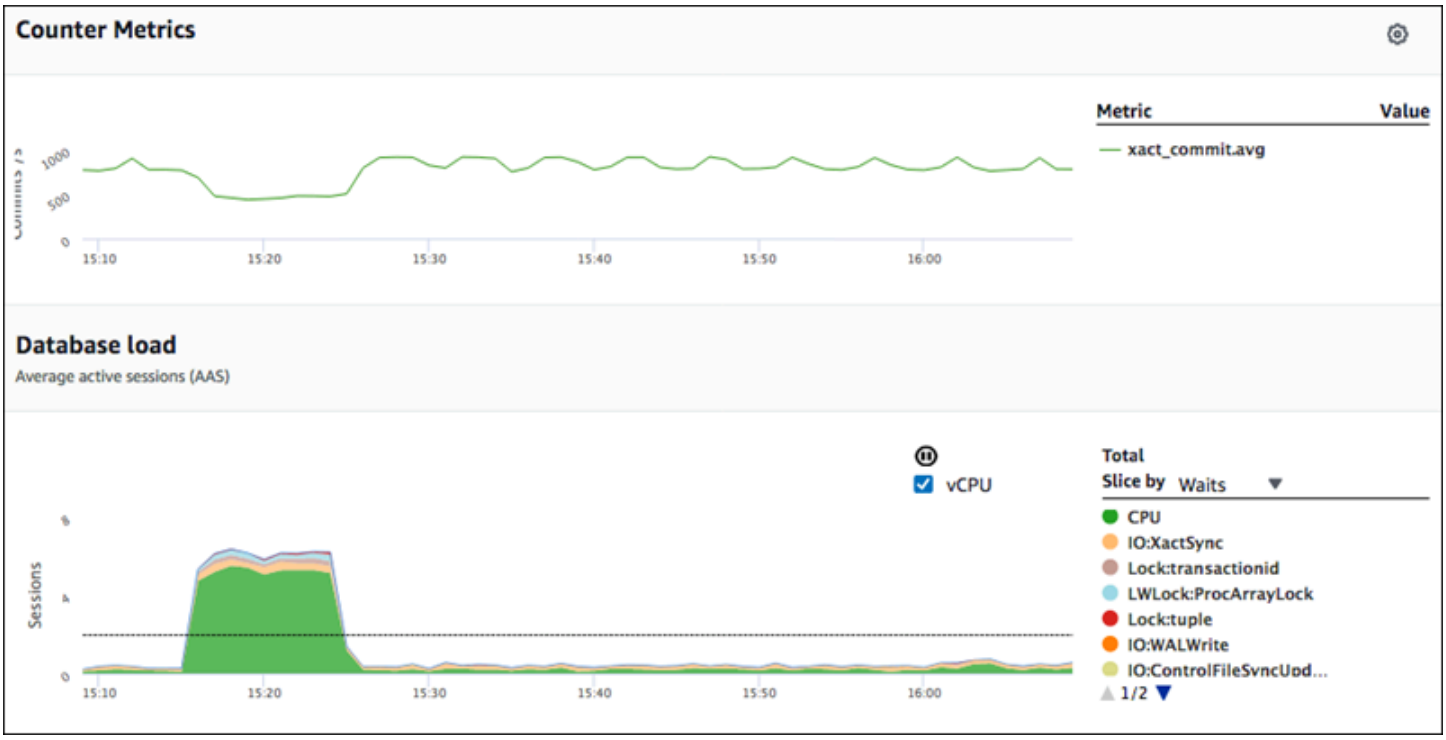
주제

- 성능 개선 도우미의 시계열 지표 조회
- 성능 개선 도우미에 대한 AWS CLI 예시

성능 개선 도우미의 시계열 지표 조회

`GetResourceMetrics` 연산은 성능 개선 도우미 데이터에서 시계열 지표를 하나 이상 조회합니다. `GetResourceMetrics`에는 지표 및 기간이 필요하고 데이터 포인트 목록이 포함된 응답을 반환합니다.

예를 들어 AWS Management Console에서 `GetResourceMetrics`는 다음 이미지에 표시된 것과 같이 [카운터 지표(Counter Metrics)] 차트와 [데이터베이스 로드(Database Load)] 차트를 채우는 데 사용됩니다.



`GetResourceMetrics`에서 반환하는 지표는 `db.load`를 제외하고 모두 표준 시계열 지표입니다. 이 지표는 Database Load(데이터베이스 로드) 차트에 표시됩니다. `db.load` 지표는 차원이라는 하위 구성 요소로 구분할 수 있다는 점에서 다른 시계열 지표와 다릅니다. 앞의 이미지에서 `db.load`는 `db.load`를 구성하는 대기 상태에 따라 구분되고 그룹화됩니다.

> **Note**
>
> GetResourceMetrics에서는 db.sampleload도 반환할 수 있지만 db.load 지표는 대부분의 경우 적절합니다.

GetResourceMetrics에서 반환하는 카운터 지표에 대한 자세한 내용은 성능 개선 도우미 카운터를 참조하십시오.

지표에 대해서는 다음 계산이 지원됩니다:

- 평균 – 일정 기간 동안 지표의 평균 값입니다. .avg를 지표 이름에 추가합니다.
- 최소 – 일정 기간 동안 지표의 최소 값입니다. .min를 지표 이름에 추가합니다.
- 최대 – 일정 기간 동안 지표의 최대 값입니다. .max를 지표 이름에 추가합니다.
- 합계 – 일정 기간 동안 지표 값의 합계입니다. .sum를 지표 이름에 추가합니다.
- 샘플 수 – 일정 기간 동안 지표가 수집된 횟수입니다. .sample_count를 지표 이름에 추가합니다.

예를 들어 지표를 300초 (5분) 동안 분당 1회씩 수집한다고 가정합시다. 각 분의 값은 1, 2, 3, 4, 5입니다. 이 경우 다음과 같은 계산 결과가 반환됩니다:

- 평균 – 3
- 최소 – 1
- 최대 – 5
- 합계 – 15
- 샘플 수 – 5

get-resource-metrics AWS CLI 명령 사용에 대한 자세한 내용은 get-resource-metrics 섹션을 참조하세요.

--metric-queries 옵션의 경우 결과를 얻고자 하는 쿼리를 한 개 이상 지정하십시오. 각 쿼리는 필수인 Metric과 선택 사항인 GroupBy 및 Filter 파라미터로 구성됩니다. 다음은 --metric-queries 옵션 사양을 보여주는 예입니다.

```
{
   "Metric": "string",
   "GroupBy": {
```

```
    "Group": "string",
    "Dimensions": ["string", ...],
    "Limit": integer
  },
  "Filter": {"string": "string"
    ...}
```

성능 개선 도우미에 대한 AWS CLI 예시

다음 섹션에서 성능 개선 도우미의 AWS Command Line Interface(AWS CLI)에 대해 자세히 알아보고 AWS CLI 예제를 사용하세요.

주제

- 성능 개선 도우미에 대한 AWS CLI의 기본 제공 도움말
- 카운터 지표 검색
- 상위 대기 이벤트에 대한 DB 평균 로드 검색
- 상위 SQL에 대한 DB 평균 로드 검색
- SQL을 기준으로 필터링된 DB 평균 로드 검색
- SQL 문의 전체 텍스트 검색
- 특정 기간에 대한 성과 분석 보고서 생성
- 성능 분석 보고서 검색
- DB 인스턴스에 대한 모든 성능 분석 보고서 나열
- 성능 분석 보고서 삭제
- 성능 분석 보고서에 태그 추가
- 성능 분석 보고서에 대한 모든 태그 나열
- 성능 분석 보고서에서 태그 삭제

성능 개선 도우미에 대한 AWS CLI의 기본 제공 도움말

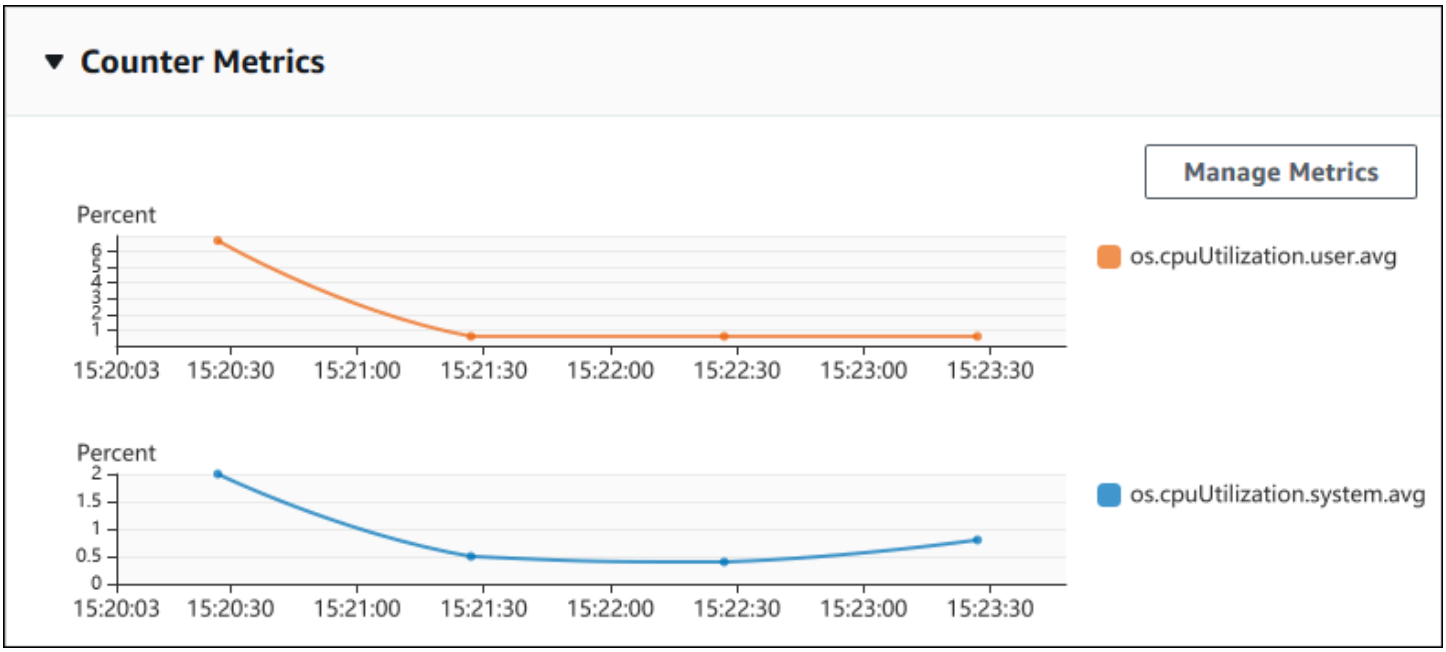
AWS CLI를 사용해 성능 개선 도우미 데이터를 볼 수 있습니다. 명령줄에 다음과 같이 입력하여 성능 개선 도우미용 AWS CLI 명령에 대한 도움말을 볼 수 있습니다.

```
aws pi help
```

AWS CLI가 설치되어 있지 않은 경우 설치에 대한 자세한 내용은 AWS CLI 사용 설명서의 AWS CLI 설치를 참조하세요.

카운터 지표 검색

다음 스크린샷은 AWS Management Console에 표시되는 카운터 지표 차트 2개를 나타낸 것입니다.



다음 예에서는 카운터 지표 차트 2개를 생성하기 위해 AWS Management Console이 사용하는 것과 동일한 데이터를 수집하는 방법을 보여줍니다.

대상 LinuxmacOS, 또는Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries '[{"Metric": "os.cpuUtilization.user.avg"  },
                      {"Metric": "os.cpuUtilization.idle.avg"}]'
```

Windows의 경우:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
```

```
   --metric-queries '[{"Metric": "os.cpuUtilization.user.avg"  },
                      {"Metric": "os.cpuUtilization.idle.avg"}]'
```

`--metrics-query` 옵션에 대해 파일을 지정하면 명령이 더 쉽게 읽히도록 할 수 있습니다. 다음 예에서는 옵션에 대해 query.json이라는 파일을 사용합니다. 이 파일의 콘텐츠는 다음과 같습니다.

```
[
    {
        "Metric": "os.cpuUtilization.user.avg"
    },
    {
        "Metric": "os.cpuUtilization.idle.avg"
    }
]
```

다음 명령을 실행하여 파일을 사용합니다.

대상 LinuxmacOS, 또는Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Windows의 경우:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

앞의 예에서는 옵션에 다음 값을 지정합니다.

- `--service-type` – Amazon RDS용 RDS
- `--identifier` – DB 인스턴스에 대한 리소스 ID입니다.

- `--start-time` 및 `--end-time` – 쿼리할 기간에 대한 ISO 8601 `DateTime` 값으로서, 지원되는 형식은 여러 가지입니다

다음과 같이 1시간 범위로 쿼리합니다:

- `--period-in-seconds` – 1분당 쿼리에 대한 `60`
- `--metric-queries` – 쿼리 2개의 배열, 각 쿼리는 지표 1개에만 해당됨.

  지표 이름에는 지표를 유용한 범주로 분류하기 위해 점이 사용되고, 마지막 요소는 함수입니다. 예시에서 함수는 각 쿼리에 대해 `avg`입니다. Amazon CloudWatch와 마찬가지로 지원되는 함수는 `min`, `max`, `total` 및 `avg`입니다.

응답은 다음과 비슷합니다.

```
{
    "Identifier": "db-XXX",
    "AlignedStartTime": 1540857600.0,
    "AlignedEndTime": 1540861200.0,
    "MetricList": [
        { //A list of key/datapoints
            "Key": {
                "Metric": "os.cpuUtilization.user.avg" //Metric1
            },
            "DataPoints": [
                //Each list of datapoints has the same timestamps and same number of
 items
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 4.0
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 4.0
                },
                {
                    "Timestamp": 1540857780.0, //Minute 3
                    "Value": 10.0
                }
                //... 60 datapoints for the os.cpuUtilization.user.avg metric
            ]
        },
```

```
        {
            "Key": {
                "Metric": "os.cpuUtilization.idle.avg" //Metric2
            },
            "DataPoints": [
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 12.0
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 13.5
                },
                //... 60 datapoints for the os.cpuUtilization.idle.avg metric
            ]
        }
    ] //end of MetricList
} //end of response
```

응답에는 Identifier, AlignedStartTime 및 AlignedEndTime이 있습니다. --period-in-seconds 값이 60인 경우 시작 및 종료 시간은 분 단위로 맞춰져 있습니다. --period-in-seconds 값이 3600인 경우 시작 및 종료 시간은 시간 단위로 맞춰져 있습니다.

응답의 MetricList에는 다수의 항목이 있는데, 각각 Key 및 DataPoints 항목이 포함되어 있습니다. 각 DataPoint에는 Timestamp 및 Value이 있습니다. 쿼리는 1시간에 걸친 분당 데이터에 대한 것이므로 각 Datapoints 목록에는 Timestamp1/Minute1, Timestamp2/Minute2 등에서 최대 Timestamp60/Minute60까지 60개의 데이터 포인트가 있습니다.

쿼리는 두 가지 카운터 지표에 대한 것이므로 MetricList 응답에는 두 개의 요소가 있습니다.

상위 대기 이벤트에 대한 DB 평균 로드 검색

다음 예는 AWS Management Console에서 누적 영역 선 그래프를 생성하는 데 사용하는 것과 동일한 쿼리입니다. 이 예에서는 최상위 7개 대기 이벤트에 따라 구분된 로드의 마지막 한 시간 db.load.avg를 검색합니다. 명령은 카운터 지표 검색의 명령과 동일합니다. 그러나 query.json 파일의 컨텐츠는 다음과 같습니다.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_event", "Limit": 7 }
    }
```

```
]
```

다음 명령을 실행합니다.

대상 LinuxmacOS, 또는Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Windows의 경우:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

이 예시에서는 최상위 7개 대기 이벤트의 db.load.avg 및 GroupBy에 대한 지표를 지정합니다. 이 예의 유효 값에 대한 자세한 내용은 성능 개선 도우미 API 참조의 DimensionGroup 단원을 참조하십시오.

응답은 다음과 비슷합니다.

```
{
    "Identifier": "db-XXX",
    "AlignedStartTime": 1540857600.0,
    "AlignedEndTime": 1540861200.0,
    "MetricList": [
        { //A list of key/datapoints
            "Key": {
                //A Metric with no dimensions. This is the total db.load.avg
                "Metric": "db.load.avg"
            },
            "DataPoints": [
```

```
                //Each list of datapoints has the same timestamps and same number of
 items
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 0.5166666666666667
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 0.38333333333333336
                },
                {
                    "Timestamp": 1540857780.0, //Minute 3
                    "Value": 0.26666666666666666
                }
                //... 60 datapoints for the total db.load.avg key
            ]
        },
        {
            "Key": {
                //Another key. This is db.load.avg broken down by CPU
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.name": "CPU",
                    "db.wait_event.type": "CPU"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 0.35
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 0.15
                },
                //... 60 datapoints for the CPU key
            ]
        },
        //... In total we have 8 key/datapoints entries, 1) total, 2-8) Top Wait Events
    ] //end of MetricList
} //end of response
```

이 응답에는 MetricList에 항목이 8개 있습니다. 총 db.load.avg에는 항목이 1개 있고, 최상위 7개 대기 이벤트 중 하나에 따라 구분된 db.load.avg에 각각에 대해서는 항목이 7개 있습니다. 첫 번째 예시와 달리 그룹화 차원이 있었기 때문에 지표에 대한 각 그룹화에는 키가 1개 있어야 합니다. 기본 카운터 지표 사용 사례처럼 각 지표에 키가 한 개만 있을 수는 없습니다.

상위 SQL에 대한 DB 평균 로드 검색

다음 예에서는 최상위 10개 SQL 문을 기준으로 db.wait_events를 그룹화합니다. SQL 문에는 두 가지 그룹이 있습니다.

- db.sql – 와 같은 SQL 문select * from customers where customer_id = 123
- db.sql_tokenized – 와 같은 토큰화된 SQL 문select * from customers where customer_id = ?

데이터베이스 성능 분석 시 파라미터만 다른 SQL 문은 하나의 로직 항목으로 간주하는 것이 도움이 될 수 있습니다. 따라서 쿼리 시에는 db.sql_tokenized를 사용할 수 있습니다. 그러나 특히 설명 계획에 관심이 있는 경우에는 때로 파라미터가 있는 전체 SQL 문을 검토하고 db.sql로 그룹화를 쿼리하는 것이 유용합니다. 토큰화된 SQL과 전체 SQL 간에는 상위-하위 관계가 있는데, 여러 개의 전체 SQL(하위)은 토큰화된 동일한 SQL(상위) 아래에 그룹화됩니다.

이 예의 명령은 상위 대기 이벤트에 대한 DB 평균 로드 검색의 명령과 유사합니다. 그러나 query.json 파일의 컨텐츠는 다음과 같습니다.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.sql_tokenized", "Limit": 10 }
    }
]
```

다음 예에는 db.sql_tokenized가 사용됩니다.

대상 LinuxmacOS, 또는Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-29T00:00:00Z \
   --end-time   2018-10-30T00:00:00Z \
```

```
    --period-in-seconds 3600 \
    --metric-queries file://query.json
```

Windows의 경우:

```
aws pi get-resource-metrics ^
    --service-type RDS ^
    --identifier db-ID ^
    --start-time 2018-10-29T00:00:00Z ^
    --end-time   2018-10-30T00:00:00Z  ^
    --period-in-seconds 3600 ^
    --metric-queries file://query.json
```

이 예에서는 24시간 동안 쿼리를 실행하는데 1시간은 초 단위로 구성됩니다.

이 예시에서는 최상위 7개 대기 이벤트의 `db.load.avg` 및 `GroupBy`에 대한 지표를 지정합니다. 이 예의 유효 값에 대한 자세한 내용은 성능 개선 도우미 API 참조의 DimensionGroup 단원을 참조하십시오.

응답은 다음과 비슷합니다.

```
{
    "AlignedStartTime": 1540771200.0,
    "AlignedEndTime": 1540857600.0,
    "Identifier": "db-XXX",

    "MetricList": [ //11 entries in the MetricList
        {
            "Key": { //First key is total
                "Metric": "db.load.avg"
            }
            "DataPoints": [ //Each DataPoints list has 24 per-hour Timestamps and a
 value
                {
                    "Value": 1.6964980544747081,
                    "Timestamp": 1540774800.0
                },
                //... 24 datapoints
            ]
        },
        {
            "Key": { //Next key is the top tokenized SQL
```

```
                "Dimensions": {
                    "db.sql_tokenized.statement": "INSERT INTO authors (id,name,email)
 VALUES\n( nextval(?)  ,?,?)",
                    "db.sql_tokenized.db_id": "pi-2372568224",
                    "db.sql_tokenized.id": "AKIAIOSFODNN7EXAMPLE"
                },
                "Metric": "db.load.avg"
            },
            "DataPoints": [ //... 24 datapoints
            ]
        },
        // In total 11 entries, 10 Keys of top tokenized SQL, 1 total key
    ] //End of MetricList
} //End of response
```

이 응답은 MetricList에 11개의 항목이 있는데(전체 1개, 최상위 토큰화 SQL 10개) 각 항목에는 시간당 DataPoints가 24개입니다.

토큰화된 SQL의 경우 각 차원 목록에 3개의 항목이 있습니다.

- db.sql_tokenized.statement – 토큰화된 SQL 문입니다.
- db.sql_tokenized.db_id – SQL 참조에 사용되는 기본 데이터베이스 ID 또는 기본 데이터베이스 ID를 사용할 수 없는 경우 성능 개선 도우미가 생성하는 합성 ID입니다. 이 예에서는 pi-2372568224 합성 ID를 반환합니다.
- db.sql_tokenized.id – 성능 개선 도우미 내부의 쿼리에 대한 ID입니다.

  AWS Management Console에서는 이 ID를 지원 ID라고 합니다. ID는 AWS Support에서 데이터베이스 문제를 해결하기 위해 조사할 수 있는 데이터이기 때문에 이 이름이 지정됩니다. AWS는 데이터의 보안 및 개인 정보를 매우 중요하게 취급하며, 거의 모든 데이터는 AWS KMS 키로 암호화되어 저장됩니다. 그러므로 AWS 내부의 어느 누구도 이 데이터를 볼 수 없습니다. 앞의 예에서 tokenized.statement와 tokenized.db_id 모두 암호화되어 저장됩니다. 데이터베이스 관련 문제가 있는 경우 AWS Support가 지원 ID를 참조하여 도움을 드릴 수 있습니다.

쿼리 시 Group에서 GroupBy을 지정하면 편리할 수 있습니다. 그러나 반환되는 데이터에 대한 더 세분화된 제어를 위해서는 차원 목록을 지정하십시오. 예를 들어 db.sql_tokenized.statement만 필요한 경우에는 query.json file에 Dimensions 속성을 추가할 수 있습니다.

```
[
    {
        "Metric": "db.load.avg",
```

```
        "GroupBy": {
            "Group": "db.sql_tokenized",
            "Dimensions":["db.sql_tokenized.statement"],
            "Limit": 10
        }
    }
]
```

SQL을 기준으로 필터링된 DB 평균 로드 검색



앞의 이미지에서는 특정 쿼리가 선택되어 있고 상위 평균 활성 세션 누적 영역 선 그래프는 이 쿼리로 범위가 지정되어 있습니다. 쿼리가 여전히 최상위 7개 전체 대기 이벤트에 대한 것이라 하더라도 응답의 값은 필터링됩니다. 필터로 인해 특정 필터의 짝이 되는 세션만 고려합니다.

이 예에서 해당되는 API 쿼리는 상위 SQL에 대한 DB 평균 로드 검색 단원의 명령과 유사합니다. 그러나 query.json 파일의 컨텐츠는 다음과 같습니다.

```
[
 {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_event", "Limit": 5  },
        "Filter": { "db.sql_tokenized.id": "AKIAIOSFODNN7EXAMPLE" }
    }
```

```
]
```

대상 LinuxmacOS, 또는Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Windows의 경우:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

응답은 다음과 비슷합니다.

```
{
    "Identifier": "db-XXX",
    "AlignedStartTime": 1556215200.0,
    "MetricList": [
        {
            "Key": {
                "Metric": "db.load.avg"
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 1.4878117913832196
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 1.192823803967328
                }
            ]
```

```
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "io",
                    "db.wait_event.name": "wait/io/aurora_redo_log_flush"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 1.1360544217687074
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 1.058051341890315
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "io",
                    "db.wait_event.name": "wait/io/table/sql/handler"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.16241496598639457
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.05163360560093349
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "synch",
```

```
                    "db.wait_event.name": "wait/synch/mutex/innodb/
aurora_lock_thread_slot_futex"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.11479591836734694
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.013127187864644107
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "CPU",
                    "db.wait_event.name": "CPU"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.05215419501133787
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.05805134189031505
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "synch",
                    "db.wait_event.name": "wait/synch/mutex/innodb/lock_wait_mutex"
                }
            },
            "DataPoints": [
                {
```

```
                    "Timestamp": 1556218800.0,
                    "Value": 0.017573696145124718
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.002333722287047841
                }
            ]
        }
    ],
    "AlignedEndTime": 1556222400.0
} //end of response
```

이 응답에서 모든 값은 query.json 파일에 지정된 토큰화된 SQL AKIAIOSFODNN7EXAMPLE의 기여에 따라 필터링됩니다. 키는 필터링된 SQL에 영향을 미친 상위 5개 대기 이벤트이므로 필터가 없는 쿼리와는 다른 순서를 따를 수 있습니다.

SQL 문의 전체 텍스트 검색

다음 예제에서는 DB 인스턴스 db-10BCD2EFGHIJ3KL4M5NO6PQRS5에 대한 SQL 문의 전체 텍스트를 검색합니다. --group은 db.sql이고 --group-identifier는 db.sql.id입니다. 이 예제에서 *my-sql-id*는 pi get-resource-metrics 또는 pi describe-dimension-keys를 호출하여 검색된 SQL ID를 나타냅니다.

다음 명령을 실행합니다.

대상 LinuxmacOS, 또는Unix:

```
aws pi get-dimension-key-details \
   --service-type RDS \
   --identifier db-10BCD2EFGHIJ3KL4M5NO6PQRS5 \
   --group db.sql \
   --group-identifier my-sql-id \
   --requested-dimensions statement
```

Windows의 경우:

```
aws pi get-dimension-key-details ^
   --service-type RDS ^
   --identifier db-10BCD2EFGHIJ3KL4M5NO6PQRS5 ^
   --group db.sql ^
   --group-identifier my-sql-id ^
```

```
   --requested-dimensions statement
```

이 예제에서는 차원 세부 정보를 사용할 수 있습니다. 따라서 성능 개선 도우미는 잘리지 않은 SQL 문의 전체 텍스트를 검색합니다.

```
{
    "Dimensions":[
    {
        "Value": "SELECT e.last_name, d.department_name FROM employees e, departments d
 WHERE e.department_id=d.department_id",
        "Dimension": "db.sql.statement",
        "Status": "AVAILABLE"
    },
    ...
    ]
}
```

특정 기간에 대한 성과 분석 보고서 생성

다음 예시에서는 db-loadtest-0 데이터베이스에 대해 1682969503 시작 시간과 1682979503 종료 시간을 사용하여 성능 분석 보고서를 생성합니다.

```
aws pi create-performance-analysis-report \
        --service-type RDS \
        --identifier db-loadtest-0 \
        --start-time 1682969503 \
        --end-time 1682979503 \
        --region us-west-2
```

응답은 보고서의 고유 식별자인 report-0234d3ed98e28fb17입니다.

```
{
   "AnalysisReportId": "report-0234d3ed98e28fb17"
}
```

성능 분석 보고서 검색

다음 예시에서는 report-0d99cc91c4422ee61 보고서에 대한 분석 보고서 세부 정보를 검색합니다.

```
aws pi get-performance-analysis-report \
```

```
--service-type RDS \
--identifier db-loadtest-0 \
--analysis-report-id report-0d99cc91c4422ee61 \
--region us-west-2
```

응답은 보고서 상태, ID, 시간 세부 정보, 인사이트를 제공합니다.

```
        {
    "AnalysisReport": {
        "Status": "Succeeded",
        "ServiceType": "RDS",
        "Identifier": "db-loadtest-0",
        "StartTime": 1680583486.584,
        "AnalysisReportId": "report-0d99cc91c4422ee61",
        "EndTime": 1680587086.584,
        "CreateTime": 1680587087.139,
        "Insights": [
           ... (Condensed for space)
        ]
    }
}
```

DB 인스턴스에 대한 모든 성능 분석 보고서 나열

다음 예시에서는 db-loadtest-0 데이터베이스에 대해 사용 가능한 모든 성능 분석 보고서를 나열합니다.

```
aws pi list-performance-analysis-reports \
--service-type RDS \
--identifier db-loadtest-0 \
--region us-west-2
```

응답에는 보고서 ID, 상태 및 기간 세부 정보와 함께 모든 보고서가 나열됩니다.

```
{
    "AnalysisReports": [
        {
            "Status": "Succeeded",
            "EndTime": 1680587086.584,
            "CreationTime": 1680587087.139,
            "StartTime": 1680583486.584,
```

```
            "AnalysisReportId": "report-0d99cc91c4422ee61"
        },
        {
            "Status": "Succeeded",
            "EndTime": 1681491137.914,
            "CreationTime": 1681491145.973,
            "StartTime": 1681487537.914,
            "AnalysisReportId": "report-002633115cc002233"
        },
        {
            "Status": "Succeeded",
            "EndTime": 1681493499.849,
            "CreationTime": 1681493507.762,
            "StartTime": 1681489899.849,
            "AnalysisReportId": "report-043b1e006b47246f9"
        },
        {
            "Status": "InProgress",
            "EndTime": 1682979503.0,
            "CreationTime": 1682979618.994,
            "StartTime": 1682969503.0,
            "AnalysisReportId": "report-01ad15f9b88bcbd56"
        }
    ]
}
```

성능 분석 보고서 삭제

다음 예시에서는 db-loadtest-0 데이터베이스에 대한 분석 보고서를 삭제합니다.

```
aws pi delete-performance-analysis-report \
--service-type RDS \
--identifier db-loadtest-0 \
--analysis-report-id report-0d99cc91c4422ee61 \
--region us-west-2
```

성능 분석 보고서에 태그 추가

다음 예시에서는 report-01ad15f9b88bcbd56 보고서에 키 name 및 값 test-tag로 태그를 추가합니다.

```
aws pi tag-resource \
```

```
--service-type RDS \
--resource-arn arn:aws:pi:us-west-2:356798100956:perf-reports/RDS/db-loadtest-0/
report-01ad15f9b88bcbd56 \
--tags Key=name,Value=test-tag \
--region us-west-2
```

성능 분석 보고서에 대한 모든 태그 나열

다음 예시에서는 report-01ad15f9b88bcbd56 보고서에 대한 태그를 모두 나열합니다.

```
aws pi list-tags-for-resource \
--service-type RDS \
--resource-arn arn:aws:pi:us-west-2:356798100956:perf-reports/RDS/db-loadtest-0/
report-01ad15f9b88bcbd56 \
--region us-west-2
```

응답에는 보고서에 추가된 모든 태그의 값과 키가 나열됩니다.

```
{
    "Tags": [
        {
            "Value": "test-tag",
            "Key": "name"
        }
    ]
}
```

성능 분석 보고서에서 태그 삭제

다음 예시에서는 report-01ad15f9b88bcbd56 보고서에서 name 태그를 삭제합니다.

```
aws pi untag-resource \
--service-type RDS \
--resource-arn arn:aws:pi:us-west-2:356798100956:perf-reports/RDS/db-loadtest-0/
report-01ad15f9b88bcbd56 \
--tag-keys name \
--region us-west-2
```

태그가 삭제된 후 list-tags-for-resource API를 호출하면 이 태그를 나열하지 않습니다.

AWS CloudTrail을 사용하여 Performance Insights 호출 로깅

성능 개선 도우미는 사용자, 역할 또는 성능 개선 도우미의 AWS 서비스에서 수행한 작업을 기록하는 서비스인 AWS CloudTrail에서 실행됩니다. CloudTrail은 성능 개선 도우미에 대한 모든 API 호출을 이벤트로 캡처합니다. 이 캡처에는 Amazon RDS 콘솔과 코드에서 성능 개선 도우미 API 작업으로의 호출이 포함됩니다.

추적을 생성하면 성능 개선 도우미를 포함해 CloudTrail 이벤트를 Amazon S3 버킷으로 지속적으로 배포할 수 있습니다. 트레일을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 데이터를 사용하여 특정 정보를 확인할 수 있습니다. 이 정보에는 성능 개선 도우미에 대한 요청, 요청한 IP 주소, 요청 주체 및 요청한 시간이 포함됩니다. 또한 추가 세부 정보도 포함되어 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

CloudTrail의 성능 개선 도우미 정보 작업

CloudTrail은 계정 생성 시 AWS 계정에서 활성화됩니다. 성능 개선 도우미에서 활동이 수행되면 해당 활동은 [이벤트 기록(Event history)]에서 CloudTrail 콘솔의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서에서 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

성능 개선 도우미의 이벤트를 포함해 AWS 계정의 이벤트를 계속 기록하려면 trail을 생선합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 AWS 지역에 추적이 적용됩니다. 추적은 AWS 파티션에 있는 모든 AWS 리전의 이벤트를 로깅하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가 분석 및 작업하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서에서 다음 주제를 참조하세요.

- 트레일 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에서 Amazon SNS 알림 구성
- 여러 리전으로부터 CloudTrail 로그 파일 받기 및 여러 계정으로부터 CloudTrail 로그 파일 받기

모든 성능 개선 도우미 작업은 CloudTrail이 기록하며 성능 개선 도우미 API 참조에 문서화됩니다. 예를 들어, `DescribeDimensionKeys` 및 `GetResourceMetrics` 작업에 대한 호출은 CloudTrail 로그 파일의 항목을 생성합니다.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. ID 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 IAM 사용자 보안 인증 정보로 했는지 여부.
- 역할 또는 페더레이션 사용자의 임시 보안 인증을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지.

자세한 설명은 CloudTrail userIdentity 요소를 참조하세요.

성능 개선 도우미 로그 파일 항목

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 원본의 단일 요청을 나타냅니다. 각 이벤트에는 요청된 작업에 대한 정보, 작업 날짜 및 시간, 요청 파라미터 등이 포함됩니다. CloudTrail 로그 파일은 퍼블릭 API 호출의 주문 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음 예는 `GetResourceMetrics`작업을 보여주는 CloudTrail 로그 항목입니다.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
         "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
    "eventTime": "2019-12-18T19:28:46Z",
    "eventSource": "pi.amazonaws.com",
    "eventName": "GetResourceMetrics",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "72.21.198.67",
    "userAgent": "aws-cli/1.16.240 Python/3.7.4 Darwin/18.7.0 botocore/1.12.230",
    "requestParameters": {
        "identifier": "db-YTDU5J5V66X7CXSCVDFD2V3SZM",
        "metricQueries": [
            {
                "metric": "os.cpuUtilization.user.avg"
            },
```

```
            {
                "metric": "os.cpuUtilization.idle.avg"
            }
        ],
        "startTime": "Dec 18, 2019 5:28:46 PM",
        "periodInSeconds": 60,
        "endTime": "Dec 18, 2019 7:28:46 PM",
        "serviceType": "RDS"
    },
    "responseElements": null,
    "requestID": "9ffbe15c-96b5-4fe6-bed9-9fccff1a0525",
    "eventID": "08908de0-2431-4e2e-ba7b-f5424f908433",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

성능 개선 도우미 API 및 인터페이스 VPC 엔드포인트(AWS PrivateLink)

AWS PrivateLink를 사용하여 VPC와 Amazon RDS 성능 개선 도우미 사이에 프라이빗 연결을 생성할 수 있습니다. 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 AWS Direct Connect 연결을 사용하지 않고 VPC에 있는 것처럼 성능 개선 도우미에 액세스할 수 있습니다. VPC의 인스턴스에서 성능 개선 도우미에 액세스하는 데 퍼블릭 IP 주소가 필요하지 않습니다.

AWS PrivateLink에서 제공되는 인터페이스 엔드포인트를 생성하여 이 프라이빗 연결을 설정합니다. 인터페이스 엔드포인트에 대해 사용 설정하는 각 서브넷에서 엔드포인트 네트워크 인터페이스를 생성합니다. 이는 성능 개선 도우미로 향하는 트래픽의 진입점 역할을 하는 요청자 관리형 네트워크 인터페이스입니다.

자세한 내용은 AWS PrivateLink 가이드의 AWS PrivateLink를 통해 AWS 서비스에 액세스를 참조하세요.

성능 개선 도우미에 대한 고려사항

성능 개선 도우미에 대한 인터페이스 엔드포인트를 설정하려면 먼저 AWS PrivateLink 안내서의 고려사항을 검토합니다.

성능 개선 도우미는 인터페이스 엔드포인트를 통해 모든 API 작업에 대한 직접 호출을 지원합니다.

기본적으로 인터페이스 엔드포인트를 통해 성능 개선 도우미에 대한 전체 액세스가 허용됩니다. 보안 그룹을 엔드포인트 네트워크 인터페이스와 연결하면 인터페이스 엔드포인트를 통해 성능 개선 도우미로 향하는 트래픽을 제어할 수 있습니다.

가용성

성능 개선 도우미 API는 현재 성능 개선 도우미를 지원하는 AWS 리전의 VPC 엔드포인트를 지원합니다. 성능 개선 도우미의 가용성에 대한 자세한 내용은 Amazon RDS에서 성능 개선 도우미를 지원하는 리전 및 DB 엔진 단원을 참조하세요.

성능 개선 도우미에 대한 인터페이스 엔드포인트 생성

Amazon VPC 콘솔 또는 AWS Command Line Interface(AWS CLI)를 사용하여 성능 개선 도우미에 대한 인터페이스 엔드포인트를 생성할 수 있습니다. 자세한 내용은 AWS PrivateLink 설명서의 인터페이스 엔드포인트 생성을 참조하십시오.

다음 서비스 이름을 사용하여 성능 개선 도우미에 대한 인터페이스 엔드포인트를 생성합니다.

인터페이스 엔드포인트에 프라이빗 DNS를 활성화하는 경우, 리전에 대한 기본 DNS 이름을 사용하여 성능 개선 도우미에 API 요청을 전송할 수 있습니다. 예: `pi.us-east-1.amazonaws.com`.

성능 개선 도우미 API에 대한 VPC 엔드포인트 정책 생성

엔드포인트 정책은 인터페이스 엔드포인트에 연결할 수 있는 IAM 리소스입니다. 기본 엔드포인트 정책을 사용하면 인터페이스 엔드포인트를 통해 성능 개선 도우미에 대한 전체 액세스가 허용됩니다. VPC에서 성능 개선 도우미에 허용되는 액세스를 제어하려면 사용자 지정 엔드포인트 정책을 인터페이스 엔드포인트에 연결합니다.

엔드포인트 정책은 다음 정보를 지정합니다.

- 작업을 수행할 수 있는 보안 주체 (AWS 계정, IAM 사용자, IAM 역할)
- 수행할 수 있는 작업.
- 작업을 수행할 수 있는 리소스.

자세한 내용은 AWS PrivateLink 가이드의 엔드포인트 정책을 사용하여 서비스에 대한 액세스 제어를 참조하세요.

예제: 성능 개선 도우미 작업에 대한 VPC 엔드포인트 정책

다음은 사용자 지정 엔드포인트 정책의 예입니다. 이 정책을 인터페이스 엔드포인트에 연결하면 나열된 성능 개선 도우미 작업에 대한 액세스 권한이 모든 리소스의 모든 보안 주체에 부여됩니다.

```
{
```

```
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "rds:CreatePerformanceAnalysisReport",
            "rds:DeletePerformanceAnalysisReport",
            "rds:GetPerformanceAnalysisReport"
         ],
         "Resource":"*"
      }
   ]
}
```

예제: 지정된 AWS 계정의 모든 액세스를 거부하는 VPC 엔드포인트 정책

다음 VPC 엔드포인트 정책은 AWS 계정 123456789012가 엔드포인트를 사용하는 리소스에 대한 모든 액세스를 거부합니다. 이 정책은 다른 계정의 모든 작업을 허용합니다.

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": "*",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": { "AWS": [ "123456789012" ] }
     }
   ]
}
```

성능 개선 도우미에 IP 주소 지정

IP 주소가 있으면 VPC 내의 리소스가 서로 통신하고, 인터넷을 통해 다른 리소스와 통신할 수 있습니다. 성능 개선 도우미는 IPv4와 IPv6 주소 지정 프로토콜을 모두 지원합니다. 기본적으로 성능 개선 도우미와 Amazon VPC는 IPv4 주소 지정 프로토콜을 사용합니다. 이 동작은 끌 수 없습니다. VPC를 생성할 때 VPC에 IPv4 CIDR 블록(프라이빗 IPv4 주소)를 지정해야 합니다.

선택적으로 IPv6 CIDR 블록을 VPC와 서브넷에 할당하고 그 블록에 속한 IPv6 주소를 서브넷의 RDS 리소스에 할당할 수도 있습니다. IPv6 프로토콜을 지원하면 지원되는 IP 주소 수가 늘어납니다. IPv6 프로토콜을 사용하면 향후에 인터넷이 성장할 때를 대비해 사용 가능한 주소를 충분히 확보할 수 있습니다. 신규 및 기존 RDS 리소스는 VPC 내에서 IPv4 및 IPv6 주소를 사용할 수 있습니다. 애플리케이션의 다른 부분에서 사용되는 두 프로토콜 간의 네트워크 트래픽을 구성, 보안 및 변환하면 운영 오버헤드가 발생할 수 있습니다. Amazon RDS 리소스의 IPv6 프로토콜을 표준화하여 네트워크 구성을 간소화할 수 있습니다. 서비스 엔드포인트 및 할당량에 대한 자세한 내용은 Amazon Relational Database Service 엔드포인트 및 할당량을 참조하세요.

Amazon RDS IP 주소 지정에 대한 자세한 내용은 Amazon RDS IP 주소 지정을 참조하세요.

Amazon DevOps Guru for Amazon RDS로 성능 이상 분석

Amazon DevOps Guru는 개발자와 운영자가 애플리케이션의 성능과 가용성을 개선하는 데 도움이 되는 완전관리형 운영 서비스입니다. DevOps Guru는 운영 문제 식별과 관련된 작업을 오프로드하므로 애플리케이션을 개선하기 위한 권장 사항을 신속하게 구현할 수 있습니다. 자세한 내용은 Amazon DevOps Guru 사용 설명서의 Amazon DevOps Guru란 무엇인가요?를 참조하세요.

DevOps Guru는 모든 Amazon RDS DB 엔진의 기존 운영 문제를 감지, 분석 및 권장 사항을 제시합니다. DevOps Guru for RDS는 RDS for PostgreSQL 데이터베이스의 성능 개선 도우미 지표에 기계 학습을 적용하여 이 기능을 확장합니다. 이러한 모니터링 기능을 통해 DevOps Guru for RDS는 성능 병목 현상을 감지 및 진단하고 특정 시정 조치를 권장할 수 있습니다. DevOps Guru for RDS는 RDS for PostgreSQL 데이터베이스에서 문제가 발생하기 전에 문제를 감지할 수 있습니다.

이제 RDS 콘솔에서 이러한 권장 사항을 확인할 수 있습니다. 자세한 내용은 Amazon RDS의 권장 사항 단원을 참조하십시오.

다음 동영상은 RDS용 DevOps 전문가 개요입니다.

이 주제에 관해 자세히 알아보려면 내부에 있는 RDS용 Amazon DevOps Guru.를 참조하세요.

주제

- DevOps Guru for RDS의 이점
- DevOps Guru for RDS 작동 방식
- DevOps Guru for RDS 설정

DevOps Guru for RDS의 이점

RDS for PostgreSQL 데이터베이스를 담당하는 경우 해당 데이터베이스에 영향을 미치는 이벤트 또는 회귀가 발생하고 있는 것을 알지 못할 수 있습니다. 이 문제에 대해 알아볼 때, 문제가 발생하는 이유나 어떻게 대응해야 할 지 모를 수도 있습니다. 데이터베이스 관리자(DBA)에게 도움을 요청하거나 타사 도구에 의존하는 대신 DevOps Guru for RDS의 권장 사항을 따를 수 있습니다.

DevOps Guru for RDS의 세부 분석을 통해 다음과 같은 이점을 얻을 수 있습니다.

빠른 진단

RDS용 DevOps Guru는 데이터베이스 원격 분석을 지속적으로 모니터링하고 분석합니다. 성능 개선 도우미, 향상된 모니터링 및 Amazon CloudWatch는 데이터베이스 인스턴스에 대한 원격 분석

데이터를 수집합니다. DevOps Guru for RDS는 통계 및 기계 학습 기술을 사용하여 이 데이터를 마이닝하고 이상을 감지합니다. 원격 측정 데이터에 대한 자세한 내용은 Amazon RDS 사용 설명서에서성능 개선 도우미를 통한 Amazon RDS 모니터링 및 Enhanced Monitoring을 사용하여 OS 지표 모니터링을 참조하세요.

빠른 해결

각 이상 현상은 성능 문제를 식별하고 조사 또는 수정 작업 방법을 제안합니다. 예를 들어 RDS용 DevOps Guru는 특정 대기 이벤트를 조사하도록 권장할 수 있습니다. 또는 애플리케이션 풀 설정을 조정하여 데이터베이스 연결 수를 제한하도록 권장할 수도 있습니다. 이러한 권장 사항을 기반으로 수동으로 문제를 해결하는 것보다 성능 문제를 더 빨리 해결할 수 있습니다.

사전 예방 인사이트

DevOps Guru for RDS는 리소스의 지표를 사용하여 문제가 더 커지기 전에 잠재적으로 문제가 될 수 있는 동작을 탐지합니다. 예를 들어 데이터베이스에서 점점 더 많은 수의 온디스크 임시 테이블을 사용하여 성능에 영향을 미치기 시작할 가능성이 있는 경우, 이를 탐지할 수 있습니다. 그런 다음 DevOps Guru는 문제가 더 커지기 전에 문제를 해결하는 데 도움이 되는 권장 사항을 제공합니다.

Amazon 엔지니어의 깊이 있는 지식과 기계 학습

성능 문제를 감지하고 병목 현상을 해결하기 위해 DevOps Guru for RDS는 기계 학습(ML)과 고급 수학 공식을 사용합니다. Amazon 데이터베이스 엔지니어들은 수년간 수십만 개의 데이터베이스를 관리한 결과를 캡슐화하는 DevOps Guru for RDS의 개발에 기여했습니다. 이러한 집단 지식을 바탕으로 DevOps Guru for RDS를 통해 모범 사례를 가르칠 수 있습니다.

DevOps Guru for RDS 작동 방식

DevOps Guru for RDS는 Amazon RDS 성능 개선 도우미로부터 RDS for PostgreSQL 데이터베이스에 대한 데이터를 수집합니다. 가장 중요한 지표는 DBLoad입니다. DevOps Guru for RDS는 성능 개선 도우미 지표를 사용하고, 기계 학습을 통해 분석하며, 대시보드에 인사이트를 게시합니다.

인사이트는 DevOps Guru가 탐지한 관련 이상의 모음입니다.

DevOps Guru for RDS에서, 이상은 RDS for PostgreSQL 데이터베이스의 정상적인 성능으로 간주되는 것과 다른 패턴입니다.

사전 예방 인사이트

사전 예방 인사이트를 통해 문제가 발생하기 전에 문제를 일으킬 수 있는 행동을 파악할 수 있습니다. 여기에는 RDS for PostgreSQL 데이터베이스에서 더 큰 문제가 발생하기 전에 문제를 해결하는

데 도움이 되는 권장 사항 및 관련 지표가 포함된 이상 항목이 포함되어 있습니다. 이러한 인사이트는 DevOps Guru 대시보드에 게시됩니다.

예를 들어 DevOps Guru는 RDS for PostgreSQL 데이터베이스에서 많은 온디스크 임시 테이블을 생성하고 있음을 감지할 수 있습니다. 이러한 추세를 해결하지 않으면 성능 문제가 발생할 수 있습니다. 각 사전 예방 인사이트에는 수정 조치에 대한 권장 사항 및 Amazon DevOps Guru의 사전 예방 인사이트를 활용하여 RDS for PostgreSQL 튜닝의 관련 주제에 대한 링크가 포함되어 있습니다. 자세한 내용은 Amazon DevOps Guru 사용 설명서의 DevOps Guru의 인사이트 활용을 참조하세요.

사후 대응 인사이트

사후 대응 인사이트는 비정상적인 동작이 발생하는 즉시 이를 식별합니다. DevOps Guru for RDS가 RDS for PostgreSQL DB 인스턴스에서 성능 문제를 발견하면 DevOps Guru 대시보드에 사후 대응 인사이트를 게시합니다. 자세한 내용은 Amazon DevOps Guru 사용 설명서의 DevOps Guru의 인사이트 활용을 참조하세요.

일반적인 이상

캐주얼 이상 항목은 사후 대응 인사이트 내에서 최상위 이상 항목입니다. 데이터베이스 로드(DB 로드)는 RDS용 DevOps 전문가의 인과 관계적 이상입니다.

이상은 심각도의 높음, 중간, 또는 낮음 수준을 할당하여 성능에 미치는 영향을 측정합니다. 자세한 내용은 Amazon DevOps Guru 사용 설명서의 DevOps Guru for RDS의 주요 개념을 참조하세요.

DevOps Guru가 DB 인스턴스에서 진행 중인 이상 현상을 감지하면, RDS 콘솔의 데이터베이스(Databases) 페이지에 알림이 표시됩니다. 또한 콘솔은 지난 24시간 동안 발생한 이상 현상을 알려줍니다. RDS 콘솔에서 이상 페이지로 이동하려면 경고 메시지에서 링크를 선택합니다. 또한 RDS 콘솔은 RDS for PostgreSQL DB 인스턴스의 페이지에 알림을 표시합니다.

문맥적 이상

컨텍스트 이상 항목은 데이터베이스 로드(DB 로드) 내의 결과로, 사후 대응 인사이트와 관련이 있습니다. 각 문맥적 이상은 조사가 필요한 특정 RDS for PostgreSQL 성능 문제를 설명합니다. 예를 들어 DevOps Guru for RDS는 CPU 용량을 늘리거나 DB 로드에 기여하는 대기 이벤트를 조사할 것을 권장할 수 있습니다.

> **Important**
>
> 프로덕션 인스턴스를 변경하기 전에 테스트 인스턴스에서 변경 사항을 테스트하는 것이 좋습니다. 이러한 방식으로 변경의 영향을 이해하게 됩니다.

자세한 내용은 Amazon DevOps Guru 사용 설명서의 Amazon RDS의 이상 현상 분석을 참조하세요.

DevOps Guru for RDS 설정

DevOps Guru for Amazon RDS가 a RDS for PostgreSQL 데이터베이스에 대한 인사이트를 게시할 수 있도록 하려면 다음 태스크를 완료합니다.

주제

- DevOps Guru for RDS에 사용되는 IAM 액세스 정책 구성
- RDS for PostgreSQL DB 인스턴스의 성능 개선 도우미 활성화
- DevOps Guru 활성화 및 리소스 적용 범위 지정

DevOps Guru for RDS에 사용되는 IAM 액세스 정책 구성

RDS 콘솔에서 DevOps Guru의 알림을 보려면 AWS Identity and Access Management(IAM) 사용자 또는 역할에 다음 정책 중 하나가 있어야 합니다.

- AWS 관리형 정책 `AmazonDevOpsGuruConsoleFullAccess`
- AWS 관리형 정책 `AmazonDevOpsGuruConsoleReadOnlyAccess` 및 다음 정책 중 하나:
  - AWS 관리형 정책 `AmazonRDSFullAccess`
  - `pi:GetResourceMetrics` 및 `pi:DescribeDimensionKeys`를 포함하는 고객 관리형 정책

자세한 내용은 Performance Insights에 대한 액세스 정책 구성 단원을 참조하십시오.

RDS for PostgreSQL DB 인스턴스의 성능 개선 도우미 활성화

DevOps Guru for RDS는 데이터를 위해 성능 개선 도우미를 사용합니다. 성능 개선 도우미가 없으면 DevOps Guru는 이상을 게시하지만 자세한 분석 및 권장 사항은 포함하지 않습니다.

RDS for PostgreSQL DB 인스턴스를 생성하거나 수정할 때 성능 개선 도우미를 활성화할 수 있습니다. 자세한 내용은 Amazon RDS의 성능 개선 도우미 설정 및 해제 단원을 참조하십시오.

DevOps Guru 활성화 및 리소스 적용 범위 지정

다음 방법 중 하나로 RDS for PostgreSQL 데이터베이스를 모니터링하도록 DevOps Guru를 활성화할 수 있습니다.

주제

- RDS 콘솔에서 DevOps Guru 활성화
- DevOps Guru 콘솔에 RDS for PostgreSQL 리소스 추가
- AWS CloudFormation을 사용하여 RDS for PostgreSQL 리소스 추가

RDS 콘솔에서 DevOps Guru 활성화

Amazon RDS 콘솔에서 여러 경로를 사용하여 DevOps Guru를 활성화할 수 있습니다.

주제

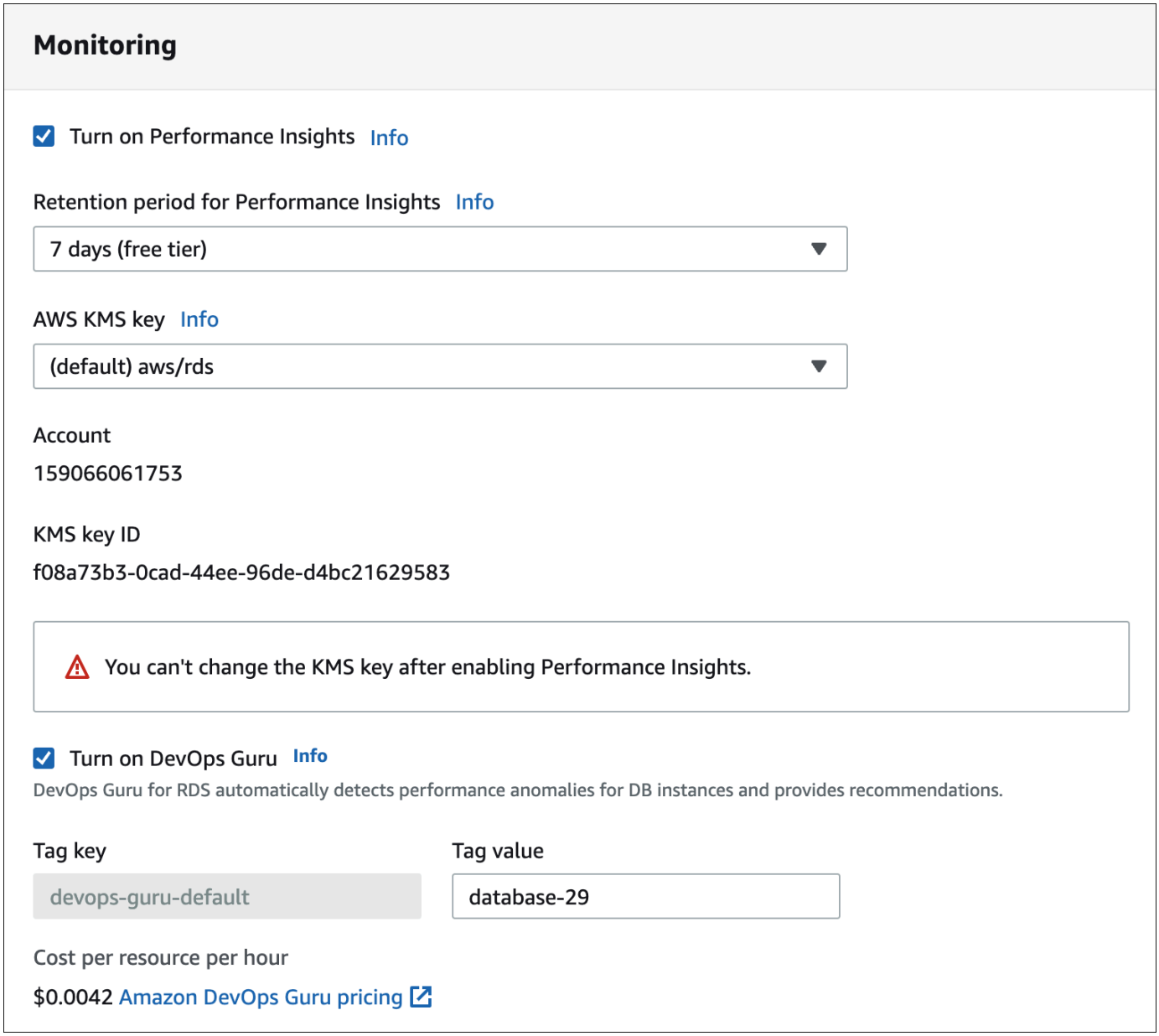
- RDS for PostgreSQL 데이터베이스를 생성할 때 DevOps Guru 활성화
- 알림 배너에서 DevOps Guru 활성화
- DevOps Guru를 활성화할 때 권한 오류에 응답

RDS for PostgreSQL 데이터베이스를 생성할 때 DevOps Guru 활성화

생성 워크플로에는 데이터베이스에 대한 DevOps Guru 적용 범위를 활성화하는 설정이 포함되어 있습니다. 이 설정은 프로덕션(Production) 템플릿을 선택할 때 기본적으로 지정됩니다.

RDS for PostgreSQL 데이터베이스를 생성할 때 DevOps Guru를 활성화하려면

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 모니터링 설정을 선택하는 단계를 제외하고 DB 인스턴스 생성에 나와 있는 단계를 최대한 수행합니다.
3. 모니터링(Monitoring)에서 성능 개선 도우미 활성화(Turn on Performance Insights)를 선택합니다. DevOps Guru for RDS가 성능 이상에 대한 자세한 분석을 제공하려면 성능 개선 도우미를 활성화해야 합니다.
4. DevOps Guru 활성화(Turn on DevOps Guru)를 선택합니다.



5. DevOps Guru가 모니터링할 수 있도록 데이터베이스에 대한 태그를 생성합니다. 다음을 따릅니다.

   - 태그 키(Tag key) 텍스트 필드에서 **`Devops-Guru-`**로 시작하는 이름을 입력합니다.
   - 태그 값(Tag value) 텍스트 필드에서 원하는 값을 입력합니다. 예를 들어, RDS for PostgreSQL 데이터베이스의 이름에 **`rds-database-1`**을 입력하는 경우 **`rds-database-1`**을 태그 값으로 입력할 수도 있습니다.

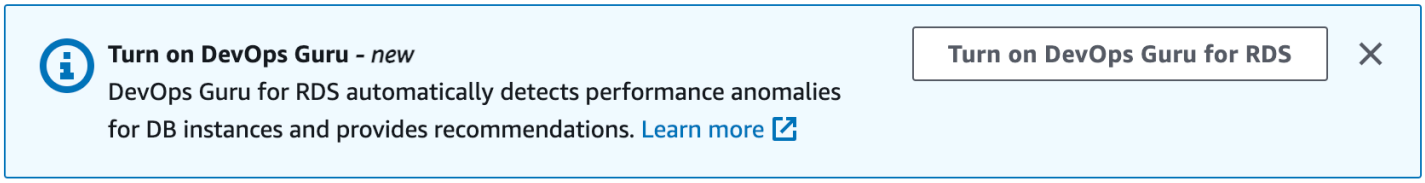
   태그에 대한 자세한 내용은 Amazon DevOps Guru 사용 설명서의 '태그를 사용하여 DevOps Guru 애플리케이션에서 리소스 식별'을 참조하세요.

6. DB 인스턴스 생성에 설명된 나머지 단계를 완료합니다.

알림 배너에서 DevOps Guru 활성화

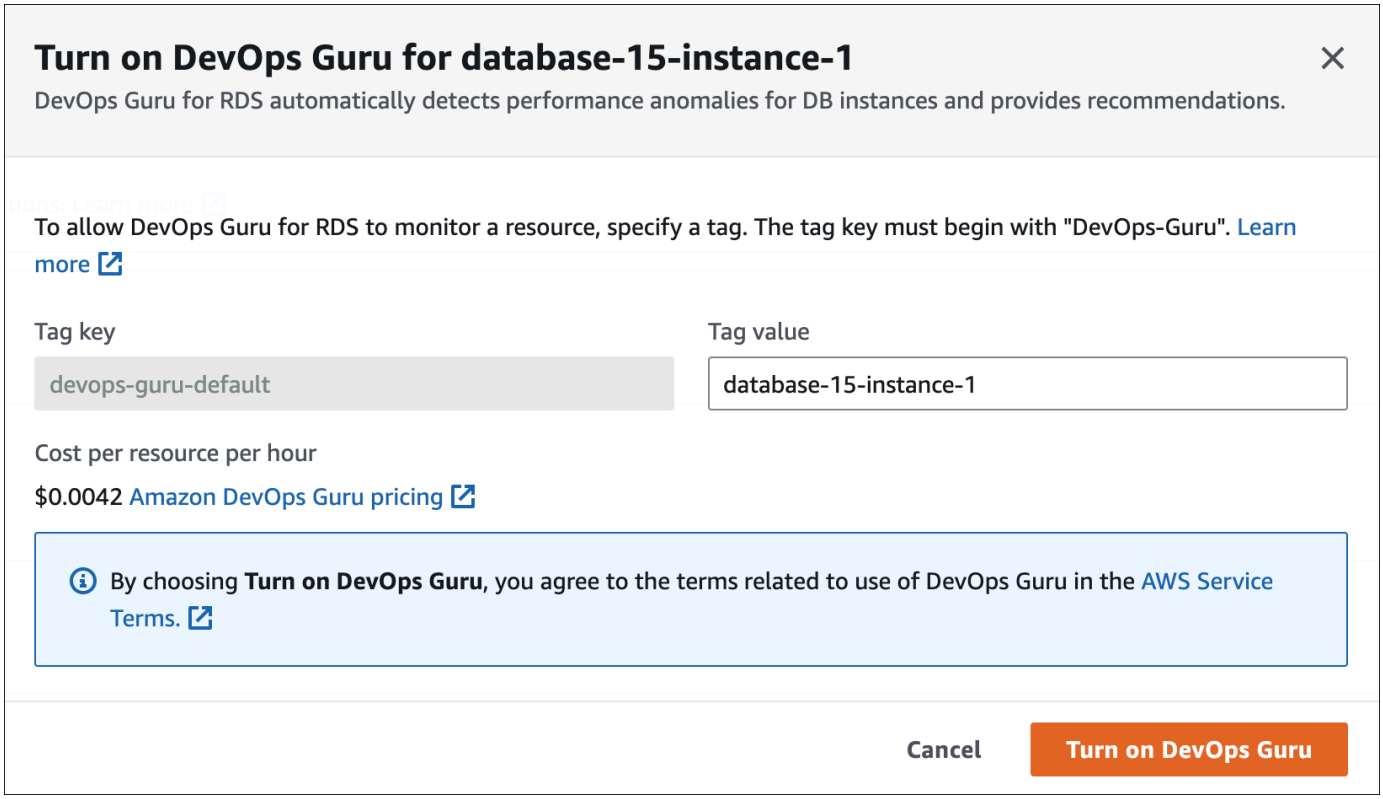
리소스가 DevOps Guru에 포함되지 않는 경우 Amazon RDS에서 다음 위치에 배너를 사용하여 알립니다.

- DB 클러스터 인스턴스의 모니터링(Monitoring) 탭
- 성능 개선 도우미 대시보드



RDS for PostgreSQL 데이터베이스에 대해 DevOps Guru를 활성화하려면

1. 배너에서 DevOps Guru for RDS 활성화를 선택합니다.
2. 태그 키 이름 및 값을 입력합니다. 태그에 대한 자세한 내용은 Amazon DevOps Guru 사용 설명서의 '태그를 사용하여 DevOps Guru 애플리케이션에서 리소스 식별'을 참조하세요.



3. DevOps Guru 활성화를 선택합니다.

DevOps Guru를 활성화할 때 권한 오류에 응답

데이터베이스를 생성할 때 RDS 콘솔에서 DevOps Guru를 활성화하면 RDS에 권한 누락에 대해 다음 배너가 표시될 수 있습니다.

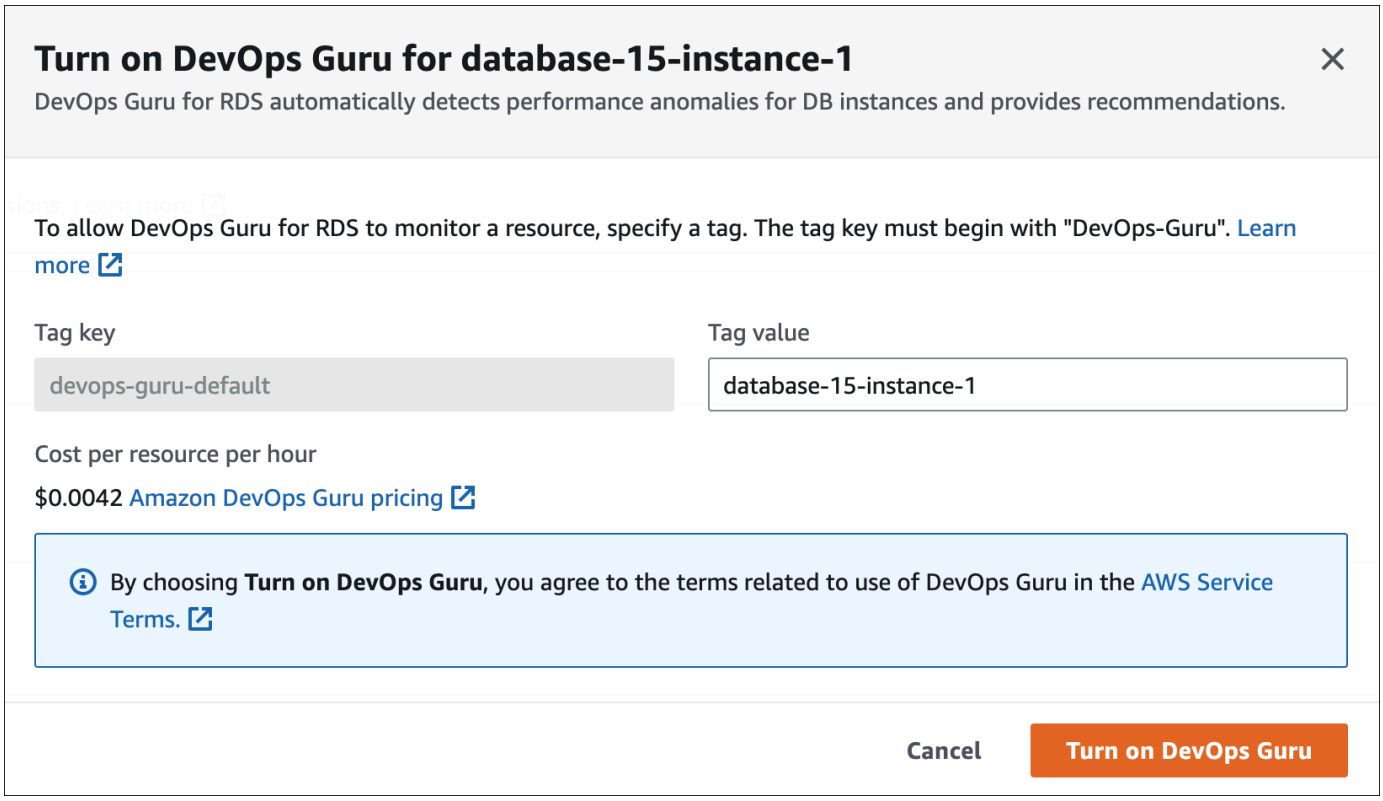


권한 오류에 응답하려면

1. IAM 사용자 또는 역할에 사용자 관리형 역할 `AmazonDevOpsGuruConsoleFullAccess`를 부여합니다. 자세한 내용은 DevOps Guru for RDS에 사용되는 IAM 액세스 정책 구성 단원을 참조하십시오.
2. RDS 콘솔을 엽니다.
3. 탐색 창에서 성능 개선 도우미를 선택합니다.
4. 방금 생성한 클러스터에서 DB 인스턴스를 선택합니다.
5. 스위치를 선택해 DevOps Guru for RDS 기능을 켭니다.

DevOps Guru for RDS

6. 태그 값을 선택합니다. 자세한 내용은 Amazon DevOps Guru 사용 설명서의 '태그를 사용하여 DevOps Guru 애플리케이션에서 리소스 식별'을 참조하세요.



7. DevOps Guru 활성화를 선택합니다.

DevOps Guru 콘솔에 RDS for PostgreSQL 리소스 추가

DevOps Guru 콘솔에서 DevOps Guru 리소스 적용 범위를 지정할 수 있습니다. Amazon DevOps Guru 사용 설명서의 DevOps Guru 리소스 적용 범위 지정에 나와 있는 단계를 따릅니다. 분석된 리소스를 편집할 때 다음 옵션 중 하나를 선택합니다.

- 모든 계정 리소스를 선택하여 AWS 계정 및 리전에서 RDS for PostgreSQL 데이터베이스를 포함하여 지원되는 모든 리소스를 분석합니다.
- CloudFormation 스택을 선택하여 지정한 스택에 있는 RDS for PostgreSQL 데이터베이스를 분석합니다. 자세한 내용은 Amazon DevOps Guru 사용 설명서의 AWS CloudFormation 스택을 사용하여 DevOps Guru 애플리케이션에서 리소스 식별을 참조하세요.
- 태그를 선택하여 태그를 지정한 RDS for PostgreSQL 데이터베이스를 분석합니다. 자세한 내용은 Amazon DevOps Guru 사용 설명서의 태그를 사용하여 DevOps Guru 애플리케이션에서 리소스 식별을 참조하세요.

자세한 내용은 Amazon DevOps Guru 사용 설명서의 DevOps Guru 활성화를 참조하세요.

AWS CloudFormation을 사용하여 RDS for PostgreSQL 리소스 추가

태그를 사용하여 RDS for PostgreSQL 리소스 적용 범위를 CloudFormation 템플릿에 추가할 수 있습니다. 다음 절차에서는 RDS for PostgreSQL DB 인스턴스와 DevOps Guru 스택 모두에 대한 CloudFormation 템플릿이 있다고 가정합니다.

CloudFormation 태그를 사용하여 RDS for PostgreSQL DB 인스턴스를 지정하는 방법

1. DB 인스턴스용 CloudFormation 템플릿에서 키/값 쌍을 사용하여 태그를 정의합니다.

   다음 예제에서는 RDS for PostgreSQL DB 인스턴스용 `Devops-guru-cfn-default`에 `my-db-instance1` 값을 할당합니다.

   ```
   MyDBInstance1:
     Type: "AWS::RDS::DBInstance"
     Properties:
       DBInstanceIdentifier: my-db-instance1
       Tags:
         - Key: Devops-guru-cfn-default
           Value: devopsguru-my-db-instance1
   ```

2. DevOps Guru 스택용 CloudFormation 템플릿에서 리소스 컬렉션 필터에 동일한 태그를 지정합니다.

   다음 예제에서는 `my-db-instance1` 태그 값을 사용하여 리소스에 대한 적용 범위를 제공하도록 DevOps Guru를 구성합니다.

   ```
   DevOpsGuruResourceCollection:
     Type: AWS::DevOpsGuru::ResourceCollection
     Properties:
       ResourceCollectionFilter:
         Tags:
           - AppBoundaryKey: "Devops-guru-cfn-default"
             TagValues:
             - "devopsguru-my-db-instance1"
   ```

   다음 예제는 애플리케이션 `Devops-guru-cfn-default` 경계 내의 모든 리소스에 대한 적용 범위를 제공합니다.

   ```
   DevOpsGuruResourceCollection:
   ```

```
  Type: AWS::DevOpsGuru::ResourceCollection
  Properties:
    ResourceCollectionFilter:
      Tags:
        - AppBoundaryKey: "Devops-guru-cfn-default"
          TagValues:
          - "*"
```

자세한 내용은 AWS CloudFormation 사용 설명서의 AWS::DevOpsGuru::ResourceCollection과 AWS::RDS::DBInstance를 참조하세요.

Enhanced Monitoring을 사용하여 OS 지표 모니터링

Enhanced Monitoring을 통해 DB 인스턴스의 운영 체제를 실시간으로 모니터링할 수 있습니다. 다른 프로세스 또는 스레드에서 CPU를 사용하는 방법을 확인하려면 Enhanced Monitoring 지표가 유용합니다.

주제

- Enhanced Monitoring 개요
- Enhanced Monitoring 설정 및 활성화
- RDS 콘솔에서 OS 지표 보기
- CloudWatch Logs을 사용하여 OS 지표 보기

Enhanced Monitoring 개요

Amazon RDS는 DB 인스턴스가 실행되는 운영 체제(OS)에 대한 측정치를 실시간으로 제공합니다. 콘솔에서 RDS DB 인스턴스에 대한 모든 시스템 지표 및 프로세스 정보를 볼 수 있습니다. 각 인스턴스에 대해 모니터링할 지표를 관리하고 요구 사항에 따라 대시보드를 사용자 지정할 수 있습니다. 향상된 모니터링 지표 설명은 향상된 모니터링의 OS 지표 섹션을 참조하세요.

RDS는 지표를 확장 모니터링에서 Amazon CloudWatch Logs 계정으로 전달합니다. CloudWatch Logs에서 CloudWatch에서 지표 필터를 생성하고 CloudWatch 대시보드에 그래프를 표시할 수 있습니다. 또한 선택한 모니터링 시스템에서 CloudWatch Logs의 Enhanced Monitoring JSON 출력을 사용할 수 있습니다. 자세한 내용은 Amazon RDS FAQ의 확장 모니터링을 참조하세요.

주제

- Enhanced Monitoring 가용성
- CloudWatch 지표와 Enhanced Monitoring 지표의 차이점
- Enhanced Monitoring 지표 보존
- Enhanced Monitoring 비용

Enhanced Monitoring 가용성

Enhanced Monitoring은 다음 데이터베이스 엔진에 사용할 수 있습니다.

- Db2
- MariaDB

- Microsoft SQL Server
- MySQL
- Oracle
- PostgreSQL

확장 모니터링은 db.m1.small 인스턴스 클래스를 제외한 모든 DB 인스턴스 클래스에서 사용할 수 있습니다.

CloudWatch 지표와 Enhanced Monitoring 지표의 차이점

하이퍼바이저는 VM(가상 머신)을 만들고 실행합니다. 하이퍼바이저를 사용하는 인스턴스는 메모리와 CPU를 가상으로 공유하여 여러 게스트 VM을 지원할 수 있습니다. CloudWatch는 DB 인스턴스의 하이퍼바이저에서 CPU 사용률에 대한 지표를 수집합니다. 반면, Enhanced Monitoring은 DB 인스턴스의 에이전트에서 지표를 수집합니다.

하이퍼바이저 계층에서는 소량의 작업만 수행하므로 CloudWatch와 Enhanced Monitoring 간의 차이점을 확인할 수 있습니다. DB 인스턴스가 더 작은 인스턴스 클래스를 사용하는 경우 차이가 커질 수 있습니다. 이 시나리오에서는 단일 물리적 인스턴스의 하이퍼바이저 계층에서 더 많은 VM(가상 머신)을 관리할 수 있습니다.

향상된 모니터링 지표 설명은 향상된 모니터링의 OS 지표 섹션을 참조하세요. CloudWatch 지표에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하세요.

Enhanced Monitoring 지표 보존

기본적으로 Enhanced Monitoring 지표는 CloudWatch Logs에서 30일간 저장됩니다. 이 보존 기간은 일반적인 CloudWatch 지표와 다릅니다.

지표가 CloudWatch Logs에 저장되는 시간을 수정하려면 CloudWatch 콘솔에서 `RDSOSMetrics` 로그 그룹의 보존을 변경하십시오. 자세한 내용은 Amazon CloudWatch Logs User Guide의 CloudWatch에서 로그 데이터 보존 기간을 변경을 참조하십시오.

Enhanced Monitoring 비용

Enhanced Monitoring 지표는 CloudWatch 지표가 아닌 CloudWatch Logs에 저장됩니다. 확장 모니터링 비용은 다음 두 가지 요인에 따라 달라집니다.

- Amazon CloudWatch Logs가 제공하는 프리 티어를 초과하는 경우에만 확장 모니터링에 대해서만 비용이 청구됩니다. 요금은 CloudWatch Logs 데이터 전송 및 스토리지 요금을 기준으로 합니다.

- RDS 인스턴스에 대해 전송되는 정보의 양은 확장 모니터링 기능에 대해 정의된 세분성에 직접적으로 비례합니다. 모니터링 간격이 작을수록 OS 측정치가 더 자주 보고되고 모니터링 비용이 증가합니다. 비용을 관리하려면 계정의 여러 인스턴스에 대해 서로 다른 세부 단위를 설정합니다.
- Enhanced Monitoring 사용 비용은 Enhanced Monitoring을 활성화한 각 DB 인스턴스에 대해 적용됩니다. 모니터링하는 DB 인스턴스의 수가 많을수록 더 많은 비용이 청구됩니다.
- 컴퓨팅 집약적인 워크로드를 지원하는 DB 인스턴스는 많은 OS 프로세스 활동이 보고되고 Enhanced Monitoring에 대한 높은 비용이 청구됩니다.

요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

Enhanced Monitoring 설정 및 활성화

Enhanced Monitoring을 사용하려면 IAM 역할을 생성한 다음 Enhanced Monitoring을 활성화해야 합니다.

주제

- Enhanced Monitoring에 대한 IAM 역할 생성
- 향상된 모니터링 설정 및 해제
- 혼동된 대리자 문제로부터 보호

Enhanced Monitoring에 대한 IAM 역할 생성

Enhanced Monitoring은 사용자를 대신하여 CloudWatch Logs에 OS 측정치 정보를 보낼 수 있는 권한이 필요합니다. AWS Identity and Access Management(IAM) 역할을 사용하여 Enhanced Monitoring에 권한을 부여합니다. 향상된 모니터링을 사용 설정할 때 이 역할을 생성하거나 미리 생성할 수 있습니다.

주제

- Enhanced Monitoring을 활성화할 때 IAM 역할 생성
- Enhanced Monitoring을 활성화하기 전에 IAM 역할 생성

Enhanced Monitoring을 활성화할 때 IAM 역할 생성

RDS 콘솔에서 Enhanced Monitoring을 활성화하면, Amazon RDS가 필요한 IAM 역할을 생성할 수 있습니다. 이 역할의 이름은 `rds-monitoring-role`입니다. RDS는 지정된 DB 인스턴스, 읽기 전용 복제본 또는 다중 AZ DB 클러스터에 이 역할을 사용합니다.

Enhanced Monitoring을 활성화할 때 IAM 역할을 생성하려면

1. 향상된 모니터링 설정 및 해제 단원의 단계를 따르십시오.
2. 역할을 선택하는 단계에서 모니터링 역할(Monitoring Role)을 기본값(Default)으로 설정합니다.

Enhanced Monitoring을 활성화하기 전에 IAM 역할 생성

Enhanced Monitoring을 활성화하기 전에, 필요한 역할을 생성할 수 있습니다. Enhanced Monitoring을 활성화할 때 새 역할의 이름을 지정합니다. AWS CLI 또는 RDS API를 사용하여 Enhanced Monitoring을 활성화할 경우 이 필수 역할을 생성해야 합니다.

확장 모니터링을 활성화하는 사용자는 `PassRole` 권한을 부여받아야 합니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 AWS 서비스에 역할을 전달할 수 있는 권한 부여에 있는 예제 2를 참조하십시오.

Amazon RDS Enhanced Monitoring에 대한 IAM 역할을 생성하려면

1. https://console.aws.amazon.com에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할을 선택합니다.
3. 역할 생성(Create role)을 선택합니다.
4. AWS 서비스(service) 탭을 선택한 다음 서비스 목록에서 RDS를 선택합니다.
5. RDS - 확장 모니터링(RDS - Enhanced Monitoring)과 다음(Next)을 차례로 선택합니다.
6. 권한 정책(Permissions policies)에 AmazonRDSEnhancedMonitoringRole이 표시되었는지 확인하고 다음(Next)을 선택합니다.
7. 역할 이름에 역할의 이름을 입력합니다. 예를 들면 **`emaccess`**를 입력합니다.

   사용자 역할에 대한 신뢰할 수 있는 엔터티는 monitoring.rds.amazonaws.com AWS 서비스입니다.
8. 역할 생성을 선택합니다.

향상된 모니터링 설정 및 해제

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 향상된 모니터링을 관리할 수 있습니다. 각 DB 인스턴스에서 지표 수집에 대해 서로 다른 세부 수준을 설정할 수 있습니다.

콘솔

DB 인스턴스, 다중 AZ DB 클러스터, 또는 읽기 전용 복제본을 생성할 때나 DB 인스턴스 또는 다중 AZ DB 클러스터를 수정할 때 향상된 모니터링을 켤 수 있습니다. 향상된 모니터링을 활성화하기 위해 DB 인스턴스 를 수정하는 경우 DB 인스턴스를 재부팅하지 않아도 변경 내용이 적용됩니다.

데이터베이스 페이지에서 다음 작업 중 하나를 수행할 때 RDS 콘솔에서 향상된 모니터링을 사용 설정할 수 있습니다.

- DB 인스턴스 또는 다중 AZ DB 클러스터 생성 - 데이터베이스 생성(Create database)을 선택합니다.
- 읽기 전용 복제본 생성(Create a read replica) - 작업(Actions)을 선택한 다음 읽기 전용 복제본 생성(Create read replica)을 선택합니다.
- DB 인스턴스 또는 다중 AZ DB 클러스터 수정 – 수정을 선택합니다.

RDS 콘솔에서 향상된 모니터링 설정 또는 해제

1. 추가 구성(Additional configuration)으로 스크롤합니다.
2. 모니터링에서 DB 인스턴스, 클러스터 또는 읽기 전용 복제본에 대해 를 선택합니다. .
3. Amazon RDS에 사용자를 대신하여 Amazon CloudWatch Logs와 통신하도록 허용하기 위해 생성한 IAM 역할에 대한 [Monitoring Role] 속성을 설정하거나, [Default]를 선택하여 RDS에서 `rds-monitoring-role` 역할을 자동으로 생성하도록 합니다.
4. 세부 수준 속성을 DB 인스턴스 또는 읽기 전용 복제본에 대한 지표가 수집되는 시점 간격(초)으로 설정합니다. [Granularity] 속성을 `1`, `5`, `10`, `15`, `30` 또는 `60` 값 중 하나로 설정할 수 있습니다.

   RDS 콘솔을 새로 고치는 최소 간격은 5초입니다. RDS 콘솔에서 단위를 1초로 설정한 경우에도 업데이트된 측정치는 5초마다 표시됩니다. CloudWatch Logs를 사용하여 1초 측정치 업데이트를 검색할 수 있습니다.

AWS CLI

AWS CLI를 사용하여 향상된 모니터링을 활성화하려면 다음 명령에서 `--monitoring-interval` 옵션을 `0` 이외의 값으로 설정하고 `--monitoring-role-arn` 옵션을 Enhanced Monitoring에 대한 IAM 역할 생성에서 생성된 역할로 설정합니다.

- create-db-instance
- create-db-instance-read-replica
- modify-db-instance

- create-db-cluster(다중 AZ DB 클러스터)
- modify-db-cluster(다중 AZ DB 클러스터)

`--monitoring-interval` 옵션은 확장 모니터링 지표가 수집되는 시점 간격(초)을 지정합니다. 이 옵션의 유효한 값은 `0`, `1`, `5`, `10`, `15`, `30` 및 `60`입니다.

AWS CLI를 사용하여 향상된 모니터링을 해제하려면 다음 명령에서 `--monitoring-interval` 옵션을 `0`으로 설정합니다.

Example

다음 예에서는 DB 인스턴스에 대해 향상된 모니터링을 설정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --monitoring-interval 30 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --monitoring-interval 30 ^
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Example

다음 예에서는 DB 인스턴스에 대해 향상된 모니터링을 설정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --monitoring-interval 30 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Windows의 경우:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --monitoring-interval 30 ^
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

RDS API

RDS API를 사용하여 향상된 모니터링을 설정하려면 `MonitoringInterval` 파라미터를 `0` 이외의 값으로 설정하고 `MonitoringRoleArn` 파라미터를 Enhanced Monitoring에 대한 IAM 역할 생성에서 생성된 역할로 설정합니다. 다음 작업에서 이러한 파라미터를 설정합니다.

- CreateDBInstance
- CreateDBInstanceReadReplica
- ModifyDBInstance
- CreateDBCluster(다중 AZ DB 클러스터)
- ModifyDBCluster(다중 AZ DB 클러스터)

`MonitoringInterval` 파라미터는 확장 모니터링 지표가 수집되는 시점 간격(초)을 지정합니다. 유효 값은 `0`, `1`, `5`, `10`, `15`, `30`, `60`입니다.

RDS API를 사용하여 향상된 모니터링을 해제하려면 `MonitoringInterval`을 `0`으로 설정합니다.

혼동된 대리자 문제로부터 보호

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. AWS에서는 교차 서비스 가장으로 인해 혼동된 대리자 문제가 발생할 수 있습니다. 교차 서비스 가장은 한 서비스(호출하는 서비스)가 다른 서비스(호출되는 서비스)를 직접적으로 호출할 때 발생할 수 있습니다. 직접적으로 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이를 방지하기 위해 AWS에서는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 보안 주체를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 제공합니다. 자세한 내용은 혼동된 대리자 문제를 참조하세요.

Amazon RDS가 다른 서비스에 제공할 수 있는 리소스에 대한 권한을 제한하려면 향상된 모니터링 역할에 대한 신뢰 정책에서 `aws:SourceArn` 및 `aws:SourceAccount` 전역 조건 컨텍스트 키를 사용하는 것이 좋습니다. 두 전역 조건 컨텍스트 키를 모두 사용하는 경우 동일한 계정 ID를 사용해야 합니다.

혼동된 대리인 문제로부터 보호하는 가장 효과적인 방법은 리소스의 전체 ARN이 포함된 `aws:SourceArn` 글로벌 조건 컨텍스트 키를 사용하는 것입니다. Amazon RDS의 경우 `aws:SourceArn`을 `arn:aws:rds:Region:my-account-id:db:dbname`으로 설정합니다.

다음 예에서는 혼동된 대리인 문제를 방지하기 위해 신뢰 정책에서 `aws:SourceArn` 및 `aws:SourceAccount` 전역 조건 컨텍스트 키를 사용합니다.

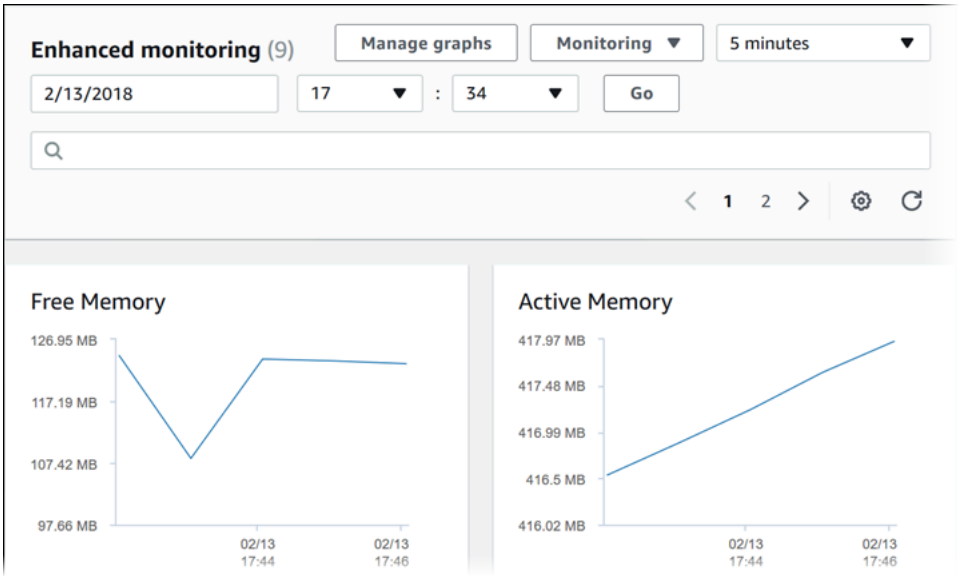
JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "monitoring.rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringLike": {
          "aws:SourceArn": "arn:aws:rds:Region:my-account-id:db:dbname"
        },
        "StringEquals": {
          "aws:SourceAccount": "my-account-id"
        }
      }
    }
  ]
}
```

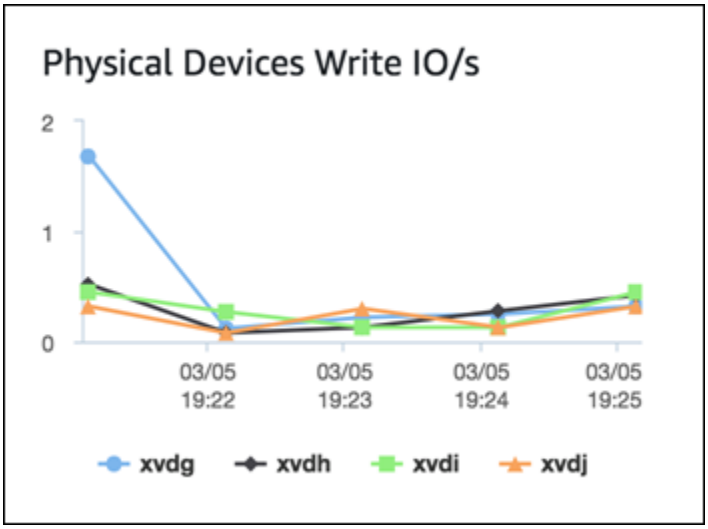
RDS 콘솔에서 OS 지표 보기

모니터링에서 확장 모니터링을 선택하면 RDS 콘솔에서 확장 모니터링이 보고하는 OS 측정치를 볼 수 있습니다.

다음 예에서는 향상된 모니터링 페이지를 보여줍니다. 향상된 모니터링 지표 설명은 향상된 모니터링의 OS 지표 섹션을 참조하세요.

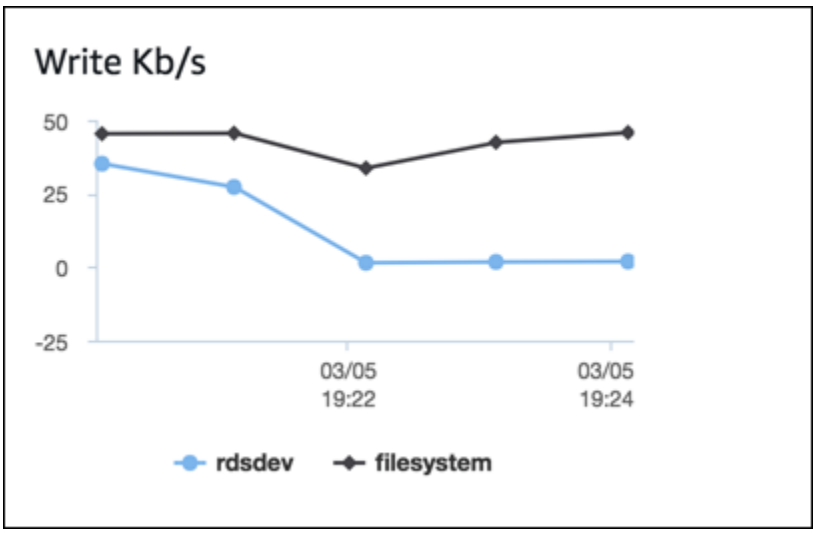


일부 DB 인스턴스에서는 DB 인스턴스의 데이터 스토리지 볼륨에 대해 한 개 이상의 디스크를 사용합니다. 이 DB 인스턴스의 Physical Devices(물리적 디바이스) 그래프에는 각 디스크에 대한 지표가 표시됩니다. 예를 들어 다음 그래프에는 디스크 4개에 대한 지표가 표시되어 있습니다.

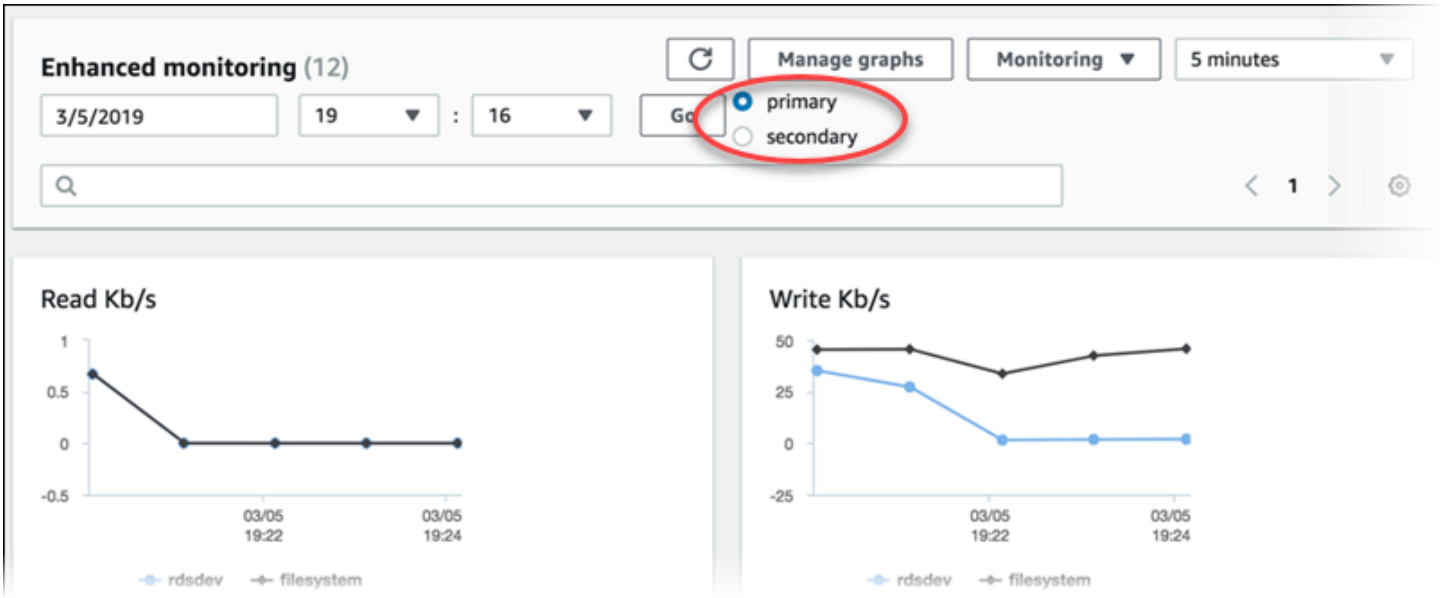


> **Note**
>
> 현재 Physical Devices(물리적 디바이스) 그래프는 Microsoft SQL Server DB 인스턴스에는 제공되지 않습니다.

집계된 디스크 I/O(Disk I/O) 및 파일 시스템(File system) 그래프를 볼 때 rdsdev 디바이스는 모든 데이터베이스 파일 및 로그가 저장되는 `/rdsdbdata` 파일 시스템과 관련이 있습니다. filesystem 디바이스는 운영 체제와 관련된 파일이 저장되는 `/` 파일 시스템(루트라고도 함)과 관련이 있습니다.



DB 인스턴스가 다중 AZ 배포인 경우 기본 DB 인스턴스 및 다중 AZ 스탠바이 복제본에 대한 OS 지표를 볼 수 있습니다. 확장 모니터링 보기에서 primary(기본)을 선택하여 기본 DB 인스턴스에 대한 OS 지표를 확인하거나 secondary(보조)를 선택하여 스탠바이 복제본에 대한 OS 지표를 확인하십시오.

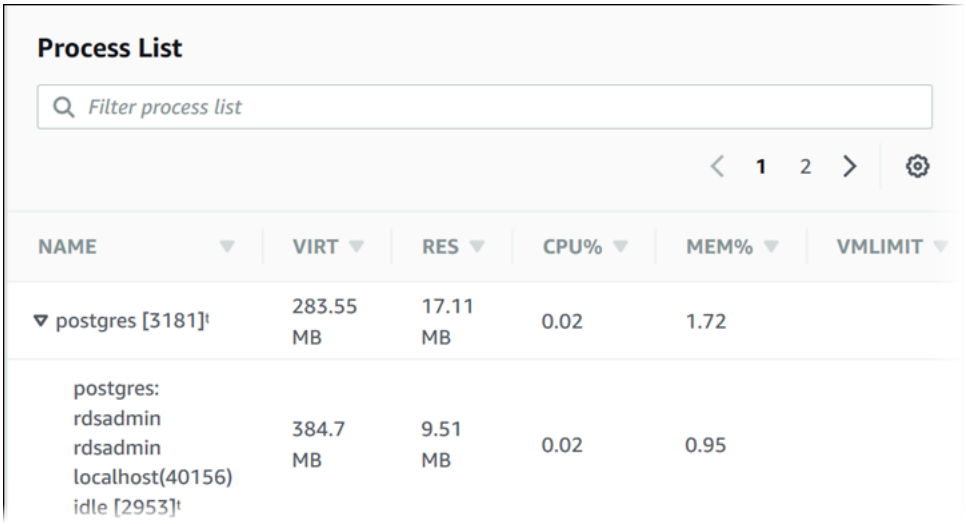


다중 AZ 배포에 대한 자세한 정보는 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 단원을 참조하십시오.

> **Note**
>
> 현재 다중 AZ 대기 복제본에 대한 OS 지표를 보는 기능은 MariaDB 인스턴스에서는 지원되지 않습니다.

DB 인스턴스에서 실행 중인 프로세스에 대한 자세한 정보를 보려면 [Monitoring]에 대해 [OS process list]를 선택합니다.

프로세스 목록 보기는 다음과 같이 표시됩니다.



프로세스 목록 보기에 표시되는 확장 모니터링 지표는 다음과 같이 구성됩니다.

- RDS child processes(RDS 하위 프로세스) – DB 인스턴스를 지원하는 RDS 프로세스(예: , MySQL DB 인스턴스의 경우 `mysqld`)를 요약하여 표시합니다. 프로세스 스레드는 상위 프로세스 아래에 중첩되어 표시됩니다. 프로세스 스레드에는 CPU 사용률만 표시됩니다. 다른 측정치는 프로세스의 모든 스레드에 대해 동일합니다. 콘솔에는 최대 100개의 프로세스와 스레드가 표시됩니다. 결과에는 CPU와 메모리를 소비하는 상위 프로세스 및 스레드가 함께 표시됩니다. 프로세스와 스레드가 각각 50개 이상씩 있는 경우 콘솔에는 각 범주의 상위 50개 소비자가 표시됩니다. 이 표시를 통해 성능에 가장 큰 영향을 미치고 있는 프로세스를 식별할 수 있습니다.
- RDS 프로세스 - RDS DB 인스턴스를 지원하는 데 필요한 RDS 관리 에이전트, 진단 모니터링 프로세스 및 기타 AWS 프로세스에서 사용되는 리소스를 요약하여 표시합니다.
- [OS processes] – 일반적으로 성능에 최소한의 영향만 미치는 커널 및 시스템 프로세스를 요약하여 표시합니다.

각 프로세스에 대해 나열되는 항목은 다음과 같습니다.

- VIRT – 프로세스의 가상 크기를 표시합니다.
- RES – 프로세스에서 사용 중인 실제 물리적 메모리를 표시합니다.
- CPU% – 프로세스에서 사용 중인 총 CPU 대역폭의 백분율을 표시합니다.
- MEM% – 프로세스에서 사용 중인 총 메모리의 백분율을 표시합니다.

RDS 콘솔에 표시되는 모니터링 데이터는 Amazon CloudWatch Logs으로부터 검색됩니다. CloudWatch Logs로부터 로그 스트림으로 DB 인스턴스용 측정치를 검색할 수도 있습니다. 자세한 내용은 CloudWatch Logs을 사용하여 OS 지표 보기 섹션을 참조하세요.

다음 기간 중에는 확장 모니터링 지표가 반환되지 않습니다.

- DB 인스턴스의 장애 조치 동안.
- DB 인스턴스의 인스턴스 클래스 변경(컴퓨팅 확장) 중.

Enhanced Monitoring 측정치는 데이터베이스 엔진이 재부팅되는 이유로만 DB 인스턴스의 재부팅 동안 반환됩니다. 운영 체제의 측정치는 계속 보고됩니다.

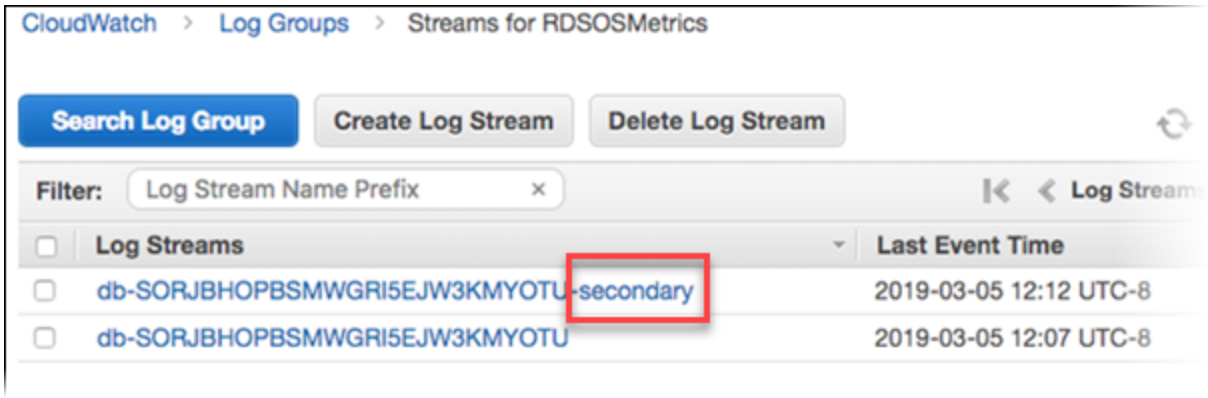
CloudWatch Logs을 사용하여 OS 지표 보기

DB 인스턴스 또는 다중 AZ DB 클러스터에 대한 향상된 모니터링을 활성화한 후 CloudWatch Logs를 사용하여 DB 인스턴스 또는 클러스터에 대한 측정치를 볼 수 있습니다. 각 로그 스트림에는 모니터링 중인 단일 DB 인스턴스가 표시됩니다. 로그 스트림 식별자는 DB 인스턴스 또는 DB 클러스터에 대한 리소스 식별자(`DbiResourceId`)입니다.

Enhanced Monitoring 로그 데이터를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 필요한 경우 DB 인스턴스 또는 다중 AZ DB 클러스터가 있는 AWS 리전을 선택합니다. 자세한 내용은 Amazon Web Services 일반 참조의 리전 및 엔드포인트를 참조하십시오.
3. 탐색 창에서 로그를 선택합니다.
4. 로그 그룹 목록에서 RDSOSMetrics를 선택합니다.

   다중 AZ DB 인스턴스 배포에서 이름에 `-secondary`가 추가된 로프 파일은 다중 AZ 스탠바이 복제본입니다.



5. 로그 스트림 목록에서 보려는 로그 스트림을 선택합니다.

Amazon RDS용 지표 참조

이 참조에서는 Amazon CloudWatch, Performance Insights 및 향상된 모니터링용 Amazon RDS 지표에 대한 설명을 확인할 수 있습니다.

주제

- Amazon RDS에 대한 Amazon CloudWatch 지표
- Amazon RDS에 대한 Amazon CloudWatch 측정기준 목록
- Amazon RDS 성능 개선 도우미를 위한 Amazon CloudWatch 지표
- 성능 개선 도우미 카운터
- 성능 개선 도우미에 대한 SQL 통계
- 향상된 모니터링의 OS 지표

Amazon RDS에 대한 Amazon CloudWatch 지표

Amazon CloudWatch 지표는 Amazon RDS 인스턴스 및 클러스터의 성능과 상태에 대한 통찰력을 제공하여 시스템 동작을 모니터링하고 데이터 기반 결정을 내릴 수 있도록 합니다. 이러한 지표는 리소스 사용률, 데이터베이스 활동 및 운영 효율성을 추적하여 인스턴스의 성능에 대한 가시성을 제공합니다.

이 참조에서는 Amazon RDS에 사용할 수 있는 특정 지표를 간략하게 설명하고 이를 해석하고 사용하여 데이터베이스 성능을 최적화하고, 문제를 해결하고, 고가용성을 보장하는 방법을 설명합니다.

Amazon RDS는 `AWS/RDS` 및 `AWS/Usage` 네임스페이스에서 Amazon CloudWatch에 지표를 게시합니다.

주제

- Amazon RDS에 대한 Amazon CloudWatch 지표
- Amazon RDS에 대한 Amazon CloudWatch 사용량 지표

Amazon RDS에 대한 Amazon CloudWatch 지표

Amazon CloudWatch의 `AWS/RDS` 네임스페이스에는 인스턴스 수준 지표가 포함되어 있습니다.

> **Note**
>
> Amazon RDS 콘솔에는 Amazon CloudWatch에 전송된 단위와 다른 단위로 지표가 표시될 수 있습니다. 예를 들어 Amazon RDS 콘솔에는 지표가 메가바이트(MB) 단위로 표시되는 반면 지표는 바이트 단위로 Amazon CloudWatch에 전송됩니다.

지표	설명	적용 대상	단위
`BinLogDiskUsage`	바이너리 로그가 사용한 디스크 공간의 양입니다. 읽기 전용 복제본을 포함하여 MySQL 및 MariaDB 인스턴스에 대해 자동 백업을 활성화한 경우 바이너리 로그가 생성됩니다.	MariaDB MySQL	바이트
`BurstBalance`	사용할 수 있는 범용 SSD(gp2) 버스트-버킷 I/O 크레딧 비율	모두	%
`CheckpointLag`	가장 최근 체크포인트 이후의 시간입니다.		초
`ConnectionAttempts`	성공 여부에 관계없이 인스턴스에 연결하려는 시도 횟수입니다.	MySQL	개수
`CPUUtilization`	CPU 사용 백분율.	모두	%
`CPUCreditUsage`	CPU 사용률을 위해 인스턴스에서 소비되는 CPU 크레딧의 수입니다. 하나의 CPU 크레딧은 1분 또는 이와 동등한 vCPU, 사용률 및 시간의 조합 동안 100%의 사용률로 실행되는 vCPU 하나에 해당됩니다. 예를 들어, CPU 크레딧 하나는 2분 동안 50%의 사용률로 실행되는 vCPU 하나 또는 2분 동안 25%의 사용률로 실행되는 vCPU 2개에 해당합니다.		크레딧(vCPU-분)

지표	설명	적용 대상	단위
	이 지표는 db.t2, db.t3, db.t4g 인스턴스에만 적용됩니다. **Note** T DB 인스턴스 클래스는 개발 및 테스트 서버 또는 기타 비프로덕션 서버에만 사용하는 것이 좋습니다. T 인스턴스 클래스에 대한 자세한 내용은 DB 인스턴스 클래스 유형 섹션을 참조하세요. CPU 크레딧 측정치는 5분 간격으로만 제공됩니다. 5분 이상의 시간을 지정할 경우 Sum 통계 대신 Average 통계를 사용하세요.		

지표	설명	적용 대상	단위
CPUCredit Balance	시작 이후 인스턴스가 누적한 획득 CPU 크레딧 수입니다. T2 스탠다드의 경우 CPUCreditBalance 에 누적된 시작 크레딧 수도 포함됩니다. 크레딧은 획득 이후에 크레딧 밸런스에 누적되고, 소비 시 크레딧 밸런스에서 소멸됩니다. 크레딧 밸런스는 최대 한도(인스턴스 크기에 따라 결정)가 있습니다. 한도에 도달하면 새로 획득한 크레딧이 모두 삭제됩니다. T2 스탠다드의 경우 시작 크레딧은 한도에 포함되지 않습니다. CPUCreditBalance 의 크레딧은 인스턴스가 기준 CPU 사용률 이상으로 버스터를 하는 데 소비할 수 있습니다. 인스턴스가 실행 중인 동안 CPUCredit Balance 의 크레딧은 만료되지 않습니다. 인스턴스가 중지되면 CPUCredit Balance 는 지속되지 않고 모든 누적된 크레딧이 삭제됩니다. CPU 크레딧 지표는 5분 간격으로만 제공됩니다. 이 지표는 db.t2, db.t3, db.t4g 인스턴스에만 적용됩니다. **Note** T DB 인스턴스 클래스는 개발 및 테스트 서버 또는 기타 비프로덕션 서버에만 사용하는 것이 좋습니다. T 인스턴스 클래스에 대한 자세한 내용은 DB 인스턴		크레딧(vCPU-분)

지표	설명	적용 대상	단위
	스 클래스 유형 섹션을 참조하세요. 시작 크레딧은 Amazon RDS에서도 Amazon EC2에서와 동일한 방식으로 작동합니다. 자세한 내용은 Linux 인스턴스용 Amazon Elastic Compute Cloud 사용 설명서의 시작 크레딧을 참조하십시오.		
CPUSurplusCreditBalance	CPUCreditBalance 값이 0일 때 무제한 인스턴스에서 소비된 잉여 크레딧의 수 입니다. 획득한 CPU 크레딧에 따라 CPUSurplusCreditBalance 값이 청산됩니다. 잉여 크레딧의 수가 인스턴스가 24시간 동안 획득할 수 있는 최대 크레딧 수를 초과한 경우 최대 값 이상으로 소비된 잉여 크레딧은 추가 요금으로 부과됩니다. CPU 크레딧 지표는 5분 간격으로만 제공됩니다.	모두	크레딧(vCPU-분)

지표	설명	적용 대상	단위
`CPUSurplusCreditsCharged`	획득한 CPU 크레딧으로 청산되지 않는 소비 잉여 크레딧의 수로, 추가 요금으로 부과됩니다. 소비된 잉여 크레딧은 다음이 발생할 때 요금이 부과됩니다. • 소비한 잉여 크레딧이 인스턴스가 24시간 동안 획득할 수 있는 최대 크레딧 수를 초과하는 경우. 해당 시간이 끝날 때 최대 값 이상으로 소비한 잉여 크레딧에 요금이 부과됩니다. • 인스턴스가 중지 또는 종료된 경우. • 인스턴스가 `unlimited` 에서 `standard`로 전환됩니다. CPU 크레딧 지표는 5분 간격으로만 제공됩니다.	모두	크레딧(vCPU-분)

지표	설명	적용 대상	단위
DatabaseConnections	데이터베이스 인스턴스에 대한 클라이언트 네트워크 연결 수입니다. 지표 값에 다음이 포함되지 않기 때문에 데이터베이스 세션 수가 지표 값보다 클 수 있습니다. • 더 이상 네트워크에 연결되어 있지 않지만 데이터베이스에 의해 정리되지 않은 세션 • 데이터베이스 엔진에 의해 자체 용도로 생성된 세션 • 데이터베이스 엔진의 병렬 실행 기능에 의해 생성된 세션 • 데이터베이스 엔진 작업 스케줄러에 의해 생성된 세션 • Amazon RDS 연결	모두	개수
DiskQueueDepth	디스크 액세스를 대기 중인 I/O(읽기/쓰기 요청) 수입니다.	모두	개수
DiskQueueDepthLogVolume	로그 볼륨 디스크 액세스를 대기 중인 I/O(읽기/쓰기 요청) 수입니다.	전용 로그 볼륨이 활성화된 DB 인스턴스	개수

지표	설명	적용 대상	단위
EBSByteBalance%	RDS 데이터베이스의 버스트 버킷에 남아 있는 처리량 크레딧의 백분율입니다. 기본 모니터링에서만 이 지표를 사용할 수 있습니다. 지표 값은 데이터베이스 파일이 포함된 볼륨만이 아니라 루트 볼륨을 포함한 모든 볼륨의 처리량을 기반으로 합니다. 이 지표를 지원하는 인스턴스 크기를 찾으려면, Amazon EC2 사용 설명서의 기본적으로 EBS 최적화 테이블에서 별표(*)가 있는 인스턴스 크기를 참조하세요. Sum 통계는 이 지표에 적용할 수 없습니다.	모두	%

지표	설명	적용 대상	단위
EBSIOBalance%	RDS 데이터베이스의 버스트 버킷에 남아 있는 I/O 크레딧의 백분율입니다. 기본 모니터링에서만 이 지표를 사용할 수 있습니다. 지표 값은 데이터베이스 파일이 포함된 볼륨만이 아니라 루트 볼륨을 포함한 모든 볼륨의 IOPS를 기반으로 합니다. 이 지표를 지원하는 인스턴스 크기를 찾으려면 Amazon EC2 사용 설명서의 Amazon EBS 최적화 인스턴스 유형을 참조하세요. Sum 통계는 이 지표에 적용할 수 없습니다. 이 지표는 BurstBalance 와 다릅니다. 이 지표를 사용하는 방법을 알아보려면 Improving application performance and reducing costs with Amazon EBS-Optimized Instance burst capability를 참조하세요.	모두	%
FailedSQLServerAgentJobsCount	최근 1분간 실패한 Microsoft SQL Server 에이전트 작업 수입니다.	Microsoft SQL Server	분당 개수
FreeableMemory	사용 가능한 RAM 크기. MariaDB, MySQL, Oracle, PostgreSQL DB 인스턴스의 경우 이 지표에서는 MemAvailable 의 /proc/meminfo 필드 값을 보고합니다.	모두	바이트

지표	설명	적용 대상	단위
FreeLocalStorage	사용 가능한 로컬 스토리지 공간 크기입니다. 이 지표는 NVMe SSD 인스턴스 스토어 볼륨이 있는 DB 인스턴스 클래스에만 적용됩니다. NVMe SSD 인스턴스 스토어 볼륨이 있는 Amazon EC2 인스턴스에 대한 자세한 내용은 인스턴스 스토어 볼륨을 참조하십시오. 동일한 RDS DB 인스턴스 클래스는 동일한 인스턴스 스토어 볼륨이 있습니다. 예를 들어 db.m6gd 및 db.r6gd DB 인스턴스 클래스에는 NVMe SSD 인스턴스 스토어 볼륨이 있습니다.		바이트
FreeLocalStoragePercent	사용 가능한 로컬 스토리지 공간의 비율입니다. 이 지표는 NVMe SSD 인스턴스 스토어 볼륨이 있는 DB 인스턴스 클래스에만 적용됩니다. NVMe SSD 인스턴스 스토어 볼륨이 있는 Amazon EC2 인스턴스에 대한 자세한 내용은 인스턴스 스토어 볼륨을 참조하십시오. 동일한 RDS DB 인스턴스 클래스는 동일한 인스턴스 스토어 볼륨이 있습니다. 예를 들어 db.m6gd 및 db.r6gd DB 인스턴스 클래스에는 NVMe SSD 인스턴스 스토어 볼륨이 있습니다.		%
FreeStorageSpace	사용 가능한 스토리지 공간 크기입니다.	모두	바이트
FreeStorageSpaceLogVolume	로그 볼륨에서 사용 가능한 스토리지 공간입니다.	전용 로그 볼륨이 활성화된 DB 인스턴스	바이트

지표	설명	적용 대상	단위
IamDbAuthConnectionRequests	DB 인스턴스에 대한 IAM 인증을 사용하는 연결 요청 수입니다.	모두	개수
MaximumUsedTransactionIDs	사용된 최대 트랜잭션 ID입니다.	PostgreSQL	개수
NetworkReceiveThroughput	DB 인스턴스 수신 네트워크 트래픽(고객 데이터베이스 트래픽과 모니터링 및 복제에 사용된 Amazon RDS 트래픽 모두 포함).	모두	초당 바이트
NetworkTransmitThroughput	DB 인스턴스 송신 네트워크 트래픽(고객 데이터베이스 트래픽과 모니터링 및 복제에 사용된 Amazon RDS 트래픽 모두 포함).	모두	초당 바이트
OldestLogicalReplicationSlotLag	Amazon RDS의 지연 크기는 소스 데이터베이스에서 트랜잭션을 커밋하고 RDS가 복제본 데이터베이스에 트랜잭션을 적용하는 시간입니다.	PostgreSQL	바이트
OldestReplicationSlotLag	수신된 WAL(Write-Ahead Log) 데이터를 기준으로 가장 지연된 복제본의 지연 크기.	PostgreSQL	바이트
ReadIOPS	초당 평균 디스크 읽기 I/O 연산 수	모두	초당 개수

지표	설명	적용 대상	단위
`ReadIOPSLocalStorage`	로컬 스토리지에 대한 초당 평균 디스크 읽기 I/O 작업 수입니다. 이 지표는 NVMe SSD 인스턴스 스토어 볼륨이 있는 DB 인스턴스 클래스에만 적용됩니다. NVMe SSD 인스턴스 스토어 볼륨이 있는 Amazon EC2 인스턴스에 대한 자세한 내용은 인스턴스 스토어 볼륨을 참조하십시오. 동일한 RDS DB 인스턴스 클래스는 동일한 인스턴스 스토어 볼륨이 있습니다. 예를 들어 db.m6gd 및 db.r6gd DB 인스턴스 클래스에는 NVMe SSD 인스턴스 스토어 볼륨이 있습니다.		초당 개수
`ReadIOPSLogVolume`	로그 볼륨에서 초당 평균 디스크 읽기 I/O 작업 수입니다.	전용 로그 볼륨이 활성화된 DB 인스턴스	초당 개수
`ReadLatency`	디스크 I/O 연산당 평균 처리 시간입니다.	모두	초
`ReadLatencyLocalStorage`	로컬 스토리지의 디스크 I/O 작업당 소요된 평균 시간입니다. 이 지표는 NVMe SSD 인스턴스 스토어 볼륨이 있는 DB 인스턴스 클래스에만 적용됩니다. NVMe SSD 인스턴스 스토어 볼륨이 있는 Amazon EC2 인스턴스에 대한 자세한 내용은 인스턴스 스토어 볼륨을 참조하십시오. 동일한 RDS DB 인스턴스 클래스는 동일한 인스턴스 스토어 볼륨이 있습니다. 예를 들어 db.m6gd 및 db.r6gd DB 인스턴스 클래스에는 NVMe SSD 인스턴스 스토어 볼륨이 있습니다.		초

지표	설명	적용 대상	단위
ReadLatencyLogVolume	로그 볼륨에서 디스크 I/O 작업당 평균 처리 시간입니다.	전용 로그 볼륨이 활성화된 DB 인스턴스	초
ReadThroughput	초당 디스크에서 읽은 평균 바이트 수입니다.	모두	초당 바이트
ReadThroughputLocalStorage	로컬 스토리지의 초당 디스크에서 읽은 평균 바이트 수입니다. 이 지표는 NVMe SSD 인스턴스 스토어 볼륨이 있는 DB 인스턴스 클래스에만 적용됩니다. NVMe SSD 인스턴스 스토어 볼륨이 있는 Amazon EC2 인스턴스에 대한 자세한 내용은 인스턴스 스토어 볼륨을 참조하십시오. 동일한 RDS DB 인스턴스 클래스는 동일한 인스턴스 스토어 볼륨이 있습니다. 예를 들어 db.m6gd 및 db.r6gd DB 인스턴스 클래스에는 NVMe SSD 인스턴스 스토어 볼륨이 있습니다.		초당 바이트
ReadThroughputLogVolume	로그 볼륨에서 초당 디스크에서 읽은 평균 바이트 수입니다.	전용 로그 볼륨이 활성화된 DB 인스턴스	초당 바이트
ReplicaLag	원본 DB 인스턴스를 기준으로 읽기 전용 복제본 DB 인스턴스의 지연 시간. MySQL, MariaDB, Oracle, PostgreSQL 및 SQL Server 읽기 전용 복제본에 적용됩니다. 다중 AZ DB 클러스터에 있는 리더 DB 인스턴스에서 가장 최근에 적용된 트랜잭션과 라이터 DB 인스턴스에서 가장 최근 트랜잭션 사이의 시간 차이입니다.		초

지표	설명	적용 대상	단위
ReplicationChannelLag	다중 소스 복제본 구성의 경우 소스 DB 인스턴스를 기준으로 다중 소스 복제본 특정 채널의 지연 시간입니다. 자세한 내용은 the section called “다중 소스 복제 채널 모니터링” 섹션을 참조하세요.	MySQL	초
ReplicationSlotDiskUsage	복제 슬롯 파일에 사용된 디스크 공간.	PostgreSQL	바이트
SwapUsage	DB 인스턴스에서 사용된 스왑 공간 크기.	MariaDB MySQL Oracle PostgreSQL	바이트
TempDbAvailableDataSpace	tempdb에서 사용 가능한 데이터 공간의 양과 tempdb가 위치한 볼륨입니다. 이 지표를 사용하여 tempdb 데이터 공간 가용성을 모니터링하고 그에 따라 용량을 계획합니다. 값이 낮으면 스토리지를 늘리거나 tempdb를 많이 사용하는 쿼리를 최적화해야 할 수 있습니다.	SQL Server	바이트
TempDbAvailableLogSpace	tempdb에서 사용 가능한 로그 공간의 양과 tempdb가 위치한 볼륨입니다. 이 지표를 사용하여 tempdb 로그 공간 가용성을 모니터링하고 트랜잭션 로그 가득 참 상태를 방지합니다. 상당한 로그 활동을 생성하는 대규모 트랜잭션이나 높은 동시성이 있는 워크로드에 필수적입니다.	SQL Server	바이트

지표	설명	적용 대상	단위
TempDbDataFileUsage	tempdb에 사용된 데이터 파일의 백분율입니다. 이 지표는 잠재적 파일 증가를 고려하지 않습니다. 이 지표를 사용하여 tempdb 데이터 파일 사용률을 모니터링하고 잠재적 성능 병목 현상을 식별합니다. 값이 높으면 대규모 임시 객체를 생성하거나 tempdb 크기를 늘리는 쿼리를 최적화해야 할 수 있습니다.	SQL Server	%
TempDbLogFileUsage	tempdb에 사용된 로그 파일의 백분율입니다. 이 지표는 잠재적 파일 증가를 고려하지 않습니다. 이 지표를 사용하여 tempdb 로그 파일 사용률을 모니터링하고 성능 문제를 방지합니다. 값이 높으면 전반적인 데이터베이스 성능에 영향을 줄 수 있는 장기 실행 트랜잭션이나 과도한 로깅 활동이 있을 수 있습니다.	SQL Server	%
TransactionLogsDiskUsage	트랜잭션 로그에 사용된 디스크 공간.	PostgreSQL	바이트
TransactionLogsGeneration	초당 생성되는 트랜잭션 로그의 크기.	PostgreSQL	초당 바이트
WriteIOPS	초당 평균 디스크 쓰기 I/O 연산 수	모두	초당 개수

지표	설명	적용 대상	단위
`WriteIOPSLocalStorage`	로컬 스토리지에서 초당 평균 디스크 쓰기 I/O 연산 수입니다. 이 지표는 NVMe SSD 인스턴스 스토어 볼륨이 있는 DB 인스턴스 클래스에만 적용됩니다. NVMe SSD 인스턴스 스토어 볼륨이 있는 Amazon EC2 인스턴스에 대한 자세한 내용은 인스턴스 스토어 볼륨을 참조하십시오. 동일한 RDS DB 인스턴스 클래스는 동일한 인스턴스 스토어 볼륨이 있습니다. 예를 들어 db.m6gd 및 db.r6gd DB 인스턴스 클래스에는 NVMe SSD 인스턴스 스토어 볼륨이 있습니다.		초당 개수
`WriteIOPSLogVolume`	로그 볼륨에서 초당 평균 디스크 쓰기 I/O 작업 수입니다.	전용 로그 볼륨이 활성화된 DB 인스턴스	초당 개수
`WriteLatency`	디스크 I/O 연산당 평균 처리 시간입니다.	모두	초
`WriteLatencyLocalStorage`	로컬 스토리지의 디스크 I/O 작업당 소요된 평균 시간입니다. 이 지표는 NVMe SSD 인스턴스 스토어 볼륨이 있는 DB 인스턴스 클래스에만 적용됩니다. NVMe SSD 인스턴스 스토어 볼륨이 있는 Amazon EC2 인스턴스에 대한 자세한 내용은 인스턴스 스토어 볼륨을 참조하십시오. 동일한 RDS DB 인스턴스 클래스는 동일한 인스턴스 스토어 볼륨이 있습니다. 예를 들어 db.m6gd 및 db.r6gd DB 인스턴스 클래스에는 NVMe SSD 인스턴스 스토어 볼륨이 있습니다.		초

지표	설명	적용 대상	단위
WriteLatencyLogVolume	로그 볼륨에서 디스크 I/O 작업당 평균 처리 시간입니다.	전용 로그 볼륨이 활성화된 DB 인스턴스	초
WriteThroughput	초당 디스크에 쓴 평균 바이트 수.	모두	초당 바이트
WriteThroughputLogVolume	로그 볼륨에서 초당 디스크에 쓴 평균 바이트 수입니다.	전용 로그 볼륨이 활성화된 DB 인스턴스	초당 바이트
WriteThroughputLocalStorage	로컬 스토리지에 디스크에 쓰여진 초당 평균 바이트 수입니다. 이 지표는 NVMe SSD 인스턴스 스토어 볼륨이 있는 DB 인스턴스 클래스에만 적용됩니다. NVMe SSD 인스턴스 스토어 볼륨이 있는 Amazon EC2 인스턴스에 대한 자세한 내용은 인스턴스 스토어 볼륨을 참조하십시오. 동일한 RDS DB 인스턴스 클래스는 동일한 인스턴스 스토어 볼륨이 있습니다. 예를 들어 db.m6gd 및 db.r6gd DB 인스턴스 클래스에는 NVMe SSD 인스턴스 스토어 볼륨이 있습니다.		초당 바이트

Amazon RDS에 대한 Amazon CloudWatch 사용량 지표

Amazon CloudWatch의 AWS/Usage 네임스페이스에는 Amazon RDS 서비스 할당량에 대한 계정 수준 사용량 지표가 포함됩니다. CloudWatch는 모든 AWS 리전의 사용량 지표를 자동으로 수집합니다.

자세한 내용은 Amazon CloudWatch 사용 설명서에서 CloudWatch 사용량 지표를 참조하세요. 할당량에 대한 자세한 내용은 Service Quotas 사용 설명서의 Amazon RDS에 대한 할당량 및 제약 조건 및 할당량 증가 요청을 참조하세요.

지표	설명	단위*
AllocatedStorage	모든 DB 인스턴스의 총 스토리지입니다. 합계에는 임시 마이그레이션 인스턴스가 제외됩니다.	GB
AuthorizationsPerDBSecurityGroup	AWS 계정의 DB 보안 그룹당 수신 규칙 수입니다. 사용된 값은 계정의 DB 보안 그룹에서 가장 많은 수신 규칙 수입니다. 계정의 다른 DB 보안 그룹은 수신 규칙 수가 더 적을 수 있습니다.	개수
CustomEndpointsPerDBCluster	AWS 계정의 DB 클러스터당 사용자 지정 엔드포인트 수입니다. 사용된 값은 계정의 DB 클러스터에서 가장 많은 사용자 지정 엔드포인트 수입니다. 계정의 다른 DB 클러스터는 사용자 지정 엔드포인트 수가 더 적을 수 있습니다.	개수
CustomEngineVersions	AWS 계정의 Amazon RDS Custom에 대한 사용자 지정 엔진 버전(CEV) 수입니다.	개수
DBClusterParameterGroups	AWS 계정에서 DB 클러스터 파라미터 그룹의 수입니다. 개수에는 기본 파라미터 그룹이 제외됩니다.	개수
DBClusterRoles	AWS Identity and Access Management의 DB 클러스터당 연결된 AWS 계정(IAM) 역할 수입니다. 사용된 값은 계정의 DB 클러스터에 연결된 IAM 역할이 가장 많은 수입니다. 계정의 다른 DB 클러스터는 연결된 IAM 역할 수가 더 적을 수 있습니다.	개수
DBClusters	AWS 계정에서 Amazon Aurora DB 클러스터의 수입니다.	개수
DBInstanceRoles	AWS Identity and Access Management의 DB 인스턴스당 연결된 AWS 계정(IAM) 역할 수입니다. 사용된 값은 계정의 DB 인스턴스에 연결된 IAM 역할이 가장 많은 수입니다. 계정의 다른 DB 인스턴스는 연결된 IAM 역할 수가 더 적을 수 있습니다.	개수
DBInstances	AWS 계정에서 DB 인스턴스의 수입니다.	개수

지표	설명	단위*
DBParameterGroups	AWS 계정에서 DB 파라미터 그룹의 수입니다. 개수에는 기본 DB 파라미터 그룹이 제외됩니다.	개수
DBSecurityGroups	AWS 계정에서 보안 그룹 수입니다. 개수에는 기본 보안 그룹과 기본 VPC 보안 그룹이 제외됩니다.	개수
DBSubnetGroups	AWS 계정에서 DB 서브넷 그룹의 수입니다. 개수에는 기본 서브넷 그룹이 제외됩니다.	개수
EventSubscriptions	AWS 계정의 이벤트 알림 구독 수입니다.	개수
Integrations	AWS 계정에 있는 Amazon Redshift와의 제로 ETL 통합 수입니다.	개수
ManualClusterSnapshots	AWS 계정에서 수동으로 생성된 DB 클러스터 스냅샷의 수입니다. 개수에는 잘못된 스냅샷이 제외됩니다.	개수
ManualSnapshots	AWS 계정에서 수동으로 생성된 DB 스냅샷의 수입니다. 개수에는 잘못된 스냅샷이 제외됩니다.	개수
OptionGroups	AWS 계정에서 옵션 그룹의 수입니다. 개수에는 기본 옵션 그룹이 제외됩니다.	개수
Proxies	AWS 계정의 RDS 프록시 수입니다.	개수
ReadReplicasPerMaster	계정의 DB 인스턴스당 읽기 전용 복제본 수입니다. 사용된 값은 계정의 DB 인스턴스에서 가장 많은 읽기 복제본 수입니다. 계정의 다른 DB 인스턴스는 읽기 전용 복제본 수가 더 적을 수 있습니다.	개수
ReservedDBInstances	AWS 계정에서 예약된 DB 인스턴스의 수입니다. 개수에는 수명 종료되거나 거절된 인스턴스가 제외합니다.	개수
SubnetsPerDBSubnetGroup	AWS 계정의 DB 서브넷 그룹당 서브넷 개수입니다. 계정의 DB 서브넷 그룹에 가장 많은 서브넷 수입니다. 계정의 다른 DB 서브넷 그룹은 서브넷 수가 더 적을 수 있습니다.	개수

> **Note**
>
> Amazon RDS는 CloudWatch에 사용량 지표 단위를 게시하지 않습니다. 단위는 설명서에만 표시됩니다.

Amazon RDS에 대한 Amazon CloudWatch 측정기준 목록

다음 표의 차원을 사용하여 Amazon RDS 지표 데이터를 필터링할 수 있습니다.

차원	다음에 대해 요청된 데이터를 필터링합니다.
`DBInstanceIdentifier`	특정 DB 인스턴스
`DatabaseClass`	데이터베이스 클래스의 모든 인스턴스 예를 들어 데이터베이스 클래스 `db.r5.large` 에 속하는 모든 인스턴스에 대한 지표를 집계할 수 있습니다.
`EngineName`	식별된 엔진 이름만 예를 들어 엔진 이름이 `postgres`인 모든 인스턴스에 대한 지표를 집계할 수 있습니다.
`SourceRegion`	지정된 리전만 예를 들어 `us-east-1` 리전의 모든 DB 인스턴스에 대한 지표를 집계할 수 있습니다.

Amazon RDS 성능 개선 도우미를 위한 Amazon CloudWatch 지표

성능 개선 도우미는 Amazon CloudWatch에 일부 지표를 자동으로 게시합니다. 동일한 데이터는 성능 개선 도우미에서 쿼리할 수 있지만 CloudWatch에 지표가 있으면 CloudWatch 경보를 더 쉽게 추가할 수 있습니다. 또한 기존 CloudWatch 대시보드에 지표를 더 쉽게 추가할 수 있습니다.

측정치	설명
DBLoad	데이터베이스에 대한 활성 세션 수입니다. 일반적으로 사용자는 활성 세션의 평균 개수에 대한 데이터를 원합니다. 성능 개선 도우미에서 이 데이터는 `db.load.avg` 로 쿼리됩니다.

측정치	설명
DBLoadCPU	대기 이벤트 유형이 CPU인 활성 세션 수입니다. 성능 개선 도우미에서 이 데이터는 db.load.avg 로 쿼리되며 대기 이벤트 유형인 CPU를 기준으로 필터링됩니다.
DBLoadNonCPU	대기 이벤트 유형이 CPU가 아닌 활성 세션 수입니다.
DBLoadRelativeToNumVCPUs	데이터베이스의 가상 CPU 수에 대한 DB 부하의 비율입니다.

> **Note**
>
> 이러한 메트릭은 DB 인스턴스에 로드가 있는 경우에만 CloudWatch에 게시됩니다.

CloudWatch 콘솔, AWS CLI 또는 CloudWatch API를 사용하여 이러한 지표를 검사할 수 있습니다. 특수 지표 수학 함수를 사용하여 다른 성능 개선 도우미 카운터 지표를 검사할 수도 있습니다. 자세한 내용은 CloudWatch에서 다른 성능 개선 도우미 카운터 지표 쿼리 섹션을 참조하세요.

예를 들어, get-metric-statistics 명령을 실행하여 DBLoad 지표에 대한 통계를 가져올 수 있습니다.

```
aws cloudwatch get-metric-statistics \
    --region us-west-2 \
    --namespace AWS/RDS \
    --metric-name DBLoad  \
    --period 60 \
    --statistics Average \
    --start-time 1532035185 \
    --end-time 1532036185 \
    --dimensions Name=DBInstanceIdentifier,Value=db-loadtest-0
```

이 예에서는 다음과 비슷한 출력이 생성됩니다.

```
{
  "Datapoints": [
  {
```

```
"Timestamp": "2021-07-19T21:30:00Z",
"Unit": "None",
"Average": 2.1
},
{
"Timestamp": "2021-07-19T21:34:00Z",
"Unit": "None",
"Average": 1.7
},
{
"Timestamp": "2021-07-19T21:35:00Z",
"Unit": "None",
"Average": 2.8
},
{
"Timestamp": "2021-07-19T21:31:00Z",
"Unit": "None",
"Average": 1.5
},
{
"Timestamp": "2021-07-19T21:32:00Z",
"Unit": "None",
"Average": 1.8
},
{
"Timestamp": "2021-07-19T21:29:00Z",
"Unit": "None",
"Average": 3.0
},
{
"Timestamp": "2021-07-19T21:33:00Z",
"Unit": "None",
"Average": 2.4
}
],
"Label": "DBLoad"
}
```


CloudWatch에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch란 무엇입니까?를 참조하세요.

CloudWatch에서 다른 성능 개선 도우미 카운터 지표 쿼리

> **Note**
>
> Database Insights의 고급 모드를 사용 설정하면 Amazon RDS는 성능 개선 도우미 카운터 지표를 Amazon CloudWatch에 게시합니다. Database Insights에서는 `DB_PERF_INSIGHTS` 지표 수학 함수를 사용할 필요가 없습니다. CloudWatch Database Insights 대시보드를 사용하여 성능 개선 도우미 카운터 지표에 대한 경보를 검색, 쿼리 및 설정할 수 있습니다.

CloudWatch에서 RDS 성능 개선 도우미 지표를 쿼리하고, 경보를 표시하고, 그래프를 작성할 수 있습니다. CloudWatch의 `DB_PERF_INSIGHTS` 지표 수학 함수를 사용하여 DB 인스턴스에 대한 정보에 액세스할 수 있습니다. 이 함수를 사용하면 CloudWatch에 직접 보고되지 않는 성능 개선 도우미 지표를 사용하여 새 시계열을 생성할 수 있습니다.

CloudWatch 콘솔의 지표 선택 화면에서 수학 추가 드롭다운 메뉴를 클릭하여 새로운 지표 수학 함수를 사용할 수 있습니다. 이를 사용하여 성능 개선 도우미 지표 또는 CloudWatch와 성능 개선 도우미 지표의 조합에 대한 경보 및 그래프를 생성할 수 있습니다. 여기에는 1분 미만의 지표에 대한 고해상도 경보가 포함됩니다. `get-metric-data` 요청에 지표 수학 표현식을 포함하여 프로그래밍 방식으로 함수를 사용할 수도 있습니다. 자세한 내용은 지표 수학 구문 및 함수 및 AWS 데이터베이스의 성능 개선 도우미 카운터 지표에 대한 경보 생성을 참조하세요.

성능 개선 도우미 카운터

카운터 지표는 Performance Insights 대시보드의 운영 체제 및 데이터베이스 성능 지표 입니다. 이 정보와 데이터베이스 로드를 연관 지으면 성능 문제를 식별하고 분석하는 데 도움이 됩니다. 지표에 통계 함수를 추가해야 지표 값을 가져올 수 있습니다. 예를 들어 `os.memory.active` 지표에 지원되는 함수는 `.avg`, `.min`, `.max`, `.sum`, `.sample_count`입니다.

카운터 지표는 1분에 한 번씩 수집됩니다. OS 지표 수집은 향상된 모니터링을 켰는지, 껐는지에 따라 달라집니다. 향상된 모니터링이 꺼져 있는 경우 OS 지표는 1분마다 한 번씩 수집됩니다. 향상된 모니터링이 켜져 있는 경우 OS 지표는 선택한 기간 동안 수집됩니다. 향상된 모니터링에 대한 자세한 내용은 향상된 모니터링 설정 및 해제 섹션을 참조하세요.

주제

- 성능 개선 도우미 운영 체제 카운터
- Amazon RDS for MariaDB 및 MySQL에 대한 성능 개선 도우미 카운터
- Amazon RDS for Microsoft SQL Server용 성능 개선 도우미 카운터

- Amazon RDS for Oracle의 성능 개선 도우미 카운터
- Amazon RDS for PostgreSQL용 성능 개선 도우미 카운터

성능 개선 도우미 운영 체제 카운터

다음 운영 체제 카운터는 os로 접두사가 붙으며, 모든 RDS 엔진(RDS for SQL Server 제외) 에 대한 성능 개선 도우미를 사용할 수 있습니다.

`ListAvailableResourceMetrics` API를 사용하여 DB 인스턴스에 대해 지원되는 카운터 지표 목록을 확인할 수 있습니다. 자세한 내용은 Amazon RDS 성능 개선 도우미 API 참조 안내서의 ListAvailableResourceMetrics를 참조하세요.

카운터	유형	Unit	지표	설명
활성	메모리	KB	os.memory.active	할당된 메모리의 양(KB)
버퍼	메모리	KB	os.memory.buffers	스토리지 디바이스에 쓰기 이전에 I/O 요청을 버퍼링하는 데 사용되는 메모리의 양(KB)
캐시됨	메모리	KB	os.memory.cached	파일 시스템 기반 I/O를 캐시하는 데 사용되는 메모리의 양(킬로바이트 단위)입니다.
DB 캐시	메모리	바이트	os.memory.db.cache	tmpfs(shmem)를 포함한 데이터베이스 프로세스별 페이지 캐시에 사용되는 메모리의 양(바이트 단위)입니다.

카운터	유형	Unit	지표	설명
DB 상주 세트 크기	메모리	바이트	os.memory.db.residentSetSize	tmpfs(shmem)를 제외한 데이터베이스 프로세스별 익명 및 스왑 캐시에 사용되는 메모리의 양(바이트 단위)입니다.
DB 스왑	메모리	바이트	os.memory.db.swap	데이터베이스 프로세스별 스왑에 사용되는 메모리의 양(바이트 단위)입니다.
더티	메모리	KB	os.memory.dirty	수정되었지만 스토리지의 관련 데이터 블록에 기록되지 않은 RAM의 메모리 페이지 양(KB)
무료	메모리	KB	os.memory.free	할당되지 않은 메모리의 양(KB)
사용 가능한 방대한 페이지	메모리	Pages	os.memory.hugePagesFree	사용 가능한 방대한 페이지 수입니다. 방대한 페이지는 Linux 커널의 기능입니다.
예약된 방대한 페이지	메모리	Pages	os.memory.hugePagesRsvd	커밋된 방대한 페이지의 수
방대한 페이지 크기	메모리	KB	os.memory.hugePagesSize	각 방대한 페이지 단위의 크기(KB)

카운터	유형	Unit	지표	설명
초과된 방대한 페이지	메모리	Pages	os.memory.hugePagesSurp	총계 대비 사용 가능한 초과 방대한 페이지 수
방대한 페이지 합계	메모리	Pages	os.memory.hugePagesTotal	방대한 페이지의 총 수입니다.
비활성	메모리	KB	os.memory.inactive	가장 적게 사용되는 메모리 페이지의 양(KB)
매핑됨	메모리	KB	os.memory.mapped	프로세스 주소 공간 내에 메모리 매핑되는 총 파일 시스템 콘텐츠 양(KB)
메모리 부족 처리 수	메모리	종료	os.memory.outOfMemoryKillCount	마지막 수집 간격 동안 발생한 OOM 처리 수입니다.
페이지 테이블	메모리	KB	os.memory.pageTables	페이지 표에 사용된 메모리의 양(KB)
슬래브	메모리	KB	os.memory.slab	재사용 가능한 커널 데이터 구조의 양(KB)
합계	메모리	KB	os.memory.total	총 메모리 양(KB)
Writeback	메모리	KB	os.memory.writeback	RAM에서 지원 스토리지에 아직 기록 중인 더티 페이지의 양(KB)

카운터	유형	Unit	지표	설명
게스트	CPU 사용률	백분율	os.cpuUtilization.guest	게스트 프로그램에서 사용 중인 CPU의 비율
유휴	CPU 사용률	백분율	os.cpuUtilization.idle	유휴 상태인 CPU의 비율
Irq	CPU 사용률	백분율	os.cpuUtilization.irq	소프트웨어 인터럽트에서 사용 중인 CPU의 비율
Nice	CPU 사용률	백분율	os.cpuUtilization.nice	가장 낮은 우선순위로 실행 중인 프로그램에서 사용 중인 CPU의 비율
도용	CPU 사용률	백분율	os.cpuUtilization.steal	다른 가상 머신에서 사용 중인 CPU의 비율
시스템	CPU 사용률	백분율	os.cpuUtilization.system	커널에서 사용 중인 CPU의 비율
합계	CPU 사용률	백분율	os.cpuUtilization.total	사용 중인 CPU의 총 비율입니다. 이 값에는 nice 값이 포함됩니다.
User	CPU 사용률	백분율	os.cpuUtilization.user	사용자 프로그램에서 사용 중인 CPU의 비율
Wait	CPU 사용률	백분율	os.cpuUtilization.wait	I/O 액세스를 대기 중인 동안 사용되지 않은 CPU의 비율

카운터	유형	Unit	지표	설명
읽기 IO PS	디스크 I/O	초당 요청	os.diskIO.<devicename>.readIOsPS	초당 읽기 작업 수
쓰기 IO PS	디스크 I/O	초당 요청	os.diskIO.<devicename>.writeIOsPS	초당 쓰기 작업 수
평균 대기열 길이	디스크 I/O	요청	os.diskIO.<devicename>.avgQueueLen	I/O 디바이스의 대기열에서 대기 중인 요청 수입니다.
평균 요청 크기	디스크 I/O	요청	os.diskIO.<devicename>.avgReqSz	I/O 디바이스의 대기열에서 대기 중인 요청 수입니다.
대기	디스크 I/O	밀리초	os.diskIO.<devicename>.await	대기열 시간과 서비스 시간을 포함하여 요청에 응답하는 데 필요한 시간(밀리초)
읽기 IO PS	디스크 I/O	요청	os.diskIO.<devicename>.readIOsPS	초당 읽기 작업 수
읽기 KB	디스크 I/O	KB	os.diskIO.<devicename>.readKb	읽은 총 KB 수

카운터	유형	Unit	지표	설명
읽기 KB PS	디스크 I/O	초당 킬로바이트	os.diskIO.<devicename>.readKbPS	초당 읽은 KB 수
Rrqm PS	디스크 I/O	초당 요청	os.diskIO.<devicename>.rrqmPS	초당 대기 중인 병합 읽기 요청 수
TPS	디스크 I/O	초당 트랜잭션 수	os.diskIO.<devicename>.tps	초당 I/O 트랜잭션 수
유틸리티	디스크 I/O	백분율	os.diskIO.<devicename>.util	요청이 발급된 CPU 시간의 비율
쓰기 KB	디스크 I/O	KB	os.diskIO.<devicename>.writeKb	기록한 총 KB 수
쓰기 KB PS	디스크 I/O	초당 킬로바이트	os.diskIO.<devicename>.writeKbPS	초당 기록한 KB 수
Wrqm PS	디스크 I/O	초당 요청	os.diskIO.<devicename>.wrqmPS	초당 대기 중인 병합 쓰기 요청 수
차단됨	업무	업무	os.tasks.blocked	차단되는 작업 수
실행 중	업무	업무	os.tasks.running	실행 중인 작업 수

카운터	유형	Unit	지표	설명
Sleeping	업무	업무	os.tasks.sleeping	절전 상태인 작업 수
Stopped	업무	업무	os.tasks.stopped	중단된 작업 수
합계	업무	업무	os.tasks.total	총 작업 수
좀비	업무	업무	os.tasks.zombie	상위 작업은 활성화되었지만 비활성 상태인 하위 작업 수
1개	로드 평균(분)	프로세스	os.loadAverageMinute.one	마지막 1분 동안 CPU 시간을 요청한 프로세스 수
15	로드 평균(분)	프로세스	os.loadAverageMinute.fifteen	마지막 15분 동안 CPU 시간을 요청한 프로세스 수
5	로드 평균(분)	프로세스	os.loadAverageMinute.five	마지막 5분 동안 CPU 시간을 요청한 프로세스 수
캐시됨	Swap	KB	os.swap.cached	캐시 메모리로 사용된 스왑 메모리의 양(KB)
무료	Swap	KB	os.swap.free	사용 가능한 스왑 메모리 양(KB)
있음	Swap	KB	os.swap.in	디스크에서 스왑된 메모리 양(KB)
Out	Swap	KB	os.swap.out	디스크로 스왑된 총 메모리 양(KB)

카운터	유형	Unit	지표	설명
합계	Swap	KB	os.swap.total	사용 가능한 총 스왑 메모리의 양(킬로바이트 단위)입니다.
최대 파일	파일 시스템	파일	os.fileSys.maxFiles	파일 시스템에 대해 생성될 수 있는 최대 파일 수
사용된 파일	파일 시스템	파일	os.fileSys.usedFiles	파일 시스템의 파일 수
사용된 파일(%)	파일 시스템	파일	os.fileSys.usedFilePercent	사용 중인 사용 가능한 파일의 비율
사용 비율	파일 시스템	백분율	os.fileSys.usedPercent	사용 중인 파일 시스템 디스크 공간의 비율
사용됨	파일 시스템	KB	os.fileSys.used	파일 시스템에서 파일에 사용된 디스크 공간의 양(KB)
합계	파일 시스템	KB	os.fileSys.total	파일 시스템에 사용 가능한 총 디스크 공간(KB)
Rx	Network	초당 바이트	os.network.rx	초당 수신된 바이트 수
Tx	Network	초당 바이트	os.network.tx	초당 업로드된 바이트 수

카운터	유형	Unit	지표	설명
ACU 사용률	일반	백분율	os.general.acuUtilization	구성된 최대 용량에서 현재 용량의 백분율입니다.
최대 구성 ACU	일반	ACU	os.general.maxConfiguredAcu	사용자가 구성한 최대 용량으로, 단위는 Aurora 용량 단위(ACU)입니다.
최소 구성 ACU	일반	ACU	os.general.minConfiguredAcu	사용자가 구성한 최소 용량(ACU 단위)입니다.
Num VCPU	일반	vCPU	os.general.numVCPUs	DB 인스턴스의 가상 CPU(vCPU) 수
서버리스 데이터베이스 용량	일반	ACU	os.general.serverlessDatabaseCapacity	인스턴스의 현재 용량(ACU 단위)입니다.

Amazon RDS for MariaDB 및 MySQL에 대한 성능 개선 도우미 카운터

Amazon RDS for MariaDB 및 MySQL에 대한 성능 개선 도우미에서 다음 데이터베이스 카운터를 사용할 수 있습니다.

주제

- RDS for MariaDB 및 RDS for MySQL용 기본 카운터
- Amazon RDS for MariaDB 및 MySQL에 대한 기본이 아닌 카운터

RDS for MariaDB 및 RDS for MySQL용 기본 카운터

기본 지표는 Amazon RDS가 아닌 데이터베이스 엔진에 의해 정의됩니다. 이러한 기본 지표에 대한 정의는 MySQL 설명서의 Server Status Variables(8.0의 경우) 및 Server Status Variables(8.4의 경우)을 참조하세요.

카운터	유형	Unit	측정치
Com_analyze	SQL	초당 쿼리 수	db.SQL.Com_analyze
Com_optimize	SQL	초당 쿼리 수	db.SQL.Com_optimize
Com_select	SQL	초당 쿼리 수	db.SQL.Com_select
Connections	SQL	MySQL 서버에 대한 분당 연결 시도 횟수(성공 또는 실패)	db.Users.Connections
Innodb_rows_deleted	SQL	초당 행	db.SQL.Innodb_rows_deleted
Innodb_rows_inserted	SQL	초당 행	db.SQL.Innodb_rows_inserted
Innodb_rows_read	SQL	초당 행	db.SQL.Innodb_rows_read
Innodb_rows_updated	SQL	초당 행	db.SQL.Innodb_rows_updated
Select_full_join	SQL	초당 쿼리 수	db.SQL.Select_full_join
Select_full_range_join	SQL	초당 쿼리 수	db.SQL.Select_full_range_join
Select_range	SQL	초당 쿼리 수	db.SQL.Select_range
Select_range_check	SQL	초당 쿼리 수	db.SQL.Select_range_check
Select_scan	SQL	초당 쿼리 수	db.SQL.Select_scan
Slow_queries	SQL	초당 쿼리 수	db.SQL.Slow_queries
Sort_merge_passes	SQL	초당 쿼리 수	db.SQL.Sort_merge_passes

카운터	유형	Unit	측정치
Sort_range	SQL	초당 쿼리 수	db.SQL.Sort_range
Sort_rows	SQL	초당 쿼리 수	db.SQL.Sort_rows
Sort_scan	SQL	초당 쿼리 수	db.SQL.Sort_scan
질문	SQL	초당 쿼리 수	db.SQL.Questions
Innodb_row_lock_time	잠금	밀리초(평균)	db.Locks.Innodb_row_lock_time
Table_locks_immediate	잠금	초당 요청	db.Locks.Table_locks_immediate
Table_locks_waited	잠금	초당 요청	db.Locks.Table_locks_waited
Aborted_clients	Users	Connections	db.Users.Aborted_clients
Aborted_connects	Users	Connections	db.Users.Aborted_connects
max_connections	Users	연결	db.User.Max_Connections
Threads_created	Users	Connections	db.Users.Threads_created
Threads_running	Users	Connections	db.Users.Threads_running
Innodb_data_writes	I/O	초당 연산 수	db.IO.Innodb_data_writes
Innodb_dblwr_writes	I/O	초당 연산 수	db.IO.Innodb_dblwr_writes
Innodb_log_write_requests	I/O	초당 연산 수	db.IO.Innodb_log_write_requests
Innodb_log_writes	I/O	초당 연산 수	db.IO.Innodb_log_writes
Innodb_pages_written	I/O	초당 페이지	db.IO.Innodb_pages_written
Created_tmp_disk_tables	Temp	초당 테이블	db.Temp.Created_tmp_disk_tables
Created_tmp_tables	Temp	초당 테이블	db.Temp.Created_tmp_tables

카운터	유형	Unit	측정치
Innodb_buffer_pool_pages_data	Cache	페이지	db.Cache.Innodb_buffer_pool_pages_data
Innodb_buffer_pool_pages_total	Cache	페이지	db.Cache.Innodb_buffer_pool_pages_total
Innodb_buffer_pool_read_requests	Cache	초당 페이지	db.Cache.Innodb_buffer_pool_read_requests
Innodb_buffer_pool_reads	Cache	초당 페이지	db.Cache.Innodb_buffer_pool_reads
Opened_tables	Cache	테이블	db.Cache.Opened_tables
Opened_table_definitions	Cache	테이블	db.Cache.Opened_table_definitions
Qcache_hits	Cache	쿼리	db.Cache.Qcache_hits

Amazon RDS for MariaDB 및 MySQL에 대한 기본이 아닌 카운터

기본이 아닌 카운터 지표는 Amazon RDS가 정의하는 카운터입니다. 기본이 아닌 지표는 특정 쿼리를 통해 얻는 지표일 수 있습니다. 기본이 아닌 지표는 파생 지표일 수 있습니다. 이 경우 비율, 적중률 또는 지연 시간에 대한 계산 시 2개 이상의 기본 카운터가 사용됩니다.

카운터	유형	Unit	지표	설명	정의
innodb_buffer_pool_hits	Cache	읽기	db.Cache.innoDB_buffer_pool_hits	InnoDB가 버퍼 풀에서 충족할 수 있었던 읽기의 수입니다.	`innodb_buffer_pool_read_requests - innodb_buffer_pool_reads`

카운터	유형	Unit	지표	설명	정의
innodb_buffer_pool_hit_rate	Cache	백분율	db.Cache.innoDB_buffer_pool_hit_rate	InnoDB가 버퍼 풀에서 InnoDB가 충족할 수 있었던 읽기의 비율입니다.	`100 * innodb_buffer_pool_read_requests / (innodb_buffer_pool_read_requests + innodb_buffer_pool_reads)`

카운터	유형	Unit	지표	설명	정의
innodb_buffer_pool_usage	Cache	백분율	db.Cache.innoDB_buffer_pool_usage	데이터(페이지)를 포함하는 InnoDB 버퍼 풀의 비율입니다. Note 압축된 테이블을 사용하는 경우 이 값은 달라질 수 있습니다. 자	`Innodb_buffer_pool_pages_data / Innodb_buffer_pool_pages_total * 100.0`

카운터	유형	Unit	지표	설명	정의
				세한 내용은 MySQl 설명서의 Server Status Variab (8.0의 경우) 및 Server Status Variab (8.4의 경우)에서 Innodffer_pageta 및 Innodffer_page	

카운터	유형	Unit	지표	설명	정의
				tal 대한 정보를 참조하세요.	0
query_cache_hit_rate	Cache	백분율	db.Cache.query_cache_hit_rate	MySQL 결과 집합 캐시(쿼리 캐시) 적중률입니다.	`Qcache_hits / (QCache_hits + Com_select) * 100`
innodb_datafile_writes_to_disk	I/O	쓰기	db.IO.innoDB_datafile_writes_to_disk	디스크에 대한 InnoDB 데이터 파일 쓰기의 수(이중 쓰기 및 재실행 로깅 쓰기 연산은 제외)입니다.	`Innodb_data_writes - Innodb_log_writes - Innodb_dblwr_writes`

카운터	유형	Unit	지표	설명	정의
innodb_rows_changed	SQL	행	db.SQL.innodb_rows_changed	총 InnoDB 행 연산입니다.	db.SQL.Innodb_rows_inserted + db.SQL.Innodb_rows_deleted + db.SQL.Innodb_rows_updated
active_transactions	트랜잭션	트랜잭션	db.Transactions.active_transactions	총 활성 트랜잭션입니다.	SELECT COUNT(1) AS active_transactions FROM INFORMATION_SCHEMA.INNODB_TRX

카운터	유형	Unit	지표	설명	정의
trx_rseg_history_len	트랜잭션	없음	db.Transactions.trx_rseg_history_len	다중 버전 동시성 제어를 구현하기 위해 InnoDB 트랜잭션 시스템에서 유지 관리하는 커밋된 트랜잭션의 실행 취소 로그 페이지 목록입니다. 로그 레코드 실행 취소 세부 정보에 대한 자세한 내용은 MySQL 설명서의 InnoDB Multi-Versioning(8.0의 경우) 및 InnoDB Multi-Versioning(8.4의 경우)을 참조하세요.	`SELECT COUNT AS trx_rseg_history_len FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME='trx_rseg_history_len'`

카운터	유형	Unit	지표	설명	정의
innodb_deadlocks	잠금	잠금	db.Locks.innodb_deadlocks	교착 상태의 총 개수입니다.	`SELECT COUNT AS innodb_deadlocks FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME='lock_deadlocks'`
innodb_lock_timeouts	잠금	잠금	db.Locks.innodb_lock_timeouts	시간을 초과한 총 잠금 수입니다.	`SELECT COUNT AS innodb_lock_timeouts FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME='lock_timeouts'`
innodb_row_lock_waits	잠금	잠금	db.Locks.innodb_row_lock_waits	대기의 원인이 된 행 잠금의 총 개수입니다.	`SELECT COUNT AS innodb_row_lock_waits FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME='lock_row_lock_waits'`

Amazon RDS for Microsoft SQL Server용 성능 개선 도우미 카운터

RDS for Microsoft SQL Server용 성능 개선 도우미에서는 다음 데이터베이스 카운터를 사용할 수 있습니다.

RDS for Microsoft SQL Server용 기본 카운터

기본 지표는 Amazon RDS가 아닌 데이터베이스 엔진에 의해 정의됩니다. Microsoft SQL Server 설명서의 Use SQL Server Objects에서 이 기본 지표의 정의를 볼 수 있습니다.

카운터	유형	Unit	측정치
전달된 레코드	액세스 메서드	초당 레코드 수	db.Access Methods.Forwarded Records
페이지 분할	액세스 메서드	초당 분할 수	db.Access Methods.Page Splits
Buffer 캐시 적중률	버퍼 관리자	비율	db.Buffer Manager.Buffer cache hit ratio
페이지 예상 수명	버퍼 관리자	초 단위 예상 수명	db.Buffer Manager.Page life expectancy
페이지 조회	버퍼 관리자	초당 조회 수	db.Buffer Manager.Page lookups
페이지 읽기 수	버퍼 관리자	초당 읽기 수	db.Buffer Manager.Page reads
페이지 쓰기 수	버퍼 관리자	초당 쓰기 수	db.Buffer Manager.Page writes
활성 트랜잭션	데이터베이스	트랜잭션	db.Databases.Active Transactions (_Total)
플러시된 로그 바이트 수	데이터베이스	초당 플러시된 바이트 수	db.Databases.Log Bytes Flushed (_Total)

카운터	유형	Unit	측정치
로그 플러시 대기 시간	데이터베이스	초당 대기 시간	db.Databases.Log Flush Waits (_Total)
로그 플러시	데이터베이스	초당 플러시 수	db.Databases.Log Flushes (_Total)
쓰기 트랜잭션	데이터베이스	초당 트랜잭션 수	db.Databases.Write Transactions (_Total)
차단된 프로세스	일반 통계	차단된 프로세스	db.General Statistics.Processes blocked
사용자 연결	일반 통계	Connections	db.General Statistics.User Connections
래치 대기 시간	래치	초당 대기 시간	db.Latches.Latch Waits
교착 상태의 수	잠금	초당 교착 상태의 수	db.Locks.Number of Deadlocks (_Total)
보류 중인 메모리 부여	메모리 관리자	메모리 부여	db.Memory Manager.Memory Grants Pending
배치 요청	SQL 통계	초당 요청	db.SQL Statistics.Batch Requests
SQL 컴파일	SQL 통계	초당 컴파일 수	db.SQL Statistics.SQL Compilations
SQL 재컴파일 수	SQL 통계	초당 재컴파일 수	db.SQL Statistics.SQL Re-Compilations

Amazon RDS for Oracle의 성능 개선 도우미 카운터

RDS for Oracle용 성능 개선 도우미에서는 다음 데이터베이스 카운터를 사용할 수 있습니다.

RDS for Oracle용 기본 카운터

기본 지표는 Amazon RDS가 아닌 데이터베이스 엔진에 의해 정의됩니다. Oracle 설명서의 통계 설명에서 이러한 기본 지표에 대한 정의를 확인하실 수 있습니다.

> **Note**
>
> `CPU used by this session` 카운터 지표의 경우 단위가 기본 100분의 1초에서 활성 세션으로 변환되어 값을 사용하기가 더 쉬워졌습니다. 예를 들어 DB 로드 차트의 CPU 전송은 CPU에 대한 수요를 나타냅니다. 카운터 지표 `CPU used by this session`은 Oracle 세션의 CPU 사용량을 나타냅니다. CPU 전송을 `CPU used by this session` 카운터 지표와 비교할 수 있습니다. CPU에 대한 수요가 사용된 CPU보다 높은 경우 세션은 CPU 시간을 기다리고 있습니다.

카운터	유형	Unit	측정치
이 세션에서 사용한 CPU	User	활성 세션	이 세션에서 사용한 db.User.CPU
클라이언트에게로 또는 클라이언트로의 SQL*Net 왕복	User	초당 왕복	클라이언트에게로 또는 클라이언트로의 db.User.SQL*Net 왕복
클라이언트에게서 SQL*Net을 통해 수신한 바이트 수	User	초당 바이트	클라이언트에게서 SQL*Net을 통해 수신한 db.User.bytes
사용자 커밋	User	초당 커밋 수	db.User.bytes 커밋
누적 로그온 수	User	초당 로그온 수	누적 db.User.logons
사용자 호출	User	초당 호출 수	db.User.bytes 호출
SQL*Net을 통해 클라이언트에게 전송된 바이트	User	초당 바이트	SQL*Net을 통해 클라이언트에게 전송된 db.User.bytes

카운터	유형	Unit	측정치
사용자 롤백	User	초당 롤백 수	db.User.user 롤백
재실행 크기	Redo	초당 바이트	db.Redo.redo 크기
구문 분석 개수(합계)	SQL	초당 구문 분석 수	db.SQL.parse 개수(합계)
구문 분석 개수(하드)	SQL	초당 구문 분석 수	db.SQL.parse 개수(하드)
획득한 테이블 스캔 행	SQL	초당 행	획득한 db.SQL.table 스캔 행
정렬(메모리)	SQL	초당 정렬	db.SQL.sorts(메모리)
정렬(디스크)	SQL	초당 정렬	db.SQL.sorts(디스크)
정렬(행)	SQL	초당 정렬	db.SQL.sorts(행)
물리적 읽기 바이트 수	Cache	초당 바이트	db.Cache.physical 읽기 바이트 수
DB 블록 GET	Cache	초당 블록 수	db.Cache.db 블록 GET
DBWR 체크포인트	Cache	분당 체크포인트	db.Cache.DBWR 체크포인트
물리적 읽기 수	Cache	초당 읽기 수	db.Cache.physical 읽기
캐시에서 일관된 GET	Cache	초당 GET의 수	캐시에서 db.Cache.consistent GET
캐시의 DB 블록 GET	Cache	초당 GET의 수	캐시의 db.Cache.db 블록 GET

카운터	유형	Unit	측정치
일관된 GET	Cache	초당 GET의 수	db.Cache.consistent GET

Amazon RDS for PostgreSQL용 성능 개선 도우미 카운터

Amazon RDS for PostgreSQL용 성능 개선 도우미에서는 다음 데이터베이스 카운터를 사용할 수 있습니다.

주제

- Amazon RDS for PostgreSQL용 기본 카운터
- Amazon RDS for PostgreSQL용 비-기본 카운터

Amazon RDS for PostgreSQL용 기본 카운터

기본 지표는 Amazon RDS가 아닌 데이터베이스 엔진에 의해 정의됩니다. PostgreSQL 설명서의 통계 보기에서 이러한 기본 지표에 대한 정의를 확인하실 수 있습니다.

카운터	유형	Unit	측정치
blks_hit	Cache	초당 블록 수	db.Cache.blks_hit
buffers_alloc	Cache	초당 블록 수	db.Cache.buffers_alloc
buffers_checkpoint	체크포인트	초당 블록 수	db.Checkpoint.buffers_checkpoint
checkpoint_sync_time	체크포인트	체크포인트당 밀리초	db.Checkpoint.checkpoint_sync_time
checkpoint_write_time	체크포인트	체크포인트당 밀리초	db.Checkpoint.checkpoint_write_time
checkpoints_req	체크포인트	분당 체크포인트	db.Checkpoint.checkpoints_req

카운터	유형	Unit	측정치
checkpoints_timed	체크포인트	분당 체크포인트	db.Checkpoint.checkpoints_timed
maxwritten_clean	체크포인트	분당 Bgwriter 클린 스톱	db.Checkpoint.maxwritten_clean
deadlocks	동시성	분당 교착 상태의 수	db.Concurrency.deadlocks
blk_read_time	I/O	밀리초	db.IO.blk_read_time
blks_read	I/O	초당 블록 수	db.IO.blks_read
buffers_backend	I/O	초당 블록 수	db.IO.buffers_backend
buffers_backend_fsync	I/O	초당 블록 수	db.IO.buffers_backend_fsync
buffers_clean	I/O	초당 블록 수	db.IO.buffers_clean
tup_deleted	SQL	초당 튜플 수	db.SQL.tup_deleted
tup_fetched	SQL	초당 튜플 수	db.SQL.tup_fetched
tup_inserted	SQL	초당 튜플 수	db.SQL.tup_inserted
tup_returned	SQL	초당 튜플 수	db.SQL.tup_returned
tup_updated	SQL	초당 튜플 수	db.SQL.tup_updated
temp_bytes	Temp	초당 바이트	db.Temp.temp_bytes
temp_files	Temp	분당 파일 수	db.Temp.temp_files
xact_commit	트랜잭션	초당 커밋 수	db.Transactions.xact_commit
xact_rollback	트랜잭션	초당 롤백 수	db.Transactions.xact_rollback
numbackends	User	Connections	db.User.numbackends

카운터	유형	Unit	측정치
archived_count	Write Ahead Log(WAL)	분당 파일 수	db.WAL.archived_count

Amazon RDS for PostgreSQL용 비-기본 카운터

기본이 아닌 카운터 지표는 Amazon RDS가 정의하는 카운터입니다. 기본이 아닌 지표는 특정 쿼리를 통해 얻는 지표일 수 있습니다. 기본이 아닌 지표는 파생 지표일 수 있습니다. 이 경우 비율, 적중률 또는 지연 시간에 대한 계산 시 2개 이상의 기본 카운터가 사용됩니다.

카운터	유형	Unit	지표	설명	정의
checkpoint_sync_latency	체크포인트	db.Checkpoint.checkpoint_sync_latency		파일이 디스크에 동기화되는 체크포인트 처리 중 일부에서 소요된 총 시간입니다.	`checkpoint_sync_time / (checkpoints_timed + checkpoints_req)`
checkpoint_write_latency	체크포인트	db.Checkpoint.checkpoint_write_latency	파일이 디스크에 쓰이는 체크포인트 처리 중 일부에서 소요된 총 시간입니다.	`checkpoint_write_time / (checkpoints_timed + checkpoints_req)`	
read_latency	I/O	db.IO.rd_later	이 인스턴스의 백엔드에서 데이터 파일 블록을 읽는 데 소요된 시간입니다.	`blk_read_time / blks_read`	

카운터	유형	Unit	지표	설명	정의
idle_in_transaction_aborted_count	상태	세션	db.state.idle_in_transaction_aborted_count	`idle in transaction (aborted)` 상태의 세션 수	해당 사항 없음
idle_in_transaction_count	상태	세션	db.state.idle_in_transaction_count	`idle in transaction` 상태의 세션 수	해당 사항 없음
idle_in_transaction_max_time	State	초	db.state.idle_in_transaction_max_time	`idle in transaction` 상태에서 가장 오래 실행되는 트랜잭션의 지속 시간(초)	해당 사항 없음
active_transactions	트랜잭션	트랜잭션	db.Transactions.active_transactions	활성 트랜잭션의 수	해당 사항 없음
blocked_transactions	트랜잭션	트랜잭션	db.Transactions.blocked_transactions	차단된 트랜잭션의 수	해당 사항 없음
oldest_active_logical_replication_slot_xid_age	트랜잭션	db.Transactions.dest_ave_logl_replication_sl_xid_a	활성 논리 복제 슬롯에서 가장 오래된 트랜잭션의 기간입니다. 자세한 내용은 논리적 복제 슬롯 섹션을 참조하세요.	-	

카운터	유형	Unit	지표	설명	정의
oldest_inactive_logical_replication_slot_xid_age	트랜잭션	db.Transactions.oldest_inactive_logical_replication_slot_xid_age	비활성 논리 복제 슬롯에서 가장 오래된 트랜잭션의 기간입니다. 자세한 내용은 논리적 복제 슬롯 섹션을 참조하세요.	-	
oldest_prepared_transaction_xid_age	트랜잭션	db.Transactions.oldest_prepared_transaction_xid_age	가장 오래된 준비된 트랜잭션의 기간입니다. 자세한 내용은 준비된 트랜잭션 섹션을 참조하세요.	-	
oldest_running_transaction_xid_age	트랜잭션	db.Transactions.oldest_running_transaction_xid_age	실행 중인 트랜잭션 중 가장 오래된 트랜잭션의 기간입니다. 자세한 내용은 가장 오랫동안 실행 중인 활성 트랜잭션의 경우 활성 문를, 가장 오래 실행 중인 유휴 트랜잭션의 경우 트랜잭션의 유휴 상태을 참조하세요.	-	

카운터	유형	Unit	지표	설명	정의
oldest_hot_standby_feedback_xid_age	트랜잭션	db.Transactions.oldest_hot_standby_feedback_xid_age	hot_standby_feedback 이 활성화된 읽기 전용 복제본에서 가장 오랫동안 실행 중인 트랜잭션의 연령입니다. 자세한 내용은 읽기 전용 복제본 섹션을 참조하세요.	-	
max_used_xact_ids	트랜잭션	트랜잭션	db.Transactions.max_used_xact_ids	vacuum되지 않은 트랜잭션의 수	해당 사항 없음
max_connections	Users	연결	db.User.Max_Connections	max_connections 파라미터에 구성된 DB 인스턴스에 허용되는 최대 연결 수	해당 사항 없음
archive_failed_count	WAL	분당 파일 수	db.WAL.archive_failed_count	WAL 파일 보관 실패 횟수(분당 파일 수)	해당 사항 없음

성능 개선 도우미에 대한 SQL 통계

SQL 통계(SQL statistics)는 성능 개선 도우미에서 수집하는 SQL 쿼리에 대한 성능 관련 지표입니다. 성능 개선 도우미는 쿼리가 실행되는 초당 통계와 각 SQL 호출에 대한 통계를 수집합니다. SQL 통계는 선택한 시간 범위의 평균입니다.

SQL 다이제스트는 주어진 패턴을 갖지만 반드시 동일한 리터럴 값을 가질 필요는 없는 모든 쿼리의 합성입니다. 다이제스트는 리터럴 값을 물음표로 바꿉니다. 예: `SELECT * FROM emp WHERE lname= ?`. 이 다이제스트에는 다음 하위 쿼리가 구성될 수 있습니다.

```
SELECT * FROM emp WHERE lname = 'Sanchez'
SELECT * FROM emp WHERE lname = 'Olagappan'
```

```
SELECT * FROM emp WHERE lname = 'Wu'
```

모든 엔진은 다이제스트 쿼리에 대한 SQL 통계를 지원합니다.

이 기능에 대한 리전, DB 엔진 및 인스턴스 클래스 지원 정보는 성능 개선 도우미 기능에 대한 Amazon RDS DB 엔진, 리전 및 인스턴스 클래스 지원 섹션을 참조하세요.

주제

- MariaDB 및 MySQL에 대한 SQL 통계
- Amazon RDS for Oracle에 대한 SQL 통계
- Amazon RDS for SQL Server에 대한 SQL 통계
- RDS PostgreSQL에 대한 SQL 통계

MariaDB 및 MySQL에 대한 SQL 통계

MariaDB 및 MySQL은 다이제스트 수준에서만 SQL 통계를 수집합니다. 명령문 수준에는 통계가 표시되지 않습니다.

주제

- MariaDB 및 MySQL에 대한 다이제스트 통계
- MariaDB 및 MySQL에 대한 초당 통계
- MariaDB 및 MySQL에 대한 호출당 통계
- MariaDB 및 MySQL에 대한 기본 통계

MariaDB 및 MySQL에 대한 다이제스트 통계

성능 개선 도우미는 `events_statements_summary_by_digest` 테이블에서 SQL 다이제스트 통계를 수집합니다. `events_statements_summary_by_digest` 테이블은 데이터베이스에 의해 관리됩니다.

다이제스트 테이블에는 제거 정책이 없습니다. 테이블이 가득 차면 AWS Management Console에 다음 메시지가 표시됩니다.

```
Performance Insights is unable to collect SQL Digest statistics on new queries because
 the table events_statements_summary_by_digest is full.
Please truncate events_statements_summary_by_digest table to clear the issue. Check the
 User Guide for more details.
```

이 상황에서는 MariaDB 및 MySQL은 SQL 쿼리를 추적하지 않습니다. 이 문제를 해결하기 위해 성능 개선 도우미는 다음 조건이 모두 충족되면 자동으로 다이제스트 테이블을 자릅니다.

- 테이블이 꽉 찼습니다.
- 성능 개선 도우미가 성능 스키마를 자동으로 관리합니다.

  자동으로 관리하려면 `performance_schema` 파라미터를 `0`으로 설정하고 [소스(Source)]를 `user` 이외의 값으로 설정해야 합니다. 성능 개선 도우미가 성능 스키마를 자동으로 관리하지 않는 경우 Amazon RDS for MariaDB 또는 MySQL에서 성능 개선 도우미의 성능 스키마 개요 섹션을 참조하세요.

AWS CLI에서 describe-db-parameters 명령을 실행하여 파라미터 값의 소스를 확인합니다.

MariaDB 및 MySQL에 대한 초당 통계

MariaDB 및 MySQL DB 인스턴스에 다음 SQL 통계를 사용할 수 있습니다.

측정치	Unit
db.sql_tokenized.stats.count_star_per_sec	초당 호출 수
db.sql_tokenized.stats.sum_timer_wait_per_sec	초당 평균 지연 시간(단위: ms)
db.sql_tokenized.stats.sum_select_full_join_per_sec	초당 전체 조인 선택
db.sql_tokenized.stats.sum_select_range_check_per_sec	초당 범위 검사 선택
db.sql_tokenized.stats.sum_select_scan_per_sec	초당 스캔 선택
db.sql_tokenized.stats.sum_sort_merge_passes_per_sec	초당 병합 패스 정렬
db.sql_tokenized.stats.sum_sort_scan_per_sec	초당 스캔 정렬
db.sql_tokenized.stats.sum_sort_range_per_sec	초당 범위 정렬

측정치	Unit
db.sql_tokenized.stats.sum_sort_rows_per_sec	초당 행 정렬
db.sql_tokenized.stats.sum_rows_affected_per_sec	초당 영향을 받는 행
db.sql_tokenized.stats.sum_rows_examined_per_sec	초당 검사된 행
db.sql_tokenized.stats.sum_rows_sent_per_sec	초당 전송된 행
db.sql_tokenized.stats.sum_created_tmp_disk_tables_per_sec	초당 생성된 임시 디스크 테이블
db.sql_tokenized.stats.sum_created_tmp_tables_per_sec	초당 생성된 임시 테이블
db.sql_tokenized.stats.sum_lock_time_per_sec	초당 잠금 시간(ms)

MariaDB 및 MySQL에 대한 호출당 통계

다음 지표에서는 SQL 문의 호출당 통계를 제공합니다.

측정치	Unit
db.sql_tokenized.stats.sum_timer_wait_per_call	호출당 평균 지연 시간(단위: ms)
db.sql_tokenized.stats.sum_select_full_join_per_call	호출당 전체 조인 선택
db.sql_tokenized.stats.sum_select_range_check_per_call	호출당 범위 검사 선택
db.sql_tokenized.stats.sum_select_scan_per_call	호출당 스캔 선택
db.sql_tokenized.stats.sum_sort_merge_passes_per_call	호출당 병합 패스 정렬

측정치	Unit
db.sql_tokenized.stats.sum_sort_scan_per_call	호출당 스캔 정렬
db.sql_tokenized.stats.sum_sort_range_per_call	호출당 범위 정렬
db.sql_tokenized.stats.sum_sort_rows_per_call	호출당 행 정렬
db.sql_tokenized.stats.sum_rows_affected_per_call	호출당 영향을 받는 행
db.sql_tokenized.stats.sum_rows_examined_per_call	호출당 검사된 행
db.sql_tokenized.stats.sum_rows_sent_per_call	호출당 전송된 행
db.sql_tokenized.stats.sum_created_tmp_disk_tables_per_call	호출당 생성된 임시 디스크 테이블
db.sql_tokenized.stats.sum_created_tmp_tables_per_call	호출당 생성된 임시 테이블
db.sql_tokenized.stats.sum_lock_time_per_call	호출당 잠금 시간(ms)

MariaDB 및 MySQL에 대한 기본 통계

MariaDB 및 MySQL DB 인스턴스에 다음 SQL 통계를 사용할 수 있습니다.

측정치	Unit
db.sql_tokenized.stats.count_star	호출
db.sql_tokenized.stats.sum_timer_wait	대기 시간(밀리초)
db.sql_tokenized.stats.sum_select_full_join	전체 조인 선택
db.sql_tokenized.stats.sum_select_range_check	범위 확인 선택
db.sql_tokenized.stats.sum_select_scan	스캔 선택

측정치	Unit
db.sql_tokenized.stats.sum_sort_merge_passes	병합 전달 정렬
db.sql_tokenized.stats.sum_sort_scan	스캔 정렬
db.sql_tokenized.stats.sum_sort_range	정렬 범위
db.sql_tokenized.stats.sum_sort_rows	행 정렬
db.sql_tokenized.stats.sum_rows_affected	영향을 받는 행
db.sql_tokenized.stats.sum_rows_examined	검사된 행
db.sql_tokenized.stats.sum_rows_sent	전송된 행
db.sql_tokenized.stats.sum_created_tmp_disk_tables	만들어진 임시 디스크 테이블
db.sql_tokenized.stats.sum_created_tmp_tables	만들어진 임시 테이블
db.sql_tokenized.stats.sum_lock_time	잠금 시간(밀리초)

Amazon RDS for Oracle에 대한 SQL 통계

Amazon RDS for Oracle은 명령문과 다이제스트 수준에서 SQL 통계를 수집합니다. 명령문 수준에서 ID 열은 `V$SQL.SQL_ID` 값을 나타냅니다. 다이제스트 수준에서 ID 열에는 `V$SQL.FORCE_MATCHING_SIGNATURE` 값이 표시됩니다.

다이제스트 레벨에서 ID가 `0`인 경우, Oracle Database는 이 문이 재사용에 적합하지 않다고 판단했습니다. 이 경우 하위 SQL 문은 서로 다른 다이제스트에 속할 수 있습니다. 그러나 문은 수집된 첫 번째 SQL 문에 대해 `digest_text` 아래에 함께 그룹화됩니다.

주제

- Oracle의 초당 통계
- Oracle의 호출당 통계
- Oracle에 대한 기본 통계

Oracle의 초당 통계

다음 지표는 Oracle SQL 쿼리에 대한 초당 통계를 제공합니다.

지표	Unit
db.sql.stats.executions_per_sec	초당 실행 수
db.sql.stats.elapsed_time_per_sec	평균 활성 실행(AAE)
db.sql.stats.rows_processed_per_sec	초당 처리된 행
db.sql.stats.buffer_gets_per_sec	초당 버퍼 GET의 수
db.sql.stats.physical_read_requests_per_sec	초당 물리적 읽기 수
db.sql.stats.physical_write_requests_per_sec	초당 물리적 쓰기 수
db.sql.stats.total_sharable_mem_per_sec	초당 공유 가능 메모리 합계(단위: 바이트)
db.sql.stats.cpu_time_per_sec	초당 CPU 시간(단위: ms)

다음 지표는 Oracle SQL 다이제스트 쿼리에 대한 초당 통계를 제공합니다.

지표	Unit
db.sql_tokenized.stats.executions_per_sec	초당 실행 수
db.sql_tokenized.stats.elapsed_time_per_sec	평균 활성 실행(AAE)
db.sql_tokenized.stats.rows_processed_per_sec	초당 처리된 행
db.sql_tokenized.stats.buffer_gets_per_sec	초당 버퍼 GET의 수
db.sql_tokenized.stats.physical_read_requests_per_sec	초당 물리적 읽기 수
db.sql_tokenized.stats.physical_write_requests_per_sec	초당 물리적 쓰기 수

지표	Unit
db.sql_tokenized.stats.total_sharable_mem_per_sec	초당 공유 가능 메모리 합계(단위: 바이트)
db.sql_tokenized.stats.cpu_time_per_sec	초당 CPU 시간(단위: ms)

Oracle의 호출당 통계

다음 지표에서는 Oracle SQL 문의 호출당 통계를 제공합니다.

지표	Unit
db.sql.stats.elapsed_time_per_exec	실행당 경과 시간(단위: ms)
db.sql.stats.rows_processed_per_exec	실행당 처리된 행
db.sql.stats.buffer_gets_per_exec	실행당 버퍼 GET의 수
db.sql.stats.physical_read_requests_per_exec	실행당 물리적 읽기 수
db.sql.stats.physical_write_requests_per_exec	실행당 물리적 쓰기 수
db.sql.stats.total_sharable_mem_per_exec	실행당 공유 가능 메모리 합계(단위: 바이트)
db.sql.stats.cpu_time_per_exec	실행당 CPU 시간(단위: ms)

다음 지표에서는 Oracle SQL ,다이제스트 쿼리의 호출당 통계를 제공합니다.

지표	Unit
db.sql_tokenized.stats.elapsed_time_per_exec	실행당 경과 시간(단위: ms)
db.sql_tokenized.stats.rows_processed_per_exec	실행당 처리된 행
db.sql_tokenized.stats.buffer_gets_per_exec	실행당 버퍼 GET의 수

지표	Unit
db.sql_tokenized.stats.physical_read_requests_per_exec	실행당 물리적 읽기 수
db.sql_tokenized.stats.physical_write_requests_per_exec	실행당 물리적 쓰기 수
db.sql_tokenized.stats.total_sharable_mem_per_exec	실행당 공유 가능 메모리 합계(단위: 바이트)
db.sql_tokenized.stats.cpu_time_per_exec	실행당 CPU 시간(단위: ms)

Oracle에 대한 기본 통계

다음 지표는 Oracle SQL 쿼리에 대한 기본 통계를 제공합니다.

지표	Unit
db.sql.stats.executions	실행 수
db.sql.stats.elapsed_time	경과 시간(밀리초)
db.sql.stats.rows_processed	처리된 행
db.sql.stats.buffer_gets	버퍼 가져오기
db.sql.stats.physical_read_requests	물리적 읽기 수
db.sql.stats.physical_write_requests	물리적 쓰기
db.sql.stats.total_sharable_mem	공유 가능한 총 메모리(바이트)
db.sql.stats.cpu_time	CPU 시간(밀리초)

다음 지표는 Oracle SQL 다이제스트 쿼리에 대한 기본 통계를 제공합니다.

지표	Unit
db.sql_tokenized.stats.executions	실행 수
db.sql_tokenized.stats.elapsed_time	경과 시간(밀리초)
db.sql_tokenized.stats.rows_processed	처리된 행
db.sql_tokenized.stats.buffer_gets	버퍼 가져오기
db.sql_tokenized.stats.physical_read_requests	물리적 읽기 수
db.sql_tokenized.stats.physical_write_requests	물리적 쓰기
db.sql_tokenized.stats.total_sharable_mem	공유 가능한 총 메모리(바이트)
db.sql_tokenized.stats.cpu_time	CPU 시간(밀리초)

Amazon RDS for SQL Server에 대한 SQL 통계

Amazon RDS for SQL Server는 명령문과 다이제스트 수준에서 SQL 통계를 수집합니다. 명령문 수준에서 ID 열은 `sql_handle` 값을 나타냅니다. 다이제스트 수준에서 ID 열에는 `query_hash` 값이 표시됩니다.

SQL Server는 몇 개의 명령문에서 `query_hash`에 대해 NULL 값을 반환합니다. 예를 들어 ALTER INDEX, CHECKPOINT, UPDATE STATISTICS, COMMIT TRANSACTION, FETCH NEXT FROM Cursor 및 몇 개의 INSERT 명령문, SELECT @<variable>, 조건문, 실행 가능한 저장 프로시저 등이 이에 해당합니다. 이 경우 `sql_handle` 값은 해당 명령문의 다이제스트 수준에서 ID로 표시됩니다.

주제

- SQL Server의 초당 통계
- SQL Server의 호출당 통계
- SQL Server의 기본 통계

SQL Server의 초당 통계

다음 지표는 SQL Server SQL 쿼리에 대한 초당 통계를 제공합니다.

지표	Unit
db.sql.stats.execution_count_per_sec	초당 실행 수
db.sql.stats.total_elapsed_time_per_sec	초당 총 경과 시간
db.sql.stats.total_rows_per_sec	초당 처리된 총 행 수
db.sql.stats.total_logical_reads_per_sec	초당 총 논리적 읽기 수
db.sql.stats.total_logical_writes_per_sec	초당 총 논리적 쓰기 수
db.sql.stats.total_physical_reads_per_sec	초당 총 물리적 읽기 수
db.sql.stats.total_worker_time_per_sec	총 CPU 시간(밀리초)

다음 지표는 SQL Server SQL 다이제스트 쿼리에 대한 초당 통계를 제공합니다.

지표	Unit
db.sql_tokenized.stats.execution_count_per_sec	초당 실행 수
db.sql_tokenized.stats.total_elapsed_time_per_sec	초당 총 경과 시간
db.sql_tokenized.stats.total_rows_per_sec	초당 처리된 총 행 수
db.sql_tokenized.stats.total_logical_reads_per_sec	초당 총 논리적 읽기 수
db.sql_tokenized.stats.total_logical_writes_per_sec	초당 총 논리적 쓰기 수
db.sql_tokenized.stats.total_physical_reads_per_sec	초당 총 물리적 읽기 수
db.sql_tokenized.stats.total_worker_time_per_sec	총 CPU 시간(밀리초)

SQL Server의 호출당 통계

다음 지표에서는 SQL Server SQL 문의 호출당 통계를 제공합니다.

지표	Unit
db.sql.stats.total_elapsed_time_per_call	실행당 총 경과 시간(단위: ms)
db.sql.stats.total_rows_per_call	실행당 처리된 총 행 수
db.sql.stats.total_logical_reads_per_call	db.sql_tokenized.stats.total_rows_per_sec
db.sql.stats.total_logical_writes_per_call	실행당 총 논리적 쓰기 수
db.sql.stats.total_physical_reads_per_call	실행당 총 물리적 읽기 수
db.sql.stats.total_worker_time_per_call	실행당 총 CPU 시간(단위: ms)

다음 지표는 SQL Server SQL 다이제스트 쿼리에 대한 호출당 통계를 제공합니다.

지표	Unit
db.sql_tokenized.stats.total_elapsed_time_per_call	실행당 총 경과 시간
db.sql_tokenized.stats.total_rows_per_call	실행당 처리된 총 행 수
db.sql_tokenized.stats.total_logical_reads_per_call	db.sql_tokenized.stats.total_rows_per_sec
db.sql_tokenized.stats.total_logical_writes_per_call	실행당 총 논리적 쓰기 수
db.sql_tokenized.stats.total_physical_reads_per_call	실행당 총 물리적 읽기 수
db.sql_tokenized.stats.total_worker_time_per_call	실행당 총 CPU 시간(단위: ms)

SQL Server의 기본 통계

다음 지표는 SQL Server SQL 쿼리에 대한 기본 통계를 제공합니다.

지표	Unit
db.sql.stats.execution_count	실행 수
db.sql.stats.total_elapsed_time	총 경과 시간(밀리초)
db.sql.stats.total_rows	처리된 총 행 수
db.sql.stats.total_logical_reads	총 논리적 읽기 수
db.sql.stats.total_logical_writes	총 논리적 쓰기 수
db.sql.stats.total_physical_reads	총 물리적 읽기 수
db.sql.stats.total_worker_time	총 CPU 시간(밀리초)

다음 지표는 SQL Server 다이제스트 쿼리에 대한 기본 통계를 제공합니다.

지표	Unit
db.sql_tokenized.stats.execution_count	실행 수
db.sql_tokenized.stats.total_elapsed_time	총 경과 시간(밀리초)
db.sql_tokenized.stats.total_rows	처리된 총 행 수
db.sql_tokenized.stats.total_logical_reads	총 논리적 읽기 수
db.sql_tokenized.stats.total_logical_writes	총 논리적 쓰기 수
db.sql_tokenized.stats.total_physical_reads	총 물리적 읽기 수
db.sql_tokenized.stats.total_worker_time	총 CPU 시간(밀리초)

RDS PostgreSQL에 대한 SQL 통계

성능 개선 도우미가 SQL 호출과 쿼리가 실행되는 초당 통계가 SQL 통계를 수집합니다. RDS for PostgreSQL은 다이제스트 수준에서만 SQL 통계를 수집합니다. 명령문 수준에는 통계가 표시되지 않습니다.

다음에서 RDS for PostgreSQL 다이제스트 수준 통계에 대한 정보를 찾을 수 있습니다.

주제

- RDS PostgreSQL에 대한 다이제스트 통계
- RDS PostgreSQL에 대한 초당 다이제스트 통계
- RDS PostgreSQL에 대한 호출당 다이제스트 통계
- RDS PostgreSQL에 대한 기본 통계

RDS PostgreSQL에 대한 다이제스트 통계

SQL 다이제스트 통계를 보려면 RDS PostgreSQL은 `pg_stat_statements` 라이브러리를 로드해야 합니다. PostgreSQL 11 이상과 호환되는 PostgreSQL DB 인스턴스의 경우 데이터베이스가 이 라이브러리를 기본값으로 로드합니다. PostgreSQL 10 이하와 호환되는 PostgreSQL DB 인스턴스의 경우 이 라이브러리를 수동으로 활성화합니다. 이 라이브러리를 수동으로 활성화하려면 DB 인스턴스와 연결된 DB 파라미터 그룹의 `pg_stat_statements`에 `shared_preload_libraries`를 추가하세요. 그런 다음 DB 인스턴스를 재부팅합니다. 자세한 내용은 Amazon RDS의 파라미터 그룹 섹션을 참조하세요.

> **Note**
>
> Performance Insights는 `pg_stat_activity`에서 잘리지 않은 쿼리에 대한 통계만 수집할 수 있습니다. 기본적으로 PostgreSQL 데이터베이스는 1,024바이트보다 긴 쿼리를 자릅니다. 쿼리 크기를 늘리려면 DB 인스턴스와 연결된 DB 파라미터 그룹에서 `track_activity_query_size` 파라미터를 변경합니다. 이 파라미터를 변경하면 DB 인스턴스를 재부팅해야 합니다.

RDS PostgreSQL에 대한 초당 다이제스트 통계

다음 SQL 다이제스트 통계는 PostgreSQL DB 인스턴스에 제공됩니다.

측정치	Unit
db.sql_tokenized.stats.calls_per_sec	초당 호출 수
db.sql_tokenized.stats.rows_per_sec	초당 행
db.sql_tokenized.stats.total_time_per_sec	초당 평균 활성 실행(AAE)
db.sql_tokenized.stats.shared_blks_hit_per_sec	초당 블록 히트 수
db.sql_tokenized.stats.shared_blks_read_per_sec	초당 블록 읽기 수
db.sql_tokenized.stats.shared_blks_dirtied_per_sec	초당 더티 블록 수
db.sql_tokenized.stats.shared_blks_written_per_sec	초당 블록 쓰기 수
db.sql_tokenized.stats.local_blks_hit_per_sec	초당 로컬 블록 히트 수
db.sql_tokenized.stats.local_blks_read_per_sec	초당 로컬 블록 읽기 수
db.sql_tokenized.stats.local_blks_dirtied_per_sec	초당 로컬 블록 더티 수
db.sql_tokenized.stats.local_blks_written_per_sec	초당 로컬 블록 쓰기 수
db.sql_tokenized.stats.temp_blks_written_per_sec	초당 임시 쓰기 수
db.sql_tokenized.stats.temp_blks_read_per_sec	초당 임시 읽기 수
db.sql_tokenized.stats.blk_read_time_per_sec	초당 평균 동시 읽기 수
db.sql_tokenized.stats.blk_write_time_per_sec	초당 평균 동시 쓰기 수

RDS PostgreSQL에 대한 호출당 다이제스트 통계

다음 지표에서는 SQL 문의 호출당 통계를 제공합니다.

측정치	Unit
db.sql_tokenized.stats.rows_per_call	호출당 행 수
db.sql_tokenized.stats.avg_latency_per_call	호출당 평균 지연 시간(단위: ms)
db.sql_tokenized.stats.shared_blks_hit_per_call	호출당 블록 히트 수
db.sql_tokenized.stats.shared_blks_read_per_call	호출당 블록 읽기 수
db.sql_tokenized.stats.shared_blks_written_per_call	호출당 블록 쓰기 수
db.sql_tokenized.stats.shared_blks_dirtied_per_call	통화 당 더티 블록 수
db.sql_tokenized.stats.local_blks_hit_per_call	호출당 로컬 블록 히트 수
db.sql_tokenized.stats.local_blks_read_per_call	호출당 로컬 블록 읽기 수
db.sql_tokenized.stats.local_blks_dirtied_per_call	호출당 로컬 블록 더티 수
db.sql_tokenized.stats.local_blks_written_per_call	호출당 로컬 블록 쓰기 수
db.sql_tokenized.stats.temp_blks_written_per_call	호출당 임시 블록 쓰기 수
db.sql_tokenized.stats.temp_blks_read_per_call	호출당 임시 블록 읽기 수
db.sql_tokenized.stats.blk_read_time_per_call	호출당 읽기 시간(단위: ms)
db.sql_tokenized.stats.blk_write_time_per_call	호출당 쓰기 시간(단위: ms)

RDS PostgreSQL에 대한 기본 통계

다음 SQL 통계는 PostgreSQL DB 인스턴스에서 사용 가능합니다.

지표	Unit
db.sql_tokenized.stats.calls	호출
db.sql_tokenized.stats.rows	행
db.sql_tokenized.stats.total_time	총 시간(밀리초)
db.sql_tokenized.stats.shared_blks_hit	블록 히트
db.sql_tokenized.stats.shared_blks_read	블록 읽기
db.sql_tokenized.stats.shared_blks_dirtied	더티 블록
db.sql_tokenized.stats.shared_blks_written	블록 쓰기
db.sql_tokenized.stats.local_blks_hit	로컬 블록 히트
db.sql_tokenized.stats.local_blks_read	로컬 블록 읽기
db.sql_tokenized.stats.local_blks_dirtied	로컬 블록 더티
db.sql_tokenized.stats.local_blks_written	로컬 블록 쓰기
db.sql_tokenized.stats.temp_blks_written	임시 쓰기
db.sql_tokenized.stats.temp_blks_read	임시 읽기
db.sql_tokenized.stats.blk_read_time	평균 동시 읽기(밀리초)
db.sql_tokenized.stats.blk_write_time	평균 동시 쓰기(밀리초)

이러한 지표에 대한 자세한 내용은 PostgreSQL 설명서의 pg_stat_statements를 참조하세요.

향상된 모니터링의 OS 지표

Amazon RDS는 DB 인스턴스가 실행되는 운영 체제(OS)에 대한 측정치를 실시간으로 제공합니다. RDS는 지표를 향상된 모니터링에서 Amazon CloudWatch Logs 계정으로 전달합니다. 다음 표에는 Amazon CloudWatch Logs에서 사용 가능한 OS 측정치가 나와 있습니다.

주제

- Db2, MariaDB, MySQL, Oracle 및 PostgreSQL에 대한 OS 지표
- Microsoft SQL Server DB 인스턴스에 대한 측정치

Db2, MariaDB, MySQL, Oracle 및 PostgreSQL에 대한 OS 지표

그룹	측정치	콘솔 이름	설명
General	engine	해당 사항 없음	DB 인스턴스에 대한 데이터베이스 엔진
	instanceID	해당 사항 없음	DB 인스턴스 식별자
	instanceResourceID	해당 사항 없음	AWS 리전에 고유한 DB 인스턴스의 변경 불가능한 식별자로, 로그 스트림 식별자로도 사용됩니다.
	numVCPUs	해당 사항 없음	DB 인스턴스의 가상 CPU 수
	timestamp	해당 사항 없음	측정치를 가져온 시간
	uptime	해당 사항 없음	DB 인스턴스가 활성 상태로 유지된 시간
	version	해당 사항 없음	OS 측정치 스트림 JSON 형식의 버전

그룹	측정치	콘솔 이름	설명
cpuUtilization	guest	CPU 게스트	게스트 프로그램에서 사용 중인 CPU의 비율
	idle	CPU 유휴	유휴 상태인 CPU의 비율
	irq	CPU IRQ	소프트웨어 인터럽트에서 사용 중인 CPU의 비율
	nice	CPU Nice	가장 낮은 우선순위로 실행 중인 프로그램에서 사용 중인 CPU의 비율
	steal	CPU Steal	다른 가상 머신에서 사용 중인 CPU의 비율
	system	CPU 시스템	커널에서 사용 중인 CPU의 비율
	total	CPU 합계	사용 중인 CPU의 총 비율입니다. 이 값에는 nice 값이 포함됩니다.
	user	CPU 사용자	사용자 프로그램에서 사용 중인 CPU의 비율
	wait	CPU 대기	I/O 액세스를 대기 중인 동안 사용되지 않은 CPU의 비율
diskIO	avgQueueLen	평균 대기열 크기	I/O 디바이스의 대기열에서 대기 중인 요청 수입니다.
	avgReqSz	평균 요청 크기	평균 요청 크기(KB)
	await	디스크 I/O 대기	대기열 시간과 서비스 시간을 포함하여 요청에 응답하는 데 필요한 시간(밀리초)
	device	해당 사항 없음	사용 중인 디스크 디바이스의 식별자
	readIOsPS	읽기 IO/초	초당 읽기 작업 수

그룹	측정치	콘솔 이름	설명
	readKb	읽기 합계	읽은 총 KB 수
	readKbPS	읽기 KB/초	초당 읽은 KB 수
	readLatency	읽기 지연 시간	읽기 I/O 요청을 제출하여 처리가 완료될 때까지 걸리는 시간(밀리초) 이 지표는 Amazon Aurora에만 사용할 수 없습니다.
	readThroughput	읽기 처리량	DB 클러스터에 요청할 때 사용되는 네트워크 처리량(초당 바이트) 이 지표는 Amazon Aurora에만 사용할 수 없습니다.
	rrqmPS	Rrqms	초당 대기 중인 병합 읽기 요청 수
	tps	TPS	초당 I/O 트랜잭션 수
	util	디스크 I/O 유틸리티	요청이 발급된 CPU 시간의 비율
	writeIOsPS	쓰기 IO/초	초당 쓰기 작업 수
	writeKb	쓰기 합계	기록한 총 KB 수
	writeKbPS	쓰기 KB/초	초당 기록한 KB 수
	writeLatency	쓰기 지연 시간	쓰기 I/O 요청을 제출하여 처리가 완료될 때까지 걸리는 평균 시간(ms) 이 지표는 Amazon Aurora에만 사용할 수 없습니다.
	writeThroughput	쓰기 처리량	DB 클러스터의 응답에 사용되는 네트워크 처리량(초당 바이트) 이 지표는 Amazon Aurora에만 사용할 수 없습니다.

그룹	측정치	콘솔 이름	설명
	wrqmPS	Wrqms	초당 대기 중인 병합 쓰기 요청 수
physicalDeviceIO	avgQueueLen	물리적 디바이스 평균 대기열 크기	I/O 디바이스의 대기열에서 대기 중인 요청 수입니다.
	avgReqSz	물리적 디바이스 평균 요청 크기	평균 요청 크기(KB)
	await	물리적 디바이스 디스크 I/O 대기	대기열 시간과 서비스 시간을 포함하여 요청에 응답하는 데 필요한 시간(밀리초)
	device	해당 사항 없음	사용 중인 디스크 디바이스의 식별자
	readIOsPS	물리적 디바이스 읽기 IO/초	초당 읽기 작업 수
	readKb	물리적 디바이스 읽기 합계	읽은 총 KB 수
	readKbPS	물리적 디바이스 읽기 Kb/초	초당 읽은 KB 수
	rrqmPS	물리적 디바이스 Rrqms	초당 대기 중인 병합 읽기 요청 수

그룹	측정치	콘솔 이름	설명
	tps	물리적 디바이스 TPS	초당 I/O 트랜잭션 수
	util	물리적 디바이스 디스크 I/O 유틸리티	요청이 발급된 CPU 시간의 비율
	writeIOsPS	물리적 디바이스 쓰기 IO/초	초당 쓰기 작업 수
	writeKb	물리적 디바이스 쓰기 합계	기록한 총 KB 수
	writeKbPS	물리적 디바이스 쓰기 Kb/초	초당 기록한 KB 수
	wrqmPS	물리적 디바이스 Wrqms	초당 대기 중인 병합 쓰기 요청 수
fileSys	maxFiles	최대 Inode	파일 시스템에 대해 생성될 수 있는 최대 파일 수
	mountPoint	해당 사항 없음	파일 시스템의 경로
	name	해당 사항 없음	파일 시스템의 이름
	total	총 파일 시스템	파일 시스템에 사용 가능한 총 디스크 공간(KB)

그룹	측정치	콘솔 이름	설명
	used	사용된 파일 시스템	파일 시스템에서 파일에 사용된 디스크 공간의 양(KB)
	usedFilePercent	사용된 Inode	사용 중인 사용 가능한 파일의 비율
	usedFiles	사용된 %	파일 시스템의 파일 수
	usedPercent	사용된 파일 시스템	사용 중인 파일 시스템 디스크 공간의 비율
loadAverageMinute	fifteen	평균 부하 15분	마지막 15분 동안 CPU 시간을 요청한 프로세스 수
	five	평균 로드 5분	마지막 5분 동안 CPU 시간을 요청한 프로세스 수
	one	평균 로드 1분	마지막 1분 동안 CPU 시간을 요청한 프로세스 수
memory	active	활성 메모리	할당된 메모리의 양(KB)
	buffers	버퍼링된 메모리	스토리지 디바이스에 쓰기 이전에 I/O 요청을 버퍼링하는 데 사용되는 메모리의 양(KB)
	cached	캐시된 메모리	파일 시스템 기반 I/O를 캐시하는 데 사용된 메모리의 양
	dirty	더티 메모리	수정되었지만 스토리지의 관련 데이터 블록에 기록되지 않은 RAM의 메모리 페이지 양(KB)
	free	여유 메모리	할당되지 않은 메모리의 양(KB)

그룹	측정치	콘솔 이름	설명
	hugePagesFree	사용 가능한 방대한 페이지	사용 가능한 방대한 페이지 수입니다. 방대한 페이지는 Linux 커널의 기능입니다.
	hugePagesRsvd	예약된 방대한 페이지	커밋된 방대한 페이지의 수
	hugePagesSize	방대한 페이지 크기	각 방대한 페이지 단위의 크기(KB)
	hugePagesSurp	초과된 방대한 페이지	총계 대비 사용 가능한 초과 방대한 페이지 수
	hugePagesTotal	방대한 페이지 합계	방대한 페이지의 총 수입니다.
	inactive	비활성 메모리	가장 적게 사용되는 메모리 페이지의 양(KB)
	mapped	매핑된 메모리	프로세스 주소 공간 내에 메모리 매핑되는 총 파일 시스템 콘텐츠 양(KB)
	pageTables	페이지 테이블	페이지 표에 사용된 메모리의 양(KB)
	slab	슬래브 메모리	재사용 가능한 커널 데이터 구조의 양(KB)
	total	메모리 합계	총 메모리 양(KB)
	writeback	다시 쓰기 메모리	RAM에서 지원 스토리지에 아직 기록 중인 더티 페이지의 양(KB)

그룹	측정치	콘솔 이름	설명
network	interface	해당 사항 없음	DB 인스턴스에 대해 사용 중인 네트워크 인터페이스의 식별자
	rx	RX	초당 수신된 바이트 수
	tx	TX	초당 업로드된 바이트 수
processList	cpuUsedPc	CPU %	프로세스에서 사용된 CPU 비율
	id	해당 사항 없음	프로세스의 식별자
	memoryUsedPc	MEM%	프로세스에서 사용된 메모리 비율
	name	해당 사항 없음	프로세스의 이름입니다.
	parentID	해당 사항 없음	프로세스의 상위 프로세스에 대한 프로세스 식별자
	rss	RES	프로세스에 할당된 RAM의 양(KB)
	tgid	해당 사항 없음	스레드 그룹 식별자이며, 스레드가 속한 프로세스 ID를 나타내는 숫자입니다. 이 식별자는 동일한 프로세스에서 스레드를 그룹화하는 데 사용됩니다.
	vss	VIRT	프로세스에 할당된 가상 메모리의 양(KB)
swap	total	Swap	사용 가능한 스왑 메모리 양(KB)
	in	스왑 인	디스크에서 스왑된 메모리 양(KB)
	out	스왑 아웃	디스크로 스왑된 총 메모리 양(KB)
	free	사용 가능한 스왑	사용 가능한 스왑 메모리 양(KB)

그룹	측정치	콘솔 이름	설명
	cached	커밋된 스왑	캐시 메모리로 사용된 스왑 메모리의 양(KB)
tasks	blocked	차단되는 작업	차단되는 작업 수
	running	실행 중인 작업	실행 중인 작업 수
	sleeping	절전 상태인 작업	절전 상태인 작업 수
	stopped	중단된 작업	중단된 작업 수
	total	총 작업	총 작업 수
	zombie	작업 좀비	상위 작업은 활성화되었지만 비활성 상태인 하위 작업 수

Microsoft SQL Server DB 인스턴스에 대한 측정치

그룹	측정치	콘솔 이름	설명
General	engine	해당 사항 없음	DB 인스턴스에 대한 데이터베이스 엔진
	instanceID	해당 사항 없음	DB 인스턴스 식별자
	instanceResourceID	해당 사항 없음	AWS 리전에 고유한 DB 인스턴스의 변경 불가능한 식별자로, 로그 스트림 식별자로도 사용됩니다.
	numVCPUs	해당 사항 없음	DB 인스턴스의 가상 CPU 수

그룹	측정치	콘솔 이름	설명
	timestamp	해당 사항 없음	측정치를 가져온 시간
	uptime	해당 사항 없음	DB 인스턴스가 활성 상태로 유지된 시간
	version	해당 사항 없음	OS 측정치 스트림 JSON 형식의 버전
cpuUtilization	idle	CPU 유휴	유휴 상태인 CPU의 비율
	kern	CPU 커널	커널에서 사용 중인 CPU의 비율
	user	CPU 사용자	사용자 프로그램에서 사용 중인 CPU의 비율
disks	name	해당 사항 없음	디스크에 대한 식별자
	totalKb	총 디스크 공간	디스크의 총 공간(KB)
	usedKb	사용된 디스크 공간	디스크에서 사용된 공간의 양(KB)
	usedPc	사용된 디스크 공간(%)	디스크에서 사용된 공간의 비율(%)
	availKb	사용 가능한 디스크 공간	디스크에서 사용 가능한 공간(KB)
	availPc	사용 가능한 디스크 공간(%)	디스크에서 사용 가능한 공간의 비율(%)
	rdCountPS	읽기/초	초당 읽기 작업 수
	rdBytesPS	읽기 KB/초	초당 읽은 바이트 수

그룹	측정치	콘솔 이름	설명
	wrCountPS	쓰기 IO/초	초당 쓰기 작업 수
	wrBytesPS	쓰기 KB/초	초당 기록한 바이트 양
memory	commitTotKb	커밋 합계	사용 중인 페이지 파일 지원 가상 주소 공간의 양 즉, 현재 약정 요금입니다. 이 값은 기본 메모리(RAM)와 디스크(페이지 파일)로 구성됩니다.
	commitLimitKb	최대 커밋	commitTotKb 측정치에 대해 가능한 최대값입니다. 이 값은 현재 페이지 파일 크기와 페이징 가능한 콘텐츠에 사용할 수 있는 물리적 메모리(페이징 불가능 영역에 할당된 RAM 제외)의 합계입니다.
	commitPeakKb	커밋 피크	운영 체제를 마지막으로 시작한 이후의 commitTotKb 측정치 중 가장 큰 값
	kernTotKb	총 커널 메모리	페이징된 풀과 페이징되지 않은 풀의 메모리 합계(KB)
	kernPagedKb	페이징된 커널 메모리	페이징된 커널 풀의 메모리 양(KB)
	kernNonpagedKb	페이징되지 않은 커널 메모리	페이징되지 않은 커널 풀의 메모리 양(KB)
	pageSize	페이지 크기	페이지 크기(KB)
	physTotKb	메모리 합계	물리적 메모리의 양(KB)
	physAvailKb	사용할 수 있는 메모리	사용 가능한 물리적 메모리의 양(KB)
	sqlServerTotKb	SQL Server 총 메모리	SQL Server에 커밋된 메모리의 양(KB)

그룹	측정치	콘솔 이름	설명
	sysCacheKb	시스템 캐시	시스템 캐시 메모리의 양(KB)
network	interface	해당 사항 없음	DB 인스턴스에 대해 사용 중인 네트워크 인터페이스의 식별자
	rdBytesPS	네트워크 읽기 KB/초	초당 수신된 바이트 수
	wrBytesPS	네트워크 쓰기 KB/초	초당 보낸 바이트 수
processList	cpuUsedPc	사용된 %	프로세스에서 사용된 CPU 비율
	memUsedPc	MEM%	프로세스에서 사용된 총 메모리 비율
	name	해당 사항 없음	프로세스의 이름입니다.
	pid	해당 사항 없음	프로세스의 식별자 Amazon RDS에서 소유한 프로세스의 경우 이 값이 표시되지 않습니다.
	ppid	해당 사항 없음	이 프로세스의 상위에 대한 프로세스 식별자입니다. 이 값은 하위 프로세스에 대해서만 표시됩니다.
	tid	해당 사항 없음	스레드 식별자입니다. 이 값은 스레드에 대해서만 표시됩니다. 소유 프로세스는 pid 값을 사용하여 식별할 수 있습니다.
	workingSetKb	해당 사항 없음	프라이빗 작업 세트의 메모리 양과 프로세스에서 사용 중이고 다른 프로세스와 공유 가능한 메모리의 양을 더한 값(KB)
	workingSetPrivKb	해당 사항 없음	프로세스에서 사용 중이지만 다른 프로세스와 공유할 수 없는 메모리의 양(KB)

그룹	측정치	콘솔 이름	설명
	workingSetShareableKb	해당 사항 없음	프로세스에서 사용 중이고 다른 프로세스와 공유할 수 있는 메모리의 양(KB)
	virtKb	해당 사항 없음	프로세스에서 사용 중인 가상 주소 공간의 양(KB)입니다. 가상 주소 공간을 사용하는 것이 반드시 디스크 또는 기본 메모리 페이지를 사용하는 것을 의미하지는 않습니다.
system	handles	핸들	시스템에서 사용 중인 핸들 수
	processes	프로세스	시스템에서 실행 중인 프로세스 수
	threads	스레드	시스템에서 실행 중인 스레드 수

Amazon RDS DB 인스턴스에서 이벤트, 로그 및 스트림 모니터링

Amazon RDS 데이터베이스 및 기타 AWS 솔루션을 모니터링할 때의 목표는 다음을 유지하는 것입니다.

- 신뢰성
- 가용성
- 성능
- 보안

Amazon RDS 인스턴스에서 지표 모니터링에서 지표를 사용하여 인스턴스를 모니터링하는 방법을 설명합니다. 완벽한 솔루션은 데이터베이스 이벤트, 로그 파일 및 활동 스트림을 모니터링해야 합니다. AWS에서는 다음과 같은 모니터링 도구를 제공합니다.

- Amazon EventBridge: 애플리케이션을 다양한 소스의 데이터와 쉽게 연결할 수 있는 서버리스 이벤트 버스 서비스입니다. EventBridge는 자체 애플리케이션, 서비스형 소프트웨어(SaaS) 애플리케이션 및 AWS 서비스의 실시간 데이터 스트림을 제공하고 그 데이터를 AWS Lambda 등의 대상으로 라우팅합니다. 이를 통해 서비스에서 발생하는 이벤트를 모니터링하고 이벤트 기반 아키텍처를 구축할 수 있습니다. 자세한 내용은 https://docs.aws.amazon.com/eventbridge/latest/userguide/Amazon EventBridge 사용 설명서를 참조하세요.
- Amazon CloudWatch Logs는 Amazon RDS 인스턴스, AWS CloudTrail, 기타 소스의 로그 파일을 모니터링, 저장 및 액세스하는 방법을 제공합니다. Amazon CloudWatch Logs는 로그 파일의 정보를 모니터링하고 특정 임계값에 도달하면 사용자에게 알릴 수 있습니다. 또한 매우 내구력 있는 스토리지에 로그 데이터를 저장할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서를 참조하십시오.
- AWS CloudTrail는 AWS 계정에서 또는 이 계정을 대신하여 수행된 API 호출 및 관련 이벤트를 캡처합니다. CloudTrail은 지정된 Amazon S3 버킷에 CloudTrail 이벤트 로그 파일을 전송합니다. 어떤 사용자 및 계정이 AWS를 호출했는지 어떤 소스 IP 주소에 호출이 이루어졌는지 언제 호출이 발생했는지 확인할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하십시오.
- 데이터베이스 활동 스트림은 DB 인스턴스에서 거의 실시간에 가까운 활동 스트림을 제공하는 Amazon RDS 기능입니다. Amazon RDS는 활동을 Amazon Kinesis Data Streams에 푸시합니다. Kinesis 스트림이 자동으로 생성됩니다. Kinesis에서 Amazon Data Firehose 및 AWS Lambda와 같은 AWS 서비스를 구성하여 스트림을 사용하고 데이터를 저장할 수 있습니다.

주제

- Amazon RDS 콘솔에서 로그, 이벤트 및 스트림 보기
- Amazon RDS 이벤트 모니터링
- Amazon RDS 로그 파일 모니터링
- AWS CloudTrail에서 Amazon RDS API 호출 모니터링
- 데이터베이스 활동 스트림을 사용하여 Amazon RDS 모니터링
- Amazon GuardDuty RDS Protection을 이용한 위협 모니터링

Amazon RDS 콘솔에서 로그, 이벤트 및 스트림 보기

Amazon RDS는 RDS 콘솔에서 로그, 이벤트 및 데이터베이스 활동 스트림에 대한 정보를 표시하기 위해 AWS 서비스와 통합합니다.

RDS DB 인스턴스의 로그 및 이벤트(Logs & events) 탭에는 다음 정보가 나와 있습니다.

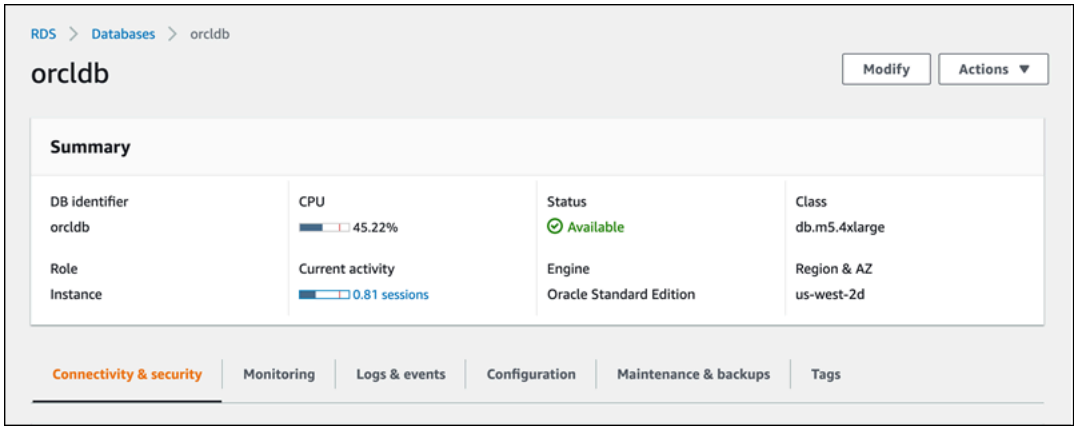
- Amazon CloudWatch 경보: DB 인스턴스에 대해 구성한 모든 지표 경보를 표시합니다. 경보를 구성하지 않은 경우 RDS 콘솔에서 생성할 수 있습니다. 자세한 내용은 Amazon CloudWatch로 Amazon RDS 지표 모니터링 단원을 참조하세요.
- 최근 이벤트(Recent events): RDS DB 인스턴스 에 대한 이벤트 요약(환경 변경 사항)을 보여줍니다. 자세한 내용은 Amazon RDS 이벤트 보기을 참조하세요.
- 로그(Logs): DB 인스턴스에서 생성된 데이터베이스 로그 파일을 표시합니다. 자세한 내용은 Amazon RDS 로그 파일 모니터링을 참조하세요.

구성(Configuration) 탭은 데이터베이스 활동 스트림에 대한 정보를 표시합니다.

RDS 콘솔에서 DB 인스턴스에 대한 로그, 이벤트 및 스트림 보기

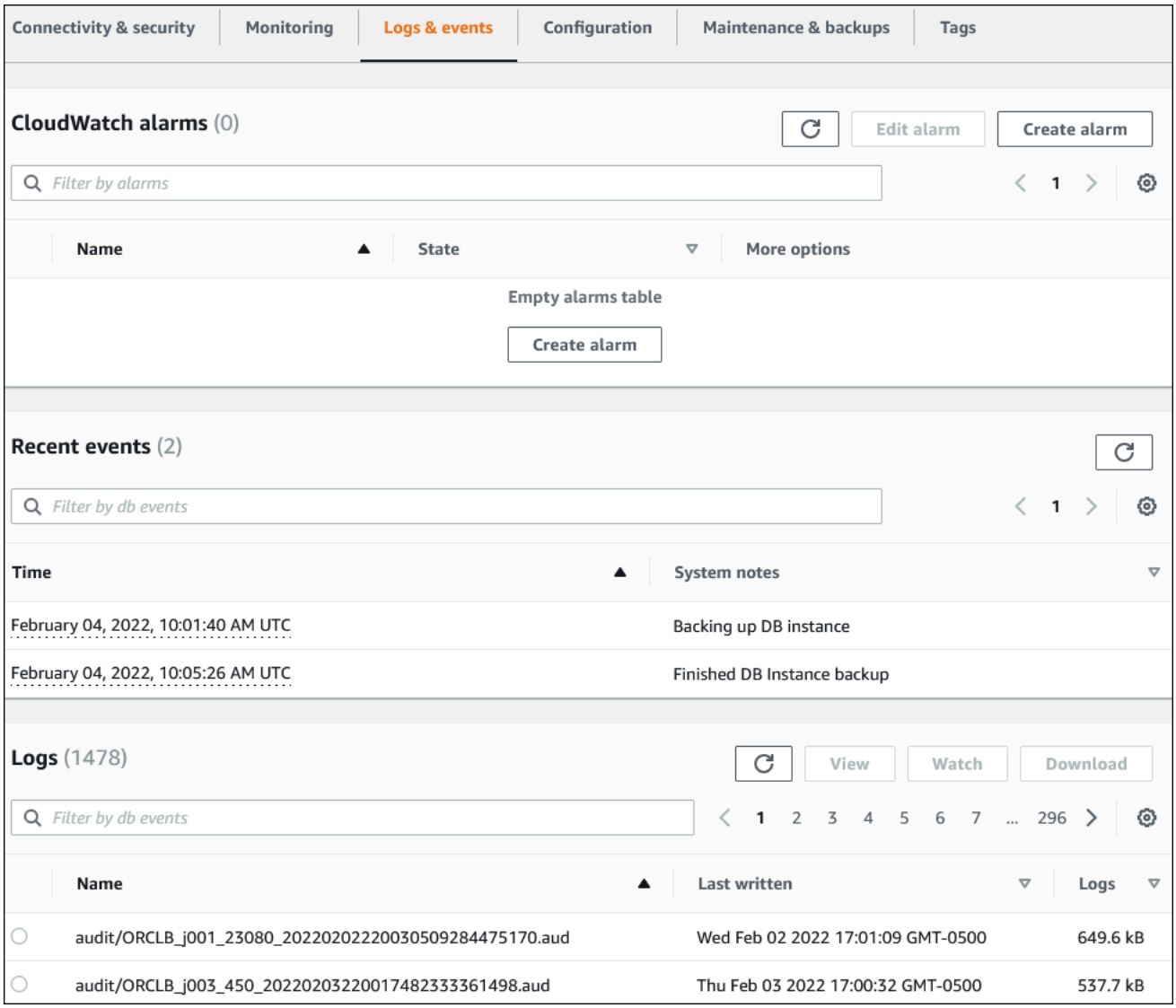
1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
3. 모니터링 하려는 DB 인스턴스의 이름을 선택합니다.

   데이터베이스 페이지가 표시됩니다. 다음 예는 의 `orclb`라는 Oracle 데이터베이스 이름을 보여줍니다.



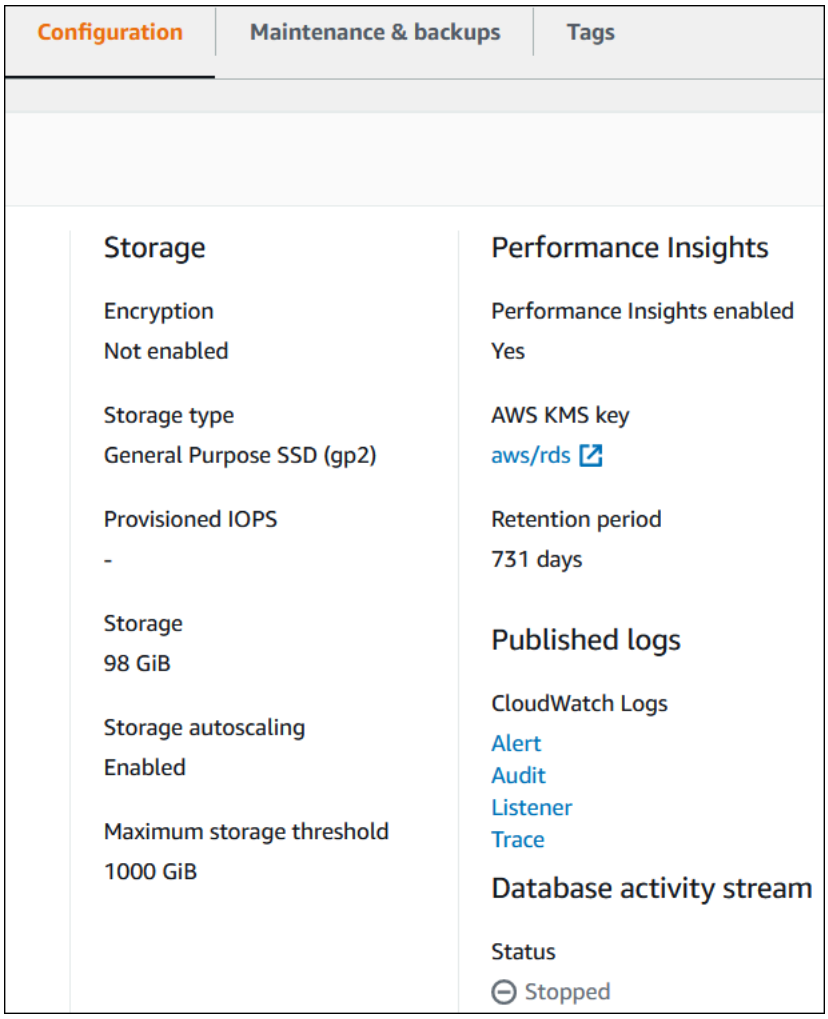
4. 로그 및 이벤트를 선택합니다.

   로그 및 이벤트 섹션이 표시됩니다.



5. 구성을 선택합니다.

   다음 예는 DB 인스턴스에 대한 데이터베이스 활동 스트림 상태를 보여줍니다.



Amazon RDS 이벤트 모니터링

이벤트는 환경의 변화를 나타냅니다. AWS 환경, 즉 SaaS 파트너 서비스나 애플리케이션 또는 사용자 지정 애플리케이션이나 서비스일 수 있습니다. RDS 이벤트에 대한 설명을 보려면 Amazon RDS 이벤트 카테고리 및 이벤트 메시지 섹션을 참조하세요.

주제

- Amazon RDS에 대한 이벤트 개요
- Amazon RDS 이벤트 보기
- Amazon RDS 이벤트 알림 작업
- Amazon RDS 이벤트에서 트리거되는 규칙 생성
- Amazon RDS 이벤트 카테고리 및 이벤트 메시지

Amazon RDS에 대한 이벤트 개요

RDS 이벤트는 Amazon RDS 환경의 변경 사항을 나타내는 이벤트입니다. 예를 들어 Amazon RDS는 DB 인스턴스 상태가 보류 중에서 실행 중으로 변경될 때 이벤트를 생성합니다. Amazon RDS는 거의 실시간으로 EventBridge에 이벤트를 전송합니다.

> **Note**
>
> Amazon RDS는 최선의 작업을 기반으로 이벤트를 방출합니다. 알림 이벤트의 순서나 존재 여부에 따라 달라지는 프로그램은 순서가 잘못되거나 누락될 수 있으므로 작성하지 않는 것이 좋습니다.

Amazon RDS는 다음 리소스와 관련된 이벤트를 기록합니다.

- DB 인스턴스

  DB 인스턴스 이벤트 목록은 DB 인스턴스 이벤트 섹션을 참조하세요.
- DB 파라미터 그룹

  DB 파라미터 그룹 이벤트 목록은 DB 파라미터 그룹 이벤트 섹션을 참조하세요.
- DB 보안 그룹

  DB 보안 그룹 이벤트 목록은 DB 보안 그룹 이벤트 섹션을 참조하세요.

- DB 스냅샷

  DB 스냅샷 이벤트 목록은 DB 스냅샷 이벤트 섹션을 참조하세요.
- RDS 프록시 이벤트

  RDS 프록시 이벤트 목록은 RDS 프록시 이벤트 섹션을 참조하세요.
- 블루/그린 배포 이벤트

  블루/그린 배포 이벤트 목록은 블루/그린 배포 이벤트 섹션을 참조하세요.

이 정보에는 다음 사항이 포함됩니다.

- 이벤트의 날짜 및 시간
- 이벤트의 소스 이름 및 소스 유형
- 이벤트와 연결된 메시지
- 이벤트 알림에는 메시지가 전송된 시점의 태그가 포함되며 이벤트가 발생한 시점의 태그는 반영되지 않을 수 있습니다.

Amazon RDS 이벤트 보기

Amazon RDS 리소스에 대해 다음 이벤트 정보를 검색할 수 있습니다.

- 리소스 이름
- 리소스 유형
- 이벤트 시간
- 이벤트 메시지 요약

AWS Management Console의 다음 부분에서 이벤트에 액세스할 수 있습니다.

- 이벤트 탭은 지난 24시간 동안의 이벤트를 표시합니다.
- 데이터베이스 탭의 로그 및 이벤트 섹션에 있는 최근 이벤트 테이블에는 최대 지난 2주 동안의 이벤트가 표시될 수 있습니다.

describe-events AWS CLI 명령 또는 DescribeEvents RDS API 작업을 사용하여 이벤트를 검색할 수도 있습니다. AWS CLI 또는 RDS API를 사용하여 이벤트를 볼 경우 최대 지난 14일 동안 발생한 이벤트를 검색할 수 있습니다.

> **Note**
>
> 더 오랜 기간 동안 이벤트를 저장해야 하는 경우 EventBridge에 Amazon RDS 이벤트를 보낼 수 있습니다. 자세한 내용은 Amazon RDS 이벤트에서 트리거되는 규칙 생성 섹션을 참조하세요.

Amazon RDS 이벤트에 대한 설명을 보려면 Amazon RDS 이벤트 카테고리 및 이벤트 메시지 섹션을 참조하세요.

요청 파라미터를 포함하여 AWS CloudTrail를 사용하여 이벤트에 대한 세부 정보에 액세스하려면 CloudTrail 이벤트 단원을 참조하십시오.

콘솔

지난 24시간 동안 발생한 모든 Amazon RDS 이벤트를 보는 방법

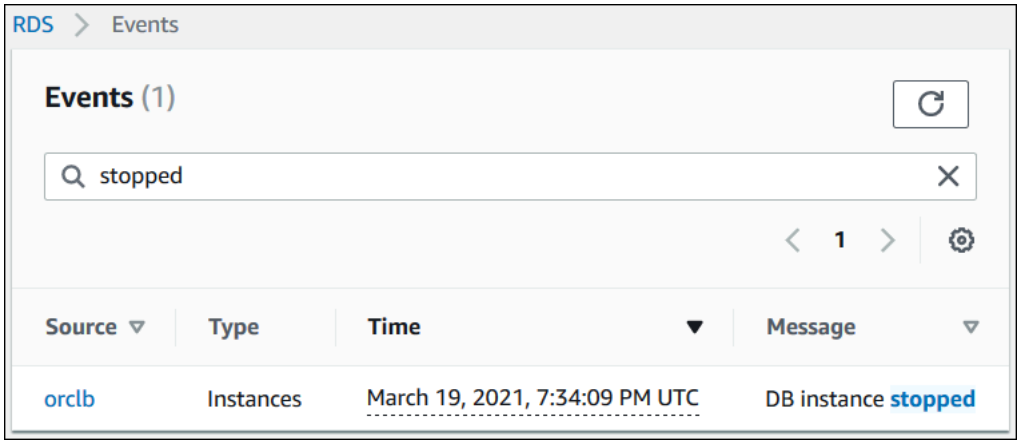
1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.

2. 탐색 창에서 [Events]를 선택합니다.

   사용 가능한 이벤트가 목록에 표시됩니다.

3. (선택 사항) 검색어를 입력하여 결과를 필터링합니다.

   다음 예에서는 **stopped** 문자로 필터링된 이벤트 목록을 보여줍니다.



AWS CLI

지난 한 시간 동안 생성된 모든 이벤트를 보려면 파라미터 없이 describe-events를 호출합니다.

```
aws rds describe-events
```

다음 샘플 출력은 DB 인스턴스가 중지되었음을 보여줍니다.

```
{
    "Events": [
        {
            "EventCategories": [
                "notification"
            ],
            "SourceType": "db-instance",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:testinst",
            "Date": "2022-04-22T21:31:00.681Z",
            "Message": "DB instance stopped",
            "SourceIdentifier": "testinst"
        }
    ]
}
```

지난 10,080분(7일) 동안의 모든 Amazon RDS 이벤트를 보려면 describe-events AWS CLI 명령을 호출하고 `--duration` 파라미터를 `10080`으로 설정합니다.

```
aws rds describe-events --duration 10080
```

다음 예에서는 DB 인스턴스 *`test-instance`*에 대해 지정된 시간 범위의 이벤트를 보여줍니다.

```
aws rds describe-events \
    --source-identifier test-instance \
    --source-type db-instance \
    --start-time 2022-03-13T22:00Z \
    --end-time 2022-03-13T23:59Z
```

다음 샘플 출력은 백업 상태를 보여줍니다.

```
{
    "Events": [
        {
            "SourceType": "db-instance",
            "SourceIdentifier": "test-instance",
            "EventCategories": [
                "backup"
            ],
            "Message": "Backing up DB instance",
            "Date": "2022-03-13T23:09:23.983Z",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance"
        },
        {
            "SourceType": "db-instance",
            "SourceIdentifier": "test-instance",
            "EventCategories": [
                "backup"
            ],
            "Message": "Finished DB Instance backup",
            "Date": "2022-03-13T23:15:13.049Z",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance"
        }
    ]
}
```

API

DescribeEvents RDS API 작업을 호출하고 `Duration` 파라미터를 `20160`으로 설정하여 지난 14일 동안 발생한 모든 Amazon RDS 인스턴스 이벤트를 볼 수 있습니다.

Amazon RDS 이벤트 알림 작업

Amazon RDS는 Amazon RDS 이벤트 발생 시 Amazon Simple Notification Service(Amazon SNS)를 사용하여 알림 서비스를 제공합니다. 이 서비스는 AWS 리전에 따라 Amazon SNS가 지원하는 알림 메시지 형식에 따라 이메일, 문자 또는 HTTP 엔드포인트 호출 등이 될 수 있습니다.

주제

- Amazon RDS 이벤트 알림 개요
- Amazon SNS 주제에 알림을 게시할 권한 부여
- Amazon RDS 이벤트 알림 구독
- Amazon RDS 이벤트 알림 태그 및 속성
- Amazon RDS 이벤트 알림 구독의 목록 표시
- Amazon RDS 이벤트 알림 구독 수정
- Amazon RDS 이벤트 알림 구독에 소스 식별자 추가
- Amazon RDS 이벤트 알림 구독에서 소스 식별자 제거
- Amazon RDS 이벤트 알림 카테고리의 목록 표시
- Amazon RDS 이벤트 알림 구독 삭제

Amazon RDS 이벤트 알림 개요

Amazon RDS는 구독 가능한 카테고리로 이벤트를 그룹화합니다. 따라서 해당 카테고리의 이벤트가 발생했을 때 이에 대한 알림 메시지를 받을 수 있습니다.

주제

- 이벤트 구독에 적합한 RDS 리소스
- Amazon RDS 이벤트 알림을 구독하는 기본 프로세스
- RDS 이벤트 알림 전송
- Amazon RDS 이벤트 알림 결제
- Amazon EventBridge를 사용한 Amazon RDS 이벤트의 예

이벤트 구독에 적합한 RDS 리소스

다음 리소스에 대한 이벤트 범주에 구독할 수 있습니다.

- DB 인스턴스
- DB 스냅샷
- DB 파라미터 그룹
- DB 보안 그룹
- RDS 프록시
- 커스텀 엔진 버전

예를 들어 임의의 DB 인스턴스에 대한 백업 카테고리를 구독할 경우 백업 관련 이벤트가 발생하여 DB 인스턴스에 영향을 끼칠 때마다 알림 메시지가 수신됩니다. 혹은 DB 인스턴스의 구성 변경 카테고리를 구독하면 DB 인스턴스가 변경될 때마다 메시지가 수신됩니다. 또한 이벤트 알림 메시지 구독이 변경되어도 알림 메시지가 수신됩니다.

여러 가지 다른 구독을 만들 수 있습니다. 예를 들어, 전체 DB 인스턴스에 대한 모든 이벤트 알림을 수신하는 하나의 구독과 DB 인스턴스의 하위 집합에 대한 중요 이벤트만 포함하는 다른 구독을 만들 수 있습니다. 두 번째 구독의 경우 필터에 하나 이상의 DB 인스턴스를 지정합니다.

Amazon RDS 이벤트 알림을 구독하는 기본 프로세스

Amazon RDS 이벤트 알림 서비스를 구독하는 프로세스는 다음과 같습니다.

1. Amazon RDS 콘솔, AWS CLI 또는 API를 사용하여 Amazon RDS 이벤트 알림 서비스 구독을 생성합니다.

   Amazon RDS에서는 Amazon SNS 주제의 ARN을 사용하여 각 구독을 식별합니다. Amazon RDS 콘솔은 구독 생성 시 ARN을 생성합니다. Amazon SNS 콘솔인 AWS CLI 또는 Amazon SNS API를 사용하여 ARN을 생성합니다.
2. Amazon RDS가 구독 생성 시 제출한 이메일 주소로 승인 이메일 또는 SMS 메시지를 전송합니다.
3. 전송된 알림의 링크를 선택하여 구독 여부를 확인합니다.
4. Amazon RDS 콘솔은 구독 상태로 내 이벤트 구독 섹션을 업데이트합니다.
5. Amazon RDS는 구독을 생성할 때 제공한 주소로 알림을 보내기 시작합니다.

Amazon SNS 사용 시 자격 증명 및 액세스 관리에 대한 자세한 내용은 Amazon Simple Notification Service 개발자 가이드의 Amazon SNS의 Identity and Access Management를 참조하세요.

AWS Lambda를 사용하여 DB 인스턴스의 이벤트 알림을 처리할 수 있습니다. 자세한 내용은 AWS Lambda 개발자 안내서의 Amazon RDS에서 AWS Lambda 사용을 참조하세요.

RDS 이벤트 알림 전송

Amazon RDS는 구독 생성 시 입력한 주소로 알림을 보냅니다. 알림에는 메시지에 대한 구조화된 메타데이터를 제공하는 메시지 속성이 포함될 수 있습니다. 메시지 속성에 대한 자세한 내용은 Amazon RDS 이벤트 카테고리 및 이벤트 메시지 단원을 참조하세요.

이벤트 알림을 전달하는 데 최대 5분 정도 걸릴 수 있습니다.

> **Important**
>
> Amazon RDS는 이벤트 스트림에서 전송된 이벤트 순서를 보장하지 않습니다. 이벤트 순서는 변경될 수 있습니다.

Amazon SNS이 구독된 HTTP 또는 HTTPS 엔드포인트에 알림을 전송할 때 엔드포인트에 전송된 POST 메시지에는 JSON 문서를 포함하는 메시지 본문이 있습니다. 자세한 내용은 Amazon Simple Notification Service 개발자 안내서의 Amazon SNS 메시지 및 JSON 형식을 참조하십시오.

문자 메시지로 알리도록 SNS를 구성할 수 있습니다. 자세한 내용은 Amazon Simple Notification Service 개발자 가이드의 모바일 문자 메시지(SMS)를 참조하세요.

구독을 삭제하지 않고 알림을 끄려면 Amazon RDS 콘솔에서 활성에 대해 아니요를 선택합니다. 또는 AWS CLI 또는 Amazon RDS API를 사용하여 `Enabled` 파라미터를 `false`(으)로 설정할 수 있습니다.

Amazon RDS 이벤트 알림 결제

Amazon RDS 이벤트 알림에 대한 결제는 Amazon SNS를 통해 이루어집니다. Amazon SNS 요금은 이벤트 알림 사용 시 적용됩니다. Amazon SNS 결제에 대한 자세한 내용은 Amazon Simple Notification Service 요금을 참조하세요.

Amazon EventBridge를 사용한 Amazon RDS 이벤트의 예

다음 예에서는 JSON 형식으로 다양한 Amazon RDS 이벤트 유형을 보여줍니다. JSON 형식으로 이벤트를 캡처하고 보는 방법을 보여 주는 자습서는 자습서: Amazon EventBridge를 사용하여 DB 인스턴스의 상태 변경 로깅 단원을 참조하세요.

주제

- DB 인스턴스 이벤트의 예
- DB 파라미터 그룹 이벤트의 예
- DB 스냅샷 이벤트의 예

DB 인스턴스 이벤트의 예

다음은 JSON 형식의 DB 인스턴스 이벤트에 대한 예제입니다. 이 이벤트는 RDS가 `my-db-instance`라는 인스턴스에 대해 다중 AZ 장애 조치를 수행했음을 보여줍니다. 이벤트 ID는 RDS-EVENT-0049입니다.

```
{
  "version": "0",
  "id": "68f6e973-1a0c-d37b-f2f2-94a7f62ffd4e",
  "detail-type": "RDS DB Instance Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-09-27T22:36:43Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:rds:us-east-1:123456789012:db:my-db-instance"
  ],
  "detail": {
    "EventCategories": [
      "failover"
    ],
    "SourceType": "DB_INSTANCE",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance",
    "Date": "2018-09-27T22:36:43.292Z",
    "Message": "A Multi-AZ failover has completed.",
    "SourceIdentifier": "my-db-instance",
    "EventID": "RDS-EVENT-0049"
  }
}
```

DB 파라미터 그룹 이벤트의 예

다음은 JSON 형식의 DB 파라미터 그룹 이벤트에 대한 예제입니다. 이 이벤트는 `time_zone` 파라미터가 파라미터 그룹 `my-db-param-group`에서 업데이트되었음을 보여줍니다. 이벤트 ID는 RDS-EVENT-0037입니다.

```
{
  "version": "0",
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",
  "detail-type": "RDS DB Parameter Group Event",
  "source": "aws.rds",
  "account": "123456789012",
```

```
  "time": "2018-10-06T12:26:13Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:rds:us-east-1:123456789012:pg:my-db-param-group"
  ],
  "detail": {
    "EventCategories": [
      "configuration change"
    ],
    "SourceType": "DB_PARAM",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:pg:my-db-param-group",
    "Date": "2018-10-06T12:26:13.882Z",
    "Message": "Updated parameter time_zone to UTC with apply method immediate",
    "SourceIdentifier": "my-db-param-group",
    "EventID": "RDS-EVENT-0037"
  }
}
```

DB 스냅샷 이벤트의 예

다음은 JSON 형식의 DB 스냅샷 이벤트에 대한 예제입니다. 이 이벤트는 `my-db-snapshot`이라는 스냅샷의 삭제를 보여줍니다. 이벤트 ID는 RDS-EVENT-0041입니다.

```
{
  "version": "0",
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",
  "detail-type": "RDS DB Snapshot Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-10-06T12:26:13Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:rds:us-east-1:123456789012:snapshot:rds:my-db-snapshot"
  ],
  "detail": {
    "EventCategories": [
      "deletion"
    ],
    "SourceType": "SNAPSHOT",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:snapshot:rds:my-db-snapshot",
    "Date": "2018-10-06T12:26:13.882Z",
    "Message": "Deleted manual snapshot",
    "SourceIdentifier": "my-db-snapshot",
```

```
    "EventID": "RDS-EVENT-0041"
  }
}
```

Amazon SNS 주제에 알림을 게시할 권한 부여

Amazon RDS 권한을 부여하여 알림을 Amazon Simple Notification Service(Amazon SNS) 주제에 게시하려면 대상 주제에 AWS Identity and Access Management(IAM) 정책을 연결하면 됩니다. 권한에 대한 자세한 내용은 Amazon Simple Notification Service 개발자 안내서의 Amazon Simple Notification Service 액세스 제어 예제 사례를 참조하세요.

기본적으로 Amazon SNS 주제에는 동일한 계정 내의 모든 Amazon RDS 리소스가 알림을 게시하도록 허용하는 정책이 있습니다. 교차 계정 알림을 허용하거나 특정 리소스에 대한 액세스를 제한하는 사용자 지정 정책을 연결할 수 있습니다.

다음은 대상 Amazon SNS 주제에 연결하는 IAM 정책의 예입니다. 지정된 접두사와 일치하는 이름을 가진 DB 인스턴스로 주제를 제한합니다. 이 정책을 사용하려면 다음 값을 지정합니다.

- `Resource` – Amazon SNS 주제에 대한 Amazon 리소스 이름(ARN)
- `SourceARN` - RDS 리소스 ARN
- `SourceAccount` – 사용자의 AWS 계정 ID

리소스 유형 및 해당 ARN 목록을 보려면 서비스 승인 참조에서 Amazon RDS에서 정의한 리소스를 참조하세요.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "events.rds.amazonaws.com"
      },
      "Action": [
        "sns:Publish"
      ],
      "Resource": "arn:aws:sns:us-east-1:123456789012:topic_name",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:rds:us-east-1:123456789012:db:prefix-*"
        },
        "StringEquals": {
```

```
          "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

Amazon RDS 이벤트 알림 구독

가장 간단한 구독 생성 방법은 RDS 콘솔입니다. CLI 또는 API를 사용하여 이벤트 알림 구독을 생성하려면 먼저 Amazon Simple Notification Service 주제를 만든 후 Amazon SNS 콘솔이나 Amazon SNS API를 통해 해당 주제를 구독해야 합니다. 또한 CLI 명령이나 API 작업을 제출할 때도 사용되기 때문에 해당 주제의 Amazon 리소스 이름(ARN)을 잊어서는 안 됩니다. SNS 주제를 새로 만들어 구독하는 방법에 대한 자세한 내용은 Amazon Simple Notification Service 개발자 안내서의 Amazon SNS 시작하기 단원을 참조하십시오.

알림 메시지를 받고 싶은 소스 유형과 이벤트를 트리거링하는 Amazon RDS 소스를 지정할 수 있습니다.

소스 유형

소스 유형입니다. 예를 들어 소스 유형은 인스턴스일 수 있습니다. 소스 유형을 선택해야 합니다.

포함할 ***###***

이벤트를 생성하는 Amazon RDS 리소스입니다. 예를 들어 특정 인스턴스 선택을 선택한 다음 myDBInstance1을 선택할 수 있습니다.

다음 테이블에서는 포함할 ***###***를 지정하거나 지정하지 않을 때의 결과를 설명합니다

포함할 리소스	설명	예
지정됨	RDS는 지정된 리소스에 대한 모든 이벤트에 대해서만 알림을 보냅니다.	소스 유형이 인스턴스이고 리소스가 myDBInstance1인 경우 RDS는 myDBInstance1에 대한 모든 이벤트에 대해서만 알립니다.
지정되지 않음	RDS는 모든 Amazon RDS 리소스에 대해 지정된 소스 유형에 대한 이벤트를 알려줍니다.	소스 유형이 인스턴스인 경우 RDS는 계정의 모든 인스턴스 관련 이벤트에 대해 알려줍니다.

Amazon SNS 주제 구독자는 기본적으로 주제에 게시된 모든 메시지를 수신합니다. 메시지의 하위 집합만 수신하려면 구독자는 주제 구독에 필터 정책을 할당해야 합니다. SNS 메시지 필터링에 대한 자세한 내용은 Amazon Simple Notification Service 개발자 안내서의 Amazon SNS 메시지 필터링을 참조하세요.

콘솔

RDS 이벤트 알림 구독 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 이벤트 구독을 선택합니다.
3. 이벤트 구독 창에서 이벤트 구독 생성을 선택합니다.
4. 다음과 같이 구독 세부 정보를 입력합니다.

   a. 이름에서 이벤트 알림 구독 이름을 입력합니다.

   b. 알림 보내기 대상에게 다음 중 하나를 실시합니다.

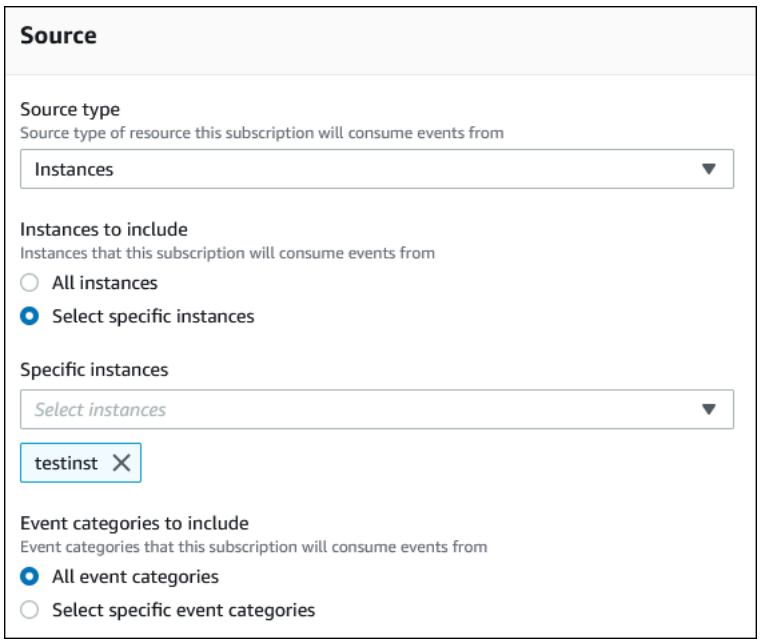
      - 새 이메일 주제를 선택합니다. 이메일 주제의 이름과 수신자 목록을 입력합니다. 기본 계정 연락처와 동일한 이메일 주소로 이벤트 구독을 구성하는 것이 좋습니다. 권장 사항, 서비스 이벤트 및 개인 건강 메시지는 다양한 채널을 사용하여 전송됩니다. 동일한 이메일 주소를 구독하면 모든 메시지가 한 위치에 통합됩니다.
      - Amazon 리소스 이름(ARN)을 선택합니다. 그런 다음 Amazon SNS 주제에 대해 기존 Amazon SNS ARN을 선택합니다.

        서버 측 암호화(SSE)에 대해 활성화된 주제를 사용하려면 AWS KMS key에 액세스하는 데 필요한 권한을 Amazon RDS에 부여하면 됩니다. 자세한 내용은 Amazon Simple Notification Service 개발자 안내서의 AWS 서비스 및 암호화된 주제의 이벤트 소스 간 호환성 활성화를 참조하세요.

   c. [Source type]에서 원본 형식을 선택합니다. 예를 들어 인스턴스 또는 파라미터 그룹을 선택합니다.

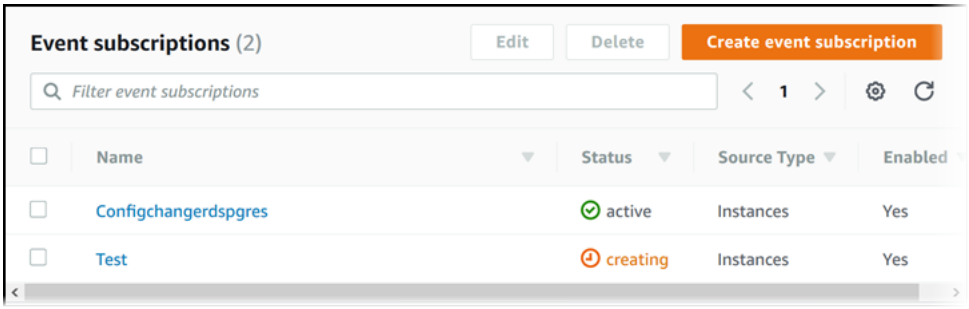
   d. 이벤트 알림을 받을 이벤트 카테고리 및 리소스를 선택합니다.

      다음 예에서는 `testinst`라는 DB 인스턴스에 대한 이벤트 알림을 구성합니다.



e. 생성을 선택합니다.

Amazon RDS 콘솔에 현재 구독 생성 중으로 나옵니다.



AWS CLI

RDS 이벤트 알림을 구독하려면 AWS CLI `create-event-subscription` 명령을 사용합니다. 다음 필수 파라미터를 포함합니다.

- `--subscription-name`
- `--sns-topic-arn`

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-event-subscription \
```

```
    --subscription-name myeventsubscription \
    --sns-topic-arn arn:aws:sns:us-east-1:123456789012:myawsuser-RDS \
    --enabled
```

Windows의 경우:

```
aws rds create-event-subscription ^
    --subscription-name myeventsubscription ^
    --sns-topic-arn arn:aws:sns:us-east-1:123456789012:myawsuser-RDS ^
    --enabled
```

API

Amazon RDS 이벤트 알림을 구독하려면 Amazon RDS API 함수 CreateEventSubscription을 호출합니다. 다음 필수 파라미터를 포함합니다.

- SubscriptionName
- SnsTopicArn

Amazon RDS 이벤트 알림 태그 및 속성

Amazon RDS에서 Amazon Simple Notification Service(SNS) 또는 Amazon EventBridge로 이벤트 알림을 전송할 경우, 알림에 메시지 속성과 이벤트 태그가 포함됩니다. RDS는 이벤트 태그가 메시지 본문에 있는 동안 메시지 속성을 메시지와 함께 별도로 전송합니다. 메시지 속성과 Amazon RDS 태그를 사용하여 메타데이터를 리소스에 추가할 수 있습니다. 이러한 태그를 DB 인스턴스에 대한 주석과 함께 수정할 수 있습니다. Amazon RDS 리소스 태그 지정에 대한 자세한 내용은 Amazon RDS 리소스에 태그 지정 단원을 참조하세요.

기본적으로 Amazon SNS 및 Amazon EventBridge는 전송되는 모든 메시지를 수신합니다. SNS 및 EventBridge는 메시지를 필터링할 수 있으며 이메일, 문자 메시지 또는 HTTP 엔드포인트로 수신되는 통화 같은 기본 커뮤니케이션 모드로 알림을 전송할 수 있습니다.

> **Note**
>
> 이메일이나 문자 메시지로 전송된 알림에는 이벤트 태그가 없습니다.

다음 표에는 주제 구독자에게 전송되는 RDS 이벤트의 메시지 속성이 나와 있습니다.

Amazon RDS 이벤트 속성	설명
EventID	RDS 이벤트 메시지의 식별자는 예를 들어 RDS-EVENT-0006입니다.
리소스	이벤트를 발생시키는 리소스의 ARN 식별자(예: `arn:aws:rds:ap-southeast-2:123456789012:db:database-1`)입니다.

RDS 태그는 서비스 이벤트의 영향을 받은 리소스에 대한 데이터를 제공합니다. RDS는 알림이 SNS 또는 EventBridge로 전송될 때 메시지 본문에 태그의 현재 상태를 추가합니다.

SNS의 필터링 메시지 속성에 대한 자세한 내용은 Amazon Simple Notification Service 개발자 안내서의 Amazon SNS 메시지 필터링을 참조하세요.

EventBridge의 이벤트 태그 필터링에 대한 자세한 내용은 Amazon EventBridge 사용 설명서에서 Amazon EventBridge 이벤트 패턴에 사용할 콘텐츠 비교 연산자를 참조하세요.

SNS에 대한 페이로드 기반 태그 필터링에 대한 자세한 내용은 Introducing payload-based message filtering for Amazon SNS를 참조하세요.

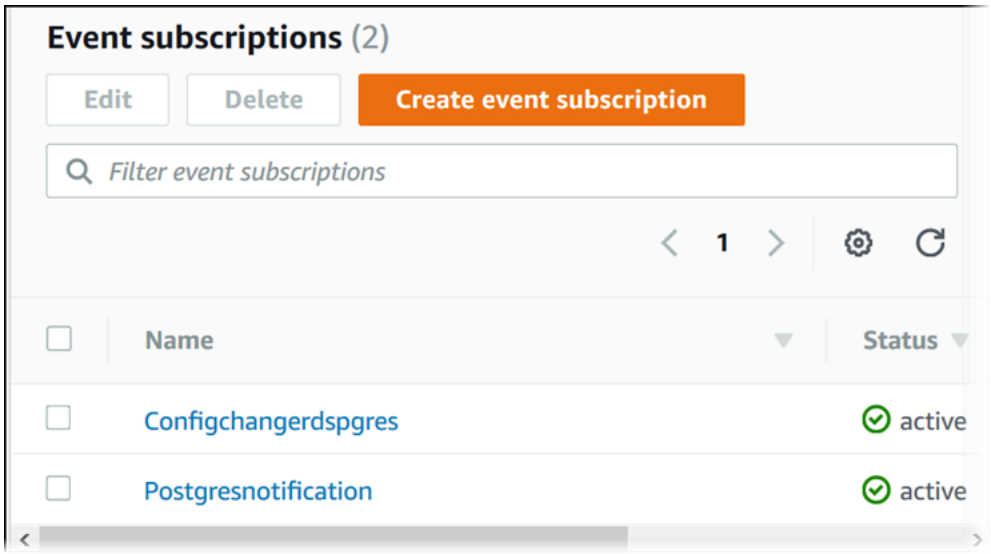
Amazon RDS 이벤트 알림 구독의 목록 표시

현재 Amazon RDS 이벤트 알림 구독을 목록으로 표시할 수 있습니다.

콘솔

현재 Amazon RDS 이벤트 알림 구독을 목록으로 표시하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Event subscriptions]를 선택합니다. [Event subscriptions] 창에 이벤트 알림 구독이 모두 표시됩니다.



AWS CLI

현재 Amazon RDS 이벤트 알림 구독을 표시하려면 AWS CLI `describe-event-subscriptions` 명령을 사용합니다.

Example

다음은 모든 이벤트 구독을 설명하는 예제입니다.

```
aws rds describe-event-subscriptions
```

다음은 `myfirsteventsubscription`을 설명하는 예제입니다.

```
aws rds describe-event-subscriptions --subscription-name myfirsteventsubscription
```

API

현재 Amazon RDS 이벤트 알림 구독을 표시하려면 Amazon RDS API DescribeEventSubscriptions 작업을 사용합니다.

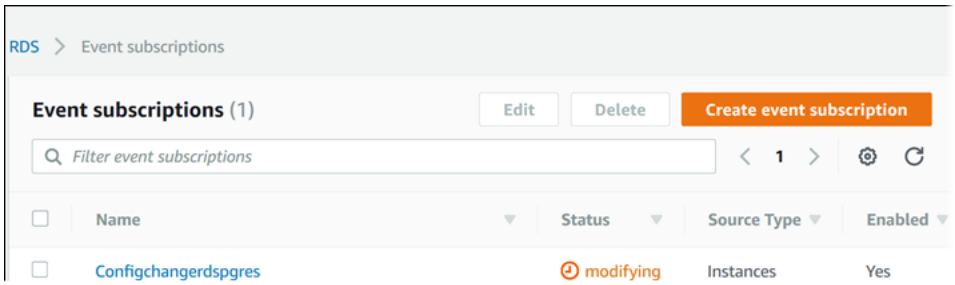
Amazon RDS 이벤트 알림 구독 수정

구독을 생성한 후에는 구독 이름, 소스 식별자, 카테고리 또는 주제 ARN을 변경할 수 있습니다.

콘솔

Amazon RDS 이벤트 알림 구독을 변경하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Event subscriptions]를 선택합니다.
3. [Event subscriptions] 창에서 수정할 구독을 선택한 다음 [Edit]를 선택합니다.
4. 대상 또는 원본 섹션에서 구독을 변경합니다.
5. 편집을 선택합니다. Amazon RDS 콘솔에 현재 구독 변경 중으로 나옵니다.



AWS CLI

Amazon RDS 이벤트 알림 구독을 수정하려면, AWS CLI `modify-event-subscription` 명령을 사용합니다. 다음 필수 파라미터를 포함합니다.

- `--subscription-name`

Example

다음 코드를 사용하여 myeventsubscription을 사용할 수 있습니다.

Linux, macOS 또는 Unix 대상:

```
aws rds modify-event-subscription \
    --subscription-name myeventsubscription \
    --enabled
```

Windows의 경우:

```
aws rds modify-event-subscription ^
    --subscription-name myeventsubscription ^
    --enabled
```

API

Amazon RDS 이벤트를 수정하려면 Amazon RDS API 작업 ModifyEventSubscription을 호출합니다. 다음 필수 파라미터를 포함합니다.

- SubscriptionName

Amazon RDS 이벤트 알림 구독에 소스 식별자 추가

소스 식별자(이벤트를 발생시키는 Amazon RDS 소스)를 기존 구독에 추가할 수 있습니다.

콘솔

소스 식별자는 Amazon RDS 콘솔에서 구독 관련 설정을 변경하면서 선택 또는 선택 해제를 통해 쉽게 추가하거나 제거할 수 있습니다. 자세한 내용은 Amazon RDS 이벤트 알림 구독 수정 섹션을 참조하세요.

AWS CLI

Amazon RDS 이벤트 알림 구독에 소스 식별자를 추가하려면 AWS CLI `add-source-identifier-to-subscription` 명령을 사용합니다. 다음 필수 파라미터를 포함합니다.

- `--subscription-name`
- `--source-identifier`

Example

다음 예제에서는 `mysqldb` 구독에 소스 식별자 `myrdseventsubscription`를 추가합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds add-source-identifier-to-subscription \
    --subscription-name myrdseventsubscription \
    --source-identifier mysqldb
```

Windows의 경우:

```
aws rds add-source-identifier-to-subscription ^
    --subscription-name myrdseventsubscription ^
    --source-identifier mysqldb
```

API

Amazon RDS 이벤트 알림 구독에 소스 식별자를 추가하려면 Amazon RDS API `AddSourceIdentifierToSubscription`을 호출합니다. 다음 필수 파라미터를 포함합니다.

- `SubscriptionName`

- SourceIdentifier

Amazon RDS 이벤트 알림 구독에서 소스 식별자 제거

임의 소스의 이벤트 알림 메시지를 더 이상 받고 싶지 않을 때는 소스 식별자(이벤트를 발생시키는 Amazon RDS 소스)를 구독에서 제거할 수 있습니다.

콘솔

소스 식별자는 Amazon RDS 콘솔에서 구독 관련 설정을 변경하면서 선택 또는 선택 해제를 통해 쉽게 추가하거나 제거할 수 있습니다. 자세한 내용은 Amazon RDS 이벤트 알림 구독 수정 섹션을 참조하세요.

AWS CLI

Amazon RDS 이벤트 알림 구독에서 소스 식별자를 제거하려면 AWS CLI `remove-source-identifier-from-subscription` 명령을 사용합니다. 다음 필수 파라미터를 포함합니다.

- `--subscription-name`
- `--source-identifier`

Example

다음 예제에서는 `mysqldb` 구독에서 소스 식별자 `myrdseventsubscription`를 제거합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds remove-source-identifier-from-subscription \
    --subscription-name myrdseventsubscription \
    --source-identifier mysqldb
```

Windows의 경우:

```
aws rds remove-source-identifier-from-subscription ^
    --subscription-name myrdseventsubscription ^
    --source-identifier mysqldb
```

API

Amazon RDS 이벤트 알림 구독에서 소스 식별자를 제거하려면 Amazon RDS API `RemoveSourceIdentifierFromSubscription` 명령을 사용합니다. 다음 필수 파라미터를 포함합니다.

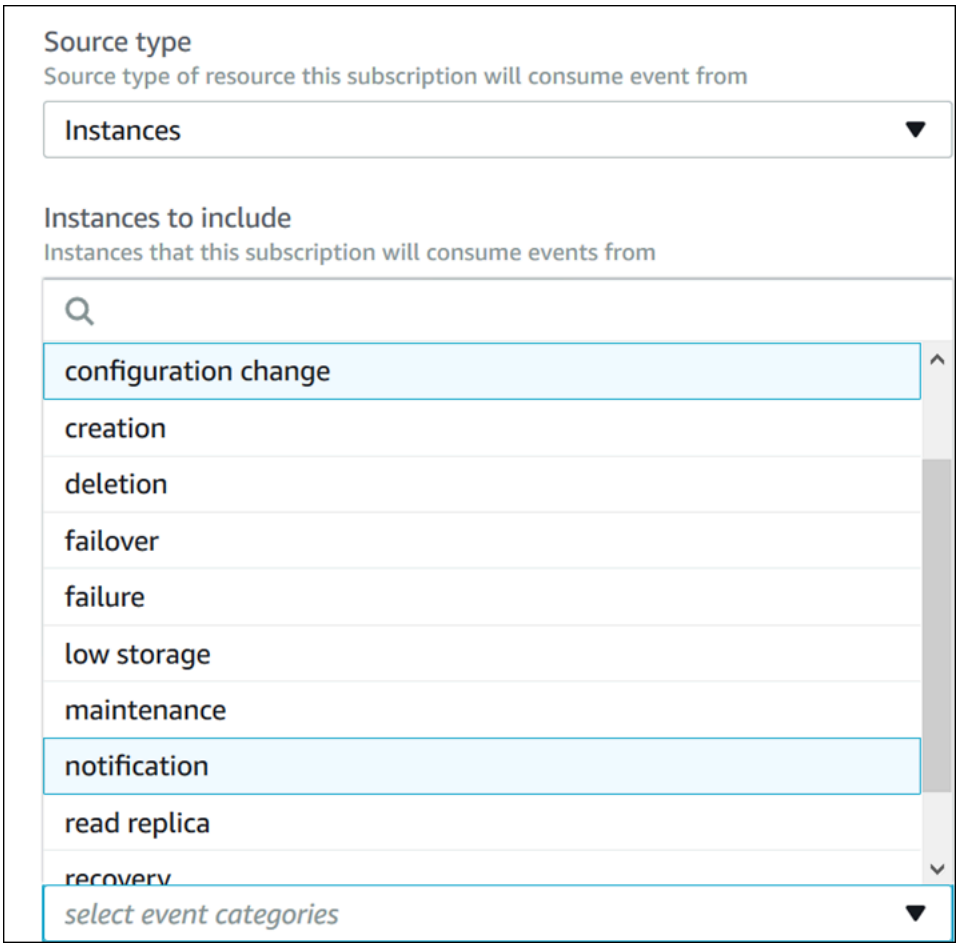
- SubscriptionName
- SourceIdentifier

Amazon RDS 이벤트 알림 카테고리의 목록 표시

리소스 유형의 이벤트는 모두 여러 카테고리로 그룹화됩니다. 이용 가능한 카테고리 목록을 보려면 다음 절차를 따릅니다.

콘솔

이벤트 알림 구독을 생성 또는 변경할 때는 이벤트 카테고리가 Amazon RDS 콘솔에 표시됩니다. 자세한 내용은 Amazon RDS 이벤트 알림 구독 수정 섹션을 참조하세요.



AWS CLI

Amazon RDS 이벤트 알림 범주를 표시하려면 AWS CLI `describe-event-categories` 명령을 사용합니다. 이 명령에는 필수 파라미터가 없습니다.

Example

```
aws rds describe-event-categories
```

API

Amazon RDS 이벤트 알림 범주를 표시하려면 Amazon RDS API `DescribeEventCategories` 명령을 사용합니다. 이 명령에는 필수 파라미터가 없습니다.

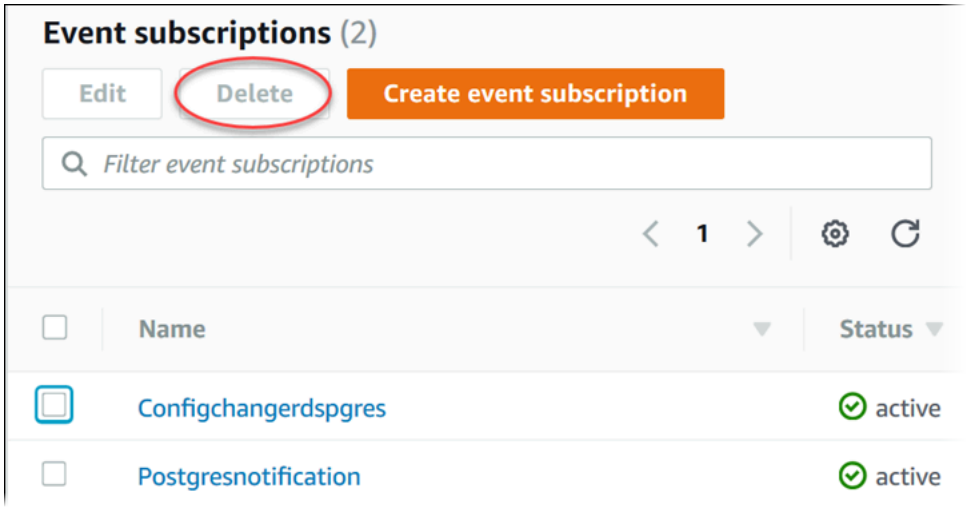
Amazon RDS 이벤트 알림 구독 삭제

필요 없는 구독은 삭제할 수 있습니다. 그러면 해당 주제의 모든 구독자에게는 구독 시 지정한 이벤트 알림 메시지가 발송되지 않습니다.

콘솔

Amazon RDS 이벤트 알림 구독을 삭제하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [DB Event Subscriptions]를 선택합니다.
3. [My DB Event Subscriptions] 창에서 삭제할 구독을 선택합니다.
4. Delete(삭제)를 선택합니다.
5. Amazon RDS 콘솔에 현재 구독 삭제 중으로 나옵니다.



AWS CLI

Amazon RDS 이벤트 알림 구독을 삭제하려면, AWS CLI `delete-event-subscription` 명령을 사용합니다. 다음 필수 파라미터를 포함합니다.

- `--subscription-name`

Example

다음은 `myrdssubscription` 구독을 삭제하는 예제입니다.

```
aws rds delete-event-subscription --subscription-name myrdssubscription
```

API

Amazon RDS 이벤트 알림 구독을 삭제하려면, RDS API DeleteEventSubscription 명령을 사용합니다. 다음 필수 파라미터를 포함합니다.

- SubscriptionName

Amazon RDS 이벤트에서 트리거되는 규칙 생성

Amazon EventBridge를 사용하면 AWS 서비스를 자동화하고 애플리케이션 가용성 문제나 리소스 변경 같은 시스템 이벤트에 응답할 수 있습니다.

주제

- Amazon RDS 이벤트를 Amazon EventBridge로 전송하는 규칙 생성
- 자습서: Amazon EventBridge를 사용하여 DB 인스턴스의 상태 변경 로깅

Amazon RDS 이벤트를 Amazon EventBridge로 전송하는 규칙 생성

관심이 있는 Amazon RDS 이벤트와 이벤트가 규칙과 일치할 때 수행할 자동화된 작업을 나타내는 단순한 규칙을 작성할 수 있습니다. AWS Lambda 함수나 Amazon SNS 주제 등 다양한 대상을 설정하고 JSON 형식으로 된 이벤트를 받을 수 있습니다. 예를 들어 DB 인스턴스가 생성되거나 삭제될 때마다 Amazon EventBridge에 이벤트를 전송하도록 Amazon RDS를 구성할 수 있습니다. 자세한 내용은 Amazon CloudWatch Events 사용 설명서 및 Amazon EventBridge 사용 설명서를 참조하세요.

RDS 이벤트에서 트리거되는 규칙 생성

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창의 이벤트 아래에서 규칙을 선택합니다.
3. 규칙 생성을 선택합니다.
4. [Event Source]에서 다음을 수행합니다.
   a. 이벤트 패턴을 선택합니다.
   b. 서비스 이름에서 Relational Database Service(RDS)를 선택합니다.
   c. 이벤트 유형(Event Type)에서 이벤트를 트리거할 Amazon RDS 리소스 유형을 선택합니다. 예를 들어, DB 인스턴스가 이벤트를 트리거하는 경우 RDS DB 인스턴스 이벤트(RDS DB Instance Event)를 선택합니다.
5. 대상에서 대상 추가를 선택한 후 선택한 유형의 이벤트가 감지되면 작동할 AWS 서비스를 선택합니다.
6. 필요할 경우 이 섹션의 다른 필드에 이 대상 유형에 관련된 정보를 입력합니다.
7. 여러 대상 유형에 대해 EventBridge에서는 대상에 이벤트를 보낼 권한이 필요합니다. 이 경우 EventBridge는 이벤트 실행에 필요한 IAM 역할을 생성할 수 있습니다.
   - IAM 역할을 자동으로 생성하려면 이 특정 리소스에 대해 새 역할 생성을 선택합니다.

- 이전에 생성한 IAM 역할을 사용하려면 기존 역할 사용을 선택합니다.

8. 선택적으로 5-7단계를 반복하여 이 규칙의 다른 대상을 추가합니다.
9. 세부 정보 구성을 선택합니다. 규칙 정의에 규칙의 이름과 규칙에 대한 설명을 입력합니다.

   규칙 이름은 이 리전 내에서 고유해야 합니다.
10. 규칙 생성을 선택합니다.

자세한 내용은 Amazon CloudWatch 사용 설명서의 이벤트에서 트리거되는 EventBridge 규칙 생성을 참조하세요.

자습서: Amazon EventBridge를 사용하여 DB 인스턴스의 상태 변경 로깅

이 자습서에서는 Amazon RDS 인스턴스의 상태 변경을 로깅하는 AWS Lambda 함수를 생성합니다. 그런 다음 기존 RDS DB 인스턴스의 상태가 변경될 때마다 함수를 실행하는 규칙을 생성합니다. 이 자습서에서는 일시적으로 종료할 수 있는 실행 중인 작은 테스트 인스턴스가 있다고 가정합니다.

> **Important**
>
> 실행 중인 프로덕션 DB 인스턴스에서 이 자습서를 수행하지 마세요.

주제

- 1단계: AWS Lambda 함수 생성
- 2단계: 규칙 생성
- 3단계: 규칙 테스트

1단계: AWS Lambda 함수 생성

Lambda 함수를 생성하여 상태 변경 이벤트를 기록합니다. 규칙을 생성할 때 이 함수를 지정합니다.

Lambda 함수를 만들려면

1. https://console.aws.amazon.com/lambda/에서 AWS Lambda 콘솔을 엽니다.
2. Lambda를 처음 사용하는 경우 시작 페이지가 표시됩니다. 지금 시작을 선택합니다. 그렇지 않은 경우에는 함수 생성을 선택합니다.
3. 새로 작성을 선택합니다.

4. 함수 생성 페이지에서 다음을 수행합니다.

   a. Lambda 함수의 이름과 설명을 입력합니다. 예를 들어 함수 이름을 **RDSInstanceStateChange**로 지정합니다.
   b. 런타임에서 Node.js 16x를 선택합니다.
   c. 아키텍처에서는 x86_64를 선택합니다.
   d. 실행 역할에서는 다음 중 하나를 수행합니다.

      - 기본 Lambda 권한을 가진 새 역할 생성을 선택합니다.
      - 기존 역할에서는 기존 역할 사용을 선택합니다. 사용하려는 역할을 선택합니다.

   e. 함수 생성을 선택합니다.
5. [RDSInstanceStateChange] 페이지에서 다음을 수행합니다.

   a. 코드 소스에서 index.js를 선택합니다.
   b. index.js 창에서 기존 코드를 삭제합니다.
   c. 다음 코드를 입력합니다.

```
console.log('Loading function');

exports.handler = async (event, context) => {
    console.log('Received event:', JSON.stringify(event));
};
```

   d. [Deploy]를 선택합니다.

2단계: 규칙 생성

Amazon RDS 인스턴스를 시작할 때마다 Lambda 함수를 실행하는 규칙을 생성합니다.

EventBridge 규칙을 만들려면

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 규칙을 선택합니다.
3. 규칙 생성을 선택합니다.
4. 규칙에 대해 이름과 설명을 입력하십시오. 예를 들면 **RDSInstanceStateChangeRule**을 입력합니다.
5. 이벤트 패턴이 있는 규칙을 선택한 후다음을 선택합니다.

6. 이벤트 소스에서 AWS 이벤트 또는 EventBridge 파트너 이벤트를 선택합니다.
7. 이벤트 패턴 섹션까지 아래로 스크롤합니다.
8. 이벤트 소스에서 AWS 서비스를 선택합니다.
9. AWS 서비스에서는 Relational Database Service(RDS)를 선택합니다.
10. 이벤트 유형에서 RDS DB 인스턴스 이벤트를 선택합니다.
11. 기본 이벤트 패턴을 그대로 둡니다. 그리고 다음을 선택합니다.
12. 대상 유형에서 AWS 서비스를 선택합니다.
13. 대상 선택에서 Lambda 함수를 선택합니다.
14. 함수에서는 생성한 Lambda 함수를 선택합니다. 다음을 선택합니다.
15. 태그 구성에서는 다음을 선택합니다.
16. 규칙의 단계를 검토하십시오. 그런 다음 규칙 생성을 선택합니다.

3단계: 규칙 테스트

규칙을 테스트하려면 RDS DB 인스턴스를 종료합니다. 인스턴스가 종료될 때까지 몇 분 기다린 후에 Lambda 함수가 호출되었는지 확인합니다.

DB 인스턴스를 중지하여 규칙을 테스트하려면

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. RDS DB 인스턴스를 중지합니다.
3. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
4. 탐색 창에서 [규칙(Rules)]을 선택하고 생성한 규칙의 이름을 선택합니다.
5. 규칙 세부 정보에서 모니터링을 선택합니다.

   그러면 Amazon CloudWatch 콘솔로 리디렉션됩니다. 리디렉션되지 않은 경우 CloudWatch에서 지표 보기를 클릭합니다.
6. [모든 지표(All metrics)]에서 생성한 규칙의 이름을 선택합니다.

   그래프에 규칙이 호출된 것으로 표시되어야 합니다.
7. 탐색 창에서 로그 그룹을 선택합니다.
8. Lambda 함수에 대한 로그 그룹 이름(/aws/lambda/*function-name*)을 선택합니다.
9. 로그 스트림 이름을 선택하여 시작한 인스턴스에서 함수를 통해 제공된 데이터를 확인합니다. 다음과 유사한 수신된 이벤트가 표시되어야 합니다.

```
{
    "version": "0",
    "id": "12a345b6-78c9-01d2-34e5-123f4ghi5j6k",
    "detail-type": "RDS DB Instance Event",
    "source": "aws.rds",
    "account": "111111111111",
    "time": "2021-03-19T19:34:09Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:rds:us-east-1:111111111111:db:testdb"
    ],
    "detail": {
        "EventCategories": [
            "notification"
        ],
        "SourceType": "DB_INSTANCE",
        "SourceArn": "arn:aws:rds:us-east-1:111111111111:db:testdb",
        "Date": "2021-03-19T19:34:09.293Z",
        "Message": "DB instance stopped",
        "SourceIdentifier": "testdb",
        "EventID": "RDS-EVENT-0087"
    }
}
```

JSON 형식의 RDS 이벤트에 대한 자세한 예는 Amazon RDS에 대한 이벤트 개요 단원을 참조하세요.

10. (선택 사항) 작업이 완료되면 Amazon RDS 콘솔을 열고 중지한 인스턴스를 시작할 수 있습니다.

Amazon RDS 이벤트 카테고리 및 이벤트 메시지

Amazon RDS는 Amazon RDS 콘솔, AWS CLI 또는 API를 사용하여 구독할 수 있는 이벤트 카테고리가 매우 많습니다.

주제

- DB 클러스터 이벤트
- DB 인스턴스 이벤트
- DB 파라미터 그룹 이벤트
- DB 보안 그룹 이벤트
- DB 스냅샷 이벤트
- DB 클러스터 스냅샷 이벤트
- RDS 프록시 이벤트
- 블루/그린 배포 이벤트
- 사용자 지정 엔진 버전 이벤트

DB 클러스터 이벤트

다음 표에는 DB 클러스터가 원본 유형일 때 이벤트 카테고리와 이벤트 목록이 나와 있습니다.

다중 AZ DB 클러스터 배포에 대한 자세한 내용은 Amazon RDS용 다중 AZ DB 클러스터 배포 섹션을 참조하세요.

Category	RDS 이벤트 ID	메시지	참고
구성 변경	RDS-EVENT-0016	마스터 보안 인증 정보를 재설정하세요.	
생성	RDS-EVENT-0170	DB 클러스터가 생성되었습니다.	
장애 조치	RDS-EVENT-0069	클러스터 장애 조치가 실패했습니다. 클러스터 인스턴스의 상태를 확인하고 다시 시도하세요.	

Category	RDS 이벤트 ID	메시지	참고
장애 조치	RDS-EVENT-0070	이전 프라이머리(##)를 다시 승격하는 중입니다.	
장애 조치	RDS-EVENT-0071	DB 인스턴스(##)로 장애 조치를 완료했습니다.	
장애 조치	RDS-EVENT-0072	DB 인스턴스(##)에 대한 동일한 AZ 장애 조치를 시작했습니다.	
장애 조치	RDS-EVENT-0073	DB 인스턴스(##)에 대한 크로스 AZ 장애 조치를 시작했습니다.	
실패	RDS-EVENT-0354	호환되지 않는 리소스 때문에 DB 클러스터를 생성할 수 없습니다. ###.	###에는 실패에 대한 세부 정보가 포함되어 있습니다.
실패	RDS-EVENT-0355	리소스 제한이 충분하지 않아 DB 클러스터를 생성할 수 없습니다. ###.	###에는 실패에 대한 세부 정보가 포함되어 있습니다.
글로벌 장애 조치	RDS-EVENT-0181	리전(##)의 DB 클러스터(##)에 대한 글로벌 전환이 시작되었습니다.	이 이벤트는 전환 작업에 해당합니다(이전 명칭: '계획된 관리형 장애 조치'). DB 클러스터에서 다른 작업이 실행되고 있기 때문에 프로세스가 지연될 수 있습니다.

Category	RDS 이벤트 ID	메시지	참고
글로벌 장애 조치	RDS-EVENT-0182	리전(##)의 이전 프라이머리 DB 클러스터(##)가 성공적으로 종료되었습니다.	이 이벤트는 전환 작업에 해당합니다(이전 명칭: '계획된 관리형 장애 조치'). 글로벌 데이터베이스의 이전 기본 인스턴스는 쓰기를 허용하지 않습니다. 모든 볼륨이 동기화됩니다.
글로벌 장애 조치	RDS-EVENT-0183	전역 클러스터 멤버 간의 데이터 동기화를 기다리는 중입니다. 현재 프라이머리 DB 클러스터보다 지연되고 있습니다. ##.	이 이벤트는 전환 작업에 해당합니다(이전 명칭: '계획된 관리형 장애 조치'). 글로벌 데이터베이스 장애 조치의 동기화 단계 중에 복제 지연이 발생합니다.
글로벌 장애 조치	RDS-EVENT-0184	리전(##)의 새로운 프라이머리 DB 클러스터(##)가 성공적으로 승격되었습니다.	이 이벤트는 전환 작업에 해당합니다(이전 명칭: '계획된 관리형 장애 조치'). 글로벌 데이터베이스의 볼륨 토폴로지가 새 기본 볼륨으로 다시 설정됩니다.
글로벌 장애 조치	RDS-EVENT-0185	리전(##)의 DB 클러스터(##)에 대한 글로벌 전환이 완료되었습니다.	이 이벤트는 전환 작업에 해당합니다(이전 명칭: '계획된 관리형 장애 조치'). 글로벌 데이터베이스 전환이 기본 DB 클러스터에서 완료됩니다. 장애 조치가 완료된 후 복제본이 온라인 상태로 전환되는 데 오랜 시간이 걸릴 수 있습니다.

Category	RDS 이벤트 ID	메시지	참고
글로벌 장애 조치	RDS-EVENT-0186	리전(##)의 DB 클러스터(##)에 대한 글로벌 전환이 취소되었습니다.	이 이벤트는 전환 작업에 해당합니다(이전 명칭: '계획된 관리형 장애 조치').
글로벌 장애 조치	RDS-EVENT-0187	리전(##)의 DB 클러스터(##)에 대한 글로벌 전환이 실패했습니다.	이 이벤트는 전환 작업에 해당합니다(이전 명칭: '계획된 관리형 장애 조치').
글로벌 장애 조치	RDS-EVENT-0238	리전(##)의 DB 클러스터(##)에 대한 글로벌 장애 조치가 완료되었습니다.	
글로벌 장애 조치	RDS-EVENT-0239	리전(##)의 DB 클러스터(##)에 대한 글로벌 장애 조치가 실패했습니다.	
글로벌 장애 조치	RDS-EVENT-0240	리전(##)의 DB 클러스터(##)에 대한 재동기화 구성 요소가 글로벌 장애 조치 후에 시작되었습니다.	
글로벌 장애 조치	RDS-EVENT-0241	리전(##)의 DB 클러스터(##)에 대한 재동기화 구성 요소가 글로벌 장애 조치 후에 완료되었습니다.	
유지 관리	RDS-EVENT-0156	DB 클러스터에 사용 가능한 DB 엔진 마이너 버전 업그레이드가 있습니다.	
유지 관리	RDS-EVENT-0176	데이터베이스 클러스터 엔진 메이저 버전이 업그레이드되었습니다.	
유지 관리	RDS-EVENT-0177	데이터베이스 클러스터 업그레이드가 진행 중입니다.	

Category	RDS 이벤트 ID	메시지	참고
유지 관리	RDS-EVENT-0286	데이터베이스 클러스터 엔진 버전 업그레이드가 시작되었습니다.	
유지 관리	RDS-EVENT-0287	운영 체제 업그레이드 요구 사항이 감지되었습니다.	
유지 관리	RDS-EVENT-0288	클러스터 운영 체제 업그레이드를 시작하는 중입니다.	
유지 관리	RDS-EVENT-0289	클러스터 운영 체제 업그레이드가 완료되었습니다.	
알림	RDS-EVENT-0172	클러스터 이름을 ##에서 ##으로 변경했습니다.	

DB 인스턴스 이벤트

다음 표는 DB 인스턴스가 소스 유형일 때 이벤트 카테고리와 이벤트 목록을 나타냅니다.

범주	RDS 이벤트 ID	메시지	참고
가용성	RDS-EVENT-0004	DB 인스턴스 종료.	
가용성	RDS-EVENT-0006	DB 인스턴스가 다시 시작되었습니다.	
가용성	RDS-EVENT-0022	MySQL을 다시 시작하는 중 오류가 발생했습니다. ###.	MySQL를 다시 시작하는 중 오류가 발생했습니다.
가용성	RDS-EVENT-0221	DB 인스턴스가 스토리지 전체 임계값에 도달했으며 데이터베이스가 종료되었습니다. 이 문제를 해결하기 위해 할당된 스토리지를 늘릴 수 있습니다.	

범주	RDS 이벤트 ID	메시지	참고
가용성	RDS-EVENT-0222	DB 인스턴스(##)의 사용 가능한 스토리지 용량이 할당된 스토리지의 ###로 낮습니다[할당된 스토리지: ##, 사용 가능한 스토리지: ##]. 사용 가능한 스토리지가 ##보다 낮은 경우 손상을 방지하기 위해 데이터베이스가 종료됩니다. 이 문제를 해결하기 위해 할당된 스토리지를 늘릴 수 있습니다.	자세한 내용은 Amazon RDS DB 인스턴스 스토리지 단원을 참조하십시오.
가용성	RDS-EVENT-0330	전용 트랜잭션 로그 볼륨의 여유 스토리지 용량이 DB 인스턴스 ##에 비해 너무 낮습니다. 로그 볼륨 여유 스토리지는 할당된 스토리지의 ###입니다. [할당된 스토리지: ##, 여유 스토리지: ##] 여유 스토리지가 ##보다 낮은 경우 손상을 방지하기 위해 데이터베이스가 종료됩니다. 전용 트랜잭션 로그 볼륨을 비활성화하여 이 문제를 해결할 수 있습니다.	자세한 내용은 전용 로그 볼륨(DLV) 단원을 참조하십시오.

범주	RDS 이벤트 ID	메시지	참고
가용성	RDS-EVENT-0331	전용 트랜잭션 로그 볼륨의 여유 스토리지 용량이 DB 인스턴스 ##에 비해 너무 낮습니다. 로그 볼륨 여유 스토리지는 프로비저닝된 스토리지의 ###입니다. [프로비저닝된 스토리지: ##, 여유 스토리지: ##] 전용 트랜잭션 로그 볼륨을 비활성화하여 이 문제를 해결할 수 있습니다.	자세한 내용은 전용 로그 볼륨(DLV) 단원을 참조하십시오.
가용성	RDS-EVENT-0396	Amazon RDS에서 내부 사용자 암호 교체 후 인스턴스의 다음 유지 관리 기간에 이 읽기 전용 복제본의 재부팅을 예약했습니다.	해당 사항 없음
백업	RDS-EVENT-0001	DB 인스턴스 백업.	
백업	RDS-EVENT-0002	DB 인스턴스 백업이 완료되었습니다.	
백업	RDS-EVENT-0086	옵션 그룹(##)을 데이터베이스 인스턴스(##)와 연결할 수 없습니다. 옵션 그룹(##)이 DB 인스턴스 클래스 및 구성에서 지원되는지 확인하세요. 지원되는 경우 모든 옵션 그룹 설정을 확인하고 다시 시도하세요.	자세한 정보는 옵션 그룹 작업 섹션을 참조하세요.
구성 변경	RDS-EVENT-0011	DBParameterGroup ##을 사용하도록 업데이트되었습니다.	

범주	RDS 이벤트 ID	메시지	참고
구성 변경	RDS-EVENT-0012	수정을 데이터베이스 인스턴스 클래스에 적용.	
구성 변경	RDS-EVENT-0014	DB 인스턴스 클래스에 대한 수정 적용이 완료되었습니다.	
구성 변경	RDS-EVENT-0016	마스터 보안 인증 정보를 재설정하세요.	
구성 변경	RDS-EVENT-0017	할당된 스토리지에 대한 수정 적용이 완료되었습니다.	
구성 변경	RDS-EVENT-0018	할당된 스토리지에 수정을 적용하는 중입니다.	
구성 변경	RDS-EVENT-0024	다중 AZ DB 인스턴스로 전환하기 위한 수정을 적용하는 중입니다.	
구성 변경	RDS-EVENT-0025	다중 AZ DB 인스턴스로 전환하기 위한 수정 적용이 완료되었습니다.	
구성 변경	RDS-EVENT-0028	자동 백업이 비활성화되었습니다.	
구성 변경	RDS-EVENT-0029	표준(단일 AZ) DB 인스턴스로 전환하기 위한 수정 적용이 완료되었습니다.	
구성 변경	RDS-EVENT-0030	표준(단일 AZ) DB 인스턴스로 전환하기 위한 수정을 적용하는 중입니다.	
구성 변경	RDS-EVENT-0032	자동 백업이 활성화되었습니다.	

범주	RDS 이벤트 ID	메시지	참고
구성 변경	RDS-EVENT-0033	마스터 사용자 이름과 일치하는 사용자가 ##명 있습니다. 특정 호스트에 연결되지 않은 사용자만 재설정합니다.	
구성 변경	RDS-EVENT-0067	암호를 재설정할 수 없습니다. 오류 정보: ###.	
구성 변경	RDS-EVENT-0078	모니터링 간격이 ##로 변경되었습니다.	Enhanced Monitoring 구성이 변경되었습니다.
구성 변경	RDS-EVENT-0092	DB 파라미터 그룹의 업데이트를 완료했습니다.	
구성 변경	RDS-EVENT-0217	할당된 스토리지에 자동 크기 조정 시작 수정 적용	
구성 변경	RDS-EVENT-0218	할당된 스토리지에 자동 크기 조정 시작 수정 적용을 마쳤습니다.	
구성 변경	RDS-EVENT-0295	스토리지 구성 업그레이드가 시작되었습니다.	
구성 변경	RDS-EVENT-0296	스토리지 구성 업그레이드가 완료되었습니다.	
구성 변경	RDS-EVENT-0332	전용 로그 볼륨이 비활성화되었습니다.	자세한 내용은 전용 로그 볼륨(DLV) 단원을 참조하십시오.
구성 변경	RDS-EVENT-0333	전용 로그 볼륨의 비활성화가 시작되었습니다.	자세한 내용은 전용 로그 볼륨(DLV) 단원을 참조하십시오.

범주	RDS 이벤트 ID	메시지	참고
구성 변경	RDS-EVENT-0334	전용 로그 볼륨의 활성화가 시작되었습니다.	자세한 내용은 전용 로그 볼륨(DLV) 단원을 참조하십시오.
구성 변경	RDS-EVENT-0335	전용 로그 볼륨이 활성화되었습니다.	자세한 내용은 전용 로그 볼륨(DLV) 단원을 참조하십시오.
구성 변경	RDS-EVENT-0383	## ##에서 memcached 플러그인을 지원하지 않습니다. RDS는 계속해서 DB 인스턴스를 업그레이드하고, 이 플러그인을 제거합니다.	MySQL 8.3.0부터 memcached 플러그인은 지원되지 않습니다. 자세한 내용은 MySQL 8.3.0의 변경 사항(2024-01-16, Innovation 릴리스)을 참조하세요.
생성	RDS-EVENT-0005	생성된 DB 인스턴스.	
삭제	RDS-EVENT-0003	DB 인스턴스가 삭제되었습니다.	
장애 조치	RDS-EVENT-0013	다중 AZ 인스턴스 장애 조치가 시작되었습니다.	예비 DB 인스턴스의 승격 원인이었던 다중 AZ 장애 조치가 시작되었습니다.
장애 조치	RDS-EVENT-0015	예비로의 다중 AZ 장애 조치가 완료되었습니다. DNS 전파에는 몇 분 정도 걸릴 수 있습니다.	예비 DB 인스턴스의 승격 원인이었던 다중 AZ 장애 조치가 완료되었습니다. DNS가 새로운 기본 DB 인스턴스로 이전하는 데 몇 분 걸릴 수 있습니다.
장애 조치	RDS-EVENT-0034	최근에 데이터베이스 인스턴스에 장애 조치가 발생하였기 때문에 사용자가 요청한 장애 조치를 중단합니다.	최근에 DB 인스턴스에 장애 조치가 발생하였기 때문에 Amazon RDS가 요청된 장애 조치를 시도하지 않습니다.

범주	RDS 이벤트 ID	메시지	참고
장애 조치	RDS-EVENT-0049	다중 AZ 인스턴스 장애 조치가 완료되었습니다.	
장애 조치	RDS-EVENT-0050	다중 AZ 인스턴스 활성화가 시작되었습니다.	성공적인 DB 인스턴스 복구 후 다중 AZ 활성화가 시작되었습니다.
장애 조치	RDS-EVENT-0051	다중 AZ 인스턴스 활성화가 완료되었습니다.	다중 AZ 활성화가 완료되었습니다. 이제 데이터베이스에 액세스할 수 있습니다.
장애 조치	RDS-EVENT-0065	부분 장애 조치에서 복구되었습니다.	
실패	RDS-EVENT-0031	DB 인스턴스가 ## 상태로 전환되었습니다. RDS는 특정 시점 복원을 시작할 것을 권장합니다.	호환되지 않는 구성 또는 기본 스토리지 문제로 인해 DB 인스턴스에 장애가 발생했습니다. DB 인스턴스에 대해 특정 시점으로 복구를 시작합니다.
실패	RDS-EVENT-0035	데이터베이스 인스턴스가 ##로 전환되었습니다. ###.	DB 인스턴스에 잘못된 파라미터가 있습니다. 예를 들어 이 인스턴스 클래스의 메모리 관련 파라미터가 너무 높게 설정되어 있어 DB 인스턴스가 시작되지 않는 경우 사용자가 할 수 있는 작업은 메모리 파라미터를 수정하고 DB 인스턴스를 재부팅하는 것입니다.

범주	RDS 이벤트 ID	메시지	참고
실패	RDS-EVENT-0036	데이터베이스 인스턴스가 ## 상태입니다. ###.	DB 인스턴스가 호환되지 않는 네트워크에 있습니다. 특정 서브넷 ID 중 일부가 잘못되었거나 존재하지 않습니다.
실패	RDS-EVENT-0058	Statspack 설치에 실패했습니다. ###.	Oracle Statspack 사용자 계정인 PERFSTAT 생성 중 오류가 발생하였습니다. STATSPACK 옵션을 추가하기 전에 해당 계정을 삭제하세요.
실패	RDS-EVENT-0079	Amazon RDS가 확장 모니터링을 위한 보안 인증 정보를 생성할 수 없으며 이 기능은 비활성화되었습니다. 이는 rds-monitoring-role이 존재하지 않고 계정에 올바르게 구성되어 있지 않기 때문일 수 있습니다. 자세한 내용은 Amazon RDS 설명서의 문제 해결 섹션을 참조하세요.	확장 모니터링을 활성화하려면 확장 모니터링 IAM 역할이 있어야 합니다. IAM 역할 생성에 대한 자세한 내용은 Amazon RDS Enhanced Monitoring에 대한 IAM 역할을 생성하려면 섹션을 참조하세요.

범주	RDS 이벤트 ID	메시지	참고
실패	RDS-EVENT-0080	Amazon RDS가 인스턴스(##)에 확장 모니터링을 구성할 수 없으며 이 기능은 비활성화되었습니다. 이는 rds-monitoring-role이 존재하지 않고 계정에 올바르게 구성되어 있지 않기 때문일 수 있습니다. 자세한 내용은 Amazon RDS 설명서의 문제 해결 섹션을 참조하세요.	구성을 변경하는 동안 오류가 발생하여 확장 모니터링이 비활성화되었습니다. 확장 모니터링 IAM 역할이 잘못 구성된 것 같습니다. 확장 모니터링 IAM 역할을 생성하는 방법에 대한 자세한 내용은 Amazon RDS Enhanced Monitoring에 대한 IAM 역할을 생성하려면 섹션을 참조하세요.
실패	RDS-EVENT-0081	Amazon RDS에서 ## 옵션에 대한 보안 인증 정보를 생성할 수 없습니다. 이는 계정에 ## IAM 역할이 올바르게 구성되지 않았기 때문입니다. 자세한 내용은 Amazon RDS 설명서의 문제 해결 섹션을 참조하세요.	SQL Server 기본 백업 및 복원을 위해 Amazon S3 버킷에 액세스하는 데 사용하는 IAM 역할이 잘못 구성되었습니다. 자세한 내용은 기본 백업 및 복원 설정 섹션을 참조하세요.

범주	RDS 이벤트 ID	메시지	참고
실패	RDS-EVENT-0165	RDS Custom DB 인스턴스가 지원 경계를 벗어났습니다.	RDS Custom DB 인스턴스를 `unsupported-configuration` 상태로 전환시키는 구성 문제는 사용자가 해결해야 합니다. AWS 인프라와 관련된 문제인 경우 콘솔 또는 AWS CLI를 사용하여 해결할 수 있습니다. 운영 체제 또는 데이터베이스 구성에 문제가 있는 경우 호스트에 로그인하여 해결하면 됩니다. 자세한 내용은 RDS Custom 지원 범위 단원을 참조하십시오.
실패	RDS-EVENT-0188	DB 인스턴스가 업그레이드할 수 없는 상태입니다. ###.	데이터 딕셔너리와 관련한 비호환성 때문에 Amazon RDS가 MySQL DB 인스턴스를 버전 5.7에서 버전 8.0으로 업그레이드할 수 없습니다. DB 인스턴스가 MySQL 버전 5.7로 롤백되었습니다. 자세한 내용은 업그레이드 실패 후 롤백 단원을 참조하십시오.
실패	RDS-EVENT-0219	DB 인스턴스가 잘못된 상태입니다. 아무 조치도 필요하지 않습니다. 자동 확장은 나중에 다시 시도합니다.	

범주	RDS 이벤트 ID	메시지	참고
실패	RDS-EVENT-0220	DB 인스턴스가 이전 규모 조정 스토리지 작업의 냉각 기간 내에 있습니다. DB 인스턴스를 최적화하고 있습니다. 이 작업은 최소 6시간이 걸립니다. 아무 조치도 필요하지 않습니다. 자동 확장은 냉각 기간 후에 다시 시도합니다.	
실패	RDS-EVENT-0223	스토리지 자동 규모 조정이 다음 이유로 스토리지의 규모를 조정할 수 없습니다. ##.	
실패	RDS-EVENT-0224	스토리지 자동 규모 조정이 보류 중인 규모 조정 스토리지 작업을 트리거했으며, 이로 인해 최대 스토리지 임계값에 도달하거나 이를 초과하게 됩니다. 최대 스토리지 임계값을 늘립니다.	
실패	RDS-EVENT-0237	DB 인스턴스에 현재 가용 영역에서 사용할 수 없는 스토리지 유형이 있습니다. 자동 확장은 나중에 다시 시도합니다.	
실패	RDS-EVENT-0254	이 고객 계정의 기본 스토리지 할당량이 한도를 초과했습니다. 인스턴스에서 규모 조정이 진행되도록 허용된 스토리지 할당량을 늘리세요.	

범주	RDS 이벤트 ID	메시지	참고
실패	RDS-EVENT-0278	DB 인스턴스 생성에 실패했습니다. ###	###에는 실패에 대한 세부 정보가 포함되어 있습니다.
실패	RDS-EVENT-0279	RDS Custom 읽기 전용 복제본의 승격이 실패했습니다. ###	###에는 실패에 대한 세부 정보가 포함되어 있습니다.
실패	RDS-EVENT-0280	사전 검사에 실패했기 때문에 RDS Custom에서 DB 인스턴스를 업그레이드하지 못했습니다. ###	###에는 실패에 대한 세부 정보가 포함되어 있습니다.
실패	RDS-EVENT-0281	사전 검사에 실패했기 때문에 RDS Custom에서 DB 인스턴스를 수정하지 못했습니다. ###	###에는 실패에 대한 세부 정보가 포함되어 있습니다.
실패	RDS-EVENT-0282	탄력적 IP 권한이 올바르지 않기 때문에 RDS Custom에서 DB 인스턴스를 수정하지 못했습니다. 탄력적 IP 주소에 `AWSRDSCustom` 태그가 지정되었는지 확인하세요.	
실패	RDS-EVENT-0283	계정에서 탄력적 IP 한도에 도달했기 때문에 RDS Custom에서 DB 인스턴스를 수정하지 못했습니다. 사용하지 않는 탄력적 IP를 해제하거나 탄력적 IP 주소 한도에 대한 할당량 증가를 요청하세요.	

범주	RDS 이벤트 ID	메시지	참고
실패	RDS-EVENT-0284	사전 검사에 실패했기 때문에 RDS Custom에서 인스턴스를 고가용성으로 변환하지 못했습니다. *###*	*###*에는 실패에 대한 세부 정보가 포함되어 있습니다.
실패	RDS-EVENT-0285	*###* 때문에 RDS Custom에서 DB 인스턴스의 최종 스냅샷을 만들지 못했습니다.	*###*에는 실패에 대한 세부 정보가 포함되어 있습니다.
실패	RDS-EVENT-0306	스토리지 구성 업그레이드에 실패했습니다. 업그레이드를 다시 시도하세요.	
실패	RDS-EVENT-0315	호환되지 않는 네트워크 데이터베이스(*##*)를 사용 가능한 상태(*###*)로 이전할 수 없음	데이터베이스 네트워킹 구성이 유효하지 않습니다. 데이터베이스는 호환되지 않는 네트워크에서 사용 가능한 네트워크로 이전이 불가능합니다.
실패	RDS-EVENT-0328	호스트를 도메인에 조인시키지 못했습니다. 인스턴스 *instancename* 의 도메인 멤버십 상태가 '실패'로 설정되었습니다.	
실패	RDS-EVENT-0329	호스트를 도메인에 조인시키지 못했습니다. 도메인 조인 프로세스 중에 Microsoft Windows에서 오류 코드 *###*를 반환했습니다. 네트워크 및 권한 구성을 확인하고 도메인 조인을 다시 시도하도록 `modify-db-instance` 요청을 진행하세요.	자체 관리형 Active Directory를 사용하는 경우 자체 관리형 Active Directory 문제 해결 섹션을 참조하세요.

범주	RDS 이벤트 ID	메시지	참고
실패	RDS-EVENT-0353	리소스 제한이 충분하지 않아 DB 인스턴스를 생성할 수 없습니다. ###.	###에는 실패에 대한 세부 정보가 포함되어 있습니다.
실패	RDS-EVENT-0356	RDS가 도메인의 Kerberos 엔드포인트를 구성하지 못했습니다. 이로 인해 DB 인스턴스에 대한 Kerberos 인증이 차단될 수 있습니다. DB 인스턴스와 도메인 컨트롤러 간의 네트워크 구성을 확인하세요.	
적은 스토리지	RDS-EVENT-0007	할당된 스토리지가 모두 소진되었습니다. 해결하려면 추가 스토리지를 할당하세요.	DB 인스턴스에 할당된 스토리지를 모두 사용했습니다. 이 문제를 해결하려면 DB 인스턴스에 스토리지를 추가 할당하세요. 자세한 내용은 RDS FAQ 단원을 참조하십시오. [Free Storage Space] 측정치를 사용하여 DB 인스턴스에 대한 스토리지 공간을 모니터링할 수 있습니다.
적은 스토리지	RDS-EVENT-0089	DB 인스턴스(##)의 사용 가능한 스토리지 용량이 프로비저닝된 스토리지의 ##%로 낮습니다[프로비저닝된 스토리지: ##, 사용 가능한 스토리지: ##]. 이 문제를 해결하려면 프로비저닝된 스토리지를 늘리는 것이 좋습니다.	DB 인스턴스가 할당된 스토리지의 90% 이상을 사용하였습니다. [Free Storage Space] 측정치를 사용하여 DB 인스턴스에 대한 스토리지 공간을 모니터링할 수 있습니다.

범주	RDS 이벤트 ID	메시지	참고
적은 스토리지	RDS-EVENT-0227	Aurora 클러스터의 스토리지가 ##테라바이트만 남아 있어 위험할 정도로 부족합니다. 클러스터의 스토리지 로드를 줄이기 위한 조치를 취하세요.	Aurora 스토리지 하위 시스템의 공간이 부족합니다.
유지 관리	RDS-EVENT-0026	DB 인스턴스에 오프라인 패치를 적용하는 중입니다.	DB 인스턴스의 오프라인 유지 관리가 진행 중입니다. 따라서 현재 DB 인스턴스는 사용할 수 없습니다.
유지 관리	RDS-EVENT-0027	DB 인스턴스에 대한 오프라인 패치 적용이 완료되었습니다.	DB 인스턴스의 오프라인 유지 관리가 완료되었습니다. 이제 DB 인스턴스를 사용할 수 있습니다.
유지 관리	RDS-EVENT-0047	데이터베이스 인스턴스가 패치되었습니다.	
유지 관리	RDS-EVENT-0155	DB 인스턴스에 사용 가능한 DB 엔진 마이너 버전 업그레이드가 있습니다.	
유지 관리	RDS-EVENT-0178	데이터베이스 인스턴스 업그레이드가 진행 중입니다.	
유지 관리	RDS-EVENT-0264	DB 엔진 버전 업그레이드를 위한 사전 점검이 시작되었습니다.	
유지 관리	RDS-EVENT-0265	DB 엔진 버전 업그레이드를 위한 사전 점검이 완료되었습니다.	
유지 관리	RDS-EVENT-0266	DB 인스턴스의 가동 중지 시간이 시작되었습니다.	

범주	RDS 이벤트 ID	메시지	참고
유지 관리	RDS-EVENT-0267	엔진 버전 업그레이드가 시작되었습니다.	
유지 관리	RDS-EVENT-0268	엔진 버전 업그레이드가 완료되었습니다.	
유지 관리	RDS-EVENT-0269	업그레이드 후 작업이 진행 중입니다.	
유지 관리	RDS-EVENT-0270	DB 엔진 버전 업그레이드에 실패했습니다. 엔진 버전 업그레이드 롤백에 성공했습니다.	
유지 관리, 실패	RDS-EVENT-0195	*message*	Oracle 표준 시간대 파일을 업데이트하지 못했습니다. 자세한 내용은 Oracle 시간대 파일 자동 업그레이드 단원을 참조하십시오.
유지 관리, 알림	RDS-EVENT-0191	시간대 파일의 새 버전을 업데이트할 수 있습니다.	RDS for Oracle DB 엔진을 업데이트하는 경우, 표준 시간대 파일 업그레이드를 선택하지 않았고 데이터베이스가 인스턴스에 사용 가능한 최신 DST 표준 시간대 파일을 사용하지 않는 경우 Amazon RDS가 이 이벤트를 생성합니다. 자세한 내용은 Oracle 시간대 파일 자동 업그레이드 단원을 참조하십시오.

범주	RDS 이벤트 ID	메시지	참고
유지 관리, 알림	RDS-EVENT-0192	표준 시간대 파일의 업데이트가 시작되었습니다.	Oracle 표준 시간대 파일의 업그레이드가 시작되었습니다. 자세한 내용은 Oracle 시간대 파일 자동 업그레이드 단원을 참조하십시오.
유지 관리, 알림	RDS-EVENT-0193	현재 표준 시간대 파일 버전에 사용할 수 있는 업데이트가 없습니다.	Oracle DB 인스턴스가 최신 표준 시간대 파일 버전을 사용하고 있으며 다음 중 하나에 해당합니다. • 최근에 TIMEZONE_FILE_AUTOUPGRADE 옵션을 추가했습니다. • 현재 Oracle DB 엔진을 업그레이드 중입니다. 자세한 내용은 Oracle 시간대 파일 자동 업그레이드 단원을 참조하십시오.
유지 관리, 알림	RDS-EVENT-0194	표준 시간대 파일의 업데이트가 완료되었습니다.	Oracle 표준 시간대 파일의 업데이트가 완료되었습니다. 자세한 내용은 Oracle 시간대 파일 자동 업그레이드 단원을 참조하십시오.
알림	RDS-EVENT-0044	*message*	운영자가 발행한 알림입니다. 자세한 내용은 이벤트 메시지 단원을 참조하십시오.

범주	RDS 이벤트 ID	메시지	참고
알림	RDS-EVENT-0048	이 인스턴스에는 먼저 업그레이드해야 하는 읽기 전용 복제본이 있으므로 데이터베이스 엔진 업그레이드가 지연됩니다.	DB 인스턴스의 패치 작업이 지연되었습니다.
알림	RDS-EVENT-0054	*message*	사용 중인 MySQL 스토리지 엔진이 InnoDB가 아닙니다. Amazon RDS는 MySQL 스토리지 엔진으로 InnoDB를 권장합니다. MySQL 스토리지 엔진에 대한 자세한 내용은 RDS for MySQL에 대해 지원되는 스토리지 엔진 섹션을 참조하세요.
알림	RDS-EVENT-0055	*message*	DB 인스턴스의 테이블 수가 Amazon RDS의 권장 모범 사례를 초과하였습니다. DB 인스턴스의 테이블 수를 줄이세요. 권장 모범 사례에 대한 자세한 내용은 Amazon RDS 기본 운영 지침 단원을 참조하십시오.
알림	RDS-EVENT-0056	*message*	DB 인스턴스의 데이터베이스 수가 Amazon RDS의 권장 모범 사례를 초과하였습니다. DB 인스턴스의 데이터베이스 수를 줄이세요. 권장 모범 사례에 대한 자세한 내용은 Amazon RDS 기본 운영 지침 단원을 참조하십시오.

범주	RDS 이벤트 ID	메시지	참고
알림	RDS-EVENT-0064	TDE 암호화 키가 성공적으로 교체되었습니다.	권장 모범 사례에 대한 자세한 내용은 Amazon RDS 기본 운영 지침 단원을 참조하십시오.
알림	RDS-EVENT-0084	DB 인스턴스를 다중 AZ로 변환할 수 없습니다. ###.	DB 인스턴스를 다중 AZ로 전환하려고 시도하였으나 다중 AZ를 지원하지 않는 인 메모리 파일 그룹이 포함되어 있습니다. 자세한 내용은 Amazon RDS for Microsoft SQL Server의 다중 AZ 배포 섹션을 참조하세요.
알림	RDS-EVENT-0087	DB 인스턴스가 중지되었습니다.	
알림	RDS-EVENT-0088	DB 인스턴스가 시작되었습니다.	
알림	RDS-EVENT-0154	DB 인스턴스가 중지 최대 허용 시간 초과로 인해 시작 중입니다.	

범주	RDS 이벤트 ID	메시지	참고
알림	RDS-EVENT-0157	DB 인스턴스 클래스를 수정할 수 없습니다. *###*.	대상 인스턴스 클래스는 원본 DB 인스턴스에 있는 데이터베이스의 수를 지원할 수 없기 때문에 RDS는 DB 인스턴스 클래스를 수정할 수 없습니다. “인스턴스에 N 데이터베이스가 있지만, 변환 후에는 N만 지원할 것입니다.”라는 오류 메시지가 표시됩니다. 자세한 내용은 Microsoft SQL Server DB 인스턴스의 한도 섹션을 참조하세요.
알림	RDS-EVENT-0158	데이터베이스 인스턴스가 업그레이드할 수 없는 상태입니다. *###*.	
알림	RDS-EVENT-0167	*message*	RDS 고객 지원 경계 구성이 변경되었습니다.
알림	RDS-EVENT-0189	RDS DB 인스턴스에 대한 gp2 버스트 밸런스 크레딧이 부족합니다. 이 문제를 해결하려면 IOPS 사용량을 줄이거나 성능 향상을 위해 스토리지 설정을 수정하세요.	RDS DB 인스턴스에 대한 gp2 버스트 밸런스 크레딧이 부족합니다. 이 문제를 해결하려면 IOPS 사용량을 줄이거나 성능 향상을 위해 스토리지 설정을 수정하세요. 자세한 내용은 Amazon Elastic Compute Cloud 사용 설명서의 I/O 크레딧 및 버스트 가능 성능을 참조하세요.

범주	RDS 이벤트 ID	메시지	참고
알림	RDS-EVENT-0225	할당된 스토리지 크기 ## #GB가 최대 스토리지 임계값 ##GB에 도달하고 있습니다. 최대 스토리지 임계값을 늘립니다.	이 이벤트는 할당된 스토리지가 최대 스토리지 임계값의 80%에 도달하면 호출됩니다. 이 이벤트를 방지하려면 최대 스토리지 임계값을 늘립니다.
알림	RDS-EVENT-0231	DB 인스턴스의 스토리지 수정에서 내부 오류가 발생했습니다. 수정 요청이 보류 중이며 나중에 다시 시도합니다.	읽기 전용 복제 프로세스에서 오류가 발생하였습니다. 자세한 내용은 이벤트 메시지 단원을 참조하십시오. 또한 DB 엔진의 읽기 전용 복제본에 대한 문제 해결 단원을 참조하십시오. • MariaDB 읽기 전용 복제본의 문제 해결 • SQL Server 읽기 전용 복제본 문제 해결 • MySQL 읽기 전용 복제본의 문제 해결 • RDS for Oracle 복제본 문제 해결

범주	RDS 이벤트 ID	메시지	참고
알림	RDS-EVENT-0253	데이터베이스가 이중 쓰기 버퍼를 사용하고 있습니다. ###. 자세한 내용은 ##용 RDS Optimized Writes 설명서를 참조하세요.	RDS Optimized Writes가 인스턴스 스토리지 구성과 호환되지 않습니다. 자세한 내용은 RDS Optimized Writes for MySQL을 통한 쓰기 성능 개선 및 Amazon RDS Optimized Writes for MariaDB를 통한 쓰기 성능 개선 단원을 참조하세요. 블루/그린 배포를 생성하여 스토리지 구성 업그레이드를 수행하여 최적화된 쓰기를 활성화할 수 있습니다.
알림	RDS-EVENT-0297	DB 인스턴스 ##의 스토리지 구성은 최대 16,384GiB를 지원합니다. 스토리지 구성 업그레이드를 수행하여 16,384GiB보다 큰 스토리지 크기를 지원합니다.	DB 인스턴스의 할당된 스토리지 크기를 16,384GiB를 초과하여 늘릴 수 없습니다. 이 제한을 극복하려면 스토리지 구성 업그레이드를 수행하세요. 자세한 내용은 DB 인스턴스의 스토리지 파일 시스템 업그레이드를 참조하세요.
알림	RDS-EVENT-0298	DB 인스턴스 ##의 스토리지 구성은 테이블 크기로 최대 2,048GiB를 지원합니다. 스토리지 구성 업그레이드를 수행하여 2,048GiB보다 큰 테이블 크기를 지원합니다.	이 제한이 적용되는 RDS MySQL 및 MariaDB 인스턴스는 테이블 크기가 2,048GiB를 초과할 수 없습니다. 이 제한을 극복하려면 스토리지 구성 업그레이드를 수행하세요. 자세한 내용은 DB 인스턴스의 스토리지 파일 시스템 업그레이드를 참조하세요.

범주	RDS 이벤트 ID	메시지	참고
알림	RDS-EVENT-0327	Amazon RDS가 암호 *SECRET ARN*을 찾지 못했습니다. *###.*	
읽기 전용 복제본	RDS-EVENT-0045	복제가 중지되었습니다.	스토리지 부족으로 DB 인스턴스 복제가 중지되었습니다. 복제를 계속하려면 스토리지의 규모를 조정하거나 다시 실행 로그의 최대 크기를 줄이세요. 크기가 *amount*MiB인 다시 실행 로그를 수용하려면 최소한 *amount*MiB의 사용 가능한 스토리지가 필요합니다.
읽기 전용 복제본	RDS-EVENT-0046	읽기 전용 복제본에 대한 복제가 재개되었습니다.	이 메시지는 읽기 전용 복제본을 처음 생성할 때 나타나거나 복제 기능의 정상 여부를 확인하는 모니터링 메시지로 나타납니다. RDS-EVENT-0045 알림 후에 이 메시지가 표시되면 오류가 표시된 후 또는 복제가 중단된 후 복제가 재개된 것입니다.
읽기 전용 복제본	RDS-EVENT-0057	복제 스트리밍이 종료되었습니다.	
읽기 전용 복제본	RDS-EVENT-0062	읽기 전용 복제본의 복제가 수동으로 중단되었습니다.	
읽기 전용 복제본	RDS-EVENT-0063	RDS가 아닌 인스턴스에서의 복제가 재설정되었습니다.	

범주	RDS 이벤트 ID	메시지	참고
읽기 전용 복제본	RDS-EVENT-0202	읽기 전용 복제본 생성이 실패했습니다.	
읽기 전용 복제본	RDS-EVENT-0357	복제 채널 ##이 시작되었습니다.	복제 채널에 대한 자세한 내용은 the section called “다중 소스 복제 구성” 섹션을 참조하세요.
읽기 전용 복제본	RDS-EVENT-0358	복제 채널 ##이 중지되었습니다.	복제 채널에 대한 자세한 내용은 the section called “다중 소스 복제 구성” 섹션을 참조하세요.
읽기 전용 복제본	RDS-EVENT-0359	복제 채널 ##이 수동으로 중지되었습니다.	복제 채널에 대한 자세한 내용은 the section called “다중 소스 복제 구성” 섹션을 참조하세요.
읽기 전용 복제본	RDS-EVENT-0360	복제 채널 ##이 재설정되었습니다.	복제 채널에 대한 자세한 내용은 the section called “다중 소스 복제 구성” 섹션을 참조하세요.
복구	RDS-EVENT-0020	DB 인스턴스 복구가 시작되었습니다. 복구 시간은 복구할 데이터 용량에 따라 달라집니다.	
복구	RDS-EVENT-0021	DB 인스턴스 복구가 완료되었습니다.	

범주	RDS 이벤트 ID	메시지	참고
복구	RDS-EVENT-0023	긴급 스냅샷 요청이 있습니다. *###*.	수동 백업을 요청하였지만 Amazon RDS가 현재 DB 스냅샷을 생성 중입니다. 따라서 Amazon RDS가 DB 스냅샷 생성을 완료한 후에 다시 요청하십시오.
복구	RDS-EVENT-0052	다중 AZ 인스턴스 복구가 시작되었습니다.	복구 시간은 복구할 데이터 용량에 따라 달라집니다.
복구	RDS-EVENT-0053	다중 AZ 인스턴스 복구가 완료되었습니다. 장애 조치 또는 활성화가 보류 중입니다.	
복구	RDS-EVENT-0066	미러링을 다시 설정하는 동안 인스턴스의 성능이 저하됩니다. *###*.	SQL Server DB 인스턴스가 미러를 재구성 중입니다. 이때 미러가 재구성될 때까지 성능이 저하됩니다. 복구 모델이 FULL이 아닌 데이터베이스가 발견되었습니다. 복구 모델이 FULL로 다시 변경된 후 미러링 복구가 시작되었습니다(<dbname>: <recovery model found>[,...]).
복구	RDS-EVENT-0166	*message*	RDS 커스텀 DB 인스턴스는 지원 경계 내에 있습니다.
복구	RDS-EVENT-0361	대기 DB 인스턴스 복구가 시작되었습니다.	복구 프로세스 중에 대기 DB 인스턴스가 다시 구축됩니다. 복구 프로세스 중에는 데이터베이스 성능이 영향을 받습니다.

범주	RDS 이벤트 ID	메시지	참고
복구	RDS-EVENT-0362	대기 DB 인스턴스 복구가 완료되었습니다.	복구 프로세스 중에 대기 DB 인스턴스가 다시 구축됩니다. 복구 프로세스 중에는 데이터베이스 성능이 영향을 받습니다.
복원	RDS-EVENT-0019	DB 인스턴스 ##에서 ##으로 복원되었습니다.	DB 인스턴스가 특정 시점으로 백업에서 복원되었습니다.
보안	RDS-EVENT-0068	인스턴스를 업데이트하기 위해 HSM 파티션 암호의 암호화를 해제하는 중입니다.	DB 인스턴스를 업데이트하기 위해. RDS가 AWS CloudHSM 파티션 암호의 암호화를 해제하는 중입니다. 자세한 내용은 AWS CloudHSM 사용 설명서의 AWS CloudHSM을 사용한 Oracle 데이터베이스 투명한 데이터 암호화(TDE)를 참조하세요.
보안 패치	RDS-EVENT-0230	DB 인스턴스의 시스템 업데이트를 사용할 수 있습니다. 업데이트 적용에 대한 자세한 내용은 RDS 사용 설명서의 'DB 인스턴스 관리'를 참조하세요.	새 운영 체제 업데이트를 사용할 수 있습니다. DB 인스턴스의 운영 체제의 새로운 버전, 마이너 버전, 운영 체제 업데이트를 사용할 수 있습니다. 업데이트에 적용에 대한 자세한 내용은 RDS DB 인스턴스의 운영 체제 업데이트 단원을 참조하세요.

DB 파라미터 그룹 이벤트

다음 표는 DB 파라미터 그룹이 소스 유형일 때 이벤트 카테고리와 이벤트 목록을 나타냅니다.

Category	RDS 이벤트 ID	메시지	참고
구성 변경	RDS-EVENT-0037	메서드 *method* 적용을 사용하여 파라미터 *name*을 *value*로 업데이트했습니다.	

DB 보안 그룹 이벤트

다음 표는 DB 보안 그룹이 소스 유형일 때 이벤트 카테고리와 이벤트 목록을 나타냅니다.

> **Note**
>
> DB 보안 그룹 EC2-Classic의 리소스입니다. EC2-Classic은 2022년 8월 15일에 사용 중지되었습니다. EC2-Classic에서 VPC로 마이그레이션하지 않은 경우 가능한 한 빨리 마이그레이션하는 것이 좋습니다. 자세한 내용은 Amazon EC2 사용 설명서의 Migrate from EC2-Classic to a VPC(EC2-Classic에서 VPC로 마이그레이션) 및 EC2-Classic Networking is Retiring – Here's How to Prepare(EC2-Classic 네트워킹 지원 중단에 대비하는 방법) 블로그 게시물을 참조하세요.

범주	RDS 이벤트 ID	메시지	참고
구성 변경	RDS-EVENT-0038	보안 그룹에 변경 사항을 적용했습니다.	
실패	RDS-EVENT-0039	###로서의 권한 부여를 취소하는 중입니다.	###가 소유한 보안 그룹이 없습니다. 보안 그룹에 대한 권한 부여가 잘못되었기 때문에 취소되었습니다.

DB 스냅샷 이벤트

다음 표는 DB 스냅샷이 소스 유형일 때 이벤트 카테고리와 이벤트 목록을 나타냅니다.

Category	RDS 이벤트 ID	메시지	참고
생성	RDS-EVENT-0040	수동 스냅샷을 생성하는 중입니다.	
생성	RDS-EVENT-0042	수동 스냅샷이 생성되었습니다.	
생성	RDS-EVENT-0090	자동 스냅샷을 생성하는 중입니다.	
생성	RDS-EVENT-0091	자동 스냅샷이 생성되었습니다.	
삭제	RDS-EVENT-0041	사용자 스냅샷이 삭제되었습니다.	
알림	RDS-EVENT-0059	리전(##)으로부터 스냅샷(##) 복사를 시작했습니다.	교차 리전 스냅샷 복사본입니다.
알림	RDS-EVENT-0060	##분 안에 리전(##)으로부터 스냅샷(##) 복사를 완료했습니다.	교차 리전 스냅샷 복사본입니다.
알림	RDS-EVENT-0061	리전(##)으로부터 ##의 스냅샷 복사 요청을 취소했습니다.	교차 리전 스냅샷 복사본입니다.
알림	RDS-EVENT-0159	스냅샷 내보내기 작업이 실패했습니다.	
알림	RDS-EVENT-0160	스냅샷 내보내기 작업이 취소되었습니다.	
알림	RDS-EVENT-0161	스냅샷 내보내기 작업이 완료되었습니다.	

Category	RDS 이벤트 ID	메시지	참고
알림	RDS-EVENT-0196	리전(##)에서 스냅샷(##) 복사를 시작했습니다.	로컬 스냅샷 복사본입니다.
알림	RDS-EVENT-0197	리전(##)에서 스냅샷(##) 복사를 완료했습니다.	로컬 스냅샷 복사본입니다.
알림	RDS-EVENT-0190	리전(##)에서 ##의 스냅샷 복사 요청을 취소했습니다.	로컬 스냅샷 복사본입니다.
복원	RDS-EVENT-0043	스냅샷(##)에서 복원되었습니다.	DB 인스턴스가 DB 스냅샷에서 복원 중입니다.

DB 클러스터 스냅샷 이벤트

다음 표에는 DB 클러스터 스냅샷이 원본 유형일 때 이벤트 카테고리와 이벤트 목록이 나와 있습니다.

범주	RDS 이벤트 ID	메시지	참고
백업	RDS-EVENT-0074	수동 클러스터 스냅샷을 생성하는 중입니다.	
백업	RDS-EVENT-0075	수동 클러스터 스냅샷이 생성되었습니다.	
백업	RDS-EVENT-0168	자동화된 클러스터 스냅샷을 생성하는 중입니다.	
백업	RDS-EVENT-0169	자동화된 클러스터 스냅샷이 생성되었습니다.	

RDS 프록시 이벤트

다음 표에서는 RDS 프록시가 소스 유형일 때 이벤트 범주와 이벤트 목록을 보여줍니다.

범주	RDS 이벤트 ID	메시지	참고
구성 변경	RDS-EVENT-0204	RDS가 DB 프록시(##)를 수정했습니다.	
구성 변경	RDS-EVENT-0207	RDS가 DB 프록시(##)의 엔드포인트를 수정했습니다.	
구성 변경	RDS-EVENT-0213	RDS가 DB 인스턴스 추가를 감지하여 DB 프록시(##)의 대상 그룹에 자동으로 추가했습니다.	
구성 변경	RDS-EVENT-0213	RDS가 DB 인스턴스(##)의 생성을 감지하여 DB 프록시(##)의 대상 그룹(##)에 자동으로 추가했습니다.	
구성 변경	RDS-EVENT-0214	RDS가 DB 인스턴스(##)의 삭제를 감지하여 DB 프록시(##)의 대상 그룹(##)에서 자동으로 제거했습니다.	
구성 변경	RDS-EVENT-0215	RDS가 DB 클러스터(##)의 삭제를 감지하여 DB 프록시(##)의 대상 그룹(##)에서 자동으로 제거했습니다.	
생성	RDS-EVENT-0203	RDS가 DB 프록시(##)를 생성했습니다.	
생성	RDS-EVENT-0206	RDS가 DB 프록시(##)에 대한 엔드포인트(##)을 생성했습니다.	
삭제	RDS-EVENT-0205	RDS가 DB 프록시(##)를 삭제했습니다.	

범주	RDS 이벤트 ID	메시지	참고
삭제	RDS-EVENT-0208	RDS가 DB 프록시(##)에 대한 엔드포인트(##)을 삭제했습니다.	
실패	RDS-EVENT-0243	서브넷(##)에서 사용할 수 있는 IP 주소가 충분하지 않기 때문에 RDS가 프록시(##)의 용량을 프로비저닝하지 못했습니다. 이러한 문제를 해결하려면 RDS 프록시 설명서의 권장 사항에 따라 서브넷에 최소한의 미사용 IP 주소 개수가 있는지 확인하세요.	권장 인스턴스 클래스 수를 확인하려면 IP 주소 용량 계획 섹션을 참조하세요.
실패	RDS-EVENT-0275	RDS가 DB 프록시 ##에 대한 일부 연결을 제한했습니다. 클라이언트에서 프록시로의 동시 연결 요청 수가 제한을 초과했습니다.	

블루/그린 배포 이벤트

다음 표는 블루/그린 배포가 소스 유형일 때 이벤트 카테고리와 이벤트 목록을 나타냅니다.

블루/그린 배포에 대한 자세한 내용은 데이터베이스 업데이트에 Amazon RDS 블루/그린 배포 사용 섹션을 참조하세요.

범주	Amazon RDS 이벤트 ID	메시지	참고
생성	RDS-EVENT-0244	블루/그린 배포 작업이 완료되었습니다. 그린 환경 데이터베이스를 추가로 수정하거	

범주	Amazon RDS 이벤트 ID	메시지	참고
		나 배포로 전환할 수 있습니다.	
실패	RDS-EVENT-0245	(소스/타겟) DB (인스턴스/클러스터)를 찾을 수 없어서 블루/그린 배포를 생성하지 못했습니다.	
삭제	RDS-EVENT-0246	블루/그린 배포가 삭제되었습니다.	
알림	RDS-EVENT-0247	##에서 ##으로 전환이 시작되었습니다.	
알림	RDS-EVENT-0248	블루/그린 배포에서 전환이 완료되었습니다.	
실패	RDS-EVENT-0249	블루/그린 배포에서 전환이 취소되었습니다.	
알림	RDS-EVENT-0250	기본/읽기 전용 복제본 ##에서 ##으로 전환이 시작되었습니다.	
알림	RDS-EVENT-0251	기본/읽기 전용 복제본 ##에서 ##으로 전환이 완료되었습니다. ##는 ##-##으로, ##은 ##로 이름을 변경했습니다.	
실패	RDS-EVENT-0252	기본/읽기 전용 복제본 ##에서 ##으로의 전환이 ##(으)로 인해 취소되었습니다.	

범주	Amazon RDS 이벤트 ID	메시지	참고
알림	RDS-EVENT-0307	##에서 ##으로의 전환을 위한 시퀀스 동기화가 시작되었습니다. 시퀀스를 사용할 때 전환하면 가동 중지가 길어질 수 있습니다.	
알림	RDS-EVENT-0308	##에서 ##으로의 전환을 위한 시퀀스 동기화가 완료되었습니다.	
실패	RDS-EVENT-0310	시퀀스 동기화에 실패하여 ##에서 ##으로의 전환을 위한 시퀀스 동기화가 취소되었습니다.	

사용자 지정 엔진 버전 이벤트

다음 표에는 사용자 지정 엔진 버전이 소스 유형일 때 이벤트 카테고리와 이벤트 목록이 나와 있습니다.

범주	Amazon RDS 이벤트 ID	메시지	참고
생성	RDS-EVENT-0316	사용자 지정 엔진 버전 ## 생성을 준비 중입니다. 전체 생성 프로세스를 완료하는 데 최대 4시간이 걸릴 수 있습니다.	
생성	RDS-EVENT-0317	사용자 지정 엔진 버전 ##을 생성하고 있습니다.	
생성	RDS-EVENT-0318	사용자 지정 엔진 버전 ##을 검증하고 있습니다.	

범주	Amazon RDS 이벤트 ID	메시지	참고
생성	RDS-EVENT-0319	사용자 지정 엔진 버전 ##이 성공적으로 생성되었습니다.	
생성	RDS-EVENT-0320	내부 문제로 인해 RDS에서 사용자 지정 엔진 버전 ##을 생성할 수 없습니다. 문제를 해결 중이며 필요한 경우 연락드리겠습니다. 추가 지원이 필요한 경우 AWS 프리미엄 지원에 문의하세요.	
실패	RDS-EVENT-0198	사용자 지정 엔진 버전(##)을 생성하지 못했습니다. ###	###에는 누락된 파일 등 실패에 대한 세부 정보가 포함되어 있습니다.
실패	RDS-EVENT-0277	사용자 지정 엔진 버전(##)을 삭제하는 중 오류가 발생했습니다. ###	###에는 실패에 대한 세부 정보가 포함되어 있습니다.
복원	RDS-EVENT-0352	특정 시점으로 복원에 지원되는 최대 데이터베이스 수가 변경되었습니다.	###에는 이벤트에 대한 세부 정보가 포함되어 있습니다.

Amazon RDS 로그 파일 모니터링

모든 RDS 데이터베이스 엔진은 감사 및 문제 해결을 위해 액세스할 수 있는 로그를 생성합니다. 로그 유형은 데이터베이스 엔진에 따라 다릅니다.

AWS Management Console, AWS Command Line Interface(AWS CLI) 또는 Amazon RDS API를 사용하여 DB 인스턴스 로그에 액세스할 수 있습니다. 트랜잭션 로그를 보거나 다운로드할 수 없습니다.

주제

- 데이터베이스 로그 파일 보기 및 나열
- 데이터베이스 로그 파일 다운로드
- 데이터베이스 로그 파일 조사
- Amazon CloudWatch Logs에 데이터베이스 로그 게시
- REST를 사용하여 로그 파일 내용 읽기
- Amazon RDS for Db2 데이터베이스 로그 파일
- MariaDB 데이터베이스 로그 파일
- Amazon RDS for Microsoft SQL Server 데이터베이스 로그 파일
- MySQL 데이터베이스 로그 파일
- Amazon RDS for Oracle 데이터베이스 로그 파일
- RDS for PostgreSQL 데이터베이스 로그 파일

데이터베이스 로그 파일 보기 및 나열

AWS Management Console을 사용하여 Amazon RDS DB 엔진에 대한 데이터베이스 로그 파일을 볼 수 있습니다. AWS CLI 또는 Amazon RDS API를 사용하여 다운로드하거나 모니터링할 수 있는 로그 파일을 나열할 수 있습니다.

> **Note**
>
> 기존 RDS for Oracle DB 인스턴스에 대한 로그 파일 목록을 볼 수 없는 경우 목록을 보려면 인스턴스를 재부팅합니다.

콘솔

데이터베이스 로그 파일을 보려면

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 보고자 하는 로그 파일을 보유한 DB 인스턴스의 이름을 선택합니다.
4. 로그 및 이벤트 탭을 선택합니다.
5. 아래로 스크롤하여 로그 섹션을 찾습니다.
6. (선택 사항) 검색어를 입력하여 결과를 필터링합니다.
7. 표시할 로그를 선택한 다음 보기(View)를 선택합니다.

AWS CLI

DB 인스턴스에 사용 가능한 데이터베이스 로그 파일을 나열하려면 AWS CLI `describe-db-log-files` 명령을 사용합니다.

다음 예에서는 `my-db-instance`라는 DB 인스턴스에 대한 로그 파일 목록을 반환합니다.

Example

```
aws rds describe-db-log-files --db-instance-identifier my-db-instance
```

RDS API

DB 인스턴스에 사용 가능한 데이터베이스 로그 파일을 나열하려면 Amazon RDS API `DescribeDBLogFiles` 작업을 사용합니다.

데이터베이스 로그 파일 다운로드

AWS Management Console, AWS CLI 또는 API를 사용하여 데이터베이스 로그 파일을 다운로드할 수 있습니다.

콘솔

데이터베이스 로그 파일을 다운로드하려면

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.

3. 보고자 하는 로그 파일을 보유한 DB 인스턴스의 이름을 선택합니다.
4. 로그 및 이벤트 탭을 선택합니다.
5. 아래로 스크롤하여 [Logs] 섹션을 찾습니다.
6. 로그 섹션에서 다운로드할 로그 옆에 있는 버튼을 선택한 다음 다운로드를 선택합니다.
7. 제공된 링크에 대한 컨텍스트(마우스 오른쪽 클릭) 메뉴를 열고 나서 [Save Link As]를 선택합니다. 로그 파일을 저장할 위치를 입력한 다음 저장을 선택합니다.



AWS CLI

데이터베이스 로그 파일을 다운로드하려면 AWS CLI 명령 download-db-log-file-portion을 사용합니다. 기본적으로 이 명령은 로그 파일의 최신 부분만을 다운로드합니다. 하지만 `--starting-token 0` 파라미터를 지정하여 전체 파일을 다운로드할 수 있습니다.

다음 예제에서는 log/ERROR.4라는 로그 파일의 내용을 다운로드하여 errorlog.txt라는 로컬 파일에 저장하는 방법을 보여줍니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds download-db-log-file-portion \
    --db-instance-identifier myexampledb \
    --starting-token 0 --output text \
    --log-file-name log/ERROR.4 > errorlog.txt
```

Windows의 경우:

```
aws rds download-db-log-file-portion ^
    --db-instance-identifier myexampledb ^
    --starting-token 0 --output text ^
    --log-file-name log/ERROR.4 > errorlog.txt
```

RDS API

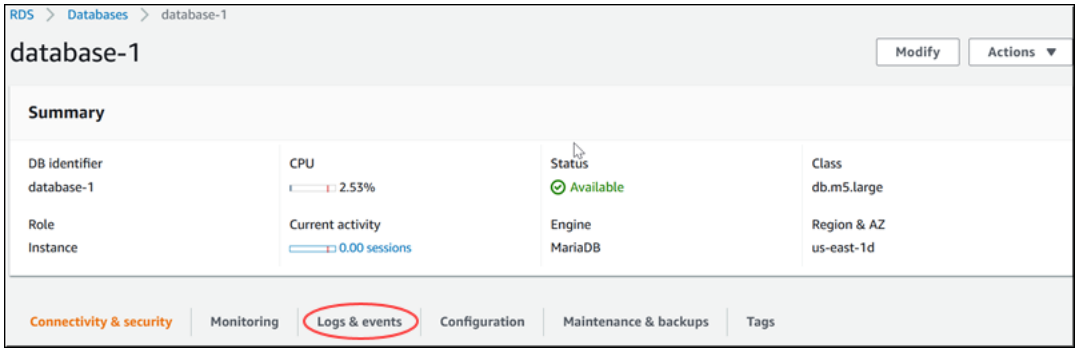
데이터베이스 로그 파일을 다운로드하려면 Amazon RDS API DownloadDBLogFilePortion 작업을 사용합니다.

데이터베이스 로그 파일 조사

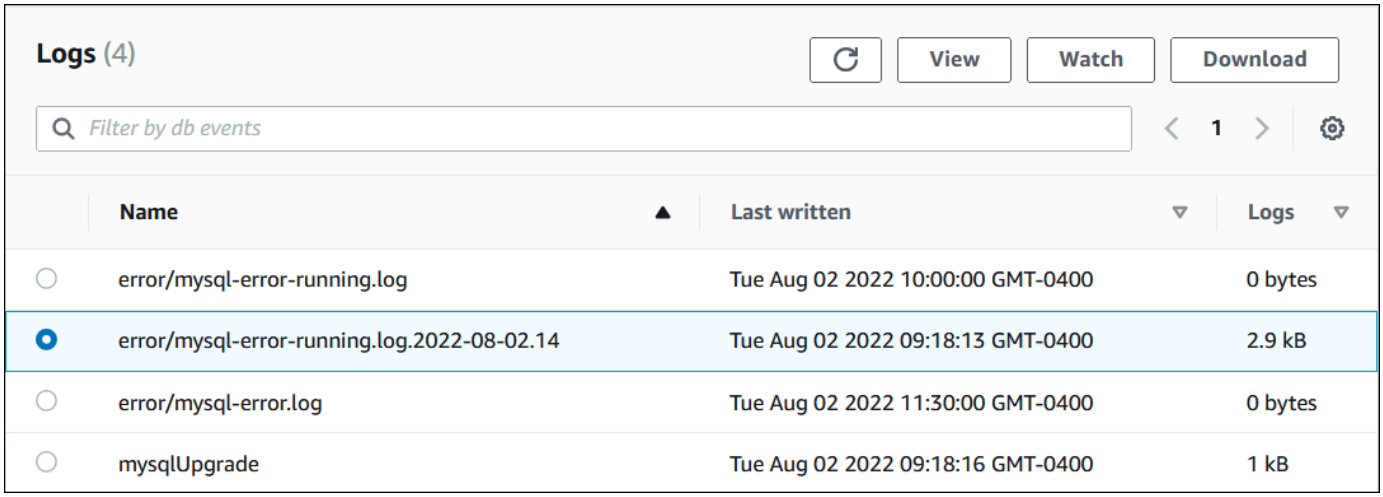
데이터베이스 로그 파일을 관찰하는 것은 UNIX 또는 Linux 시스템에서 파일을 추적하는 것과 같습니다. AWS Management Console을 사용하여 로그 파일을 볼 수 있습니다. RDS는 5초마다 로그 테일을 새로 고칩니다.

데이터베이스 로그 파일을 조사하려면

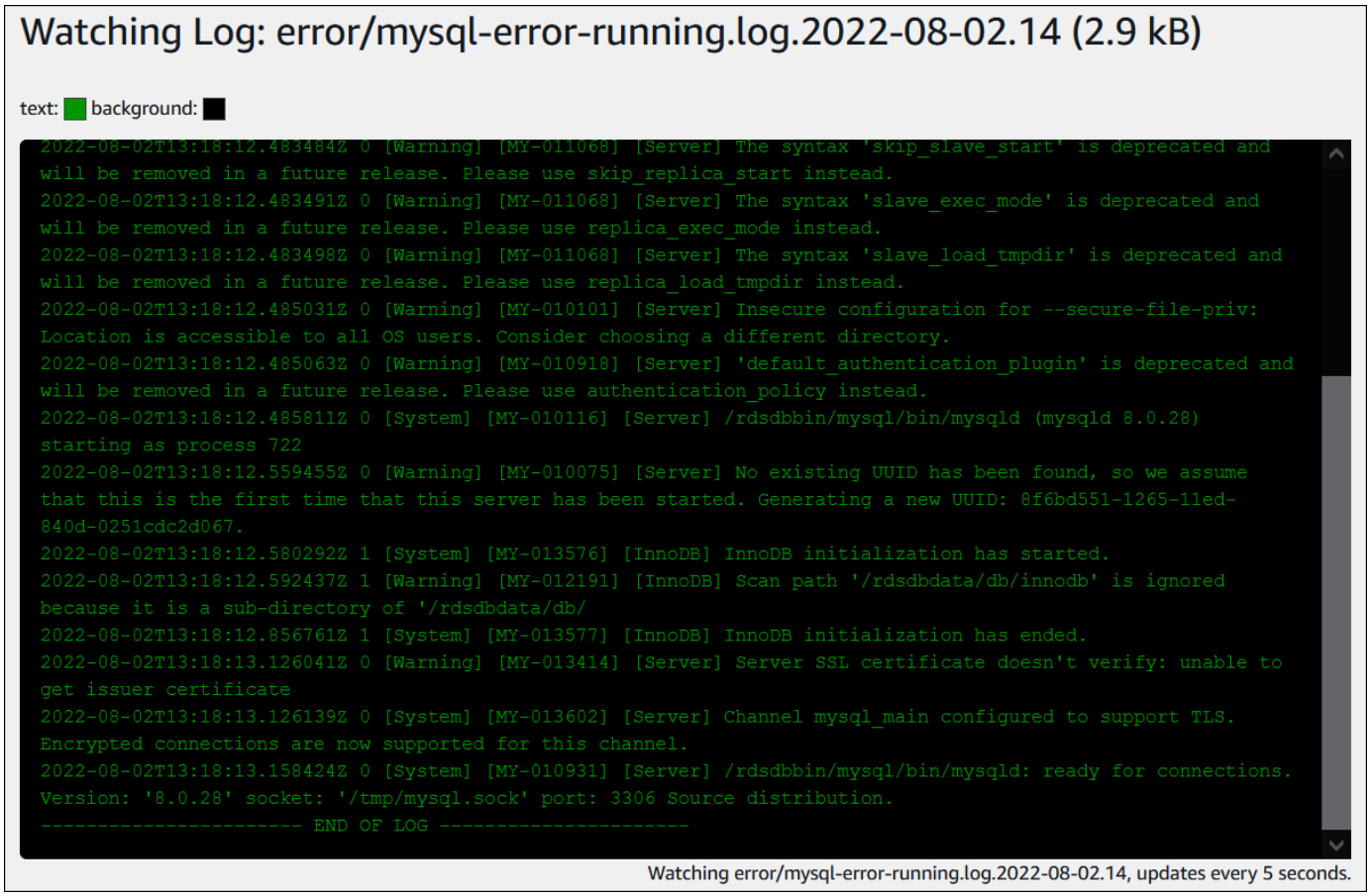
1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 보고자 하는 로그 파일을 보유한 DB 인스턴스의 이름을 선택합니다.
4. 로그 및 이벤트 탭을 선택합니다.



5. 로그 섹션에서 로그 파일을 선택한 다음 보기를 선택합니다.



RDS는 다음 MySQL 예제와 같이 로그의 끝부분을 보여줍니다.



Amazon CloudWatch Logs에 데이터베이스 로그 게시

온프레미스 데이터베이스에서 데이터베이스 로그는 파일 시스템에 있습니다. Amazon RDS는 DB 인스턴스의 파일 시스템에 있는 데이터베이스 로그에 대한 호스트 액세스를 제공하지 않습니다. 이러한

이유로 Amazon RDS를 사용하면 데이터베이스 로그를 Amazon CloudWatch Logs로 내보낼 수 있습니다. CloudWatch Logs를 통해 로그 데이터에 대한 실시간 분석을 수행할 수 있습니다. 또한 내구성이 뛰어난 스토리지에 데이터를 저장하고, CloudWatch Logs Agent로 데이터를 관리할 수 있습니다.

주제

- RDS와 CloudWatch Logs에 대한 통합 개요
- CloudWatch Logs에 게시할 로그 결정
- CloudWatch Logs에 게시할 로그 지정
- CloudWatch Logs에서 로그 검색 및 필터링

RDS와 CloudWatch Logs에 대한 통합 개요

CloudWatch Logs에서 로그 스트림은 동일한 소스를 공유하는 일련의 로그 이벤트입니다. CloudWatch Logs에서 각 별도의 로그 소스가 별도의 로그 스트림을 구성합니다. 로그 그룹은 동일한 보존 기간, 모니터링 및 액세스 제어 설정을 공유하는 로그 스트림 그룹입니다.

Amazon RDS는 DB 인스턴스 로그 레코드를 로그 그룹으로 지속적으로 스트리밍합니다. 예를 들어 게시하는 각 유형의 로그에 대한 로그 그룹 `/aws/rds/instance/`*`instance_name`*`/`*`log_type`*이 있습니다. 이 로그 그룹은 로그를 생성하는 데이터베이스 인스턴스와 동일한 AWS 리전에 있습니다.

보존 기간을 지정하지 않는 한 AWS는 CloudWatch Logs에 게시된 로그 데이터를 무기한 보존합니다. 자세한 내용은 CloudWatch Logs에서 로그 데이터 보존 기간 변경을 참조하세요.

CloudWatch Logs에 게시할 로그 결정

각 RDS 데이터베이스 엔진은 자체 로그 세트를 지원합니다. 데이터베이스 엔진 옵션에 대해 알아보려면 다음 주제를 검토하십시오.

- the section called “Amazon CloudWatch Logs에 Db2 로그 게시”
- the section called “Amazon CloudWatch Logs에 MariaDB 로그 게시”
- the section called “Amazon CloudWatch Logs에 MySQL 로그 게시”
- the section called “Amazon CloudWatch Logs에 Oracle 로그 게시”
- the section called “Amazon CloudWatch Logs에 PostgreSQL 로그 게시”
- the section called “Amazon CloudWatch Logs에 SQL Server 로그 게시”

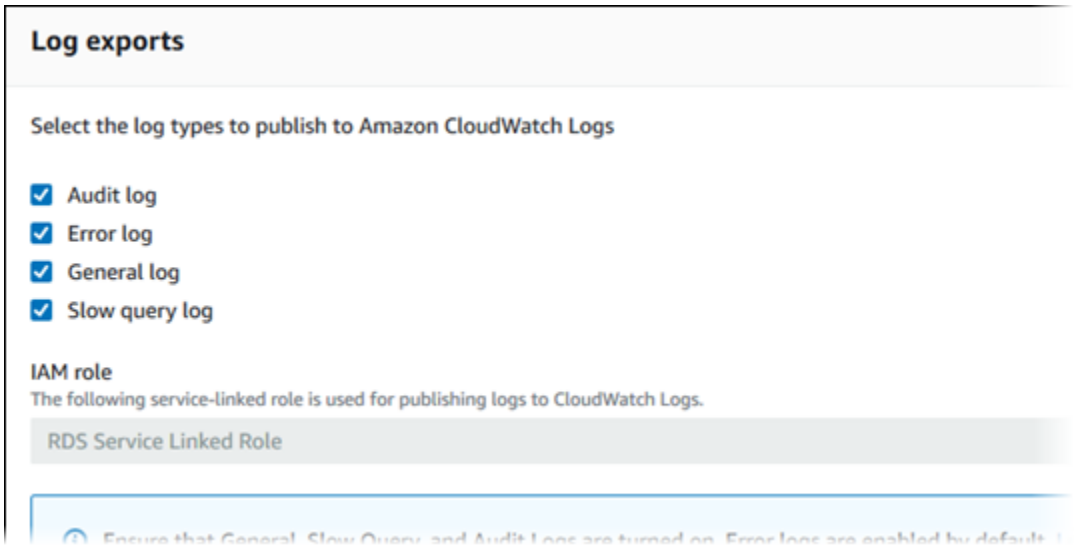
CloudWatch Logs에 게시할 로그 지정

콘솔에 게시할 로그를 지정합니다. AWS Identity and Access Management(IAM)에 서비스 연결 역할이 있는지 확인합니다. 서비스 연결 역할에 대한 자세한 내용은 Amazon RDS에 서비스 연결 역할 사용를 참조하세요.

게시할 로그 지정

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 다음 중 하나를 수행하세요.
   - 데이터베이스 생성을 선택합니다.
   - 목록에서 데이터베이스를 선택한 다음 수정을 선택합니다.
4. 로그 내보내기에서 게시할 로그를 선택합니다.

   다음 예시에서는 RDS for MySQL DB 인스턴스에 대한 감사 로그, 오류 로그, 일반 로그, 및 느린 쿼리 로그를 지정합니다.



CloudWatch Logs에서 로그 검색 및 필터링

CloudWatch Logs 콘솔을 이용하여 지정된 기준을 충족하는 로그 항목을 검색할 수 있습니다. CloudWatch Logs 콘솔로 연결되는 RDS 콘솔이나 CloudWatch Logs 콘솔에서 직접 로그에 액세스할 수 있습니다.

콘솔을 이용하여 로그 항목 검색

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. DB 인스턴스를 고르십시오.
4. 구성을 선택합니다.
5. 게시된 로그에서 보려는 데이터베이스 로그를 선택합니다.

CloudWatch Logs 콘솔을 사용하여 플로우 로그 레코드 검색

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그 그룹을 선택합니다.
3. 필터 상자에 **/aws/rds**를 입력합니다.
4. 로그 그룹에서 검색할 로그 스트림이 포함된 로그 그룹의 이름을 선택합니다.
5. 로그 스트림에서 검색할 로그 스트림의 이름을 선택합니다.
6. 로그 이벤트에서 사용할 필터 구문을 입력합니다.

자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 로그 데이터 검색 및 필터링을 참조하십시오. RDS 로그를 모니터링하는 방법을 설명하는 블로그의 자습서는 Amazon CloudWatch Logs, AWS Lambda 및 Amazon SNS를 사용하여 Amazon RDS에 대한 사전 예방적 데이터베이스 모니터링 구축을 참조하십시오.

REST를 사용하여 로그 파일 내용 읽기

Amazon RDS는 DB 인스턴스 로그 파일 액세스를 허용하는 REST 엔드포인트를 제공합니다. Amazon RDS 로그 파일 내용을 스트리밍하는 애플리케이션을 작성해야 하는 경우 유용합니다.

구문은 다음과 같습니다.

```
GET /v13/downloadCompleteLogFile/DBInstanceIdentifier/LogFileName HTTP/1.1
Content-type: application/json
host: rds.region.amazonaws.com
```

다음 파라미터는 필수 파라미터입니다.

- *DBInstanceIdentifier* — 다운로드하려는 로그 파일이 있는 DB 인스턴스에 고객이 할당하는 이름입니다.

- *LogFileName* — 다운로드할 로그 파일의 이름입니다.

응답에는 스트림으로 요청된 로그 파일의 내용이 포함됩니다.

다음 예제에서는 us-west-2 리전에 sample-sql로 명명된 DB 인스턴스에 대해 log/ERROR.6으로 명명된 로그 파일을 다운로드합니다.

```
GET /v13/downloadCompleteLogFile/sample-sql/log/ERROR.6 HTTP/1.1
host: rds.us-west-2.amazonaws.com
X-Amz-Security-Token: AQoDYXdzEIH//////////
wEa0AIXLhngC5zp9CyB1R6abwKrXHVR5efnAVN3XvR7IwqKYalFSn6UyJuEFTft9nObglx4QJ+GXV9cpACkETq=
X-Amz-Date: 20140903T233749Z
X-Amz-Algorithm: AWS4-HMAC-SHA256
X-Amz-Credential: AKIADQKE4SARGYLE/20140903/us-west-2/rds/aws4_request
X-Amz-SignedHeaders: host
X-Amz-Content-SHA256: e3b0c44298fc1c229afbf4c8996fb92427ae41e4649b934de495991b7852b855
X-Amz-Expires: 86400
X-Amz-Signature: 353a4f14b3f250142d9afc34f9f9948154d46ce7d4ec091d0cdabbcf8b40c558
```

존재하지 않는 DB 인스턴스를 지정하는 경우 응답에 다음 오류가 포함됩니다.

- DBInstanceNotFound — *DBInstanceIdentifier*는 기존 DB 인스턴스를 참조하지 않습니다. (HTTP 상태 코드: 404)

Amazon RDS for Db2 데이터베이스 로그 파일

Amazon RDS 콘솔, AWS CLI 또는 RDS API를 사용하여 RDS for Db2 진단 로그와 알림 로그에 액세스할 수 있습니다. 파일 기반 데이터베이스 로그 보기, 다운로드 및 조사 방법에 대한 자세한 내용은 Amazon RDS 로그 파일 모니터링 단원을 참조하십시오.

주제

- 보존 일정
- Amazon CloudWatch Logs에 Db2 로그 게시

보존 일정

로그 파일은 날마다, 그리고 DB 인스턴스가 다시 시작될 때마다 교환됩니다. 다음은 Amazon RDS의 RDS for Db2 로그에 대한 보존 일정입니다.

로그 유형	보존 일정
진단 로그	Db2는 인스턴스 수준 구성의 보존 설정을 벗어난 로그를 삭제합니다. Amazon RDS는 `diagsize` 파라미터를 1,000으로 설정합니다.
알림 로그	Db2는 인스턴스 수준 구성의 보존 설정을 벗어난 로그를 삭제합니다. Amazon RDS는 `diagsize` 파라미터를 1,000으로 설정합니다.

Amazon CloudWatch Logs에 Db2 로그 게시

RDS for Db2를 사용하면 Amazon CloudWatch Logs에 직접 진단 로그 이벤트와 알림 로그 이벤트를 게시할 수 있습니다. CloudWatch Logs로 로그 데이터를 분석한 다음 CloudWatch를 사용하여 경보를 만들고 지표를 봅니다.

CloudWatch Logs을 사용하여 다음 작업을 할 수 있습니다.

- 정의한 보존 기간 만큼, 내구성이 뛰어난 스토리지 공간에 로그를 저장하십시오.
- 로그 데이터를 검색하고 필터링합니다.
- 계정 간에 로그 데이터를 공유합니다.
- Amazon S3로 로그를 내보냅니다.
- Amazon OpenSearch Service로 데이터를 스트리밍합니다.

- Amazon Kinesis Data Streams를 이용해 로그 데이터를 실시간으로 처리합니다. 자세한 내용은 SQL 애플리케이션용 Amazon Managed Service for Apache Flink 개발자 안내서의 Amazon CloudWatch Logs로 작업을 참조하세요.

Amazon RDS는 각 RDS for Db2 데이터베이스 로그를 로그 그룹에 개별적인 데이터베이스 스트림으로 게시합니다. 예를 들어, 진단 로그와 알림 로그를 게시하면 진단 데이터는 `/aws/rds/instance/my_instance/diagnostic` 로그 그룹의 진단 로그 스트림에 저장되고 알림 로그 데이터는 `/aws/rds/instance/my_instance/notify` 로그 그룹에 저장됩니다.

> **Note**
>
> RDS for Db2 로그를 CloudWatch Logs에 게시하는 기능은 기본적으로 활성화되지 않습니다. 자체 조정 메모리 관리자(STMM) 및 옵티마이저 통계 로그 게시는 지원되지 않습니다. RDS for Db2 로그를 CloudWatch Logs에 게시하는 기능은 아시아 태평양(홍콩)을 제외한 모든 리전에 지원됩니다.

콘솔

AWS Management Console에서 RDS for Db2 로그를 CloudWatch Logs에 게시하는 방법

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 다음 변경하려는 DB 인스턴스를 선택합니다.
3. 수정을 선택합니다.
4. 로그 내보내기 섹션에서 CloudWatch Logs에 게시하기 시작할 로그를 선택합니다.

   diag.log, notify.log 또는 둘 다를 선택할 수 있습니다.
5. [Continue]를 선택한 후, 요약 페이지에서 [Modify DB Instance]를 선택합니다.

AWS CLI

RDS for Db2 로그를 게시하기 위해 다음 파라미터와 함께 `modify-db-instance` 명령을 사용할 수 있습니다.

- `--db-instance-identifier`
- `--cloudwatch-logs-export-configuration`

> **Note**
>
> `--cloudwatch-logs-export-configuration` 옵션에 대한 변경 사항은 항상 DB 인스턴스에 즉시 적용됩니다. 따라서 `--apply-immediately` 및 `--no-apply-immediately` 옵션은 지정해도 아무런 효과가 없습니다.

또한 다음 명령을 사용하여 RDS for Db2 로그를 게시할 수 있습니다.

- `create-db-instance`
- `restore-db-instance-from-db-snapshot`
- `restore-db-instance-to-point-in-time`

Example

다음 예에서는 CloudWatch Logs 게시가 활성화된 RDS for Db2 DB 인스턴스를 만듭니다. `--enable-cloudwatch-logs-exports` 값은 `diag.log`, `notify.log` 또는 둘 다 포함할 수 있는 문자열의 JSON 배열입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --enable-cloudwatch-logs-exports '["diag.log","notify.log"]' \
    --db-instance-class db.m4.large \
    --engine db2-se
```

Windows의 경우:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --enable-cloudwatch-logs-exports "[\"diag.log\",\"notify.log\"]" ^
    --db-instance-class db.m4.large ^
    --engine db2-se
```

> **Note**
>
> Windows 명령 프롬프트를 사용하는 경우 백슬래시(\)를 접두사로 추가하여 JSON 코드에서 큰 따옴표(")를 이스케이프해야 합니다.

Example

다음 예에서는 CloudWatch Logs에 로그 파일을 게시하도록 기존 RDS for Db2 DB 인스턴스를 수정합니다. `--cloudwatch-logs-export-configuration` 값은 JSON 객체입니다. 이 객체의 키는 `EnableLogTypes`입니다. 값은 `diag.log`, `notify.log` 또는 둘 다 포함할 수 있는 문자열의 배열입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["diag.log","notify.log"]}'
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration "{\"EnableLogTypes\":[\"diag.log\",
\"notify.log\"]}"
```

> **Note**
>
> Windows 명령 프롬프트를 사용하는 경우 백슬래시(\)를 접두사로 추가하여 JSON 코드에서 큰 따옴표(")를 이스케이프해야 합니다.

Example

다음 예에서는 CloudWatch Logs에 로그 파일을 게시하는 기능을 비활성화하도록 기존 RDS for Db2 DB 인스턴스를 수정합니다. `--cloudwatch-logs-export-configuration` 값은 JSON 객체입니다. 이 객체의 키는 `DisableLogTypes`입니다. 값은 `diag.log`, `notify.log` 또는 둘 다 포함할 수 있는 문자열의 배열입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"DisableLogTypes":["diag.log"]}'
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration "{\"DisableLogTypes\":[\"diag.log\"]}"
```

> **Note**
>
> Windows 명령 프롬프트를 사용하는 경우 백슬래시(\)를 접두사로 추가하여 JSON 코드에서 큰 따옴표(")를 이스케이프해야 합니다.

MariaDB 데이터베이스 로그 파일

MariaDB 오류 로그, 느린 쿼리 로그, IAM 데이터베이스 인증 오류 로그, 일반 로그를 모니터링할 수 있습니다. MariaDB 오류 로그는 기본적으로 생성됩니다. DB 파라미터 그룹에서 파라미터를 설정하여 느린 쿼리 및 일반 로그를 생성할 수 있습니다. Amazon RDS는 모든 MariaDB 로그 파일을 교체합니다. 각 유형의 교체 간격은 다음과 같이 지정됩니다.

Amazon RDS 콘솔, Amazon RDS API, Amazon RDS CLI 또는 AWS SDK를 통해 MariaDB 로그를 직접 모니터링할 수 있습니다. 또한, 주 데이터베이스에 있는 데이터베이스 테이블로 로그를 전송하고 그 테이블을 쿼리하여 MariaDB 로그에 액세스할 수 있습니다. mysqlbinlog 유틸리티를 사용하여 이진 로그를 다운로드할 수 있습니다.

파일 기반 데이터베이스 로그 보기, 다운로드 및 조사 방법에 대한 자세한 내용은 Amazon RDS 로그 파일 모니터링 단원을 참조하십시오.

주제

- MariaDB 오류 로그 액세스
- MariaDB 느린 쿼리 및 일반 로그 액세스
- Amazon CloudWatch Logs에 MariaDB 로그 게시
- MariaDB의 로그 교체 및 보존
- 테이블 기반 MariaDB 로그 관리
- MariaDB 바이너리 로깅 구성
- MariaDB 이진 로그 액세스
- MariaDB 바이너리 로그 주석 활성화

MariaDB 오류 로그 액세스

MariaDB 오류 로그는 `<host-name>.err` 파일에 기록됩니다. Amazon RDS 콘솔을 사용하여 이 파일을 보거나 Amazon RDS API, Amazon RDS CLI 또는 AWS SDK를 사용하여 로그를 검색할 수 있습니다. `<host-name>.err` 파일은 5분마다 플러시되고 그 내용이 `mysql-error-running.log`에 추가됩니다. 그런 다음, `mysql-error-running.log` 파일은 1시간마다 순환되고 지난 24시간 동안 매시간 생성된 파일이 보존됩니다. 각 로그 파일이 생성된 시간(UTC)이 파일 이름에 추가됩니다. 로그 파일에는 타임스탬프도 포함되어 있어, 로그 항목이 작성된 시간을 확인하는 데 도움이 됩니다.

MariaDB에서는 시작, 종료 및 오류 발생 시에만 오류 로그에 데이터가 기록됩니다. DB 인스턴스는 오류 로그에 새 항목이 기록되지 않는 상태로 몇 시간이나 며칠씩 작동할 수 있습니다. 최근 항목이 보이지 않으면 이는 서버에서 로그에 입력된 오류가 발생하지 않았기 때문입니다.

MariaDB 느린 쿼리 및 일반 로그 액세스

DB 파라미터 그룹에서 파라미터를 설정하면 MariaDB 느린 쿼리 로그와 일반 로그를 파일이나 데이터베이스 테이블에 기록할 수 있습니다. DB 파라미터 그룹의 생성 및 변경에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오. Amazon RDS 콘솔에서 또는 Amazon RDS API, AWS CLI, AWS SDK를 사용하여 느린 쿼리 로그 또는 일반 로그를 보려면 먼저 이러한 파라미터를 설정해야 합니다.

이 목록에 있는 파라미터를 사용하여 MariaDB 로깅을 제어할 수 있습니다.

- `slow_query_log` 또는 `log_slow_query`: 느린 쿼리 로그를 만들려면 1로 설정합니다. 기본값은 0입니다.
- `general_log`: 일반 로그를 만들려면 1로 설정합니다. 기본값은 0입니다.
- `long_query_time` 또는 `log_slow_query_time`: 빠르게 실행되는 쿼리가 느린 쿼리 로그에 기록되지 않도록 하려면 로그에 기록할 쿼리의 최단 실행 시간 값(초)을 지정합니다. 기본값은 10초이고, 최소값은 0초입니다. log_output = FILE인 경우에는 마이크로초 단위까지 부동 소수점 값을 지정할 수 있습니다. log_output = TABLE인 경우에는 초 단위로 정수 값을 지정해야 합니다. 실행 시간이 `long_query_time` 또는 `log_slow_query_time` 값을 초과하는 쿼리만 로그에 기록됩니다. 예를 들어, `long_query_time` 또는 `log_slow_query_time`을 0.1로 설정하면 100밀리초 미만의 시간 동안 작동하는 쿼리가 로그에 기록되지 않습니다.
- `log_queries_not_using_indexes`: 인덱스를 사용하지 않는 모든 쿼리를 느린 쿼리 로그에 기록하려면 이 파라미터를 1로 설정합니다. 기본값은 0입니다. 인덱스를 사용하지 않는 쿼리는 실행 시간이 `long_query_time` 파라미터의 값보다 짧아도 로그에 기록됩니다.
- `log_output` *`option`*: `log_output` 파라미터에 대해 다음 옵션 중 하나를 지정할 수 있습니다.
  - TABLE(기본값)– `mysql.general_log` 테이블에는 일반 쿼리를, `mysql.slow_log` 테이블에는 느린 쿼리를 씁니다.
  - FILE– 파일 시스템에 일반 쿼리 로그와 느린 쿼리 로그를 모두 씁니다. 로그 파일은 매시간 순환됩니다.
  - NONE– 로깅을 비활성화합니다.

로깅을 사용하는 경우, Amazon RDS는 테이블 로그를 순환하거나 로그 파일을 정기적으로 삭제합니다. 이러한 예방 조치를 취하면 데이터베이스 사용에 방해가 되거나 성능에 영향을 미치는 큰 로그 파일이 생성될 가능성을 줄일 수 있습니다. FILE 및 TABLE 로깅 접근 방식 교체 및 삭제는 다음과 같습니다.

- FILE 로깅을 사용하는 경우, 로그 파일은 매시간 검사되며 24시간 이상 지난 로그 파일은 삭제됩니다. 경우에 따라 삭제 후 나머지 로그 파일의 총 크기가 DB 인스턴스에 할당된 공간 중 2%의 임계값을 초과할 수 있습니다. 이러한 경우 로그 파일의 전체 크기가 임계값 이하로 작아질 때까지 가장 큰 로그 파일부터 차례대로 삭제됩니다.
- TABLE 로깅이 활성화된 경우 경우에 따라 로그 테이블이 24시간마다 순환됩니다. 테이블 로그에서 사용되는 공간이 할당된 스토리지 공간 중 20% 이상을 차지하면 이 교체가 발생합니다. 모든 로그를 합쳤을 때 크기가 10GB를 초과하는 경우에도 교체됩니다. DB 인스턴스에 대해 사용된 공간의 양이 DB 인스턴스의 할당된 스토리지 공간 중 90% 이상을 차지할 경우, 로그 교체를 위한 임계값은 줄어듭니다. 테이블 로그에서 사용되는 공간이 할당된 스토리지 공간 중 10% 이상을 차지하면 로그 테이블이 교체됩니다. 모든 로그를 합쳤을 때 크기가 10GB를 초과하는 경우에도 교체됩니다.

  로그 테이블이 순환되면 현재 로그 테이블은 백업 로그 테이블에 복사되며 현재 로그 테이블의 해당 항목들은 제거됩니다. 백업 로그 테이블이 이미 존재할 경우, 현재 로그 테이블이 백업으로 복사되기 전에 백업 로그 테이블이 삭제됩니다. 필요하다면 백업 로그 테이블을 쿼리할 수 있습니다. `mysql.general_log` 테이블에 대한 백업 로그 테이블 이름은 `mysql.general_log_backup`으로 지정됩니다. `mysql.slow_log` 테이블에 대한 백업 로그 테이블 이름은 `mysql.slow_log_backup`으로 지정됩니다.

  `mysql.general_log` 절차를 호출하면 `mysql.rds_rotate_general_log` 테이블을 순환할 수 있습니다. `mysql.slow_log` 절차를 호출하면 `mysql.rds_rotate_slow_log` 테이블을 순환할 수 있습니다.

  데이터베이스 버전 업그레이드가 진행되는 동안 테이블 로그가 순환됩니다.

Amazon RDS는 TABLE 및 FILE 로그 순환을 Amazon RDS 이벤트에 기록하고 사용자에게 알림 메시지를 보냅니다.

Amazon RDS 콘솔, Amazon RDS API, Amazon RDS CLI 또는 AWS SDK에서 로그를 사용하여 작업하려면 `log_output` 파라미터를 FILE로 설정합니다. MariaDB 오류 로그와 같이, 이런 로그 파일은 매시간 순환됩니다. 이전의 24시간 동안 생성된 로그 파일이 보존됩니다.

느린 쿼리 및 일반 로그에 대한 자세한 내용은 MariaDB 문서에서 다음 단원을 참조하십시오.

- 느린 쿼리 로그
- 일반 쿼리 로그

Amazon CloudWatch Logs에 MariaDB 로그 게시

Amazon CloudWatch Logs의 로그 그룹에 로그 데이터를 게시하도록 MariaDB DB 인스턴스를 구성할 수 있습니다. CloudWatch Logs를 통해 로그 데이터에 대한 실시간 분석을 수행할 수 있고, CloudWatch를 사용하여 경보를 만들고 지표를 볼 수 있습니다. CloudWatch Logs를 사용하여 내구성이 뛰어난 스토리지에 로그 레코드를 저장할 수 있습니다.

Amazon RDS는 각 MariaDB 데이터베이스 로그를 로그 그룹에 개별적인 데이터베이스 스트림으로 게시합니다. 예를 들어 느린 쿼리 포함하도록 내보내기 함수를 구성한다고 가정합니다. 그러면 느린 쿼리 데이터가 `/aws/rds/instance/my_instance/slowquery` 로그 그룹의 느린 쿼리 로그 스트림에 저장됩니다.

오류 로그는 기본적으로 활성화됩니다. 다음 표에는 기타 MariaDB 로그의 요구 사항이 요약되어 있습니다.

로그	요구 사항
감사 로그	DB 인스턴스는 `MARIADB_AUDIT_PLUGIN` 옵션과 함께 사용자 지정 옵션 그룹을 사용해야 합니다.
일반 로그	DB 인스턴스는 파라미터 설정 `general_log = 1`과 함께 사용자 지정 파라미터 그룹을 사용하여 일반 로그를 활성화해야 합니다.
느린 쿼리 로그	DB 인스턴스는 파라미터 설정 `slow_query_log = 1` 또는 `log_slow_query = 1`과 함께 사용자 지정 파라미터 그룹을 사용하여 느린 쿼리 로그를 활성화해야 합니다.
IAM 데이터베이스 인증 오류 로그	DB 인스턴스를 만들거나 수정하여 DB 인스턴스의 `iam-db-auth-error` 로그 유형을 사용 설정해야 합니다.
로그 출력	DB 인스턴스는 파라미터 설정 `log_output = FILE` 과 함께 사용자 지정 파라미터 그룹을 사용하여 로그를 파일 시스템에 쓰고 CloudWatch Logs에 게시해야 합니다.

콘솔

콘솔에서 CloudWatch Logs에 MariaDB 로그를 게시하려면

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 다음 변경하려는 DB 인스턴스를 선택합니다.
3. 수정을 선택합니다.
4. 로그 내보내기 섹션에서 CloudWatch Logs에 게시하기 시작할 로그를 선택합니다.
5. [Continue]를 선택한 후, 요약 페이지에서 [Modify DB Instance]를 선택합니다.

AWS CLI

AWS CLI를 사용하여 MariaDB 로그를 게시할 수 있습니다. 다음 파라미터로 `modify-db-instance` 명령을 호출할 수 있습니다.

- `--db-instance-identifier`
- `--cloudwatch-logs-export-configuration`

> **Note**
>
> `--cloudwatch-logs-export-configuration` 옵션에 대한 변경 사항은 항상 DB 인스턴스에 즉시 적용됩니다. 따라서 `--apply-immediately` 및 `--no-apply-immediately` 옵션은 지정해도 아무런 효과가 없습니다.

또 다음 AWS CLI 명령을 호출해 MariaDB 로그를 게시할 수 있습니다.

- `create-db-instance`
- `restore-db-instance-from-db-snapshot`
- `restore-db-instance-from-s3`
- `restore-db-instance-to-point-in-time`

다음 옵션으로 AWS CLI 명령 중 하나를 실행합니다.

- `--db-instance-identifier`
- `--enable-cloudwatch-logs-exports`

- `--db-instance-class`
- `--engine`

실행하는 AWS CLI 명령에 따라 다른 옵션이 필요할 수 있습니다.

Example

다음 예에서는 CloudWatch Logs에 로그 파일을 게시하도록 기존 MariaDB DB 인스턴스를 수정합니다. `--cloudwatch-logs-export-configuration` 값은 JSON 객체입니다. 이 객체에 대한 키는 `EnableLogTypes`이며, 해당 값은 `audit`, `error`, `general` 및 `slowquery`의 조합을 사용하는 문자열의 배열입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["audit","error","general","slowquery"]}'
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["audit","error","general","slowquery"]}'
```

Example

다음 명령은 CloudWatch Logs에 로그 파일을 게시하도록 MariaDB DB 인스턴스를 만듭니다. `--enable-cloudwatch-logs-exports` 값은 문자열의 JSON 배열입니다. 문자열은 `audit`, `error`, `general` 및 `slowquery`의 조합일 수 있습니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --enable-cloudwatch-logs-exports '["audit","error","general","slowquery"]' \
```

```
    --db-instance-class db.m4.large \
    --engine mariadb
```

Windows의 경우:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --enable-cloudwatch-logs-exports '["audit","error","general","slowquery"]' ^
    --db-instance-class db.m4.large ^
    --engine mariadb
```

RDS API

RDS API를 사용하여 MariaDB 로그를 게시할 수 있습니다. 다음 파라미터와 함께 ModifyDBInstance 작업을 호출할 수 있습니다.

- DBInstanceIdentifier
- CloudwatchLogsExportConfiguration

> **Note**
>
> CloudwatchLogsExportConfiguration 파라미터에 대한 변경 사항은 항상 DB 인스턴스에 즉시 적용됩니다. 따라서 ApplyImmediately 파라미터는 지정해도 아무런 효과가 없습니다.

또한 다음 RDS API 작업을 호출해 MariaDB 로그를 게시할 수 있습니다.

- CreateDBInstance
- RestoreDBInstanceFromDBSnapshot
- RestoreDBInstanceFromS3
- RestoreDBInstanceToPointInTime

다음 파라미터로 RDS API 작업 중 하나를 실행합니다.

- DBInstanceIdentifier

- EnableCloudwatchLogsExports
- Engine
- DBInstanceClass

실행하는 AWS CLI 명령에 따라 다른 파라미터가 필요할 수 있습니다.

MariaDB의 로그 교체 및 보존

로깅을 사용하는 경우, Amazon RDS는 테이블 로그를 순환하거나 로그 파일을 정기적으로 삭제합니다. 이러한 예방 조치를 취하면 데이터베이스 사용에 방해가 되거나 성능에 영향을 미치는 큰 로그 파일이 생성될 가능성을 줄일 수 있습니다.

MariaDB 느린 쿼리 로그, 오류 로그 및 일반 로그 파일 크기는 DB 인스턴스에 대해 할당된 스토리지 공간의 2% 이하로 제한됩니다. 이 임계값을 유지하기 위해 로그는 매시간 자동으로 순환되면서 24시간 이상 지난 로그 파일은 제거됩니다. 오래된 로그 파일을 제거한 후 로그 파일의 총 크기가 임계값을 초과하는 경우에는 로그 파일의 전체 크기가 임계값 이하로 작아질 때까지 가장 큰 로그 파일부터 차례대로 삭제됩니다.

Amazon RDS는 10MB보다 큰 IAM 데이터베이스 인증 오류 로그 파일을 교체합니다. Amazon RDS는 5일 이상 또는 100MB 이상의 IAM 데이터베이스 인증 오류 로그 파일을 제거합니다.

테이블 기반 MariaDB 로그 관리

일반 및 느린 쿼리 로그를 DB 인스턴스상의 테이블로 전송할 수 있습니다. 이렇게 하려면 DB 파라미터 그룹을 만들고 log_output 서버 파라미터를 TABLE로 설정합니다. 그러면 일반 쿼리는 mysql.general_log 테이블에, 느린 쿼리는 mysql.slow_log 테이블에 로그가 기록됩니다. 이들 테이블을 쿼리하여 로그 정보에 액세스할 수 있습니다. 이 로깅을 활성화하면 데이터베이스에 기록되는 데이터의 양이 증가하여 성능이 저하될 수 있습니다.

일반 로그와 느린 쿼리 로그는 모두 기본적으로 비활성화됩니다. 표에 대한 로깅을 활성화하려면 다음 서버 파라미터도 1로 설정해야 합니다.

- general_log
- slow_query_log 또는 log_slow_query

관련 파라미터를 0으로 설정하여 각 로깅 활동을 해제할 때까지 로그 테이블은 계속 커집니다. 흔히 대량의 데이터가 시간 경과에 따라 누적되어 할당된 스토리지 공간 중 상당한 비율을 사용할 수 있습니다. Amazon RDS에서는 로그 테이블을 자를 수 없지만 테이블의 콘텐츠를 이동할 수 있습니다. 테이블

을 순환하면 그 내용이 백업 테이블에 저장된 다음 빈 로그 테이블이 새로 생성됩니다. 다음 명령줄 프로시저로 로그 테이블을 수동으로 순환시킬 수 있으며, 이때 명령 프롬프트는 PROMPT>로 표시됩니다.

```
PROMPT> CALL mysql.rds_rotate_slow_log;
PROMPT> CALL mysql.rds_rotate_general_log;
```

오래된 데이터를 완전히 제거하고 디스크 공간을 회수하려면 알맞은 프로시저를 두 번 연속으로 호출합니다.

MariaDB 바이너리 로깅 구성

이진 로그는 MariaDB 서버 인스턴스의 데이터 수정에 대한 정보를 포함하는 로그 파일 세트입니다. 이진 로그에는 다음과 같은 정보가 포함되어 있습니다.

- 테이블 생성 또는 행 수정과 같은 데이터베이스 변경 사항을 설명하는 이벤트
- 데이터를 업데이트한 각 문의 기간에 대한 정보
- 데이터를 업데이트할 수 있었지만 업데이트하지 않은 문에 대한 이벤트

이진 로그는 복제 중 전송된 문을 기록합니다. 일부 복구 작업에도 필요합니다. 자세한 내용은 MariaDB 설명서에서 Binary Log를 참조하세요.

자동 백업 기능은 MariaDB에 대해 이진 로깅을 설정할지 아니면 해제할지를 결정합니다. 다음과 같은 옵션이 있습니다:

이진 로깅 설정

백업 보존 기간을 0이 아닌 양수 값으로 설정합니다.

이진 로깅 해제

백업 보존 기간을 0으로 설정합니다.

자세한 내용은 자동 백업 활성화 단원을 참조하십시오.

Amazon RDS의 MariaDB는 행 기반, 설명문 기반 및 혼합 바이너리 로깅 형식을 지원합니다. 기본 이진 로깅 형식은 혼합입니다. 다양한 MariaDB 이진 로그 형식에 관한 세부 정보는 MariaDB 설명서에서 이진 로그 형식 단원을 참조하십시오.

복제를 사용할 계획이라면 이진 로깅 형식이 중요합니다. 이진 로깅 형식이 원본에 기록되고 복제 대상으로 전송되는 데이터 변경 내용의 레코드를 결정하기 때문입니다. 복제와 관련된 다양한 바이너

리 로깅 형식의 장/단점에 대한 자세한 내용은 MySQL 설명서의 Advantages and Disadvantages of Statement-Based and Row-Based Replication을 참조하십시오.

> **Important**
>
> 이진 로깅 형식을 행 기반으로 설정하면 이진 로그 파일이 매우 커질 수 있습니다. 큰 이진 로그 파일은 DB 인스턴스가 사용할 수 있는 스토리지의 양이 줄어들게 합니다. 또한 DB 인스턴스의 복원 작업 수행에 필요한 시간이 늘어나게 할 수도 있습니다.
> 설명문 기반 복제는 원본 DB 인스턴스와 읽기 전용 복제본 간의 불일치를 초래할 수 있습니다. 자세한 내용은 MariaDB 설명서의 Unsafe Statements for Statement-based Replication을 참조하십시오.
> 이진 로깅을 활성화하면 DB 인스턴스에 대한 평균 디스크 쓰기 I/O 작업 수가 증가합니다. `WriteIOPS` CloudWatch 지표를 사용하여 IOPS 사용량을 모니터링할 수 있습니다.

MariaDB 이진 로깅 형식을 설정하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.
3. 수정할 DB 인스턴스에 사용되는 파라미터 그룹을 선택합니다.

   기본 파라미터 그룹을 수정할 수 없습니다. DB 인스턴스에서 기본 파라미터 그룹을 사용 중인 경우 새 파라미터 그룹을 생성하여 DB 인스턴스와 연결합니다.

   DB 파라미터 그룹에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.
4. 파라미터 그룹 작업에서 편집을 선택합니다.
5. `binlog_format` 파라미터를 선택한 바이너리 로깅 형식(ROW, STATEMENT 또는 MIXED)으로 설정합니다.

   DB 인스턴스의 백업 보존 기간을 0으로 설정하여 이진 로깅을 끌 수 있지만, 이렇게 하면 일일 자동 백업이 비활성화됩니다. 자동 백업을 비활성화하면 `log_bin` 세션 변수가 꺼지거나 비활성화됩니다. 그러면 RDS for MariaDB DB 인스턴스에 대한 이진 로깅이 비활성화되며, 이는 다시 `binlog_format` 세션 변수를 데이터베이스의 기본값인 `ROW` 값으로 재설정합니다. 백업을 비활성화하지 않는 것이 좋습니다. 백업 보존 기간 설정에 대한 자세한 정보는 DB 인스턴스에 대한 설정 섹션을 참조하세요.
6. 변경 내용 저장을 선택하여 업데이트를 DB 파라미터 그룹에 저장합니다.

binlog_format 파라미터는 RDS for MariaDB에서 동적이므로 DB 인스턴스를 재부팅하지 않고 변경 사항을 적용할 수 있습니다.

> **Important**
>
> DB 파라미터 그룹을 변경하면 해당 파라미터 그룹을 사용하는 모든 DB 인스턴스에 영향을 미칩니다. AWS 리전의 다양한 MariaDB DB 인스턴스에 대해 서로 다른 바이너리 로깅 형식을 지정하려면 DB 인스턴스에서 서로 다른 DB 파라미터 그룹을 사용해야 합니다. 이러한 파라미터 그룹은 다양한 로깅 형식을 식별합니다. 각 DB 인스턴스에 적절한 DB 파라미터 그룹을 할당합니다.

MariaDB 이진 로그 액세스

mysqlbinlog 유틸리티를 사용하여 MariaDB DB 인스턴스로부터 텍스트 형식의 이진 로그를 다운로드할 수 있습니다. 이진 로그는 로컬 컴퓨터로 다운로드됩니다. mysqlbinlog 유틸리티 사용에 대한 자세한 내용은 MariaDB 설명서에서 mysqlbinlog 사용을 참조하십시오.

Amazon RDS 인스턴스에 대해 mysqlbinlog 유틸리티를 실행하려면 다음 옵션을 사용합니다.

- --read-from-remote-server 옵션을 지정합니다.
- --host: 인스턴스의 엔드포인트에서 DNS 이름을 지정합니다.
- --port: 인스턴스에서 사용되는 포트를 지정합니다.
- --user: 복제 슬레이브 권한이 부여된 MariaDB 사용자를 지정합니다.
- --password: 사용자의 암호를 지정하거나, 유틸리티에서 암호 입력을 요구하는 메시지가 표시되도록 암호 값을 생략합니다.
- --result-file: 출력을 수신할 로컬 파일을 지정합니다.
- 하나 이상의 이진 로그 파일의 이름을 지정합니다. 사용 가능한 로그의 목록을 획득하려면 SQL 명령 SHOW BINARY LOGS를 사용합니다.

mysqlbinlog 옵션에 대한 자세한 내용은 MariaDB 설명서에서 mysqlbinlog 옵션을 참조하십시오.

다음은 그 예제입니다.

대상 LinuxmacOS, 또는Unix:

```
mysqlbinlog \
```

```
    --read-from-remote-server \
    --host=mariadbinstance1.1234abcd.region.rds.amazonaws.com \
    --port=3306  \
    --user ReplUser \
    --password <password> \
    --result-file=/tmp/binlog.txt
```

Windows의 경우:

```
mysqlbinlog ^
    --read-from-remote-server ^
    --host=mariadbinstance1.1234abcd.region.rds.amazonaws.com ^
    --port=3306  ^
    --user ReplUser ^
    --password <password> ^
    --result-file=/tmp/binlog.txt
```

Amazon RDS는 보통은 최대한 빨리 이진 파일을 삭제합니다. 하지만 mysqlbinlog가 액세스할 수 있도록 인스턴스에서 이진 파일을 여전히 사용할 수 있어야 합니다. RDS의 이진수 로그 보관 시간을 지정하려면 `mysql.rds_set_configuration` 저장 프로시저를 사용합니다. 로그를 다운로드할 수 있는 충분한 기간을 지정합니다. 보존 기간을 설정한 후, DB 인스턴스의 스토리지 사용량을 모니터링하여 보존된 이진 로그가 너무 많은 스토리지를 차지하지 않도록 합니다.

다음 예제에서는 보존 기간을 1일로 설정합니다.

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

현재 설정을 표시하려면 `mysql.rds_show_configuration` 저장 프로시저를 사용합니다.

```
call mysql.rds_show_configuration;
```

MariaDB 바이너리 로그 주석 활성화

MariaDB DB 인스턴스에서, `Annotate_rows` 이벤트를 사용하여 행 이벤트로 인해 발생한 SQL 쿼리의 복사본으로 해당 행 이벤트에 주석을 달 수 있습니다. 이 접근 방식은 RDS for MySQL DB 인스턴스에서 `binlog_rows_query_log_events` 파라미터를 활성화하는 것과 비슷한 기능을 제공합니다.

사용자 지정 파라미터 그룹을 생성하고 `binlog_annotate_row_events` 파라미터를 **1**로 설정하여 이진 로그 주석을 전역에서 활성화할 수 있습니다. 또한 `SET SESSION binlog_annotate_row_events = 1` 호출로 세션 수준에서 주석을 활성화할 수도 있습니다. 복제

본 인스턴스에서 바이너리 로깅이 사용되는 경우 replicate_annotate_row_events를 사용하여 바이너리 로그 주석을 복제본 인스턴스로 복제합니다. 이러한 설정을 사용하는 데는 특별한 권한이 필요하지 않습니다.

다음 예는 MariaDB의 행 기반 트랜잭션입니다. 트랜잭션 격리 수준을 읽기 커밋으로 설정하면 행 기반 로깅 사용이 트리거됩니다.

```
CREATE DATABASE IF NOT EXISTS test;
USE test;
CREATE TABLE square(x INT PRIMARY KEY, y INT NOT NULL) ENGINE = InnoDB;
SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
BEGIN
INSERT INTO square(x, y) VALUES(5, 5 * 5);
COMMIT;
```

주석이 없을 경우, 트랜잭션의 이진 로그 항목은 다음과 같습니다.

```
BEGIN
/*!*/;
# at 1163
# at 1209
#150922  7:55:57 server id 1855786460  end_log_pos 1209         Table_map:
 `test`.`square` mapped to number 76
#150922  7:55:57 server id 1855786460  end_log_pos 1247         Write_rows: table id 76
 flags: STMT_END_F
### INSERT INTO `test`.`square`
### SET
###   @1=5
###   @2=25
# at 1247
#150922  7:56:01 server id 1855786460  end_log_pos 1274         Xid = 62
COMMIT/*!*/;
```

다음 문은 이 동일한 트랜잭션에 대해 세션 수준의 주석을 활성화하고, 트랜잭션을 커밋한 후 다시 비활성화합니다.

```
CREATE DATABASE IF NOT EXISTS test;
USE test;
CREATE TABLE square(x INT PRIMARY KEY, y INT NOT NULL) ENGINE = InnoDB;
SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET SESSION binlog_annotate_row_events = 1;
BEGIN;
```

```
INSERT INTO square(x, y) VALUES(5, 5 * 5);
COMMIT;
SET SESSION binlog_annotate_row_events = 0;
```

주석이 있는 경우, 트랜잭션의 이진 로그 항목은 다음과 같습니다.

```
BEGIN
/*!*/;
# at 423
# at 483
# at 529
#150922  8:04:24 server id 1855786460  end_log_pos 483  Annotate_rows:
#Q> INSERT INTO square(x, y) VALUES(5, 5 * 5)
#150922  8:04:24 server id 1855786460  end_log_pos 529  Table_map: `test`.`square`
 mapped to number 76
#150922  8:04:24 server id 1855786460  end_log_pos 567  Write_rows: table id 76 flags:
 STMT_END_F
### INSERT INTO `test`.`square`
### SET
###   @1=5
###   @2=25
# at 567
#150922  8:04:26 server id 1855786460  end_log_pos 594  Xid = 88
COMMIT/*!*/;
```

Amazon RDS for Microsoft SQL Server 데이터베이스 로그 파일

Amazon RDS 콘솔 또는 AWS CLI 또는 RDS API를 사용하여 Microsoft SQL Server 오류 로그, 에이전트 로그, 추적 파일 및 덤프 파일에 액세스할 수 있습니다. 파일 기반 데이터베이스 로그 보기, 다운로드 및 조사 방법에 대한 자세한 내용은 Amazon RDS 로그 파일 모니터링 단원을 참조하십시오.

보존 일정

로그 파일은 날마다, 그리고 DB 인스턴스가 다시 시작될 때마다 교환됩니다. 다음은 Amazon RDS의 Microsoft SQL Server 로그에 대한 보존 일정입니다.

로그 유형	보존 일정
오류 로그	최대 30개의 오류 로그가 보존됩니다. Amazon RDS는 7일이 경과한 오류 로그를 삭제할 수도 있습니다.
에이전트 로그	최대 10개의 에이전트 로그가 보존됩니다. Amazon RDS에서는 7일이 경과한 에이전트 로그를 삭제할 수 있습니다.
추적 파일	추적 파일은 DB 인스턴스의 추적 파일 보존 기간에 따라 보존됩니다. 추적 파일의 기본 보존 기간은 7일입니다. DB 인스턴스의 추적 파일 보존 기간을 수정하려면 추적 및 덤프 파일의 보존 기간 설정 단원을 참조하십시오.
덤프 파일	덤프 파일은 DB 인스턴스의 덤프 파일 보존 기간에 따라 보존됩니다. 덤프 파일의 기본 보존 기간은 7일입니다. DB 인스턴스의 덤프 파일 보존 기간을 수정하려면 추적 및 덤프 파일의 보존 기간 설정 단원을 참조하십시오.

rds_read_error_log 프로시저를 사용하여 SQL Server 오류 로그 보기

Amazon RDS 저장 프로시저 `rds_read_error_log`를 사용하여 오류 로그 및 에이전트 로그를 볼 수 있습니다. 자세한 내용은 오류 및 에이전트 로그 보기 섹션을 참조하세요.

Amazon CloudWatch Logs에 SQL Server 로그 게시

Amazon RDS for SQL Server의 경우, 오류 및 에이전트 로그 이벤트를 Amazon CloudWatch Logs에 직접 게시할 수 있습니다. CloudWatch Logs로 로그 데이터를 분석한 다음 CloudWatch를 사용하여 경보를 만들고 지표를 봅니다.

CloudWatch Logs을 사용하여 다음 작업을 할 수 있습니다.

- 정의한 보존 기간 만큼, 내구성이 뛰어난 스토리지 공간에 로그를 저장하십시오.
- 로그 데이터를 검색하고 필터링합니다.
- 계정 간에 로그 데이터를 공유합니다.
- Amazon S3로 로그를 내보냅니다.
- Amazon OpenSearch Service로 데이터를 스트리밍합니다.
- Amazon Kinesis Data Streams를 이용해 로그 데이터를 실시간으로 처리합니다. 자세한 내용은 SQL 애플리케이션용 Amazon Managed Service for Apache Flink 개발자 안내서의 Amazon CloudWatch Logs로 작업을 참조하세요.

Amazon RDS는 각 SQL Server 데이터베이스 로그를 로그 그룹에 개별적인 데이터베이스 스트림으로 게시합니다. 예를 들어, 에이전트 로그 및 오류 로그를 게시하면 오류 데이터가 `/aws/rds/instance/my_instance.node1/error` 로그 그룹의 오류 로그 스트림에 저장되고 에이전트 로그 데이터가 `/aws/rds/instance/my_instance.node1/agent` 로그 그룹에 저장됩니다.

다중 AZ DB 인스턴스의 경우 Amazon RDS는 데이터베이스 로그를 로그 그룹에 별도의 두 스트림으로 게시합니다. 예를 들어 오류 로그를 게시하는 경우 오류 데이터가 오류 로그 스트림 `/aws/rds/instance/my_instance.node1/error`와 `/aws/rds/instance/my_instance.node2/error`에 각각 저장됩니다. 로그 스트림은 장애 조치 중에 변경되지 않으며 각 노드의 오류 로그 스트림에는 기본 또는 보조 인스턴스의 오류 로그가 포함될 수 있습니다. 다중 AZ를 사용하면 DB 인스턴스 장애 조치와 같은 이벤트 데이터를 저장하기 위해 `/aws/rds/instance/my_instance/rds-events`에 대한 로그 스트림이 자동으로 생성됩니다.

> **Note**
>
> CloudWatch Logs의 SQL Server 로그 게시는 기본적으로 활성화되지 않습니다. 추적 및 덤프 파일 게시는 지원되지 않습니다. CloudWatch Logs에 SQL Server 로그 게시는 아시아 태평양(홍콩)을 제외한 모든 리전에 지원됩니다.

콘솔

AWS Management Console에서 SQL Server DB 로그를 CloudWatch Logs에 게시하려면

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 다음 변경하려는 DB 인스턴스를 선택합니다.
3. 수정을 선택합니다.

4. 로그 내보내기 섹션에서 CloudWatch Logs에 게시하기 시작할 로그를 선택합니다.

   에이전트 로그(Agent log), 오류 로그(Error log) 또는 둘 다를 선택할 수 있습니다.

5. [Continue]를 선택한 후, 요약 페이지에서 [Modify DB Instance]를 선택합니다.

AWS CLI

SQL Server 로그를 게시하기 위해 다음 파라미터와 함께 `modify-db-instance` 명령을 사용할 수 있습니다.

- `--db-instance-identifier`
- `--cloudwatch-logs-export-configuration`

> **Note**
>
> `--cloudwatch-logs-export-configuration` 옵션에 대한 변경 사항은 항상 DB 인스턴스에 즉시 적용됩니다. 따라서 `--apply-immediately` 및 `--no-apply-immediately` 옵션은 지정해도 아무런 효과가 없습니다.

또한 다음 명령을 사용하여 SQL Server 로그를 게시할 수 있습니다.

- `create-db-instance`
- `restore-db-instance-from-db-snapshot`
- `restore-db-instance-to-point-in-time`

Example

다음 예에서는 CloudWatch Logs 게시가 활성화된 SQL Server DB 인스턴스를 생성합니다. `--enable-cloudwatch-logs-exports` 값은 `error`, `agent` 또는 둘 다 포함할 수 있는 문자열의 JSON 배열입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --enable-cloudwatch-logs-exports '["error","agent"]' \
    --db-instance-class db.m4.large \
```

```
    --engine sqlserver-se
```

Windows의 경우:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --enable-cloudwatch-logs-exports "[\"error\",\"agent\"]" ^
    --db-instance-class db.m4.large ^
    --engine sqlserver-se
```

> **Note**
>
> Windows 명령 프롬프트를 사용하는 경우 백슬래시(\)를 접두사로 추가하여 JSON 코드에서 큰 따옴표(")를 이스케이프해야 합니다.

Example

다음 예에서는 CloudWatch Logs에 로그 파일을 게시하도록 기존 SQL Server DB 인스턴스를 수정합니다. `--cloudwatch-logs-export-configuration` 값은 JSON 객체입니다. 이 객체의 키는 `EnableLogTypes`입니다. 값은 `error`, `agent` 또는 둘 다 포함할 수 있는 문자열의 배열입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":["error","agent"]}'
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration "{\"EnableLogTypes\":[\"error\",\"agent\"]}"
```

> **Note**
>
> Windows 명령 프롬프트를 사용하는 경우 백슬래시(\)를 접두사로 추가하여 JSON 코드에서 큰 따옴표(")를 이스케이프해야 합니다.

Example

다음 예제는 기존 SQL Server DB 인스턴스를 수정하여 CloudWatch Logs에 에이전트 로그 파일 게시를 비활성화합니다. `--cloudwatch-logs-export-configuration` 값은 JSON 객체입니다. 이 객체의 키는 `DisableLogTypes`입니다. 값은 `error`, `agent` 또는 둘 다 포함할 수 있는 문자열의 배열입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"DisableLogTypes":["agent"]}'
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration "{\"DisableLogTypes\":[\"agent\"]}"
```

> **Note**
>
> Windows 명령 프롬프트를 사용하는 경우 백슬래시(\)를 접두사로 추가하여 JSON 코드에서 큰 따옴표(")를 이스케이프해야 합니다.

MySQL 데이터베이스 로그 파일

Amazon RDS 콘솔, Amazon RDS API, AWS CLI 또는 AWS SDK를 통해 MySQL 로그를 직접 모니터링할 수 있습니다. 또한, 주 데이터베이스에 있는 데이터베이스 테이블로 로그를 전송하고 그 테이블을 쿼리하여 MySQL 로그에 액세스할 수 있습니다. mysqlbinlog 유틸리티를 사용하여 이진 로그를 다운로드할 수 있습니다.

파일 기반 데이터베이스 로그 보기, 다운로드 및 조사 방법에 대한 자세한 내용은 Amazon RDS 로그 파일 모니터링 단원을 참조하십시오.

주제

- RDS for MySQL 데이터베이스 로그 개요
- Amazon CloudWatch Logs에 MySQL 로그 게시
- MySQL 로그 출력을 표로 보내기
- 단일 AZ 데이터베이스의 RDS for MySQL 이진 로깅 구성
- 다중 AZ DB 클러스터에 대한 MySQL 이진 로깅 구성
- MySQL 이진 로그 액세스

RDS for MySQL 데이터베이스 로그 개요

다음 유형의 RDS for MySQL 로그 파일을 모니터링할 수 있습니다.

- 오류 로그
- 느린 쿼리 로그
- 일반 로그
- 감사 로그
- 인스턴스 로그
- IAM 데이터베이스 인증 오류 로그

RDS for MySQL 오류 로그는 기본적으로 생성됩니다. DB 파라미터 그룹에서 파라미터를 설정하여 느린 쿼리 및 일반 로그를 생성할 수 있습니다.

주제

- RDS for MySQL 오류 로그
- RDS for MySQL 느린 쿼리 로그 및 일반 로그

- MySQL 감사 로그
- RDS for MySQL용 로그 교체 및 보존
- 다시 실행 로그의 크기 제한

RDS for MySQL 오류 로그

RDS for MySQL은 `mysql-error.log` 파일에 오류를 기록합니다. 각 로그 파일이 생성된 시간(UTC)이 파일 이름에 추가됩니다. 로그 파일에는 타임스탬프도 포함되어 있어, 로그 항목이 작성된 시간을 확인하는 데 도움이 됩니다.

RDS for MySQL에서는 시작, 종료 및 오류 발생 시에만 오류 로그에 데이터가 로그됩니다. DB 인스턴스는 오류 로그에 새 항목이 기록되지 않는 상태로 몇 시간이나 며칠씩 작동할 수 있습니다. 최근 항목이 보이지 않으면 이는 서버에서 로그에 입력될 만한 오류가 발생하지 않았기 때문입니다.

기본적으로 오류 로그는 오류와 같은 예기치 않은 이벤트만 표시되도록 필터링됩니다. 그러나 오류 로그에는 쿼리 진행률과 같이 표시되지 않는 몇 가지 추가 데이터베이스 정보도 포함되어 있습니다. 따라서 실제 오류가 없더라도 진행 중인 데이터베이스 작업으로 인해 오류 로그의 크기가 커질 수 있습니다. 그리고 AWS Management Console에서 오류 로그의 특정 크기(바이트 또는 킬로바이트)를 볼 수 있지만 다운로드할 때 0바이트일 수 있습니다.

RDS for MySQL은 5분마다 `mysql-error.log`를 디스크에 씁니다. 또한, 로그의 콘텐츠를 `mysql-error-running.log`에 추가합니다.

RDS for MySQL은 `mysql-error-running.log` 파일을 1시간마다 교체합니다. 또한, 지난 2주 동안 생성된 로그를 보존합니다.

> **Note**
>
> 로그 보존 기간은 Amazon RDS와 Aurora 간에 다릅니다.

RDS for MySQL 느린 쿼리 로그 및 일반 로그

RDS for MySQL 느린 쿼리 로그와 일반 로그를 파일이나 데이터베이스 테이블에 기록할 수 있습니다. 그러려면 DB 파라미터 그룹의 파라미터를 설정합니다. DB 파라미터 그룹의 생성 및 변경에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오. 이런 파라미터를 설정해야 Amazon RDS 콘솔에서 느린 쿼리 로그 또는 일반 로그를 볼 수 있습니다. 아니면 Amazon RDS API, Amazon RDS CLI 또는 AWS SDK를 사용하여 볼 수 있습니다.

이 목록에 있는 파라미터를 사용하여 RDS for MySQL 로깅을 제어할 수 있습니다.

- `slow_query_log`: 느린 쿼리 로그를 만들려면 1로 설정합니다. 기본값은 0입니다.
- `general_log`: 일반 로그를 만들려면 1로 설정합니다. 기본값은 0입니다.
- `long_query_time`: 빠르게 실행되는 쿼리가 느린 쿼리 로그에 기록되지 않도록 하려면 로깅할 쿼리의 최단 런타임 값(초)을 지정합니다. 기본값은 10초이고, 최소값은 0초입니다. log_output = FILE인 경우에는 마이크로초 단위까지 부동 소수점 값을 지정할 수 있습니다. log_output = TABLE인 경우에는 초 단위로 정수 값을 지정해야 합니다. 런타임이 `long_query_time` 값을 초과하는 쿼리만 로깅됩니다. 예를 들어 `long_query_time`을 0.1로 설정하면 100밀리초 미만의 시간 동안 작동하는 쿼리가 로그에 기록되지 않습니다.
- `log_queries_not_using_indexes`: 인덱스를 사용하지 않는 모든 쿼리를 느린 쿼리 로그에 기록하려면 1로 설정합니다. 인덱스를 사용하지 않는 쿼리는 런타임이 `long_query_time` 파라미터의 값보다 짧아도 로깅됩니다. 기본값은 0입니다.
- `log_output` *option*: `log_output` 파라미터에 대해 다음 옵션 중 하나를 지정할 수 있습니다.
  - TABLE(기본값) – 일반 쿼리를 `mysql.general_log`테이블에 쓰고 느린 쿼리를 `mysql.slow_log` 테이블에 씁니다.
  - FILE – 일반 쿼리 로그와 느린 쿼리 로그를 모두 파일 시스템에 씁니다.
  - NONE – 로깅을 비활성화합니다.

느린 쿼리 데이터가 Amazon CloudWatch Logs에 표시되려면 다음 조건을 충족해야 합니다.

- 느린 쿼리 로그를 포함하도록 CloudWatch Logs를 구성해야 합니다.
- `slow_query_log`를 활성화해야 합니다.
- `log_output`를 `FILE`로 설정해야 합니다.
- 쿼리는 `long_query_time`에 구성된 시간보다 오래 걸릴 수 있습니다.

느린 쿼리 및 일반 로그에 대한 자세한 내용은 MySQL 문서에서 다음 항목을 참조하십시오.

- 느린 쿼리 로그
- 일반 쿼리 로그

MySQL 감사 로그

감사 로그에 액세스하려면 DB 인스턴스가 `MARIADB_AUDIT_PLUGIN` 옵션과 함께 사용자 지정 옵션 그룹을 사용해야 합니다. 자세한 내용은 MySQL에 대한 MariaDB 감사 플러그인 지원 섹션을 참조하세요.

RDS for MySQL용 로그 교체 및 보존

로깅을 사용하는 경우, Amazon RDS는 테이블 로그를 순환하거나 로그 파일을 정기적으로 삭제합니다. 이러한 예방 조치를 취하면 데이터베이스 사용에 방해가 되거나 성능에 영향을 미치는 큰 로그 파일이 생성될 가능성을 줄일 수 있습니다. RDS for MySQL은 교체 및 삭제를 다음과 같이 처리합니다.

- MySQL 느린 쿼리 로그, 오류 로그 및 일반 로그 파일 크기는 DB 인스턴스에 대해 할당된 스토리지 공간의 2% 이하로 제한됩니다. 이 임계값을 유지하기 위해 매시간 로그가 자동으로 교체됩니다. MySQL은 2주 이상 지난 로그 파일을 제거합니다. 오래된 로그 파일을 제거한 후 로그 파일의 총 크기가 임계값을 초과하는 경우에는 로그 파일의 전체 크기가 임계값 이하로 작아질 때까지 가장 오래된 로그 파일부터 차례대로 삭제됩니다.
- `FILE` 로깅을 사용하는 경우 로그 파일은 매시간 검사되며 2주 이상 지난 로그 파일은 삭제됩니다. 경우에 따라 삭제 후 나머지 로그 파일의 총 크기가 DB 인스턴스에 할당된 공간 중 2%의 임계값을 초과할 수 있습니다. 이 경우 로그 파일의 전체 크기가 임계값 이하로 작아질 때까지 가장 오래된 로그 파일부터 차례대로 삭제됩니다.
- `TABLE` 로깅이 활성화된 경우 경우에 따라 로그 테이블이 24시간마다 순환됩니다. 테이블 로그에서 사용되는 공간이 할당된 스토리지 공간 중 20% 이상을 차지하면 이 교체가 발생합니다. 모든 로그를 합쳤을 때 크기가 10GB를 초과하는 경우에도 교체됩니다. DB 인스턴스에 대해 사용된 공간의 양이 DB 인스턴스의 할당된 스토리지 공간 중 90% 이상을 차지할 경우, 로그 순환을 위한 임계값은 줄어듭니다. 테이블 로그에서 사용되는 공간이 할당된 스토리지 공간 중 10% 이상을 차지하면 로그 테이블이 교체됩니다. 모든 로그를 합쳤을 때 크기가 10GB를 초과하는 경우에도 교체됩니다. 공간 확보를 위해 로그 테이블이 교체될 때 알림을 받기 위해 `low storage` 이벤트 범주를 구독할 수 있습니다. 자세한 내용은 Amazon RDS 이벤트 알림 작업 섹션을 참조하세요.

  로그 테이블이 교체되면 현재 로그 테이블이 먼저 백업 로그 테이블에 복사됩니다. 그런 다음 현재 로그 테이블의 항목이 제거됩니다. 백업 로그 테이블이 이미 존재할 경우, 현재 로그 테이블이 백업으로 복사되기 전에 백업 로그 테이블이 삭제됩니다. 필요하다면 백업 로그 테이블을 쿼리할 수 있습니다. `mysql.general_log` 테이블에 대한 백업 로그 테이블 이름은 `mysql.general_log_backup`으로 지정됩니다. `mysql.slow_log` 테이블에 대한 백업 로그 테이블 이름은 `mysql.slow_log_backup`으로 지정됩니다.

  `mysql.general_log` 절차를 호출하면 `mysql.rds_rotate_general_log` 테이블을 순환할 수 있습니다. `mysql.slow_log` 절차를 호출하면 `mysql.rds_rotate_slow_log` 테이블을 순환할 수 있습니다.

  데이터베이스 버전 업그레이드가 진행되는 동안 테이블 로그가 순환됩니다.

Amazon RDS 콘솔, Amazon RDS API, Amazon RDS CLI 또는 AWS SDK에서 로그를 사용하여 작업하려면 `log_output` 파라미터를 FILE로 설정합니다. MySQL 오류 로그와 같이, 이런 로그 파일은 매시간 순환됩니다. 이전 2주 동안 생성된 로그 파일은 유지됩니다. 보존 기간은 Amazon RDS와 Aurora 간에 다릅니다.

다시 실행 로그의 크기 제한

RDS for MySQL 버전 8.0.32 이하에서는 이 파라미터의 기본값이 256MB입니다. 이 양은 `innodb_log_file_size` 파라미터의 기본값(128MB)에 `innodb_log_files_in_group` 파라미터의 기본값(2)을 곱하여 산출됩니다. 자세한 내용은 Amazon RDS for MySQL의 파라미터 구성 모범 사례, 1부: 성능과 관련된 파라미터를 참조하세요.

RDS for MySQL 버전 8.0.33 이상 마이너 버전에서 Amazon RDS는 `innodb_log_file_size` 파라미터 대신 `innodb_redo_log_capacity` 파라미터를 사용합니다. `innodb_redo_log_capacity` 파라미터의 Amazon RDS 기본값은 2GB입니다. 자세한 내용은 MySQL 설명서의 MySQL 8.0.30 변경 사항에 관한 문서를 참조하세요.

MySQL 8.4부터 Amazon RDS는 기본적으로 `innodb_dedicated_server` 파라미터를 활성화합니다. `innodb_dedicated_server` 파라미터를 사용하면 데이터베이스 엔진이 `innodb_buffer_pool_size` 및 `innodb_redo_log_capacity` 파라미터를 계산합니다. 자세한 내용은 MySQL 8.4에서 버퍼 풀 크기 및 다시 실행 로그 용량 구성 섹션을 참조하세요.

Amazon CloudWatch Logs에 MySQL 로그 게시

Amazon CloudWatch Logs의 로그 그룹에 로그 데이터를 게시하도록 MySQL DB 인스턴스를 구성할 수 있습니다. CloudWatch Logs를 통해 로그 데이터에 대한 실시간 분석을 수행할 수 있고, CloudWatch를 사용하여 경보를 만들고 지표를 볼 수 있습니다. CloudWatch Logs를 사용하여 내구성이 뛰어난 스토리지에 로그 레코드를 저장할 수 있습니다.

Amazon RDS는 각 MySQL 데이터베이스 로그를 로그 그룹에 개별적인 데이터베이스 스트림으로 게시합니다. 예를 들어 느린 쿼리 로그를 포함하도록 내보내기 함수를 구성하면 느린 쿼리 데이터가 `/aws/rds/instance/`*`my_instance`*`/slowquery` 로그 그룹의 느린 쿼리 로그 스트림에 저장됩니다.

오류 로그는 기본적으로 활성화됩니다. 다음 표에는 기타 MySQL 로그의 요구 사항이 요약되어 있습니다.

로그	요구 사항
감사 로그	DB 인스턴스는 `MARIADB_AUDIT_PLUGIN` 옵션과 함께 사용자 지정 옵션 그룹을 사용해야 합니다.
일반 로그	DB 인스턴스는 파라미터 설정 `general_log = 1`과 함께 사용자 지정 파라미터 그룹을 사용하여 일반 로그를 활성화해야 합니다.
느린 쿼리 로그	DB 인스턴스는 파라미터 설정 `slow_query_log = 1` 과 함께 사용자 지정 파라미터 그룹을 사용하여 느린 쿼리 로그를 활성화해야 합니다.
IAM 데이터베이스 인증 오류 로그	DB 인스턴스를 만들거나 수정하여 DB 인스턴스의 `iam-db-auth-error` 로그 유형을 사용 설정해야 합니다.
로그 출력	DB 인스턴스는 파라미터 설정 `log_output = FILE` 과 함께 사용자 지정 파라미터 그룹을 사용하여 로그를 파일 시스템에 쓰고 CloudWatch Logs에 게시해야 합니다.

콘솔

콘솔을 사용하여 MySQL 로그를 CloudWatch Logs에 게시하려면

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 다음 변경하려는 DB 인스턴스를 선택합니다.
3. 수정을 선택합니다.
4. 로그 내보내기 섹션에서 CloudWatch Logs에 게시하기 시작할 로그를 선택합니다.
5. [Continue]를 선택한 후, 요약 페이지에서 [Modify DB Instance]를 선택합니다.

AWS CLI

AWS CLI를 사용하여 MySQL 로그를 게시할 수 있습니다. 다음 파라미터로 modify-db-instance 명령을 호출할 수 있습니다.

- --db-instance-identifier
- --cloudwatch-logs-export-configuration

> **Note**
>
> --cloudwatch-logs-export-configuration 옵션에 대한 변경 사항은 항상 DB 인스턴스에 즉시 적용됩니다. 따라서 --apply-immediately 및 --no-apply-immediately 옵션은 지정해도 아무런 효과가 없습니다.

또 다음 AWS CLI 명령을 호출해 MySQL 로그를 게시할 수 있습니다.

- create-db-instance
- restore-db-instance-from-db-snapshot
- restore-db-instance-from-s3
- restore-db-instance-to-point-in-time

다음 옵션으로 AWS CLI 명령 중 하나를 실행합니다.

- --db-instance-identifier
- --enable-cloudwatch-logs-exports
- --db-instance-class
- --engine

실행하는 AWS CLI 명령에 따라 다른 옵션이 필요할 수 있습니다.

Example

다음 예에서는 CloudWatch Logs에 로그 파일을 게시하도록 기존 MySQL DB 인스턴스를 수정합니다. --cloudwatch-logs-export-configuration 값은 JSON 객체입니다. 이 객체에 대한 키는

EnableLogTypes이며, 해당 값은 audit, error, general 및 slowquery의 조합을 사용하는 문자열의 배열입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["audit","error","general","slowquery"]}'
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["audit","error","general","slowquery"]}'
```

Example

다음 예에서는 CloudWatch Logs에 로그 파일을 게시하도록 MySQL DB 인스턴스를 생성합니다. --enable-cloudwatch-logs-exports 값은 문자열의 JSON 배열입니다. 문자열은 audit, error, general 및 slowquery의 조합일 수 있습니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --enable-cloudwatch-logs-exports '["audit","error","general","slowquery"]' \
    --db-instance-class db.m4.large \
    --engine MySQL
```

Windows의 경우:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --enable-cloudwatch-logs-exports '["audit","error","general","slowquery"]' ^
    --db-instance-class db.m4.large ^
    --engine MySQL
```

RDS API

RDS API를 사용하여 MySQL 로그를 게시할 수 있습니다. 다음 파라미터로 `ModifyDBInstance` 작업을 호출할 수 있습니다.

- `DBInstanceIdentifier`
- `CloudwatchLogsExportConfiguration`

> **Note**
>
> `CloudwatchLogsExportConfiguration` 파라미터에 대한 변경 사항은 항상 DB 인스턴스에 즉시 적용됩니다. 따라서 `ApplyImmediately` 파라미터는 지정해도 아무런 효과가 없습니다.

또한 다음 RDS API 작업을 호출해 MySQL 로그를 게시할 수 있습니다.

- `CreateDBInstance`
- `RestoreDBInstanceFromDBSnapshot`
- `RestoreDBInstanceFromS3`
- `RestoreDBInstanceToPointInTime`

다음 파라미터로 RDS API 작업 중 하나를 실행합니다.

- `DBInstanceIdentifier`
- `EnableCloudwatchLogsExports`
- `Engine`
- `DBInstanceClass`

실행하는 AWS CLI 명령에 따라 다른 파라미터가 필요할 수 있습니다.

MySQL 로그 출력을 표로 보내기

DB 파라미터 그룹을 만들고 log_output 서버 파라미터를 TABLE로 설정하여 일반 및 느린 쿼리 로그를 DB 인스턴스 상의 테이블로 전송할 수 있습니다. 그러면 일반 쿼리는 mysql.general_log 테이블에, 느린 쿼리는 mysql.slow_log 테이블에 로그가 기록됩니다. 이들 테이블을 쿼리하여 로그 정보에 액세스할 수 있습니다. 이 로깅을 활성화하면 데이터베이스에 기록되는 데이터의 양이 증가하여 성능이 저하될 수 있습니다.

일반 로그와 느린 쿼리 로그는 모두 기본적으로 비활성화됩니다. 테이블에 대한 로깅을 활성화하려면 general_log 및 slow_query_log 서버 파라미터도 1로 설정해야 합니다.

관련 파라미터를 0으로 설정하여 각 로깅 활동을 해제할 때까지 로그 테이블은 계속 커집니다. 흔히 대량의 데이터가 시간 경과에 따라 누적되어 할당된 스토리지 공간 중 상당한 비율을 사용할 수 있습니다. Amazon RDS에서는 로그 테이블을 자를 수 없지만 테이블의 콘텐츠를 이동할 수 있습니다. 테이블을 순환하면 그 내용이 백업 테이블에 저장된 다음 빈 로그 테이블이 새로 생성됩니다. 다음 명령줄 프로시저로 로그 테이블을 수동으로 순환시킬 수 있으며, 이때 명령 프롬프트는 PROMPT>로 표시됩니다.

```
PROMPT> CALL mysql.rds_rotate_slow_log;
PROMPT> CALL mysql.rds_rotate_general_log;
```

오래된 데이터를 완전히 제거하고 디스크 공간을 회수하려면 알맞은 프로시저를 두 번 연속으로 호출합니다.

단일 AZ 데이터베이스의 RDS for MySQL 이진 로깅 구성

이진 로그는 MySQL 서버 인스턴스의 데이터 수정에 대한 정보를 포함하는 로그 파일 세트입니다. 이진 로그에는 다음과 같은 정보가 포함되어 있습니다.

- 테이블 생성 또는 행 수정과 같은 데이터베이스 변경 사항을 설명하는 이벤트
- 데이터를 업데이트한 각 문의 기간에 대한 정보
- 데이터를 업데이트할 수 있었지만 업데이트하지 않은 문에 대한 이벤트

이진 로그는 복제 중 전송된 문을 기록합니다. 일부 복구 작업에도 필요합니다. 자세한 내용은 MySQL 설명서의 바이너리 로그 단원을 참조하십시오.

자동 백업 기능은 MySQL에 대해 이진 로깅을 설정할지 아니면 해제할지를 결정합니다. 다음과 같은 옵션이 있습니다:

이진 로깅 설정

백업 보존 기간을 0이 아닌 양수 값으로 설정합니다.

이진 로깅 해제

백업 보존 기간을 0으로 설정합니다.

자세한 내용은 자동 백업 활성화 단원을 참조하십시오.

Amazon RDS의 MySQL은 행 기반, 문 기반, 혼합 바이너리 로깅 형식을 지원합니다. 특정한 binlog 형식이 필요한 경우가 아니면 혼합하는 것이 좋습니다. 다양한 MySQL 이진 로그 형식에 대한 자세한 내용은 MySQL 설명서의 Binary Logging Formats을 참조하세요.

복제를 사용하려는 경우 바이너리 로깅 형식은 원본에 기록되고 복제 대상으로 전송되는 데이터 변경 내용의 레코드를 확인하므로 중요합니다. 복제와 관련된 다양한 바이너리 로깅 형식의 장/단점에 대한 자세한 내용은 MySQL 설명서의 Advantages and Disadvantages of Statement-Based and Row-Based Replication을 참조하십시오.

> **Important**
>
> MySQL 8.0.34에서 MySQL이 `binlog_format` 파라미터를 더 이상 사용하지 않습니다. 이후 MySQL 버전에서 MySQL은 파라미터를 제거하고 행 기반 복제만 지원할 계획입니다. 따라서 새 MySQL 복제 설정에 행 기반 로깅을 사용하는 것이 좋습니다. 자세한 내용은 MySQL 설명서의 binlog_format을 참조하세요.
>
> MySQL 버전 8.0 및 8.4는 `binlog_format` 파라미터를 수락합니다. 이 파라미터를 사용하면 MySQL에서 사용 중단 경고가 표시됩니다. 향후 메이저 릴리스에서 MySQL로부터 `binlog_format` 파라미터가 제거될 예정입니다.
>
> 명령문 기반 복제를 수행하면 원본 DB 인스턴스 와 읽기 전용 복제본 간에 불일치가 발생할 수 있습니다. 자세한 내용은 MySQL 설명서의 Determination of Safe and Unsafe Statements in Binary Logging을 참조하십시오.
>
> 이진 로깅을 활성화하면 DB 인스턴스에 대한 평균 디스크 쓰기 I/O 작업 수가 증가합니다. `WriteIOPS` CloudWatch 지표를 사용하여 IOPS 사용량을 모니터링할 수 있습니다.

MySQL 이진 로깅 형식을 설정하려면

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.

3. 수정할 DB 인스턴스와 연결된 DB 파라미터 그룹을 선택합니다.

   기본 파라미터 그룹을 수정할 수 없습니다. DB 인스턴스 가 기본 파라미터 그룹을 사용하고 있는 경우 새 파라미터 그룹을 생성하여 DB 인스턴스 와 연결합니다.

   파라미터 그룹에 대한 자세한 정보는 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.
4. 작업에서 편집을 선택합니다.
5. `binlog_format` 파라미터를 선택한 이진 로깅 형식(`ROW`, `STATEMENT`, 또는 `MIXED`)으로 설정합니다.

   DB 인스턴스의 백업 보존 기간을 0으로 설정하여 이진 로깅을 끌 수 있지만, 이렇게 하면 일일 자동 백업이 비활성화됩니다. 자동 백업을 비활성화하면 `log_bin` 세션 변수가 꺼지거나 비활성화됩니다. 그러면 RDS for MySQL DB 인스턴스에 대한 이진 로깅이 비활성화되며, 이는 다시 `binlog_format` 세션 변수를 데이터베이스의 기본값인 `ROW` 값으로 재설정합니다. 백업을 비활성화하지 않는 것이 좋습니다. 백업 보존 기간 설정에 대한 자세한 정보는 DB 인스턴스에 대한 설정 섹션을 참조하세요.
6. 변경 내용 저장을 선택하여 업데이트를 DB 파라미터 그룹에 저장합니다.

`binlog_format` 파라미터는 RDS for MySQL에서 동적이므로 DB 인스턴스를 재부팅하지 않고 변경 사항을 적용할 수 있습니다. (Aurora MySQL에서 이 파라미터는 정적입니. 자세한 내용은 Aurora MySQL 바이너리 로깅 구성을 참조하세요.)

> **Important**
>
> DB 파라미터 그룹을 변경하면 해당 파라미터 그룹을 사용하는 모든 DB 인스턴스에 영향을 미칩니다. AWS 리전의 다양한 MySQL DB 인스턴스에 대해 서로 다른 바이너리 로깅 형식을 지정하려면 DB 인스턴스에서 서로 다른 DB 파라미터 그룹을 사용해야 합니다. 이러한 파라미터 그룹은 다양한 로깅 형식을 식별합니다. 각 DB 인스턴스에 적절한 DB 파라미터 그룹을 할당합니다.

다중 AZ DB 클러스터에 대한 MySQL 이진 로깅 구성

Amazon RDS for MySQL 다중 AZ DB 클러스터의 이진 로깅은 복제, 시점 복구 및 감사를 지원하기 위해 모든 데이터베이스 변경 사항을 기록합니다. 다중 AZ DB 클러스터에서 이진 로그는 세컨더리 노드를 프라이머리 노드와 동기화하여 가용 영역 간 데이터 일관성을 보장하고 원활한 장애 조치를 활성화합니다.

이진 로깅을 최적화하기 위해 Amazon RDS는 이진 로그 트랜잭션 압축을 지원하므로 이진 로그에 대한 스토리지 요구 사항이 줄어들고 복제 효율성이 향상됩니다.

주제

- 다중 AZ DB 클러스터에 대한 이진 로그 트랜잭션 압축
- 다중 AZ DB 클러스터에 대한 이진 로그 트랜잭션 압축 구성

다중 AZ DB 클러스터에 대한 이진 로그 트랜잭션 압축

이진 로그 트랜잭션 압축은 zstd 알고리즘을 사용하여 이진 로그에 저장된 트랜잭션 데이터의 크기를 줄입니다. 활성화되면 MySQL 데이터베이스 엔진은 트랜잭션 페이로드를 단일 이벤트로 압축하여 I/O 및 스토리지 오버헤드를 최소화합니다. 이 기능은 데이터베이스 성능을 개선하고, 이진 로그 크기를 줄이며, 다중 AZ DB 클러스터에서 로그를 관리하고 복제하기 위한 리소스 사용을 최적화합니다.

Amazon RDS는 다음 파라미터를 통해 RDS for MySQL 다중 AZ DB 클러스터에 대한 이진 로그 트랜잭션 압축을 제공합니다.

- `binlog_transaction_compression` - 활성화되면(1) 데이터베이스 엔진은 트랜잭션 페이로드를 압축하여 단일 이벤트로 이진 로그에 기록합니다. 이렇게 하면 스토리지 사용량과 I/O 오버헤드가 줄어듭니다. 파라미터는 기본적으로 비활성화되어 있습니다.
- `binlog_transaction_compression_level_zstd` - 이진 로그 트랜잭션에 대한 zstd 압축 수준을 구성합니다. 값이 높을수록 압축률이 증가하여 스토리지 요구 사항은 더 줄어들지만 압축을 위한 CPU 및 메모리 사용량은 증가합니다. 기본값은 3이며 범위는 1~22입니다.

이러한 파라미터를 사용하면 워크로드 특성 및 리소스 가용성에 따라 이진 로그 압축을 미세 조정할 수 있습니다. 자세한 내용은 MySQL 설명서에서 Binary Log Transaction Compression을 참조하세요.

이진 로그 트랜잭션 압축의 주요 이점은 다음과 같습니다.

- 압축은 특히 트랜잭션이 크거나 쓰기 볼륨이 많은 워크로드의 경우 이진 로그의 크기를 줄입니다.
- 이진 로그가 작을수록 네트워크 및 I/O 오버헤드가 줄어들어 복제 성능이 향상됩니다.
- `binlog_transaction_compression_level_zstd` 파라미터는 압축률과 리소스 소비 간의 균형을 제어합니다.

다중 AZ DB 클러스터에 대한 이진 로그 트랜잭션 압축 구성

RDS for MySQL 다중 AZ DB 클러스터에 대한 이진 로그 트랜잭션 압축을 구성하려면 워크로드 요구 사항에 맞게 관련 클러스터 파라미터 설정을 수정합니다.

콘솔

이진 로그 트랜잭션 압축을 활성화하는 방법

1. DB 클러스터 파라미터 그룹을 수정하여 `binlog_transaction_compression` 파라미터를 1로 설정합니다.
2. (선택 사항) 워크로드 요구 사항 및 리소스 가용성에 따라 `binlog_transaction_compression_level_zstd` 파라미터 값을 조정합니다.

자세한 내용은 the section called “DB 클러스터 파라미터 그룹의 파라미터 수정” 단원을 참조하십시오.

AWS CLI

AWS CLI를 사용하여 이진 로그 트랜잭션 압축을 구성하려면 modify-db-cluster-parameter-group 명령을 사용합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name your-cluster-parameter-group \
  --parameters
 "ParameterName=binlog_transaction_compression,ParameterValue=1,ApplyMethod=pending-
reboot"
```

Windows의 경우:

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name your-cluster-parameter-group ^
  --parameters
 "ParameterName=binlog_transaction_compression,ParameterValue=1,ApplyMethod=pending-
reboot"
```

RDS API

Amazon RDS API를 사용하여 이진 로그 트랜잭션 압축을 구성하려면 `ModifyDBClusterParameterGroup` 작업을 사용합니다.

MySQL 이진 로그 액세스

mysqlbinlog 유틸리티를 사용하여 RDS for MySQL DB 인스턴스에서 이진 로그를 다운로드하거나 스트리밍할 수 있습니다. 이진 로그를 로컬 컴퓨터로 다운로드하면 mysql 유틸리티를 사용하여 로그를 재생하는 것과 같은 작업을 수행할 수 있습니다. mysqlbinlog 유틸리티 사용에 대한 자세한 내용은 MySQL 설명서의 mysqlbinlog를 사용하여 이진 로그 파일 백업을 참조하세요.

Amazon RDS 인스턴스에 대해 mysqlbinlog 유틸리티를 실행하려면 다음 옵션을 사용합니다.

- `--read-from-remote-server` - 필수입니다.
- `--host` – 인스턴스의 엔드포인트에서 DNS 이름.
- `--port` – 인스턴스에서 사용되는 포트.
- `--user` - `REPLICATION SLAVE` 권한이 부여된 MySQL 사용자.
- `--password` – MySQL 사용자의 암호. 또는 유틸리티에서 암호 입력을 요구하는 메시지가 표시되도록 암호 값을 생략.
- `--raw` - 파일을 이진 형식으로 다운로드합니다.
- `--result-file` - 원시 출력을 수신할 로컬 파일.
- `--stop-never` - 이진 로그 파일을 스트리밍합니다.
- `--verbose` - `ROW` binlog 형식을 사용할 경우 이 옵션을 포함하면 행 이벤트를 유사 SQL 문으로 볼 수 있습니다. `--verbose` 옵션에 대한 자세한 내용은 MySQL 설명서의 mysqlbinlog 행 이벤트 표시를 참조하세요.
- 하나 이상의 이진 로그 파일의 이름을 지정합니다. 사용 가능한 로그의 목록을 획득하려면 SQL 명령 `SHOW BINARY LOGS`를 사용합니다.

mysqlbinlog 옵션에 대한 자세한 내용은 MySQL 설명서의 mysqlbinlog - 이진 로그 파일 처리용 유틸리티를 참조하세요.

다음 예시에서는 mysqlbinlog 유틸리티를 사용하는 방법을 보여줍니다.

대상 LinuxmacOS, 또는Unix:

```
mysqlbinlog \
```

```
    --read-from-remote-server \
    --host=MySQLInstance1.cg034hpkmmjt.region.rds.amazonaws.com \
    --port=3306  \
    --user ReplUser \
    --password \
    --raw \
    --verbose \
    --result-file=/tmp/ \
    binlog.00098
```

Windows의 경우:

```
mysqlbinlog ^
    --read-from-remote-server ^
    --host=MySQLInstance1.cg034hpkmmjt.region.rds.amazonaws.com ^
    --port=3306  ^
    --user ReplUser ^
    --password ^
    --raw ^
    --verbose ^
    --result-file=/tmp/ ^
    binlog.00098
```

mysqlbinlog 유틸리티가 바이너리 로그에 액세스할 수 있도록 DB 인스턴스에서 바이너리 로그를 계속 사용할 수 있어야 합니다. 가용성을 보장하기 위해 mysql.rds_set_configuration 저장 프로시저를 사용하고 로그를 다운로드할 수 있는 충분한 시간을 지정합니다. 이 구성이 설정되지 않은 경우 Amazon RDS가 이진 로그를 가능한 한 빨리 제거하므로 mysqlbinlog 유틸리티에서 검색하는 바이너리 로그에 간극이 발생합니다.

다음 예제에서는 보존 기간을 1일로 설정합니다.

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

현재 설정을 표시하려면 mysql.rds_show_configuration 저장 프로시저를 사용합니다.

```
call mysql.rds_show_configuration;
```

Amazon RDS for Oracle 데이터베이스 로그 파일

Amazon RDS 콘솔 또는 API를 사용하여 Oracle 알림 로그, 감사 파일 및 추적 파일에 액세스할 수 있습니다. 파일 기반 데이터베이스 로그 보기, 다운로드 및 조사 방법에 대한 자세한 내용은 Amazon RDS 로그 파일 모니터링 단원을 참조하십시오.

제공되는 Oracle 감사 파일은 표준 Oracle 감사 파일입니다. Amazon RDS는 Oracle FGA(세밀한 감사) 기능을 지원합니다. 하지만 로그 액세스는 `SYS.FGA_LOG$` 테이블에 저장되고 `DBA_FGA_AUDIT_TRAIL` 보기를 통해 액세스 가능한 FGA 이벤트에 대한 액세스를 제공하지 않습니다.

DB 인스턴스에 사용 가능한 Oracle 로그 파일을 나열하는 `DescribeDBLogFiles` API 작업에서는 `MaxRecords` 파라미터를 무시하고 최대 1,000개의 레코드를 반환합니다. 호출은 `LastWritten`을(를) 밀리초 단위의 POSIX 날짜로 반환합니다.

주제

- 보존 일정
- Oracle 추적 파일을 사용한 작업
- Amazon CloudWatch Logs에 Oracle 로그 게시
- 알림 로그 및 Listener 로그에 액세스하기

보존 일정

로그 파일이 매우 커질 경우 Oracle 데이터베이스 엔진이 커진 파일들을 순환시킬 수 있습니다. 감사 또는 추적 파일을 보존하려면 해당 파일을 다운로드해야 합니다. 파일을 로컬에 저장하면 Amazon RDS 스토리지 비용이 절감되고 데이터에 사용할 수 있는 공간이 늘어납니다.

다음 표에서는 Amazon RDS의 Oracle 알림 로그, 감사 파일 및 추적 파일에 대한 보존 일정을 보여줍니다.

로그 유형	보존 일정
알림 로그	텍스트 알림 로그는 매일 교체되며 Amazon RDS에서 30일간 보존합니다. XML 알림 로그는 7일 이상 보관됩니다. `ALERTLOG` 보기를 사용하여 이 로그에 액세스할 수 있습니다.
감사 파일	감사 파일의 기본 보존 기간은 7일입니다. Amazon RDS는 7일이 경과한 감사 파일을 삭제할 수 있습니다.

로그 유형	보존 일정
추적 파일	추적 파일의 기본 보존 기간은 7일입니다. Amazon RDS는 7일이 경과한 추적 파일을 삭제할 수 있습니다.
리스너 로그	리스너 로그의 기본 보존 기간은 7일입니다. Amazon RDS는 7일이 경과한 리스너 로그를 삭제할 수 있습니다.

> **Note**
>
> 감사 파일과 추적 파일의 보존 구성은 동일합니다.

Oracle 추적 파일을 사용한 작업

추적 파일을 만들고 새로 고치고 액세스하고 삭제하기 위한 Amazon RDS 절차의 설명을 확인할 수 있습니다.

주제

- 파일 나열
- 추적 파일 생성 및 세션 추적
- 추적 파일 검색
- 추적 파일 제거

파일 나열

`background_dump_dest` 경로 내에 있는 임의의 파일에 대한 액세스를 허용하기 위해 두 절차 중 하나를 사용할 수 있습니다. 첫 번째 절차는 현재 `background_dump_dest`에 있는 모든 파일의 목록을 포함한 보기를 새로 고칩니다.

```
EXEC rdsadmin.manage_tracefiles.refresh_tracefile_listing;
```

보기가 새로 고쳐지면 다음 보기를 쿼리하여 결과에 액세스합니다.

```
SELECT * FROM rdsadmin.tracefile_listing;
```

이전 프로세스를 대체하는 프로세스는 FROM table을 사용하여 테이블과 같은 형식으로 되어 있는 관계형 데이터가 아닌 데이터를 스트리밍하여 데이터베이스 디렉터리 내용을 나열하는 것입니다.

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.listdir('BDUMP'));
```

다음 쿼리는 로그 파일의 텍스트를 표시합니다.

```
SELECT text FROM
 TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP','alert_dbname.log.date'));
```

읽기 전용 복제본에서 V$DATABASE.DB_UNIQUE_NAME을 쿼리하여 BDUMP 디렉터리의 이름을 가져옵니다. 고유 이름이 DATABASE_B인 경우 BDUMP 디렉터리는 BDUMP_B입니다. 다음 예제에서는 복제본에서 BDUMP 이름을 쿼리한 다음 이 이름을 사용하여 alert_DATABASE.log.2020-06-23의 콘텐츠를 쿼리합니다.

```
SELECT 'BDUMP' || (SELECT regexp_replace(DB_UNIQUE_NAME,'.*(_[A-Z])', '\1') FROM V
$DATABASE) AS BDUMP_VARIABLE FROM DUAL;

BDUMP_VARIABLE
--------------
BDUMP_B

SELECT TEXT FROM
 table(rdsadmin.rds_file_util.read_text_file('BDUMP_B','alert_DATABASE.log.2020-06-23'));
```

추적 파일 생성 및 세션 추적

ALTER SESSION에 대한 제한이 없으므로, Oracle에서 추적 파일을 생성하는 다양한 표준 메서드를 Amazon RDS DB 인스턴스에도 그대로 사용할 수 있습니다. 더 높은 액세스 권한이 필요한 추적 파일에 대해서는 다음 프로시저가 제공됩니다.

Oracle 메서드	Amazon RDS 메서드
oradebug hanganalyze 3	EXEC rdsadmin.manage_tracefiles.hanganalyze;
oradebug dump systemstate 266	EXEC rdsadmin.manage_tracefiles.dump_systemstate;

여러 표준 메서드를 사용하여 Amazon RDS의 Oracle DB 인스턴스에 연결된 개별 세션을 추적할 수 있습니다. 세션 추적 기능을 활성화하기 위해 DBMS_SESSION 및 DBMS_MONITOR 등 Oracle에서 제공한 PL/SQL 패키지의 하위 프로그램을 실행할 수 있습니다. 자세한 내용은 Oracle 문서의 세션 추적 기능 활성화 단원을 참조하십시오.

추적 파일 검색

Amazon RDS에서 관리되는 외부 테이블의 표준 SQL 쿼리를 사용하여 background_dump_dest에 있는 추적 파일을 검색할 수 있습니다. 이 메서드를 사용하려면 특정 추적 파일에 대한 이 테이블의 위치를 설정하는 프로시저를 실행해야 합니다.

예를 들어 이전에 언급한 rdsadmin.tracefile_listing 보기를 사용하여 시스템 상의 모든 추적 파일을 나열할 수 있습니다. 그러면 다음 프로시저를 사용하여 의도한 추적 파일을 가리키도록 tracefile_table 보기를 설정할 수 있습니다.

```
EXEC
 rdsadmin.manage_tracefiles.set_tracefile_table_location('CUST01_ora_3260_SYSTEMSTATE.trc');
```

다음 예제에서는 제공된 파일로 위치가 설정된 현재 스키마에서 외부 테이블을 만듭니다. SQL 쿼리를 사용하여 콘텐츠를 로컬 파일로 가져올 수 있습니다.

```
SPOOL /tmp/tracefile.txt
SELECT * FROM tracefile_table;
SPOOL OFF;
```

추적 파일 제거

추적 파일이 누적되면서 디스크 공간을 낭비할 수 있습니다. Amazon RDS는 기본적으로 7일 이상 지난 추적 파일 및 덤프 파일을 제거합니다. show_configuration 절차를 사용하여 추적 파일 보존 기간을 보고 설정할 수 있습니다. 구성 결과를 볼 수 있도록 SET SERVEROUTPUT ON 명령을 실행해야 합니다.

다음 예제에서는 현재 추적 파일 보존 기간을 표시한 다음, 새 추적 파일 보존 기간을 설정합니다.

```
# Show the current tracefile retention
SQL> EXEC rdsadmin.rdsadmin_util.show_configuration;
NAME:tracefile retention
VALUE:10080
DESCRIPTION:tracefile expiration specifies the duration in minutes before tracefiles in
 bdump are automatically deleted.
```

```
# Set the tracefile retention to 24 hours:
SQL> EXEC rdsadmin.rdsadmin_util.set_configuration('tracefile retention',1440);
SQL> commit;

#show the new tracefile retention
SQL> EXEC rdsadmin.rdsadmin_util.show_configuration;
NAME:tracefile retention
VALUE:1440
DESCRIPTION:tracefile expiration specifies the duration in minutes before tracefiles in
 bdump are automatically deleted.
```

주기적인 제거 프로세스 외에, background_dump_dest에서 파일을 수동으로 제거할 수 있습니다. 다음 예제에서는 5분 이상 지난 모든 파일을 제거하는 방법을 보여줍니다.

```
EXEC rdsadmin.manage_tracefiles.purge_tracefiles(5);
```

(.trc와 같은 파일 확장명을 포함하지 않고) 특정 패턴과 일치하는 모든 파일을 제거할 수도 있습니다. 다음 예제에서는 "SCHPOC1_ora_5935"로 시작하는 모든 파일을 제거하는 방법을 보여줍니다.

```
EXEC rdsadmin.manage_tracefiles.purge_tracefiles('SCHPOC1_ora_5935');
```

Amazon CloudWatch Logs에 Oracle 로그 게시

Amazon CloudWatch Logs의 로그 그룹에 로그 데이터를 게시하도록 RDS for Oracle DB 인스턴스를 구성할 수 있습니다. CloudWatch Logs를 통해 로그 데이터에 대한 분석을 수행할 수 있고, CloudWatch를 사용하여 경보를 만들고 측정치를 볼 수 있습니다. CloudWatch Logs를 사용하여 내구성이 뛰어난 스토리지에 로그 레코드를 저장할 수 있습니다.

Amazon RDS는 각 Oracle 데이터베이스 로그를 로그 그룹에 개별적인 데이터베이스 스트림으로 게시합니다. 예를 들어 감사 로그를 포함하도록 내보내기 함수를 구성하면 감사 데이터가 /aws/rds/instance/my_instance/audit 로그 그룹의 감사 로그 스트림에 저장됩니다. 다음 표에는 Amazon CloudWatch Logs에 로그를 게시하기 위한 RDS for Oracle의 요구 사항이 요약되어 있습니다.

로그 이름	요구 사항	기본값
알림 로그	없음. 이 로그를 비활성화할 수 없습니다.	활성화됨

로그 이름	요구 사항	기본값
추적 로그	`trace_enabled` 파라미터를 TRUE로 설정하거나 기본값으로 그대로 두세요.	TRUE
감사 로그	`audit_trail` 파라미터를 다음 허용값 중 하나로 설정합니다. `{ none | os | db [, extended] | xml [, extended] }`	none
리스너 로그	없음. 이 로그를 비활성화할 수 없습니다.	활성화됨
Oracle Management Agent 로그	없음. 이 로그를 비활성화할 수 없습니다.	활성화됨

이 Oracle Management Agent 로그는 다음 테이블에 나와 있는 로그 그룹으로 구성됩니다.

로그 이름	CloudWatch 로그 그룹
emctl.log	oemagent-emctl
emdctlj.log	oemagent-emdctlj
gcagent.log	oemagent-gcagent
gcagent_errors.log	oemagent-gcagent-errors
emagent.nohup	oemagent-emagent-nohup
secure.log	oemagent-secure

자세한 내용은 Oracle 설명서에서 Management Agent 로그 및 추적 파일 찾기를 참조하세요.

콘솔

AWS Management Console에서 Oracle DB 로그를 CloudWatch Logs에 게시하려면

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.

2. 탐색 창에서 데이터베이스를 선택한 다음 변경하려는 DB 인스턴스를 선택합니다.
3. 수정을 선택합니다.
4. 로그 내보내기 섹션에서 CloudWatch Logs에 게시하기 시작할 로그를 선택합니다.
5. [Continue]를 선택한 후, 요약 페이지에서 [Modify DB Instance]를 선택합니다.

AWS CLI

Oracle 로그를 게시하기 위해 다음 파라미터와 함께 `modify-db-instance` 명령을 사용할 수 있습니다.

- `--db-instance-identifier`
- `--cloudwatch-logs-export-configuration`

> **Note**
>
> `--cloudwatch-logs-export-configuration` 옵션에 대한 변경 사항은 항상 DB 인스턴스에 즉시 적용됩니다. 따라서 `--apply-immediately` 및 `--no-apply-immediately` 옵션은 지정해도 아무런 효과가 없습니다.

또한 다음 명령을 사용하여 Oracle 로그를 게시할 수 있습니다.

- `create-db-instance`
- `restore-db-instance-from-db-snapshot`
- `restore-db-instance-from-s3`
- `restore-db-instance-to-point-in-time`

Example

다음 예에서는 CloudWatch Logs 게시가 활성화된 Oracle DB 인스턴스를 생성합니다. `--cloudwatch-logs-export-configuration` 값은 문자열의 JSON 배열입니다. 문자열은 `alert`, `audit`, `listener` 및 `trace`의 조합일 수 있습니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance \
```

```
    --db-instance-identifier mydbinstance \
    --cloudwatch-logs-export-configuration
 '["trace","audit","alert","listener","oemagent"]' \
    --db-instance-class db.m5.large \
    --allocated-storage 20 \
    --engine oracle-ee \
    --engine-version 19.0.0.0.ru-2024-04.rur-2024-04.r1 \
    --license-model bring-your-own-license \
    --master-username myadmin \
    --manage-master-user-password
```

Windows의 경우:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration trace alert audit listener oemagent ^
    --db-instance-class db.m5.large ^
    --allocated-storage 20 ^
    --engine oracle-ee ^
    --engine-version 19.0.0.0.ru-2024-04.rur-2024-04.r1 ^
    --license-model bring-your-own-license ^
    --master-username myadmin ^
    --manage-master-user-password
```

Example

다음 예에서는 CloudWatch Logs에 로그 파일을 게시하도록 기존 Oracle DB 인스턴스를 수정합니다. `--cloudwatch-logs-export-configuration` 값은 JSON 객체입니다. 이 객체에 대한 키는 `EnableLogTypes`이며, 해당 값은 `alert`, `audit`, `listener` 및 `trace`의 조합을 사용하는 문자열의 배열입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["trace","alert","audit","listener","oemagent"]}'
```

Windows의 경우:

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

```
    --db-instance-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration EnableLogTypes=\"trace\",\"alert\",\"audit
\",\"listener\",\"oemagent\"
```

Example

다음 예에서는 CloudWatch Logs에 감사 및 리스너 로그 파일 게시를 비활성화하도록 기존 Oracle DB 인스턴스를 수정합니다. `--cloudwatch-logs-export-configuration` 값은 JSON 객체입니다. 이 객체에 대한 키는 `DisableLogTypes`이며, 해당 값은 `alert`, `audit`, `listener` 및 `trace`의 조합을 사용하는 문자열의 배열입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"DisableLogTypes":["audit","listener"]}'
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration DisableLogTypes=\"audit\",\"listener\"
```

RDS API

RDS API를 사용하여 Oracle DB 로그를 게시할 수 있습니다. 다음 파라미터로 `ModifyDBInstance` 작업을 호출할 수 있습니다.

- `DBInstanceIdentifier`
- `CloudwatchLogsExportConfiguration`

> **Note**
>
> `CloudwatchLogsExportConfiguration` 파라미터에 대한 변경 사항은 항상 DB 인스턴스에 즉시 적용됩니다. 따라서 `ApplyImmediately` 파라미터는 지정해도 아무런 효과가 없습니다.

또한 다음 RDS API 작업을 호출하여 Oracle 로그를 게시할 수도 있습니다.

- CreateDBInstance
- RestoreDBInstanceFromDBSnapshot
- RestoreDBInstanceFromS3
- RestoreDBInstanceToPointInTime

다음 파라미터로 RDS API 작업 중 하나를 실행합니다.

- DBInstanceIdentifier
- EnableCloudwatchLogsExports
- Engine
- DBInstanceClass

실행하는 RDS 명령에 따라 다른 파라미터가 필요할 수 있습니다.

알림 로그 및 Listener 로그에 액세스하기

Amazon RDS 콘솔을 사용하여 알림 로그를 볼 수 있습니다. 다음 SQL 문도 사용할 수 있습니다.

```
SELECT message_text FROM alertlog;
```

Amazon CloudWatch Logs를 사용하여 Listener 로그에 액세스하세요.

> **Note**
>
> Oracle은 알림 및 리스너 로그가 Amazon RDS 보기에서 이들 로그를 사용할 수 없게 되는 시점인 10MB를 초과할 때 이들 로그를 순환시킵니다.

RDS for PostgreSQL 데이터베이스 로그 파일

다음 유형의 로그 파일을 모니터링할 수 있습니다.

- PostgreSQL 로그
- 업그레이드 로그
- IAM 데이터베이스 인증 오류 로그

> **Note**
>
> IAM 데이터베이스 인증 오류 로그를 활성화하려면 먼저 RDS for PostgreSQL DB 인스턴스에 IAM 데이터베이스 인증을 활성화해야 합니다. IAM 데이터베이스 인증 활성화에 대한 자세한 내용은 IAM 데이터베이스 인증의 활성화 및 비활성화 섹션을 참조하세요.

RDS for PostgreSQL은 기본 PostgreSQL 로그 파일에 데이터베이스 활동을 로깅합니다. 온프레미스 PostgreSQL DB 인스턴스의 경우 이러한 메시지가 `log/postgresql.log`에 로컬로 저장됩니다. RDS for PostgreSQL DB 인스턴스의 경우 Amazon RDS 인스턴스에서 로그 파일을 사용할 수 있습니다. 또한 이러한 로그는 AWS Management Console을 통해 액세스할 수 있으며, 여기에서 해당 로그를 보거나 다운로드할 수 있습니다. 기본 로깅 수준에서는 로그인 실패, 치명적인 서버 오류, 교착 상태 및 쿼리 실패를 캡처합니다.

파일 기반 데이터베이스 로그 보기, 다운로드 및 조사 방법에 대한 자세한 내용은 Amazon RDS 로그 파일 모니터링 섹션을 참조하세요. PostgreSQL 로그에 대한 자세한 내용은 Working with Amazon RDS and Aurora PostgreSQL logs: Part 1(RDS 및 Aurora PostgreSQL 로그로 작업하기: 파트 1) 및 Working with Amazon RDS and Aurora PostgreSQL logs: Part 2(RDS 및 Aurora PostgreSQL 로그로 작업하기: 파트 2)를 참조하세요.

이 주제에서 설명하는 표준 PostgreSQL 로그 외에 RDS for PostgreSQL은 PostgreSQL Auidt 확장(`pgAudit`)도 지원합니다. 대부분의 규제 대상 산업 및 정부 기관은 법적 요구 사항을 준수하기 위해 데이터 변경에 대한 감사 로그 또는 감사 추적을 보존해야 합니다. pgAudit 설치 및 사용에 대한 자세한 내용은 pgAudit를 사용하여 데이터베이스 활동 로깅 섹션을 참조하세요.

주제

- RDS for PostgreSQL에서의 로깅을 위한 파라미터
- RDS for PostgreSQL DB 인스턴스에 쿼리 로깅을 활성화합니다.
- Amazon CloudWatch Logs에 PostgreSQL 로그 게시

RDS for PostgreSQL에서의 로깅을 위한 파라미터

다양한 파라미터를 수정하여 RDS for PostgreSQL DB 인스턴스의 로깅 동작을 사용자 지정할 수 있습니다. 다음 표에서는 로그 저장 기간, 로그 교체 시기, 로그를 CSV(쉼표로 구분된 값) 형식으로 출력할지 여부 등 다양한 파라미터를 확인할 수 있습니다. 다른 설정에서 STDERR로 전송된 텍스트 출력도 찾을 수 있습니다. 수정 가능한 파라미터의 설정을 변경하려면 에 대한 사용자 지정 DB 파라미터 그룹을 사용하세요. RDS for PostgreSQL 인스턴스 자세한 내용은 Amazon RDS DB 인스턴스용 DB 파라미터 그룹. 섹션을 참조하세요.

파라미터	Default	설명
log_destination	stderr	로그에 대한 출력 형식을 설정합니다. 기본값은 `stderr`이지만 설정에 `csvlog`를 추가하여 CSV(쉼표로 구분된 값)를 지정할 수도 있습니다. 자세한 내용은 로그 대상 설정(`stderr`, `csvlog`) 섹션을 참조하세요.
log_filename	postgresql.log.%Y-%m-%d-%H	로그 파일 이름의 패턴을 지정합니다. 이 파라미터는 기본값 외에도 `postgresql.log.%Y-%m-%d` 및 `postgresql.log.%Y-%m-%d-%H%M` 파일 이름 패턴을 지원합니다.
log_line_prefix	%t:%r:%u@%d:[%p]:	시간(%t), 원격 호스트(%r), 사용자(%u), 데이터베이스(%d) 및 프로세스 ID(%p)를 기록하기 위해 `stderr`에 기록되는 각 로그 줄의 접두사를 정의합니다.
log_rotation_age	60	몇 분 후에 로그 파일이 자동으로 교체됩니다. 이 값을 1분에서 1,440분 사이의 값으로 변경할 수 있습니다. 자세한 내용은 로그 파일 회전 설정 섹션을 참조하세요.
log_rotation_size	–	로그 교체가 자동으로 이루어질 때의 크기(kB)입니다. 기본적으로 로그는 `log_rotation_age` 파라미터에 따라 교체되므로 이 파라미터는 사용되지 않습니다. 자세한 내용은 로그 파일 회전 설정를 참조하세요.

파라미터	Default	설명
rds.log_retention_period	4320	지정된 시간(분)보다 오래된 PostgreSQL 로그가 삭제됩니다. 기본값인 4,320분으로 지정하면 3일이 지난 로그 파일이 삭제됩니다. 자세한 내용은 로그 보존 기간 설정 섹션을 참조하세요.

애플리케이션 문제를 식별하기 위해 로그에서 쿼리 실패, 로그인 실패, 교착 상태 및 치명적인 서버 오류를 찾아볼 수 있습니다. 예를 들어, 레거시 애플리케이션을 Oracle에서 Amazon RDS PostgreSQL로 변환했지만 일부 쿼리가 올바르게 변환되지 않았다고 가정합시다. 이러한 잘못된 형식의 쿼리는 오류 메시지를 생성하고, 로그에서 오류 메시지를 찾아 문제를 식별할 수 있습니다. 쿼리 로깅에 대한 자세한 내용은 RDS for PostgreSQL DB 인스턴스에 쿼리 로깅을 활성화합니다. 섹션을 참조하세요.

다음 주제에서 PostgreSQL 로깅의 기본 세부 정보를 제어하는 다양한 파라미터에 관한 정보를 찾을 수 있습니다.

주제

- 로그 보존 기간 설정
- 로그 파일 회전 설정
- 로그 대상 설정(stderr, csvlog)
- log_line_prefix 파라미터 이해하기

로그 보존 기간 설정

`rds.log_retention_period` 파라미터는 RDS for PostgreSQL DB 인스턴스가 로그 파일을 보관하는 기간을 지정합니다. 기본 설정은 3일(4,320분)이지만 1일(1,440분)에서 7일(10,080분) 사이로 이 값을 설정할 수 있습니다. RDS for PostgreSQL DB 인스턴스에 일정 기간 동안 로그 파일을 보관할 수 있는 충분한 스토리지가 있는지 확인하세요.

로그를 정기적으로 Amazon CloudWatch Logs에 게시하여 로그가 에서 제거된 후에도 시스템 데이터를 보고 분석할 수 있도록 하는 것이 좋습니다. RDS for PostgreSQL DB 인스턴스 자세한 내용은 Amazon CloudWatch Logs에 PostgreSQL 로그 게시 섹션을 참조하세요.

로그 파일 회전 설정

Amazon RDS는 기본적으로 매시간 새 로그 파일을 생성합니다. 타이밍은 `log_rotation_age` 파라미터로 제어됩니다. 이 파라미터의 기본값은 60(분)이지만 1분에서 24시간(1,440분) 사이로 설정할 수

있습니다. 교체 시간이 되면 새 로그 파일이 생성됩니다. 파일의 이름은 log_filename 파라미터로 지정된 패턴에 따라 결정됩니다.

로그 파일은 log_rotation_size 파라미터에 지정된 대로 크기에 따라 교체될 수도 있습니다. 이 파라미터는 로그가 지정된 크기(KB)에 도달하면 교체되도록 지정합니다. RDS for PostgreSQL DB 인스턴스의 경우 log_rotation_size가 설정되지 않습니다. 즉, 지정된 값이 없습니다. 그러나 이 파라미터는 0~2,097,151KB(킬로바이트) 사이로 설정할 수 있습니다.

로그 파일 이름은 log_filename 파라미터에 지정된 파일 이름 패턴을 기반으로 합니다. 이 파라미터에 사용할 수 있는 설정은 다음과 같습니다.

- postgresql.log.%Y-%m-%d - 로그 파일 이름의 기본 형식입니다. 로그 파일 이름에 년, 월, 일이 포함됩니다.
- postgresql.log.%Y-%m-%d-%H - 로그 파일 이름 형식에 시간이 포함됩니다.

자세한 내용은 PostgreSQL 설명서의 log_rotation_age 및 log_rotation_size 섹션을 참조하세요.

로그 대상 설정(**stderr**, **csvlog**)

기본적으로 Amazon RDS PostgreSQL은 표준 오류(stderr) 형식으로 로그를 생성합니다. 이 형식이 log_destination 파라미터에 대한 기본 설정입니다. 각 메시지에는 log_line_prefix 파라미터에 지정된 패턴을 사용하여 접두사가 붙습니다. 자세한 내용은 log_line_prefix 파라미터 이해하기 섹션을 참조하세요.

RDS for PostgreSQL도 로그를 csvlog 형식으로 생성할 수 있습니다. csvlog는 로그 데이터를 쉼표로 구분된 값(CSV) 데이터로 분석하는 데 유용합니다. 예를 들어 외부 테이블로 로그를 사용하기 위해 log_fdw 확장을 사용한다고 가정하겠습니다. stderr 로그 파일에 만들어진 외부 테이블에는 로그 이벤트 데이터가 있는 하나의 열이 포함되어 있습니다. log_destination 파라미터에 csvlog를 추가하면 외부 테이블의 여러 열에 대한 구분이 있는 CSV 형식의 로그 파일을 얻게 됩니다. 이제 로그를 더 쉽게 정렬하고 분석할 수 있습니다. log_fdw를 csvlog와 함께 사용하는 방법을 알아보려면 log_fdw 확장으로 SQL을 사용하여 DB 로그에 액세스 섹션을 참조하세요.

이 파라미터에 csvlog를 지정하는 경우 stderr 및 csvlog 파일이 모두 생성된다는 점에 유의하세요. rds.log_retention_period 및 로그 스토리지와 회전율에 영향을 주는 다른 설정을 고려하여 로그에서 사용하는 스토리지를 모니터링해야 합니다. stderr 및 csvlog를 사용하는 경우 로그에서 소비하는 스토리지가 두 배 이상 늘어납니다.

log_destination에 csvlog를 추가하고 stderr로만 되돌리려면 파라미터를 재설정해야 합니다. 이렇게 하려면 Amazon RDS 콘솔을 연 다음 인스턴스에 대한 사용자 지정 DB 파라미터 그룹을 엽니다. log_destination 파라미터를 선택하고 파라미터 편집을 선택한 다음 재설정을 선택합니다.

로깅 구성에 대한 자세한 내용은 Amazon RDS 및 Aurora PostgreSQL 로그 작업: 1부를 참조하세요.

log_line_prefix 파라미터 이해하기

stderr 로그 형식은 각 로그 메시지에 log_line_prefix 파라미터에 지정된 세부 정보를 접두사로 지정합니다. 기본값은 다음과 같습니다.

```
%t:%r:%u@%d:[%p]:t
```

Aurora PostgreSQL 버전 16부터는 다음을 선택할 수도 있습니다.

```
%m:%r:%u@%d:[%p]:%l:%e:%s:%v:%x:%c:%q%a
```

stderr로 전송되는 각 로그 항목에는 선택한 값에 따라 다음 정보가 포함됩니다.

- %t - 밀리초 단위를 제외한 로그 항목 시간
- %m - 밀리초 단위를 포함한 로그 항목 시간
- %r - 원격 호스트 주소
- %u@%d - 사용자 이름 @ 데이터베이스 이름
- [%p] - 프로세스 ID(사용 가능한 경우)
- %l - 세션당 로그 행 번호
- %e – SQL 오류 코드
- %s – 프로세스 시작 타임스탬프
- %v - 가상 트랜잭션 ID
- %x – 트랜잭션 ID
- %c – 세션 ID
- %q - 세션이 아닌 종결자
- %a – 애플리케이션 이름

RDS for PostgreSQL DB 인스턴스에 쿼리 로깅을 활성화합니다.

다음 테이블에 나열된 파라미터 중 일부를 설정하여 쿼리, 잠금을 기다리는 쿼리, 체크포인트 및 기타 여러 세부 정보를 포함하여 데이터베이스 활동에 대한 보다 자세한 정보를 수집할 수 있습니다. 이 주제에서는 쿼리 로깅에 중점을 둡니다.

파라미터	Default	설명
log_connections	–	성공한 연결을 모두 기록합니다.
log_disconnections	–	각 세션의 끝과 기간을 로깅합니다.
log_checkpoints	1	각 체크포인트를 기록합니다.
log_lock_waits	–	오랜 잠금 대기 시간을 기록합니다. 이 파라미터는 기본적으로 설정되어 있지 않습니다.
log_min_duration_sample	–	(밀리초) 문 샘플이 로그되는 최소 실행 시간을 설정합니다. 샘플 크기는 `log_statement_sample_rate` 파라미터를 사용하여 설정합니다.
log_min_duration_statement	–	지정된 시간 이상 실행되는 모든 SQL 문은 로깅됩니다. 이 파라미터는 기본적으로 설정되어 있지 않습니다. 이 파라미터를 활성화하면 최적화되지 않은 쿼리를 찾는 데 도움이 될 수 있습니다.
log_statement	–	기록할 문 유형을 설정합니다. 기본적으로 이 파라미터는 설정되어 있지 않지만 `all`, `ddl` 또는 `mod`로 변경하여 로깅하려는 SQL 문 유형을 지정할 수 있습니다. 이 파라미터에 none 이외의 다른 것을 지정하면 추가 단계를 수행하여 로그 파일에 암호가 노출되지 않도록 해야 합니다. 자세한 내용은 쿼리 로깅 사용 시 암호 노출 위험 완화 섹션을 참조하세요.
log_statement_sample_rate	–	로깅되도록 `log_min_duration_sample` 에 지정된 시간을 초과하는 문의 비율

파라미터	Default	설명
		로, 0.0에서 1.0 사이의 부동 소수점 값으로 표시됩니다.
log_statement_stats	–	누적 성능 통계를 서버 로그에 기록합니다.

로깅을 사용하여 성능이 느린 쿼리 찾기

SQL 문 및 쿼리를 로깅하여 성능이 느린 쿼리를 찾을 수 있습니다. 이 섹션에 설명된 대로 `log_statement` 및 `log_min_duration` 파라미터 설정을 수정하여 이 기능을 활성화합니다. RDS for PostgreSQL DB 인스턴스S에 대한 쿼리 로깅을 활성화하기 전에 로그에서 암호가 노출될 수 있는 사항과 위험을 완화하는 방법을 알고 있어야 합니다. 자세한 내용은 쿼리 로깅 사용 시 암호 노출 위험 완화 섹션을 참조하세요.

다음에서 `log_statement` 및 `log_min_duration` 파라미터에 대한 참조 정보를 확인할 수 있습니다.

log_statement

이 파라미터는 로그에 전송해야 하는 SQL 문의 유형을 지정합니다. 기본값은 none입니다. 이 파라미터를 `all`, `ddl` 또는 `mod`로 변경할 경우 로그에 암호가 노출될 위험을 줄이려면 권장 조치를 취해야 합니다. 자세한 내용은 쿼리 로깅 사용 시 암호 노출 위험 완화 섹션을 참조하세요.

모두

모든 문을 로깅합니다. 이 설정은 디버깅 용도로 사용하는 것이 좋습니다.

ddl

모든 데이터 정의 언어(DDL) 문(CREATE, ALTER, DROP 등)을 로깅합니다.

mod

NSERT, UPDATE, DELETE 등 데이터를 수정하는 모든 DDL 문과 데이터 조작 언어(DML) 문을 로깅합니다.

없음

SQL 문이 로깅되지 않습니다. 로그에서 암호가 노출될 위험을 피하려면 이 설정을 사용하는 것이 좋습니다.

log_min_duration_statement

지정된 시간 이상 실행되는 모든 SQL 문은 로깅됩니다. 이 파라미터는 기본적으로 설정되어 있지 않습니다. 이 파라미터를 활성화하면 최적화되지 않은 쿼리를 찾는 데 도움이 될 수 있습니다.

–1–2147483647

문이 로깅되는 런타임의 밀리초(ms) 수입니다.

쿼리 로깅 설정

이 단계에서는 RDS for PostgreSQL DB 인스턴스가 사용자 지정 DB 파라미터 그룹을 사용한다고 가정합니다.

1. `log_statement` 파라미터를 `all`로 설정합니다. 다음 예에서는 이 파라미터가 설정에서 `postgresql.log`에 기록되는 정보를 보여줍니다.

```
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:LOG: statement:
 SELECT feedback, s.sentiment,s.confidence
FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
ORDER BY s.confidence DESC;
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:LOG: QUERY
 STATISTICS
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:DETAIL: ! system
 usage stats:
! 0.017355 s user, 0.000000 s system, 0.168593 s elapsed
! [0.025146 s user, 0.000000 s system total]
! 36644 kB max resident size
! 0/8 [0/8] filesystem blocks in/out
! 0/733 [0/1364] page faults/reclaims, 0 [0] swaps
! 0 [0] signals rcvd, 0/0 [0/0] messages rcvd/sent
! 19/0 [27/0] voluntary/involuntary context switches
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:STATEMENT: SELECT
 feedback, s.sentiment,s.confidence
FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
ORDER BY s.confidence DESC;
2022-10-05 22:05:56 UTC:52.95.4.1(11335):postgres@labdb:[3639]:ERROR: syntax error
 at or near "ORDER" at character 1
2022-10-05 22:05:56 UTC:52.95.4.1(11335):postgres@labdb:[3639]:STATEMENT: ORDER BY
 s.confidence DESC;
---------------------- END OF LOG ----------------------
```

2. `log_min_duration_statement` 파라미터를 설정합니다. 다음 예제에서는 이 파라미터가 `postgresql.log`로 설정되어 있을 때 1 파일에 기록되는 정보를 보여줍니다.

   `log_min_duration_statement` 파라미터에 지정된 기간을 초과하는 쿼리가 로깅됩니다. 다음은 그 한 예입니다. Amazon RDS 콘솔에서 RDS for PostgreSQL DB 인스턴스에 대한 로그 파일을 볼 수 있습니다.

```
2022-10-05 19:05:19 UTC:52.95.4.1(6461):postgres@labdb:[6144]:LOG: statement: DROP
 table comments;
2022-10-05 19:05:19 UTC:52.95.4.1(6461):postgres@labdb:[6144]:LOG: duration:
 167.754 ms
2022-10-05 19:08:07 UTC::@:[355]:LOG: checkpoint starting: time
2022-10-05 19:08:08 UTC::@:[355]:LOG: checkpoint complete: wrote 11 buffers
 (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=1.013 s, sync=0.006 s,
 total=1.033 s; sync files=8, longest=0.004 s, average=0.001 s; distance=131028 kB,
 estimate=131028 kB
---------------------- END OF LOG ----------------------
```

쿼리 로깅 사용 시 암호 노출 위험 완화

암호가 노출되지 않도록 `log_statement`를 none으로 설정하는 것이 좋습니다. `log_statement`를 `all`, `ddl`, 또는 `mod`로 설정하는 경우 다음 단계 중 하나 이상을 수행하는 것이 좋습니다.

- 클라이언트의 경우 민감한 정보를 암호화합니다. 자세한 내용은 PostgreSQL 설명서의 암호화 옵션에 관한 문서를 참조하세요. `CREATE` 및 `ALTER` 문의 `ENCRYPTED`(및 `UNENCRYPTED`) 옵션을 사용합니다. 자세한 내용은 PostgreSQL 설명서에서 CREATE USER를 참조하세요.
- RDS for PostgreSQL DB 인스턴스의 경우 PostgreSQL Auditing(pgAudit) 확장을 설정하고 사용합니다. 이 확장은 로그로 전송된 CREATE 및 ALTER 문에서 민감한 정보를 삭제합니다. 자세한 내용은 pgAudit를 사용하여 데이터베이스 활동 로깅 섹션을 참조하세요.
- CloudWatch 로그에 대한 액세스를 제한합니다.
- IAM 등의 보다 강력한 인증 메커니즘을 사용합니다.

Amazon CloudWatch Logs에 PostgreSQL 로그 게시

매우 내구력 있는 스토리지에 PostgreSQL 로그 레코드를 저장하려면 Amazon CloudWatch Logs를 사용하면 됩니다. CloudWatch Logs를 사용하면 로그 데이터에 대한 실시간 분석을 수행하고 CloudWatch를 통해 지표를 보고 경보를 생성할 수도 있습니다. 예를 들어, `log_statement`를 `ddl`로 설정하면 DDL 문이 실행될 때마다 경고하도록 경보를 설정할 수 있습니다. RDS for PostgreSQL DB

인스턴스를 생성하는 동안 PostgreSQL 로그가 CloudWatch Logs에 업로드되도록 선택할 수 있습니다. 이때 로그를 업로드하지 않도록 선택한 경우 나중에 인스턴스를 수정하여 해당 시점부터 로그 업로드를 시작하도록 할 수 있습니다. 즉, 기존 로그는 업로드되지 않습니다. 수정된 RDS for PostgreSQL DB 인스턴스에 생성된 새 로그만 업로드됩니다.

현재 사용 가능한 모든 RDS for PostgreSQL 버전에서는 CloudWatch Logs에 로그 파일 게시를 지원합니다. 자세한 내용은 Amazon RDS for PostgreSQL 릴리스 정보에서 Amazon RDS for PostgreSQL 업데이트 내용을 참조하세요.

CloudWatch Logs로 작업하려면 로그 그룹에 로그 데이터를 게시하도록 RDS for PostgreSQL DB 인스턴스를 구성하십시오.

RDS for PostgreSQL용 CloudWatch Logs에 다음 로그 유형을 게시할 수 있습니다.

- PostgreSQL 로그
- 업그레이드 로그
- IAM 데이터베이스 인증 오류 로그

구성을 완료하고 나면 Amazon RDS에서 CloudWatch 로그 그룹 내 로그 스트림에 로그 이벤트를 게시합니다. 예를 들면 PostgreSQL 로그 데이터가 로그 그룹 `/aws/rds/instance/my_instance/postgresql` 안에 저장됩니다. 로그를 보려면 https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.

콘솔

콘솔을 사용하여 PostgreSQL 로그를 CloudWatch Logs에 게시하려면

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 수정할 DB 인스턴스를 선택한 다음 수정을 선택합니다.
4. 로그 내보내기 섹션에서 CloudWatch Logs에 게시하기 시작할 로그를 선택합니다.

   로그 내보내기 섹션은 CloudWatch Logs에 게시하는 기능을 지원하는 PostgreSQL 버전에서만 제공됩니다.
5. [Continue]를 선택한 후, 요약 페이지에서 [Modify DB Instance]를 선택합니다.

AWS CLI

AWS CLI를 통해 PostgreSQL 로그를 게시할 수 있습니다. 다음 파라미터로 modify-db-instance 명령을 호출할 수 있습니다.

- --db-instance-identifier
- --cloudwatch-logs-export-configuration

> **Note**
>
> --cloudwatch-logs-export-configuration 옵션에 대한 변경 사항은 항상 DB 인스턴스에 즉시 적용됩니다. 따라서 --apply-immediately 및 --no-apply-immediately 옵션은 지정해도 아무런 효과가 없습니다.

또한 다음 CLI 명령을 호출하여 PostgreSQL 로그를 게시할 수도 있습니다.

- create-db-instance
- restore-db-instance-from-db-snapshot
- restore-db-instance-to-point-in-time

다음 옵션으로 CLI 명령 중 하나를 실행합니다.

- --db-instance-identifier
- --enable-cloudwatch-logs-exports
- --db-instance-class
- --engine

실행하는 CLI 명령에 따라 다른 옵션이 필요할 수 있습니다.

Example 로그를 CloudWatch Logs에 게시하도록 인스턴스 수정

다음 예에서는 CloudWatch Logs에 로그 파일을 게시하도록 기존 PostgreSQL DB 인스턴스를 수정합니다. --cloudwatch-logs-export-configuration 값은 JSON 객체입니다. 이 객체에 대한 키는 EnableLogTypes이며, 해당 값은 postgresql 및 upgrade의 조합을 사용하는 문자열의 배열입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":["postgresql",
 "upgrade"]}'
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["postgresql","upgrade"]}'
```

Example 로그를 CloudWatch Logs에 게시하도록 인스턴스 생성

다음 예에서는 CloudWatch Logs에 로그 파일을 게시하도록 PostgreSQL DB 인스턴스를 생성합니다. `--enable-cloudwatch-logs-exports` 값은 문자열의 JSON 배열입니다. 문자열은 `postgresql` 및 `upgrade`의 조합일 수 있습니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --enable-cloudwatch-logs-exports '["postgresql","upgrade"]' \
    --db-instance-class db.m4.large \
    --engine postgres
```

Windows의 경우:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --enable-cloudwatch-logs-exports '["postgresql","upgrade"]' ^
    --db-instance-class db.m4.large ^
    --engine postgres
```

RDS API

RDS API를 사용하여 PostgreSQL 로그를 게시할 수 있습니다. 다음 파라미터로 `ModifyDBInstance` 작업을 호출할 수 있습니다.

- `DBInstanceIdentifier`
- `CloudwatchLogsExportConfiguration`

> **Note**
>
> `CloudwatchLogsExportConfiguration` 파라미터에 대한 변경 사항은 항상 DB 인스턴스에 즉시 적용됩니다. 따라서 `ApplyImmediately` 파라미터는 지정해도 아무런 효과가 없습니다.

또한 다음 RDS API 작업을 호출해 PostgreSQL 로그를 게시할 수 있습니다.

- `CreateDBInstance`
- `RestoreDBInstanceFromDBSnapshot`
- `RestoreDBInstanceToPointInTime`

다음 파라미터로 RDS API 작업 중 하나를 실행합니다.

- `DBInstanceIdentifier`
- `EnableCloudwatchLogsExports`
- `Engine`
- `DBInstanceClass`

실행하는 작업에 따라 다른 파라미터가 필요할 수 있습니다.

AWS CloudTrail에서 Amazon RDS API 호출 모니터링

AWS CloudTrail은 AWS 계정을 감사하는 데 도움이 되는 AWS 서비스입니다. AWS CloudTrail을 생성하면 AWS 계정에서 활성화됩니다. CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

주제

- CloudTrail를 Amazon RDS와 통합
- Amazon RDS 로그 파일 항목

CloudTrail를 Amazon RDS와 통합

모든 Amazon RDS 작업은 CloudTrail에 의해 로깅됩니다. CloudTrail은 Amazon RDS에서 사용자, 역할 또는 AWS 서비스가 수행한 작업의 기록을 제공합니다.

CloudTrail 이벤트

CloudTrail은 Amazon RDS에 대한 API 호출을 이벤트로 캡처합니다. 이벤트는 모든 소스로부터의 단일 요청을 나타내며 요청 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보가 들어 있습니다. 이벤트에는 Amazon RDS 콘솔 호출과 Amazon RDS API 작업에 대한 코드 호출이 포함됩니다.

Amazon RDS 작업은 이벤트 기록(Event history)의 CloudTrail 이벤트에 기록됩니다. CloudTrail 콘솔을 사용하여 AWS 리전에서 지난 90일간 기록된 API 활동 및 이벤트를 확인할 수 있습니다. 자세한 내용은 CloudTrail 이벤트 기록에서 이벤트 보기를 참조하세요.

CloudTrail 추적

Amazon RDS에 대한 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. 추적은 지정된 Amazon S3 버킷에 이벤트를 전송할 수 있도록 하는 구성입니다. CloudTrail은 일반적으로 계정 활동 15분 이내에 로그 파일을 전송합니다.

> **Note**
>
> 트레일을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다.

AWS 계정에 대해 두 가지 추적 유형(모든 리전에 적용되는 추적 또는 한 리전에 적용되는 추적)을 생성할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 지역에 추적이 적용됩니다.

또는 CloudTrail 로그에서 수집된 이벤트 데이터를 추가 분석 및 처리하도록 다른 AWS서비스를 구성할 수 있습니다. 자세한 내용은 다음을 참조하세요.

- 추적 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에 대한 Amazon SNS 알림 구성
- 여러 리전에서 CloudTrail 로그 파일 수신 및 여러 계정에서 CloudTrail 로그 파일 수신

Amazon RDS 로그 파일 항목

CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. CloudTrail 로그 파일은 퍼블릭 API 호출에 대한 순서 지정된 스택 추적이 아니기 때문에 특정 순서로 표시되지 않습니다.

다음은 `CreateDBInstance` 작업을 보여 주는 CloudTrail 로그 항목이 나타낸 예제입니다.

```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
    "eventTime": "2018-07-30T22:14:06Z",
    "eventSource": "rds.amazonaws.com",
    "eventName": "CreateDBInstance",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-cli/1.15.42 Python/3.6.1 Darwin/17.7.0 botocore/1.10.42",
    "requestParameters": {
        "enableCloudwatchLogsExports": [
            "audit",
            "error",
            "general",
            "slowquery"
        ],
        "dBInstanceIdentifier": "test-instance",
        "engine": "mysql",
```

```
        "masterUsername": "myawsuser",
        "allocatedStorage": 20,
        "dBInstanceClass": "db.m1.small",
        "masterUserPassword": "****"
    },
    "responseElements": {
        "dBInstanceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance",
        "storageEncrypted": false,
        "preferredBackupWindow": "10:27-10:57",
        "preferredMaintenanceWindow": "sat:05:47-sat:06:17",
        "backupRetentionPeriod": 1,
        "allocatedStorage": 20,
        "storageType": "standard",
        "engineVersion": "8.0.28",
        "dbInstancePort": 0,
        "optionGroupMemberships": [
            {
                "status": "in-sync",
                "optionGroupName": "default:mysql-8-0"
            }
        ],
        "dBParameterGroups": [
            {
                "dBParameterGroupName": "default.mysql8.0",
                "parameterApplyStatus": "in-sync"
            }
        ],
        "monitoringInterval": 0,
        "dBInstanceClass": "db.m1.small",
        "readReplicaDBInstanceIdentifiers": [],
        "dBSubnetGroup": {
            "dBSubnetGroupName": "default",
            "dBSubnetGroupDescription": "default",
            "subnets": [
                {
                    "subnetAvailabilityZone": {"name": "us-east-1b"},
                    "subnetIdentifier": "subnet-cbfff283",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1e"},
                    "subnetIdentifier": "subnet-d7c825e8",
                    "subnetStatus": "Active"
                },
```

```
                {
                    "subnetAvailabilityZone": {"name": "us-east-1f"},
                    "subnetIdentifier": "subnet-6746046b",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1c"},
                    "subnetIdentifier": "subnet-bac383e0",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1d"},
                    "subnetIdentifier": "subnet-42599426",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1a"},
                    "subnetIdentifier": "subnet-da327bf6",
                    "subnetStatus": "Active"
                }
            ],
            "vpcId": "vpc-136a4c6a",
            "subnetGroupStatus": "Complete"
        },
        "masterUsername": "myawsuser",
        "multiAZ": false,
        "autoMinorVersionUpgrade": true,
        "engine": "mysql",
        "cACertificateIdentifier": "rds-ca-2015",
        "dbiResourceId": "db-ETDZIIXHEWY5N7GXVC4SH7H5IA",
        "dBSecurityGroups": [],
        "pendingModifiedValues": {
            "masterUserPassword": "****",
            "pendingCloudwatchLogsExports": {
                "logTypesToEnable": [
                    "audit",
                    "error",
                    "general",
                    "slowquery"
                ]
            }
        },
        "dBInstanceStatus": "creating",
        "publiclyAccessible": true,
```

```
        "domainMemberships": [],
        "copyTagsToSnapshot": false,
        "dBInstanceIdentifier": "test-instance",
        "licenseModel": "general-public-license",
        "iAMDatabaseAuthenticationEnabled": false,
        "performanceInsightsEnabled": false,
        "vpcSecurityGroups": [
            {
                "status": "active",
                "vpcSecurityGroupId": "sg-f839b688"
            }
        ]
    },
    "requestID": "daf2e3f5-96a3-4df7-a026-863f96db793e",
    "eventID": "797163d3-5726-441d-80a7-6eeb7464acd4",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


앞의 예의 `userIdentity` 요소에 표시된 것처럼 모든 이벤트 또는 로그 항목에는 요청을 생성한 사용자에 대한 정보가 포함됩니다. 자격 증명 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 IAM 사용자 보안 인증 정보로 했는지 여부.
- 역할 또는 페더레이션 사용자의 임시 보안 인증을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지.

`userIdentity`에 대한 자세한 내용은 CloudTrail userIdentity 요소를 참조하십시오. `CreateDBInstance` 및 기타 Amazon RDS 작업에 대한 자세한 내용은 Amazon RDS API 참조를 참조하세요.

데이터베이스 활동 스트림을 사용하여 Amazon RDS 모니터링

데이터베이스 활동 스트림을 사용하면 데이터베이스 활동 스트림을 거의 실시간으로 모니터링할 수 있습니다.

주제

- 데이터베이스 활동 스트림 개요
- Oracle 데이터베이스 통합 감사 구성
- Amazon RDS for Microsoft SQL Server에 대한 감사 정책 구성
- 데이터베이스 활동 스트림 시작
- Amazon RDS에 대한 데이터베이스 활동 스트림 수정
- 데이터베이스 활동 스트림 상태 가져오기
- 데이터베이스 활동 스트림 중지
- 데이터베이스 활동 스트림 모니터링
- 데이터베이스 활동 스트림의 IAM 정책 예제

데이터베이스 활동 스트림 개요

Amazon RDS 데이터베이스 관리자는 데이터베이스를 보호하고 규정 준수 및 규제 요건을 충족해야 합니다. 한 가지 전략은 데이터베이스 활동 스트림을 모니터링 도구와 통합하는 것입니다. 이렇게 하면 데이터베이스에서 감사 활동에 대한 경보를 모니터링하고 설정할 수 있습니다.

보안 위협은 외부 및 내부 위협입니다. 내부 위협으로부터 보호하기 위해 데이터베이스 활동 스트림 기능을 구성하여 데이터 스트림에 대한 관리자 액세스를 제어할 수 있습니다. Amzon RDS DBA는 스트림의 수집, 전송, 저장 및 처리에 대한 액세스 권한이 없습니다.

목차

- 데이터베이스 활동 스트림 작동 방식
- Oracle 데이터베이스 및 SQL Server 데이터베이스에서의 감사
  - Oracle 데이터베이스의 통합 감사
  - Microsoft SQL Server에서의 감사
  - Oracle Database 및 SQL Server에 대한 기본이 아닌 감사 필드
  - DB 파라미터 그룹 재정의
- 데이터베이스 활동 스트림에 대한 비동기식 모드

- 데이터베이스 활동 스트림의 요구 사항 및 제한 사항
- 리전 및 버전 사용 가능 여부
- 데이터베이스 활동 스트림이 지원되는 DB 인스턴스 클래스

데이터베이스 활동 스트림 작동 방식

Amazon RDS는 활동을 Amazon Kinesis 데이터 스트림에 거의 실시간으로 푸시합니다. Kinesis 스트림이 자동으로 생성됩니다. Kinesis에서 Amazon Data Firehose 및 AWS Lambda와 같은 AWS 서비스를 구성하여 스트림을 사용하고 데이터를 저장할 수 있습니다.

> **Important**
>
> Amazon RDS에서 데이터베이스 활동 스트림은 무료 기능이지만, Amazon Kinesis는 데이터 스트림에 대한 요금을 부과합니다. 자세한 내용은 Amazon Kinesis Data Streams 요금을 참조하세요.

규정 준수 관리용 애플리케이션이 데이터베이스 활동 스트림을 사용하도록 구성할 수 있습니다. 이 애플리케이션은 스트림을 사용하여 데이터베이스에 대한 경보를 생성하고 활동 감사를 수행할 수 있습니다.

Amazon RDS는 다중 AZ 배포에서 데이터베이스 활동 스트림을 지원합니다. 이 경우 데이터베이스 활동 스트림은 기본 인스턴스와 대기 인스턴스를 모두 감사합니다.

Oracle 데이터베이스 및 SQL Server 데이터베이스에서의 감사

감사는 구성된 데이터베이스 작업의 모니터링 및 기록입니다. Amazon RDS는 기본적으로 데이터베이스 활동을 캡처하지 않습니다. 사용자가 데이터베이스에서 직접 감사 정책을 생성하고 관리합니다.

주제

- Oracle 데이터베이스의 통합 감사
- Microsoft SQL Server에서의 감사
- Oracle Database 및 SQL Server에 대한 기본이 아닌 감사 필드
- DB 파라미터 그룹 재정의

Oracle 데이터베이스의 통합 감사

Oracle 데이터베이스에서 통합 감사 정책은 사용자 동작의 측면을 감사하는 데 사용할 수 있는 명명된 감사 설정 그룹입니다. 정책은 단일 사용자의 활동을 감사하는 것만큼 간단할 수 있습니다. 또한 조건을 사용하는 복잡한 감사 정책을 생성할 수도 있습니다.

Oracle 데이터베이스는 SYS 감사 레코드를 포함한 감사 레코드를 통합 감사 추적에 기록합니다. 예를 들어, INSERT 문 중에 오류가 발생하는 경우 표준 감사는 오류 번호와 실행된 SQL을 보여줍니다. 감사 추적은 AUDSYS 스키마에 있는 읽기 전용 테이블에 상주합니다. UNIFIED_AUDIT_TRAIL 데이터 사전 보기를 쿼리하여 이러한 레코드에 액세스합니다.

일반적으로 데이터베이스 활동 스트림을 다음과 같이 구성합니다.

1. CREATE AUDIT POLICY 명령을 사용하여 Oracle 데이터베이스 감사 정책을 생성합니다.

   Oracle 데이터베이스는 감사 레코드를 생성합니다.
2. AUDIT POLICY 명령을 사용하여 감사 정책을 활성화합니다.
3. 데이터베이스 활동 스트림을 구성합니다.

   Oracle 데이터베이스 감사 정책과 일치하 는 활동만 캡처되어 Amazon Kinesis 데이터 스트림으로 전송됩니다. 데이터베이스 활동 스트림이 활성화된 경우 Oracle 데이터베이스 관리자는 감사 정책을 변경하거나 감사 로그를 제거할 수 없습니다.

통합 감사 정책에 대해 자세히 알아보려면 Oracle Database Security Guide에서 About Auditing Activities with Unified Audit Policies and AUDIT를 참조하세요.

Microsoft SQL Server에서의 감사

데이터베이스 활동 스트림은 SQLAudit 기능을 사용하여 SQL Server 데이터베이스를 감사합니다.

RDS for SQL Server 인스턴스에는 다음이 포함되어 있습니다.

- 서버 감사 - SQL Server 감사는 서버 또는 데이터베이스 수준 작업의 단일 인스턴스와 모니터링할 작업 그룹을 수집합니다. 서버 수준 감사인 RDS_DAS_AUDIT 및 RDS_DAS_AUDIT_CHANGES는 RDS에서 관리합니다.
- 서버 감사 사양 - 서버 감사 사양은 서버 수준 이벤트를 기록합니다. RDS_DAS_SERVER_AUDIT_SPEC 사양을 수정할 수 있습니다. 이 사양은 서버 감사 RDS_DAS_AUDIT과 연결되어 있습니다. RDS_DAS_CHANGES_AUDIT_SPEC 사양은 RDS에서 관리합니다.

- 데이터베이스 감사 사양 - 데이터베이스 감사 사양은 데이터베이스 수준 이벤트를 기록합니다. 데이터베이스 감사 사양 RDS_DAS_DB_<name>을 생성하여 RDS_DAS_AUDIT 서버 감사에 연결할 수 있습니다.

콘솔 또는 CLI를 사용하여 데이터베이스 활동 스트림을 구성할 수 있습니다. 일반적으로 데이터베이스 활동 스트림을 다음과 같이 구성합니다.

1. (선택 사항) CREATE DATABASE AUDIT SPECIFICATION 명령을 사용하여 데이터베이스 감사 사양을 생성하고 이를 RDS_DAS_AUDIT 서버 감사에 연결합니다.
2. (선택 사항) ALTER SERVER AUDIT SPECIFICATION 명령을 사용하여 서버 감사 사양을 수정하고 정책을 정의합니다.
3. 데이터베이스 및 서버 감사 정책을 활성화합니다. 예시:

   ```
   ALTER DATABASE AUDIT SPECIFICATION [<Your database specification>] WITH
   (STATE=ON)

   ALTER SERVER AUDIT SPECIFICATION [RDS_DAS_SERVER_AUDIT_SPEC] WITH
   (STATE=ON)
   ```
4. 데이터베이스 활동 스트림을 구성합니다.

   서버 및 데이터베이스 감사 정책과 일치하는 활동만 캡처되어 Amazon Kinesis 데이터 스트림으로 전송됩니다. 데이터베이스 활동 스트림이 활성화되고 정책이 잠겨 있는 경우 데이터베이스 관리자는 감사 정책을 변경하거나 감사 로그를 제거할 수 없습니다.

   > **Important**
   >
   > 특정 데이터베이스에 대한 데이터베이스 감사 사양이 활성화되어 있고 정책이 잠긴 상태이면 데이터베이스를 삭제할 수 없습니다.

SQL Server 감사에 자세한 내용은 Microsoft SQL Server 설명서의 SQL Server Audit 구성 요소를 참조하세요.

Oracle Database 및 SQL Server에 대한 기본이 아닌 감사 필드

데이터베이스 활동 스트림을 시작하면 모든 데이터베이스 이벤트는 해당하는 활동 스트림 이벤트를 생성합니다. 예를 들어 데이터베이스 사용자는 SELECT 및 INSERT 문을 실행합니다. 데이터베이스는 이러한 이벤트를 감사하여 Amazon Kinesis 데이터 스트림으로 전송합니다.

이 이벤트는 스트림에서 JSON 객체로 표현됩니다. JSON 객체는 databaseActivityEventList 배열이 있는 DatabaseActivityMonitoringRecord를 포함합니다. 배열의 미리 정의된 필드는 class, clientApplication 및 command를 포함합니다.

기본적으로 활동 스트림은 엔진 기본 감사 필드를 포함하지 않습니다. engineNativeAuditFields JSON 객체에 이러한 추가 필드가 포함되도록 Amazon RDS for Oracle 및 SQL Server를 구성할 수 있습니다.

Oracle Database에서 통합 감사 추적의 대부분 이벤트는 RDS 데이터 활동 스트림의 필드에 매핑됩니다. 예를 들어, 통합 감사의 UNIFIED_AUDIT_TRAIL.SQL_TEXT 필드는 데이터베이스 활동 스트림의 commandText 필드에 매핑됩니다. 그러나 OS_USERNAME과 같은 Oracle 데이터베이스 감사 필드는 데이터베이스 활동 스트림에 있는 미리 정의된 필드에 매핑되지 않습니다.

SQL Server에서 SQLAudit에 의해 기록되는 대부분의 이벤트 필드는 RDS 데이터베이스 활동 스트림의 필드에 매핑됩니다. 예를 들어, 통합 감사의 sys.fn_get_audit_file에 있는 code 필드는 데이터베이스 활동 스트림의 commandText 필드에 매핑됩니다. 그러나 permission_bitmask와 같은 SQL Server 데이터베이스 감사 필드는 데이터베이스 활동 스트림에 있는 미리 정의된 필드에 매핑되지 않습니다.

databaseActivityEventList에 대한 자세한 내용은 데이터베이스 활동 스트림에 대한 databaseActivityEventList JSON 배열 섹션을 참조하세요.

DB 파라미터 그룹 재정의

일반적으로 파라미터 그룹을 연결하여 RDS for Oracle에서 통합 감사를 활성화합니다. 그러나 데이터베이스 활동 스트림에는 추가 구성이 필요합니다. 고객 경험을 개선하기 위해 Amazon RDS는 다음을 수행합니다.

- 활동 스트림을 활성화하면 RDS for Oracle은 파라미터 그룹의 감사 파라미터를 무시합니다.
- 활동 스트림을 비활성화하면 RDS for Oracle은 감사 파라미터 무시를 중단합니다.

SQL Server에 대한 데이터베이스 활동 스트림은 SQL Audit 옵션에서 설정한 파라미터와는 독립적입니다.

데이터베이스 활동 스트림에 대한 비동기식 모드

Amazon RDS의 활동 스트림은 항상 비동기식입니다. 데이터베이스 세션이 활동 스트림 이벤트를 생성하면 세션은 즉시 정상 활동으로 되돌아갑니다. Amazon RDS는 백그라운드에서 내구성 있는 레코드에 활동 스트림 이벤트를 생성합니다.

백그라운드 태스크에서 오류가 발생하면 Amazon RDS가 이벤트를 생성합니다. 이 이벤트는 활동 스트림 이벤트 레코드가 분실되었을 수있는 기간의 시작과 끝을 나타냅니다. 비동기 모드는 활동 스트림의 정확성보다 데이터베이스 성능을 우선시합니다.

데이터베이스 활동 스트림의 요구 사항 및 제한 사항

RDS에서 데이터베이스 활동 스트림에는 다음과 같은 요구 사항과 제한 사항이 있습니다.

- 데이터베이스 활동 스트림에는 Amazon Kinesis를 사용해야 합니다.
- 데이터베이스 활동 스트림은 항상 암호화되므로 AWS Key Management Service(AWS KMS)를 사용해야 합니다.
- Amazon Kinesis 데이터 스트림에 추가 암호화를 적용하는 것은 이미 AWS KMS 키를 사용하여 암호화된 데이터베이스 활동 스트림과 호환되지 않습니다.
- 사용자가 직접 감사 정책을 생성하고 관리합니다. Amazon Aurora와 달리 RDS for Oracle은 기본적으로 데이터베이스 활동을 캡처하지 않습니다.
- 사용자가 직접 감사 정책 또는 사양을 생성하고 관리합니다. Amazon Aurora와 달리 Amazon RDS는 기본적으로 데이터베이스 활동을 캡처하지 않습니다.
- 다중 AZ 배포에서 기본 DB 인스턴스에서만 데이터베이스 활동 스트림을 시작합니다. 활동 스트림은 기본 DB 인스턴스와 대기 DB 인스턴스를 모두 자동으로 감사합니다. 장애 조치 중에는 추가 단계가 필요 없습니다.
- DB 인스턴스의 이름을 변경해도 새 Kinesis 스트림이 생성되지는 않습니다.
- CDB는 RDS for Oracle에 대해 지원되지 않습니다.
- 읽기 전용 복제본은 지원되지 않습니다.

리전 및 버전 사용 가능 여부

기능 가용성 및 해당 지원은 각 데이터베이스 엔진의 특정 버전 및 AWS 리전마다 다릅니다. 데이터베이스 활동 스트림의 버전 및 지역 가용성에 대한 자세한 내용은 Amazon RDS에서 데이터베이스 활동 스트림을 지원하는 리전 및 DB 엔진 단원을 참조하세요.

데이터베이스 활동 스트림이 지원되는 DB 인스턴스 클래스

RDS for Oracle의 경우 다음 DB 인스턴스 클래스와 함께 데이터베이스 활동 스트림을 사용할 수 있습니다.

- db.m4.*large

- db.m5.*large
- db.m5d.*large
- db.m6i.*large
- db.r4.*large
- db.r5.*large
- db.r5.*large.tpc*.mem*x
- db.r5b.*large
- db.r5b.*large.tpc*.mem*x
- db.r5d.*large
- db.r6i.*large
- db.r6i.*large.tpc*.mem*x
- db.x2idn.*large
- db.x2iedn.*large
- db.x2iezn.*large
- db.z1d.*large

RDS for sQL Server의 경우 다음 DB 인스턴스 클래스와 함께 데이터베이스 활동 스트림을 사용할 수 있습니다.

- db.m4.*large
- db.m5.*large
- db.m5d.*large
- db.m6i.*large
- db.r4.*large
- db.r5.*large
- db.r5b.*large
- db.r5d.*large
- db.r6i.*large
- db.x1e.*large
- db.x2iedn.*large
- db.z1d.*large

인스턴스 클래스 유형에 대한 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요.

Oracle 데이터베이스 통합 감사 구성

데이터베이스 활동 스트림에 사용할 통합 감사를 구성하는 경우 다음과 같은 상황이 발생할 수 있습니다.

- Oracle 데이터베이스에 통합 감사가 구성되지 않았습니다.

  이 경우 `CREATE AUDIT POLICY` 명령을 사용하여 새 정책을 생성하고 `AUDIT POLICY` 명령을 사용하여 정책을 활성화합니다. 다음 예시에서는 특정 권한 및 역할을 가진 사용자를 모니터링하는 정책을 만들고 활성화합니다.

  ```
  CREATE AUDIT POLICY table_pol
  PRIVILEGES CREATE ANY TABLE, DROP ANY TABLE
  ROLES emp_admin, sales_admin;

  AUDIT POLICY table_pol;
  ```

  전체 지침은 Oracle 데이터베이스 문서에서 Configuring Audit Policies를 참조하세요.
- 통합 감사는 Oracle 데이터베이스에 대해 구성됩니다.

  데이터베이스 활동 스트림을 활성화하면 RDS for Oracle에서 기존 감사 데이터를 자동으로 지웁니다. 또한 감사 추적 권한도 취소됩니다. RDS for Oracle은 더 이상 다음을 수행하지 않습니다.

  - 통합 감사 추적 레코드 삭제
  - 통합 감사 정책 추가, 삭제 또는 수정
  - 마지막으로 아카이빙된 타임스탬프 업데이트

  > **Important**
  >
  > 데이터베이스 활동 스트림을 활성화하기 전에 감사 데이터를 백업하는 것이 좋습니다.

  `UNIFIED_AUDIT_TRAIL` 보기에 대한 설명은 UNIFIED_AUDIT_TRAIL을 참조하세요. Oracle Support 계정이 있는 경우 How To Purge The UNIFIED AUDIT TRAIL을 참조하세요.

Amazon RDS for Microsoft SQL Server에 대한 감사 정책 구성

SQL Server 데이터베이스 인스턴스에는 Amazon RDS에서 관리하는 서버 감사(RDS_DAS_AUDIT)가 있습니다. 정책을 정의하여 서버 감사 사양(RDS_DAS_SERVER_AUDIT_SPEC)에 서버 이벤트를 기록할 수 있습니다. 데이터베이스 감사 사양(예: RDS_DAS_DB_<name>)을 생성하고 데이터베이스 이벤트를 기록하는 정책을 정의할 수 있습니다. 서버 및 데이터베이스 수준 감사 작업 그룹 목록은 Microsoft SQL Server 설명서의 SQL Server 감사 작업 그룹 및 작업을 참조하세요.

기본 서버 정책은 실패한 로그인과 데이터베이스 활동 스트림에 대한 데이터베이스 또는 서버 감사 사양의 변경만 모니터링합니다.

감사 및 감사 사양에 대한 제한 사항은 다음과 같습니다.

- 데이터베이스 활동 스트림이 잠긴 상태일 때는 서버 또는 데이터베이스 감사 사양을 수정할 수 없습니다.
- 서버 감사 RDS_DAS_AUDIT 사양을 수정할 수 없습니다.
- SQL Server 감사 RDS_DAS_CHANGES 또는 관련 서버 감사 사양 RDS_DAS_CHANGES_AUDIT_SPEC을 수정할 수 없습니다.
- 데이터베이스 감사 사양을 만들 때는 RDS_DAS_DB_<name> 형식(예: RDS_DAS_DB_databaseActions)을 사용해야 합니다.

> **Important**
>
> 소규모 인스턴스 클래스의 경우 모든 데이터를 감사하지 않고 필요한 데이터만 감사하는 것이 좋습니다. 이렇게 하면 데이터베이스 활동 스트림이 이러한 인스턴스 클래스 성능에 미치는 영향을 줄일 수 있습니다.

다음 샘플 코드는 서버 감사 사양 RDS_DAS_SERVER_AUDIT_SPEC을 수정하고 로그아웃 및 성공적인 로그인 작업을 감사합니다.

```
ALTER SERVER AUDIT SPECIFICATION [RDS_DAS_SERVER_AUDIT_SPEC]
      WITH (STATE=OFF);
ALTER SERVER AUDIT SPECIFICATION [RDS_DAS_SERVER_AUDIT_SPEC]
      ADD (LOGOUT_GROUP),
      ADD (SUCCESSFUL_LOGIN_GROUP)
      WITH (STATE = ON );
```

다음 샘플 코드는 데이터베이스 감사 사양 RDS_DAS_DB_database_spec을 생성하여 서버 감사 RDS_DAS_AUDIT에 첨부합니다.

```
USE testDB;
CREATE DATABASE AUDIT SPECIFICATION [RDS_DAS_DB_database_spec]
     FOR SERVER AUDIT [RDS_DAS_AUDIT]
     ADD ( INSERT, UPDATE, DELETE
          ON testTable BY testUser )
     WITH (STATE = ON);
```

감사 사양을 구성한 후에는 사양 RDS_DAS_SERVER_AUDIT_SPEC 및 RDS_DAS_DB_<name>의 상태가 ON으로 설정되어 있는지 확인합니다. 이제 해당 사양이 감사 데이터를 데이터베이스 활동 스트림으로 보낼 수 있습니다.

데이터베이스 활동 스트림 시작

DB 인스턴스 에 대해 활동 스트림을 시작하면 감사 정책에 구성한 각 데이터베이스 활동 이벤트가 활동 스트림 이벤트를 생성합니다. CONNECT 및 SELECT 같은 SQL 명령은 액세스 이벤트를 생성합니다. CREATE 및 INSERT 같은 SQL 명령은 변경 이벤트를 생성합니다.

Important

Oracle DB 인스턴스에 대해 활동 스트림을 활성화하면 기존 감사 데이터를 지웁니다. 또한 감사 추적 권한도 취소됩니다. 스트림이 활성화된 경우 RDS for Oracle은 더 이상 다음 작업을 수행할 수 없습니다.

- 통합 감사 추적 레코드 삭제
- 통합 감사 정책 추가, 삭제 또는 수정
- 마지막으로 아카이빙된 타임스탬프 업데이트

Console

데이터베이스 활동 스트림을 시작하려면

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.

3. 활동 스트림을 시작하려는 Amazon RDS 데이터베이스 인스턴스를 선택합니다. 다중 AZ 배포에서 기본 인스턴스에서만 스트림을 시작합니다. 활동 스트림은 기본 및 대기 DB 인스턴스를 모두 감사합니다.
4. 작업의 경우 Start activity stream(활동 스트림 시작)을 선택합니다.

   데이터베이스 활동 스트림 시작: ## 창이 나타납니다. 여기서 ##은 사용자의 RDS 인스턴스입니다.
5. 다음 설정을 입력합니다.

   - AWS KMS key의 경우, AWS KMS keys 목록에서 키를 선택합니다.

     RDS for Oracle이 KMS 키를 사용하여 키를 암호화하면 암호화된 키가 데이터베이스 활동을 암호화합니다. 기본 키가 아닌 KMS 키를 선택합니다. 암호화 키 및 AWS KMS에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서에서 AWS Key Management Service란 무엇입니까?를 참조하세요.
   - 데이터베이스 활동 이벤트에서 엔진 고유 감사 필드 활성화를 선택하여 엔진 고유의 감사 필드를 포함합니다.
   - [즉시(Immediately))를 선택합니다.

     [즉시(Immediately)]를 선택하면 RDS 인스턴스가 바로 다시 시작됩니다. [다음 유지 관리 기간 중(During the next maintenance window)]을 선택하면 RDS 인스턴스는 즉시 다시 시작되지 않습니다. 이 경우 데이터베이스 활동 스트림은 다음 유지 관리 기간까지 시작되지 않습니다.
6. 데이터베이스 활동 스트림 시작을 선택합니다.

   데이터베이스에 대한 상태는 활동 스트림이 시작 중임을 보여줍니다.

> **Note**
>
> `You can't start a database activity stream in this configuration` 오류가 발생하는 경우 RDS 인스턴스가 지원되는 인스턴스 클래스를 사용하고 있는지 데이터베이스 활동 스트림이 지원되는 DB 인스턴스 클래스에서 확인하세요.

AWS CLI

DB 인스턴스에 대해 데이터베이스 활동 스트림을 시작하려면 start-activity-stream AWS CLI 명령을 사용하여 데이터베이스를 구성합니다.

- `--resource-arn` *arn* - DB 인스턴스의 Amazon 리소스 이름(ARN)을 지정합니다.
- `--kms-key-id` *key* - 데이터베이스 활동 스트림에서 메시지를 암호화하기 위한 KMS 키 식별자를 지정합니다. AWS KMS 키 식별자는 AWS KMS key의 키 ARN, 키 ID, 별칭 ARN 또는 별칭 이름입니다.
- `--engine-native-audit-fields-included` - 데이터 스트림에 엔진별 감사 필드를 포함합니다. 이러한 필드를 제외하려면 `--no-engine-native-audit-fields-included`(default)을 지정합니다.

다음 예에서는 비동기 모드에서 DB 인스턴스에 대한 데이터베이스 활동 스트림을 시작합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds start-activity-stream \
    --mode async \
    --kms-key-id my-kms-key-arn \
    --resource-arn my-instance-arn \
    --engine-native-audit-fields-included \
    --apply-immediately
```

Windows의 경우:

```
aws rds start-activity-stream ^
    --mode async ^
    --kms-key-id my-kms-key-arn ^
    --resource-arn my-instance-arn ^
    --engine-native-audit-fields-included ^
    --apply-immediately
```

Amazon RDS API

DB 인스턴스에 대해 데이터베이스 활동 스트림을 시작하려면 StartActivityStream 작업을 사용하여 인스턴스를 구성합니다.

아래 파라미터를 사용하여 작업을 호출하세요.

- `Region`
- `KmsKeyId`
- `ResourceArn`
- `Mode`
- `EngineNativeAuditFieldsIncluded`

Amazon RDS에 대한 데이터베이스 활동 스트림 수정

활동 스트림이 시작될 때 Amazon RDS 감사 정책을 사용자 지정하는 것이 좋습니다. 활동 스트림을 중지하여 시간과 데이터를 잃지 않으려면 감사 정책 상태를 다음 설정 중 하나로 변경할 수 있습니다.

잠김(기본값)

데이터베이스의 감사 정책은 읽기 전용입니다.

잠금 해제됨

데이터베이스의 감사 정책은 읽기/쓰기가 가능합니다.

기본 단계는 다음과 같습니다.

1. 감사 정책 상태를 잠금 해제된 상태로 수정합니다.
2. 감사 정책을 사용자 정의합니다.
3. 감사 정책 상태를 잠김 상태로 수정합니다.

콘솔

활동 스트림의 감사 정책 상태를 수정하려면

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 작업에서는 데이터베이스 활동 스트림 수정을 선택합니다.

   데이터베이스 활동 스트림 시작: ***name*** 창이 나타납니다. 여기서 *name*은 사용자의 RDS 인스턴스입니다.
4. 다음 옵션 중 하나를 선택합니다.

잠김

감사 정책을 잠그면 읽기 전용으로 바뀝니다. 정책을 잠금 해제하거나 활동 스트림을 중지하지 않는 한 감사 정책을 편집할 수 없습니다.

잠금 해제됨

감사 정책을 잠금 해제하면 읽기/쓰기가 가능합니다. 활동 스트림이 시작되는 동안 감사 정책을 편집할 수 있습니다.

5. DB 활동 스트림 수정을 선택합니다.

   Amazon RDS 데이터베이스에 대한 상태는 활동 스트림이 구성 중임을 보여줍니다.

6. (선택 사항) DB 인스턴스 링크를 선택합니다. 그런 다음 구성 탭을 선택합니다.

   이 감사 정책 상태 필드는 다음 값 중 하나를 표시합니다.

   - 잠김
   - 잠금 해제됨
   - 잠금 정책
   - 잠금 해제 정책

AWS CLI

데이터베이스 인스턴스의 활동 스트림 상태를 수정하려면 modify-activity-stream AWS CLI 명령을 사용하세요.

옵션	필수?	Description
`--resource-arn` *my-instance-ARN*	예	RDS 데이터베이스 인스턴스의 Amazon 리소스 이름(ARN)입니다.
`--audit-policy-state`	아니요	인스턴스의 데이터베이스 활동 스트림에 대한 감사 정책의 새 상태는 `locked` 또는 `unlocked`입니다.

다음 예는 *my-instance-ARN*에서 시작된 활동 스트림에 대한 감사 정책을 잠금 해제합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-activity-stream \
    --resource-arn my-instance-ARN \
    --audit-policy-state unlocked
```

Windows의 경우:

```
aws rds modify-activity-stream ^
    --resource-arn my-instance-ARN ^
    --audit-policy-state unlocked
```

다음 예에서는 *my-instance* 인스턴스를 설명합니다. 부분 샘플 출력은 감사 정책이 잠금 해제되었음을 보여줍니다.

```
aws rds describe-db-instances --db-instance-identifier my-instance

{
    "DBInstances": [
        {
            ...
            "Engine": "oracle-ee",
            ...
            "ActivityStreamStatus": "started",
            "ActivityStreamKmsKeyId": "ab12345e-1111-2bc3-12a3-ab1cd12345e",
            "ActivityStreamKinesisStreamName": "aws-rds-das-db-
AB1CDEFG23GHIJK4LMNOPQRST",
            "ActivityStreamMode": "async",
            "ActivityStreamEngineNativeAuditFieldsIncluded": true,
            "ActivityStreamPolicyStatus": "unlocked",
            ...
        }
    ]
}
```

RDS API

데이터베이스 활동 스트림의 정책 상태를 수정하려면 ModifyActivityStream 작업을 사용합니다.

아래 파라미터를 사용하여 작업을 호출하세요.

- `AuditPolicyState`
- `ResourceArn`

데이터베이스 활동 스트림 상태 가져오기

콘솔 또는 AWS CLI를 사용하여 Amazon RDS 데이터베이스 인스턴스에 대한 활동 스트림의 상태를 가져올 수 있습니다.

콘솔

데이터베이스 활동 스트림 상태 가져오기

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [데이터베이스(Databases)]를 선택하고 DB 인스턴스 링크를 선택합니다.
3. 구성 탭을 선택하고 Database activity stream(데이터베이스 활동 스트림)에서 상태를 확인하십시오.

AWS CLI

describe-db-instances CLI 요청에 대한 응답으로 데이터베이스 인스턴스에 대한 활동 스트림 구성을 가져올 수 있습니다.

다음 예는 *my-instance*를 설명합니다.

```
aws rds --region my-region describe-db-instances --db-instance-identifier my-db
```

다음 예에서는 JSON 응답을 보여 줍니다. 다음 필드가 표시됩니다.

- `ActivityStreamKinesisStreamName`
- `ActivityStreamKmsKeyId`
- `ActivityStreamStatus`
- `ActivityStreamMode`
- `ActivityStreamPolicyStatus`

```
{
    "DBInstances": [
        {
            ...
            "Engine": "oracle-ee",
            ...
```

```
            "ActivityStreamStatus": "starting",
            "ActivityStreamKmsKeyId": "ab12345e-1111-2bc3-12a3-ab1cd12345e",
            "ActivityStreamKinesisStreamName": "aws-rds-das-db-
AB1CDEFG23GHIJK4LMNOPQRST",
            "ActivityStreamMode": "async",
            "ActivityStreamEngineNativeAuditFieldsIncluded": true,
            "ActivityStreamPolicyStatus": locked",
            ...
        }
    ]
}
```

RDS API

DescribeDBInstances 작업에 대한 응답으로 데이터베이스에 대한 활동 스트림 구성을 가져올 수 있습니다.

데이터베이스 활동 스트림 중지

콘솔 또는 AWS CLI를 사용하여 활동 스트림을 중지할 수 있습니다.

Amazon RDS 데이터베이스 인스턴스를 삭제하면 활동 스트림이 중지되고 기본 Amazon Kinesis 스트림이 자동으로 삭제됩니다.

콘솔

활동 스트림을 끄려면

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 데이터베이스 활동 스트림을 중지하려는 데이터베이스를 선택하세요.
4. 작업의 경우 Stop activity stream(작업 스트림 중지)을 선택합니다. Database Activity Stream(데이터베이스 활동 스트림) 창이 나타납니다.

    a. [즉시(Immediately))를 선택합니다.

    [즉시(Immediately)]를 선택하면 RDS 인스턴스가 바로 다시 시작됩니다. [다음 유지 관리 기간 중(During the next maintenance window)]을 선택하면 RDS 인스턴스는 즉시 다시 시작되지 않습니다. 이 경우 데이터베이스 활동 스트림은 다음 유지 관리 기간까지 중지되지 않습니다.

    b. [Continue]를 선택합니다.

AWS CLI

데이터베이스에 대한 데이터베이스 활동 스트림을 중지하려면 AWS CLI 명령 stop-activity-stream을 사용하여 DB 인스턴스를 구성하세요. `--region` 파라미터를 사용하여 DB 인스턴스에 대한 AWS 리전을 식별합니다. `--apply-immediately` 파라미터는 선택 항목입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds --region MY_REGION \
    stop-activity-stream \
    --resource-arn MY_DB_ARN \
    --apply-immediately
```

Windows의 경우:

```
aws rds --region MY_REGION ^
    stop-activity-stream ^
    --resource-arn MY_DB_ARN ^
    --apply-immediately
```

RDS API

데이터베이스에 대해 데이터베이스 활동 스트림을 중지하려면 StopActivityStream 작업을 사용하여 DB 인스턴스를 구성합니다. `Region` 파라미터를 사용하여 DB 인스턴스에 대한 AWS 리전을 식별합니다. `ApplyImmediately` 파라미터는 선택 항목입니다.

데이터베이스 활동 스트림 모니터링

데이터베이스 활동 스트림은 활동을 모니터링하고 보고합니다. 활동 스트림이 수집되어 Amazon Kinesis에 전송됩니다. Kinesis에서 활동 스트림을 모니터링하거나 다른 서비스 및 애플리케이션이 추가 분석을 위해 활동 스트림을 사용할 수 있습니다. AWS CLI 명령 `describe-db-instances` 또는 RDS API `DescribeDBInstances` 작업을 사용하여 기본 Kinesis 스트림 이름을 찾을 수 있습니다.

Amazon RDS는 다음과 같이 Kinesis 스트림을 관리합니다.

- Amazon RDS는 24시간 보존 기간으로 Kinesis 스트림을 자동으로 생성합니다.
- Amazon RDS는 필요한 경우 Kinesis 스트림 크기를 조정합니다.
- 데이터베이스 활동 스트림을 중지하거나 DB 인스턴스를 삭제하면 Amazon RDS에서 Kinesis 스트림을 삭제합니다.

다음 활동 범주가 모니터링되고 활동 스트림 감사 로그에 기록됩니다.

- SQL 명령 – 모든 SQL 명령이 감사되고 PL/SQL에서 준비된 문, 내장 함수 및 함수도 제공됩니다. 저장 프로시저에 대한 호출이 감사됩니다. 저장 프로시저 또는 함수 내에서 발급된 모든 SQL 문도 감사됩니다.
- 다른 데이터베이스 정보 – 모니터링되는 활동에는 전체 SQL 문, DML 명령의 영향을 받은 행의 행 수, 액세스된 객체 및 고유한 데이터베이스 이름이 포함됩니다. 데이터베이스 활동 스트림은 바인딩 변수와 저장 프로시저 파라미터도 모니터링합니다.

  > **Important**
  >
  > 각 문의 전체 SQL 텍스트는 중요한 데이터를 포함하여 활동 스트림 감사 로그에 표시됩니다. 그러나 데이터베이스 사용자 암호는 다음 SQL 문에서와 같이 Oracle이 컨텍스트에서 판별할 수 있는 경우 수정됩니다.
  >
  > ```
  > ALTER ROLE role-name WITH password
  > ```

- 연결 정보 – 모니터링되는 활동에는 세션 및 네트워크 정보, 서버 프로세스 ID 및 종료 코드가 포함됩니다.

DB 인스턴스를 모니터링하는 동안 활동 스트림에 오류가 발생하면 RDS 이벤트를 통해 사용자에게 알립니다.

다음 섹션에서는 데이터베이스 활동 스트림에 액세스하고, 이를 감사 및 처리할 수 있습니다.

주제

- Amazon Kinesis에서 활동 스트림에 액세스
- 데이터베이스 활동 스트림에 대한 감사 로그 내용 및 예제
- 데이터베이스 활동 스트림에 대한 databaseActivityEventList JSON 배열
- AWS SDK를 사용하여 데이터베이스 활동 스트림 처리

Amazon Kinesis에서 활동 스트림에 액세스

데이터베이스에 대해 활동 스트림을 활성화하면 Kinesis 스트림이 생성됩니다. Kinesis에서 데이터베이스 활동을 실시간으로 모니터링할 수 있습니다. 데이터베이스 활동을 추가 분석하려면 Kinesis 스트림을 소비자 애플리케이션에 연결하면 됩니다. 또한 IBM의 Security Guardium 또는 Imperva의

SecureSphere Database Audit and Protection과 같은 규정 준수 관리 애플리케이션에 스트림을 연결할 수 있습니다.

RDS 콘솔 또는 Kinesis 콘솔에서 Kinesis 스트림에 액세스할 수 있습니다.

RDS 콘솔을 사용하여 Kinesis에서 활동 스트림에 액세스하는 방법

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 활동 스트림을 시작한 Amazon RDS 데이터베이스 인스턴스를 선택합니다.
4. Configuration(구성)을 선택합니다.
5. 데이터베이스 활동 스트림에서 Kinesis 스트림 아래의 링크를 선택합니다.
6. Kinesis 콘솔에서 모니터링을 선택하여 데이터베이스 활동 관찰을 시작합니다.

Kinesis 콘솔을 사용하여 Kinesis에서 활동 스트림에 액세스하는 방법

1. https://console.aws.amazon.com/kinesis에서 Kinesis 콘솔을 엽니다.
2. Kinesis 스트림 목록에서 활동 스트림을 선택합니다.

   활동 스트림의 이름에는 접두사 `aws-rds-das-db-`와 그 뒤의 데이터베이스의 리소스 ID가 포함됩니다. 다음은 예제입니다.

   ```
   aws-rds-das-db-NHVOV4PCLWHGF52NP
   ```

   Amazon RDS 콘솔을 사용하여 데이터베이스의 리소스 ID를 찾으려면 데이터베이스 목록에서 DB 인스턴스를 선택한 다음 구성(Configuration) 탭을 선택합니다.

   AWS CLI를 사용하여 활동 스트림의 전체 Kinesis 스트림 이름을 찾으려면 describe-db-instances CLI 요청을 사용하고 응답에서 `ActivityStreamKinesisStreamName` 값을 기록합니다.
3. 데이터베이스 활동을 관찰하려면 모니터링을 선택하십시오.

Amazon Kinesis 사용에 대한 자세한 내용은 Amazon Kinesis Data Streams이란 무엇입니까?를 참조하십시오.

데이터베이스 활동 스트림에 대한 감사 로그 내용 및 예제

모니터링되는 이벤트는 데이터베이스 활동 스트림에 JSON 문자열로 표시됩니다. 구조는 `DatabaseActivityMonitoringRecord`를 포함하는 JSON 객체로 구성되며, 여기에는 `databaseActivityEventList` 활동 이벤트 배열이 포함됩니다.

> **Note**
>
> 데이터베이스 활동 스트림의 경우 `paramList` JSON 배열에는 최대 절전 모드 애플리케이션의 null 값이 포함되지 않습니다.

주제

- 활동 스트림 감사 로그 예제
- DatabaseActivityMonitoringRecords JSON 객체
- databaseActivityEvents JSON 객체

활동 스트림 감사 로그 예제

다음은 활동 이벤트 레코드의 해독된 JSON 감사 로그 샘플입니다.

Example CONNECT SQL 문 의 활동 이벤트 레코드

다음 활동 이벤트 레코드는 Oracle DB의 JDBC 씬 클라이언트(`clientApplication`)에서 `CONNECT` SQL 문(`command`)을 사용하여 로그인한 것을 보여줍니다.

```
{
    "class": "Standard",
    "clientApplication": "JDBC Thin Client",
    "command": "LOGON",
    "commandText": null,
    "dbid": "0123456789",
    "databaseName": "ORCL",
    "dbProtocol": "oracle",
    "dbUserName": "TEST",
    "endTime": null,
    "errorMessage": null,
    "exitCode": 0,
    "logTime": "2021-01-15 00:15:36.233787",
    "netProtocol": "tcp",
```

```
    "objectName": null,
    "objectType": null,
    "paramList": [],
    "pid": 17904,
    "remoteHost": "123.456.789.012",
    "remotePort": "25440",
    "rowCount": null,
    "serverHost": "987.654.321.098",
    "serverType": "oracle",
    "serverVersion": "19.0.0.0.ru-2020-01.rur-2020-01.r1.EE.3",
    "serviceName": "oracle-ee",
    "sessionId": 987654321,
    "startTime": null,
    "statementId": 1,
    "substatementId": null,
    "transactionId": "0000000000000000",
    "engineNativeAuditFields": {
        "UNIFIED_AUDIT_POLICIES": "TEST_POL_EVERYTHING",
        "FGA_POLICY_NAME": null,
        "DV_OBJECT_STATUS": null,
        "SYSTEM_PRIVILEGE_USED": "CREATE SESSION",
        "OLS_LABEL_COMPONENT_TYPE": null,
        "XS_SESSIONID": null,
        "ADDITIONAL_INFO": null,
        "INSTANCE_ID": 1,
        "DBID": 123456789
        "DV_COMMENT": null,
        "RMAN_SESSION_STAMP": null,
        "NEW_NAME": null,
        "DV_ACTION_NAME": null,
        "OLS_PROGRAM_UNIT_NAME": null,
        "OLS_STRING_LABEL": null,
        "RMAN_SESSION_RECID": null,
        "OBJECT_PRIVILEGES": null,
        "OLS_OLD_VALUE": null,
        "XS_TARGET_PRINCIPAL_NAME": null,
        "XS_NS_ATTRIBUTE": null,
        "XS_NS_NAME": null,
        "DBLINK_INFO": null,
        "AUTHENTICATION_TYPE": "(TYPE\u003d(DATABASE));(CLIENT ADDRESS\u003d((ADDRESS
\u003d(PROTOCOL\u003dtcp)(HOST\u003d205.251.233.183)(PORT\u003d25440))));",
        "OBJECT_EDITION": null,
        "OLS_PRIVILEGES_GRANTED": null,
        "EXCLUDED_USER": null,
```

```
        "DV_ACTION_OBJECT_NAME": null,
        "OLS_LABEL_COMPONENT_NAME": null,
        "EXCLUDED_SCHEMA": null,
        "DP_TEXT_PARAMETERS1": null,
        "XS_USER_NAME": null,
        "XS_ENABLED_ROLE": null,
        "XS_NS_ATTRIBUTE_NEW_VAL": null,
        "DIRECT_PATH_NUM_COLUMNS_LOADED": null,
        "AUDIT_OPTION": null,
        "DV_EXTENDED_ACTION_CODE": null,
        "XS_PACKAGE_NAME": null,
        "OLS_NEW_VALUE": null,
        "DV_RETURN_CODE": null,
        "XS_CALLBACK_EVENT_TYPE": null,
        "USERHOST": "a1b2c3d4e5f6.amazon.com",
        "GLOBAL_USERID": null,
        "CLIENT_IDENTIFIER": null,
        "RMAN_OPERATION": null,
        "TERMINAL": "unknown",
        "OS_USERNAME": "sumepate",
        "OLS_MAX_READ_LABEL": null,
        "XS_PROXY_USER_NAME": null,
        "XS_DATASEC_POLICY_NAME": null,
        "DV_FACTOR_CONTEXT": null,
        "OLS_MAX_WRITE_LABEL": null,
        "OLS_PARENT_GROUP_NAME": null,
        "EXCLUDED_OBJECT": null,
        "DV_RULE_SET_NAME": null,
        "EXTERNAL_USERID": null,
        "EXECUTION_ID": null,
        "ROLE": null,
        "PROXY_SESSIONID": 0,
        "DP_BOOLEAN_PARAMETERS1": null,
        "OLS_POLICY_NAME": null,
        "OLS_GRANTEE": null,
        "OLS_MIN_WRITE_LABEL": null,
        "APPLICATION_CONTEXTS": null,
        "XS_SCHEMA_NAME": null,
        "DV_GRANTEE": null,
        "XS_COOKIE": null,
        "DBPROXY_USERNAME": null,
        "DV_ACTION_CODE": null,
        "OLS_PRIVILEGES_USED": null,
        "RMAN_DEVICE_TYPE": null,
```

```
        "XS_NS_ATTRIBUTE_OLD_VAL": null,
        "TARGET_USER": null,
        "XS_ENTITY_TYPE": null,
        "ENTRY_ID": 1,
        "XS_PROCEDURE_NAME": null,
        "XS_INACTIVITY_TIMEOUT": null,
        "RMAN_OBJECT_TYPE": null,
        "SYSTEM_PRIVILEGE": null,
        "NEW_SCHEMA": null,
        "SCN": 5124715
    }
}
```

다음 활동 이벤트 레코드는 SQL Server DB의 로그인 실패를 보여줍니다.

```
{
    "type": "DatabaseActivityMonitoringRecord",
    "clusterId": "",
    "instanceId": "db-4JCWQLUZVFYP7DIWP6JVQ77O3Q",
    "databaseActivityEventList": [
        {
            "class": "LOGIN",
            "clientApplication": "Microsoft SQL Server Management Studio",
            "command": "LOGIN FAILED",
            "commandText": "Login failed for user 'test'. Reason: Password did not
 match that for the login provided. [CLIENT: local-machine]",
            "databaseName": "",
            "dbProtocol": "SQLSERVER",
            "dbUserName": "test",
            "endTime": null,
            "errorMessage": null,
            "exitCode": 0,
            "logTime": "2022-10-06 21:34:42.7113072+00",
            "netProtocol": null,
            "objectName": "",
            "objectType": "LOGIN",
            "paramList": null,
            "pid": null,
            "remoteHost": "local machine",
            "remotePort": null,
            "rowCount": 0,
            "serverHost": "172.31.30.159",
            "serverType": "SQLSERVER",
```

```
            "serverVersion": "15.00.4073.23.v1.R1",
            "serviceName": "sqlserver-ee",
            "sessionId": 0,
            "startTime": null,
            "statementId": "0x1eb0d1808d34a94b9d3dcf5432750f02",
            "substatementId": 1,
            "transactionId": "0",
            "type": "record",
            "engineNativeAuditFields": {
                "target_database_principal_id": 0,
                "target_server_principal_id": 0,
                "target_database_principal_name": "",
                "server_principal_id": 0,
                "user_defined_information": "",
                "response_rows": 0,
                "database_principal_name": "",
                "target_server_principal_name": "",
                "schema_name": "",
                "is_column_permission": false,
                "object_id": 0,
                "server_instance_name": "EC2AMAZ-NFUJJNO",
                "target_server_principal_sid": null,
                "additional_information": "<action_info "xmlns=\"http://
schemas.microsoft.com/sqlserver/2008/sqlaudit_data\"><pooled_connection>0</
pooled_connection><error>0x00004818</error><state>8</state><address>local machine</
address><PasswordFirstNibbleHash>B</PasswordFirstNibbleHash></action_info>"-->,
                "duration_milliseconds": 0,
                "permission_bitmask": "0x00000000000000000000000000000000",
                "data_sensitivity_information": "",
                "session_server_principal_name": "",
                "connection_id": "98B4F537-0F82-49E3-AB08-B9D33B5893EF",
                "audit_schema_version": 1,
                "database_principal_id": 0,
                "server_principal_sid": null,
                "user_defined_event_id": 0,
                "host_name": "EC2AMAZ-NFUJJNO"
            }
        }
    ]
}
```

> **Note**
>
> 데이터베이스 활동 스트림이 활성화되지 않은 경우 JSON 문서의 마지막 필드는 `"engineNativeAuditFields": { }`입니다.

Example CREATE TABLE 문의 활동 이벤트 레코드

다음 예시는 Oracle 데이터베이스에 대한 CREATE TABLE 이벤트를 보여줍니다.


```
{
    "class": "Standard",
    "clientApplication": "sqlplus@ip-12-34-5-678 (TNS V1-V3)",
    "command": "CREATE TABLE",
    "commandText": "CREATE TABLE persons(\n    person_id NUMBER GENERATED BY DEFAULT AS
 IDENTITY,\n    first_name VARCHAR2(50) NOT NULL,\n    last_name VARCHAR2(50) NOT NULL,
\n    PRIMARY KEY(person_id)\n)",
    "dbid": "0123456789",
    "databaseName": "ORCL",
    "dbProtocol": "oracle",
    "dbUserName": "TEST",
    "endTime": null,
    "errorMessage": null,
    "exitCode": 0,
    "logTime": "2021-01-15 00:22:49.535239",
    "netProtocol": "beq",
    "objectName": "PERSONS",
    "objectType": "TEST",
    "paramList": [],
    "pid": 17687,
    "remoteHost": "123.456.789.0",
    "remotePort": null,
    "rowCount": null,
    "serverHost": "987.654.321.01",
    "serverType": "oracle",
    "serverVersion": "19.0.0.0.ru-2020-01.rur-2020-01.r1.EE.3",
    "serviceName": "oracle-ee",
    "sessionId": 1234567890,
    "startTime": null,
    "statementId": 43,
    "substatementId": null,
    "transactionId": "090011007F0D0000",
    "engineNativeAuditFields": {
```

```
        "UNIFIED_AUDIT_POLICIES": "TEST_POL_EVERYTHING",
        "FGA_POLICY_NAME": null,
        "DV_OBJECT_STATUS": null,
        "SYSTEM_PRIVILEGE_USED": "CREATE SEQUENCE, CREATE TABLE",
        "OLS_LABEL_COMPONENT_TYPE": null,
        "XS_SESSIONID": null,
        "ADDITIONAL_INFO": null,
        "INSTANCE_ID": 1,
        "DV_COMMENT": null,
        "RMAN_SESSION_STAMP": null,
        "NEW_NAME": null,
        "DV_ACTION_NAME": null,
        "OLS_PROGRAM_UNIT_NAME": null,
        "OLS_STRING_LABEL": null,
        "RMAN_SESSION_RECID": null,
        "OBJECT_PRIVILEGES": null,
        "OLS_OLD_VALUE": null,
        "XS_TARGET_PRINCIPAL_NAME": null,
        "XS_NS_ATTRIBUTE": null,
        "XS_NS_NAME": null,
        "DBLINK_INFO": null,
        "AUTHENTICATION_TYPE": "(TYPE\u003d(DATABASE));(CLIENT ADDRESS\u003d((PROTOCOL
\u003dbeq)(HOST\u003d123.456.789.0)));",
        "OBJECT_EDITION": null,
        "OLS_PRIVILEGES_GRANTED": null,
        "EXCLUDED_USER": null,
        "DV_ACTION_OBJECT_NAME": null,
        "OLS_LABEL_COMPONENT_NAME": null,
        "EXCLUDED_SCHEMA": null,
        "DP_TEXT_PARAMETERS1": null,
        "XS_USER_NAME": null,
        "XS_ENABLED_ROLE": null,
        "XS_NS_ATTRIBUTE_NEW_VAL": null,
        "DIRECT_PATH_NUM_COLUMNS_LOADED": null,
        "AUDIT_OPTION": null,
        "DV_EXTENDED_ACTION_CODE": null,
        "XS_PACKAGE_NAME": null,
        "OLS_NEW_VALUE": null,
        "DV_RETURN_CODE": null,
        "XS_CALLBACK_EVENT_TYPE": null,
        "USERHOST": "ip-10-13-0-122",
        "GLOBAL_USERID": null,
        "CLIENT_IDENTIFIER": null,
        "RMAN_OPERATION": null,
```

```
        "TERMINAL": "pts/1",
        "OS_USERNAME": "rdsdb",
        "OLS_MAX_READ_LABEL": null,
        "XS_PROXY_USER_NAME": null,
        "XS_DATASEC_POLICY_NAME": null,
        "DV_FACTOR_CONTEXT": null,
        "OLS_MAX_WRITE_LABEL": null,
        "OLS_PARENT_GROUP_NAME": null,
        "EXCLUDED_OBJECT": null,
        "DV_RULE_SET_NAME": null,
        "EXTERNAL_USERID": null,
        "EXECUTION_ID": null,
        "ROLE": null,
        "PROXY_SESSIONID": 0,
        "DP_BOOLEAN_PARAMETERS1": null,
        "OLS_POLICY_NAME": null,
        "OLS_GRANTEE": null,
        "OLS_MIN_WRITE_LABEL": null,
        "APPLICATION_CONTEXTS": null,
        "XS_SCHEMA_NAME": null,
        "DV_GRANTEE": null,
        "XS_COOKIE": null,
        "DBPROXY_USERNAME": null,
        "DV_ACTION_CODE": null,
        "OLS_PRIVILEGES_USED": null,
        "RMAN_DEVICE_TYPE": null,
        "XS_NS_ATTRIBUTE_OLD_VAL": null,
        "TARGET_USER": null,
        "XS_ENTITY_TYPE": null,
        "ENTRY_ID": 12,
        "XS_PROCEDURE_NAME": null,
        "XS_INACTIVITY_TIMEOUT": null,
        "RMAN_OBJECT_TYPE": null,
        "SYSTEM_PRIVILEGE": null,
        "NEW_SCHEMA": null,
        "SCN": 5133083
    }
}
```

다음 예시는 SQL Server 데이터베이스의 CREATE TABLE 이벤트를 보여줍니다.

```
{
    "type": "DatabaseActivityMonitoringRecord",
```

```
    "clusterId": "",
    "instanceId": "db-4JCWQLUZVFYP7DIWP6JVQ77O3Q",
    "databaseActivityEventList": [
        {
            "class": "SCHEMA",
            "clientApplication": "Microsoft SQL Server Management Studio - Query",
            "command": "ALTER",
            "commandText": "Create table [testDB].[dbo].[TestTable2](\r\ntextA
 varchar(6000),\r\n    textB varchar(6000)\r\n)",
            "databaseName": "testDB",
            "dbProtocol": "SQLSERVER",
            "dbUserName": "test",
            "endTime": null,
            "errorMessage": null,
            "exitCode": 1,
            "logTime": "2022-10-06 21:44:38.4120677+00",
            "netProtocol": null,
            "objectName": "dbo",
            "objectType": "SCHEMA",
            "paramList": null,
            "pid": null,
            "remoteHost": "local machine",
            "remotePort": null,
            "rowCount": 0,
            "serverHost": "172.31.30.159",
            "serverType": "SQLSERVER",
            "serverVersion": "15.00.4073.23.v1.R1",
            "serviceName": "sqlserver-ee",
            "sessionId": 84,
            "startTime": null,
            "statementId": "0x5178d33d56e95e419558b9607158a5bd",
            "substatementId": 1,
            "transactionId": "4561864",
            "type": "record",
            "engineNativeAuditFields": {
                "target_database_principal_id": 0,
                "target_server_principal_id": 0,
                "target_database_principal_name": "",
                "server_principal_id": 2,
                "user_defined_information": "",
                "response_rows": 0,
                "database_principal_name": "dbo",
                "target_server_principal_name": "",
                "schema_name": "",
```

```
                "is_column_permission": false,
                "object_id": 1,
                "server_instance_name": "EC2AMAZ-NFUJJNO",
                "target_server_principal_sid": null,
                "additional_information": "",
                "duration_milliseconds": 0,
                "permission_bitmask": "0x00000000000000000000000000000000",
                "data_sensitivity_information": "",
                "session_server_principal_name": "test",
                "connection_id": "EE1FE3FD-EF2C-41FD-AF45-9051E0CD983A",
                "audit_schema_version": 1,
                "database_principal_id": 1,
                "server_principal_sid":
 "0x0105000000000005150000000bdc2795e2d0717901ba6998cf4010000",
                "user_defined_event_id": 0,
                "host_name": "EC2AMAZ-NFUJJNO"
            }
        }
    ]
}
```

Example SELECT 문의 활동 이벤트 레코드

다음 예시는 Oracle DB에 대한 SELECT 이벤트를 보여줍니다.

```
{
    "class": "Standard",
    "clientApplication": "sqlplus@ip-12-34-5-678 (TNS V1-V3)",
    "command": "SELECT",
    "commandText": "select count(*) from persons",
    "databaseName": "1234567890",
    "dbProtocol": "oracle",
    "dbUserName": "TEST",
    "endTime": null,
    "errorMessage": null,
    "exitCode": 0,
    "logTime": "2021-01-15 00:25:18.850375",
    "netProtocol": "beq",
    "objectName": "PERSONS",
    "objectType": "TEST",
    "paramList": [],
    "pid": 17687,
    "remoteHost": "123.456.789.0",
    "remotePort": null,
```

```
    "rowCount": null,
    "serverHost": "987.654.321.09",
    "serverType": "oracle",
    "serverVersion": "19.0.0.0.ru-2020-01.rur-2020-01.r1.EE.3",
    "serviceName": "oracle-ee",
    "sessionId": 1080639707,
    "startTime": null,
    "statementId": 44,
    "substatementId": null,
    "transactionId": null,
    "engineNativeAuditFields": {
        "UNIFIED_AUDIT_POLICIES": "TEST_POL_EVERYTHING",
        "FGA_POLICY_NAME": null,
        "DV_OBJECT_STATUS": null,
        "SYSTEM_PRIVILEGE_USED": null,
        "OLS_LABEL_COMPONENT_TYPE": null,
        "XS_SESSIONID": null,
        "ADDITIONAL_INFO": null,
        "INSTANCE_ID": 1,
        "DV_COMMENT": null,
        "RMAN_SESSION_STAMP": null,
        "NEW_NAME": null,
        "DV_ACTION_NAME": null,
        "OLS_PROGRAM_UNIT_NAME": null,
        "OLS_STRING_LABEL": null,
        "RMAN_SESSION_RECID": null,
        "OBJECT_PRIVILEGES": null,
        "OLS_OLD_VALUE": null,
        "XS_TARGET_PRINCIPAL_NAME": null,
        "XS_NS_ATTRIBUTE": null,
        "XS_NS_NAME": null,
        "DBLINK_INFO": null,
        "AUTHENTICATION_TYPE": "(TYPE\u003d(DATABASE));(CLIENT ADDRESS\u003d((PROTOCOL
\u003dbeq)(HOST\u003d123.456.789.0)));",
        "OBJECT_EDITION": null,
        "OLS_PRIVILEGES_GRANTED": null,
        "EXCLUDED_USER": null,
        "DV_ACTION_OBJECT_NAME": null,
        "OLS_LABEL_COMPONENT_NAME": null,
        "EXCLUDED_SCHEMA": null,
        "DP_TEXT_PARAMETERS1": null,
        "XS_USER_NAME": null,
        "XS_ENABLED_ROLE": null,
        "XS_NS_ATTRIBUTE_NEW_VAL": null,
```

```
        "DIRECT_PATH_NUM_COLUMNS_LOADED": null,
        "AUDIT_OPTION": null,
        "DV_EXTENDED_ACTION_CODE": null,
        "XS_PACKAGE_NAME": null,
        "OLS_NEW_VALUE": null,
        "DV_RETURN_CODE": null,
        "XS_CALLBACK_EVENT_TYPE": null,
        "USERHOST": "ip-12-34-5-678",
        "GLOBAL_USERID": null,
        "CLIENT_IDENTIFIER": null,
        "RMAN_OPERATION": null,
        "TERMINAL": "pts/1",
        "OS_USERNAME": "rdsdb",
        "OLS_MAX_READ_LABEL": null,
        "XS_PROXY_USER_NAME": null,
        "XS_DATASEC_POLICY_NAME": null,
        "DV_FACTOR_CONTEXT": null,
        "OLS_MAX_WRITE_LABEL": null,
        "OLS_PARENT_GROUP_NAME": null,
        "EXCLUDED_OBJECT": null,
        "DV_RULE_SET_NAME": null,
        "EXTERNAL_USERID": null,
        "EXECUTION_ID": null,
        "ROLE": null,
        "PROXY_SESSIONID": 0,
        "DP_BOOLEAN_PARAMETERS1": null,
        "OLS_POLICY_NAME": null,
        "OLS_GRANTEE": null,
        "OLS_MIN_WRITE_LABEL": null,
        "APPLICATION_CONTEXTS": null,
        "XS_SCHEMA_NAME": null,
        "DV_GRANTEE": null,
        "XS_COOKIE": null,
        "DBPROXY_USERNAME": null,
        "DV_ACTION_CODE": null,
        "OLS_PRIVILEGES_USED": null,
        "RMAN_DEVICE_TYPE": null,
        "XS_NS_ATTRIBUTE_OLD_VAL": null,
        "TARGET_USER": null,
        "XS_ENTITY_TYPE": null,
        "ENTRY_ID": 13,
        "XS_PROCEDURE_NAME": null,
        "XS_INACTIVITY_TIMEOUT": null,
        "RMAN_OBJECT_TYPE": null,
```

```
        "SYSTEM_PRIVILEGE": null,
        "NEW_SCHEMA": null,
        "SCN": 5136972
    }
}
```

다음 예시는 SQL Server DB에 대한 SELECT 이벤트를 보여줍니다.

```
{
    "type": "DatabaseActivityMonitoringRecord",
    "clusterId": "",
    "instanceId": "db-4JCWQLUZVFYP7DIWP6JVQ77O3Q",
    "databaseActivityEventList": [
        {
            "class": "TABLE",
            "clientApplication": "Microsoft SQL Server Management Studio - Query",
            "command": "SELECT",
            "commandText": "select * from [testDB].[dbo].[TestTable]",
            "databaseName": "testDB",
            "dbProtocol": "SQLSERVER",
            "dbUserName": "test",
            "endTime": null,
            "errorMessage": null,
            "exitCode": 1,
            "logTime": "2022-10-06 21:24:59.9422268+00",
            "netProtocol": null,
            "objectName": "TestTable",
            "objectType": "TABLE",
            "paramList": null,
            "pid": null,
            "remoteHost": "local machine",
            "remotePort": null,
            "rowCount": 0,
            "serverHost": "172.31.30.159",
            "serverType": "SQLSERVER",
            "serverVersion": "15.00.4073.23.v1.R1",
            "serviceName": "sqlserver-ee",
            "sessionId": 62,
            "startTime": null,
            "statementId": "0x03baed90412f564fad640ebe51f89b99",
            "substatementId": 1,
            "transactionId": "4532935",
            "type": "record",
```

```
            "engineNativeAuditFields": {
                "target_database_principal_id": 0,
                "target_server_principal_id": 0,
                "target_database_principal_name": "",
                "server_principal_id": 2,
                "user_defined_information": "",
                "response_rows": 0,
                "database_principal_name": "dbo",
                "target_server_principal_name": "",
                "schema_name": "dbo",
                "is_column_permission": true,
                "object_id": 581577110,
                "server_instance_name": "EC2AMAZ-NFUJJNO",
                "target_server_principal_sid": null,
                "additional_information": "",
                "duration_milliseconds": 0,
                "permission_bitmask": "0x00000000000000000000000000000001",
                "data_sensitivity_information": "",
                "session_server_principal_name": "test",
                "connection_id": "AD3A5084-FB83-45C1-8334-E923459A8109",
                "audit_schema_version": 1,
                "database_principal_id": 1,
                "server_principal_sid":
 "0x0105000000000005150000000bdc2795e2d0717901ba6998cf4010000",
                "user_defined_event_id": 0,
                "host_name": "EC2AMAZ-NFUJJNO"
            }
        }
    ]
}
```


DatabaseActivityMonitoringRecords JSON 객체

데이터베이스 작업 이벤트 레코드는 다음 정보가 포함된 JSON 객체에 있습니다.

JSON 필드	데이터 형식	설명
`type`	string	JSON 레코드 형식입니다. 이 값은 `DatabaseActivityMonitoringRecords` 입니다.

JSON 필드	데이터 형식	설명
`version`	string	데이터베이스 작업 모니터링 레코드의 버전입니다. Oracle DB는 버전 1.3을 사용하고 SQL Server는 버전 1.4를 사용합니다. 이러한 엔진 버전에는 `engineNativeAuditFields` JSON 객체가 도입됩니다.
databaseActivityEvents	문자열	작업 이벤트를 포함하는 JSON 객체입니다.
키	문자열	databaseActivityEventList JSON 배열를 해독하는 데 사용되는 암호화 키

databaseActivityEvents JSON 객체

`databaseActivityEvents` JSON 객체에는 다음과 같은 정보가 포함되어 있습니다.

JSON 레코드의 최상위 필드

감사 로그의 각 이벤트는 JSON 형식의 레코드 내에 래핑됩니다. 이 레코드에는 다음 필드가 포함되어 있습니다.

type

이 필드는 항상 값이 `DatabaseActivityMonitoringRecords`입니다.

version

이 필드는 데이터베이스 활동 스트림 데이터 프로토콜 또는 계약 버전을 나타냅니다. 이는 사용 가능한 필드를 정의합니다.

databaseActivityEvents

하나 이상의 활동 이벤트를 나타내는 암호화된 문자열입니다. base64 바이트 배열로 표현됩니다. 문자열을 해독하면 결과는 이 단원의 예제와 같이 필드가 있는 JSON 형식의 레코드입니다.

키

`databaseActivityEvents` 문자열을 암호화하는 데 사용되는 암호화된 데이터 키입니다. 이 키는 데이터베이스 활동 스트림을 시작할 때 제공한 AWS KMS key와(과) 동일합니다.

다음 예제에서는 이 레코드의 형식을 보여줍니다.

```
{
  "type":"DatabaseActivityMonitoringRecords",
  "version":"1.3",
  "databaseActivityEvents":"encrypted audit records",
  "key":"encrypted key"
}
```

```
          "type":"DatabaseActivityMonitoringRecords",
          "version":"1.4",
          "databaseActivityEvents":"encrypted audit records",
          "key":"encrypted key"
```

`databaseActivityEvents` 필드의 내용을 해독하려면 다음 단계를 수행합니다.

1. 데이터베이스 활동 스트림을 시작할 때 제공한 키를 사용하여 `key` JSON 필드의 값을 복호화합니다. 이렇게 하면 데이터 암호화 키가 일반 텍스트로 반환됩니다.
2. Base64로 `databaseActivityEvents` JSON 필드의 값을 디코딩하여 감사 페이로드의 암호화 텍스트를 이진 형식으로 가져옵니다.
3. 첫 번째 단계에서 디코딩한 데이터 암호화 키를 사용하여 이진 암호화 텍스트를 해독합니다.
4. 해독된 페이로드의 압축을 풉니다.
   - 암호화된 페이로드가 `databaseActivityEvents` 필드에 있습니다.
   - `databaseActivityEventList` 필드에는 감사 레코드 배열이 포함되어 있습니다. 배열의 `type` 필드는 `record` 또는 `heartbeat`일 수 있습니다.

감사 로그 활동 이벤트 레코드는 다음 정보가 포함된 JSON 객체입니다.

JSON 필드	데이터 형식	설명
`type`	string	JSON 레코드 형식입니다. 이 값은 `DatabaseActivityMonitoringRecord` 입니다.

JSON 필드	데이터 형식	설명
`instanceId`	string	DB 인스턴스 리소스 식별자입니다. DB 인스턴스 속성 `DbiResourceId` 에 해당합니다.
databaseActivityEventList JSON 배열	string	활동 감사 레코드 또는 하트비트 메시지의 배열입니다.

데이터베이스 활동 스트림에 대한 databaseActivityEventList JSON 배열

감사 로그 페이로드는 암호화된 `databaseActivityEventList` JSON 배열입니다. 다음 표에는 감사 로그의 복호화된 `DatabaseActivityEventList` 배열에 있는 각 활동 이벤트의 필드가 알파벳 순으로 나열되어 있습니다.

Oracle 데이터베이스에서 통합 감사가 활성화된 경우 이 새 감사 추적에 감사 레코드가 채워집니다. `UNIFIED_AUDIT_TRAIL` 보기는 감사 추적에서 감사 레코드를 검색하여 감사 레코드를 테이블 형식으로 표시합니다. 데이터베이스 활동 스트림을 시작하려면 `UNIFIED_AUDIT_TRAIL`의 열이 `databaseActivityEventList` 배열의 필드에 매핑됩니다.

> **Important**
>
> 이벤트 구조는 변경될 수 있습니다. Amazon RDS는 향후 활동 이벤트에 새 필드를 추가할 수 있습니다. JSON 데이터를 구문 분석하는 애플리케이션에서 코드는 알 수 없는 필드 이름에 대해 무시할 수 있는지 또는 적절한 작업을 수행할 수 있는지 확인합니다.

Amazon RDS for Oracle에 대한 databaseActivityEventList 필드

Amazon RDS for Oracle에 대한 `databaseActivityEventList` 필드는 다음과 같습니다.

필드	데이터 형식	소스	설명
class	string	UNIFIED_AUDIT_TRAIL 의 AUDIT_TYPE 열	활동 이벤트의 클래스입니다. 이는 UNIFIED_AUDIT_TRAIL 보기의 AUDIT_TYPE 열에 해당합니다. Amazon RDS for Oracle에 대한 유효한 값은 다음과 같습니다. • Standard • FineGrainedAudit • XS • Database Vault • Label Security • RMAN_AUDIT • Datapump • Direct path API 자세한 내용은 Oracle 문서의 UNIFIED_AUDIT_TRAIL을 참조하세요.
clientApplication	string	CLIENT_PROGRAM_NAME 의 UNIFIED_AUDIT_TRAIL	클라이언트가 보고한 대로 클라이언트가 연결에 사용한 애플리케이션입니다. 클라이언트는 이 정보를 제공할 필요가 없으므로 값은 null일 수 있

필드	데이터 형식	소스	설명
			습니다. 샘플 값은 JDBC Thin Client입니다.
command	string	UNIFIED_AUDIT_TRAIL 의 ACTION_NAME 열	사용자가 실행한 작업 이름입니다. 전체 작업을 이해하려면 명령 이름과 AUDIT_TYPE 값을 숙지합니다. 샘플 값은 ALTER DATABASE입니다.
commandText	string	UNIFIED_AUDIT_TRAIL 의 SQL_TEXT 열	이벤트와 연결된 SQL 문 샘플 값은 ALTER DATABASE BEGIN BACKUP입니다.
databaseName	string	V$DATABASE 의 NAME 열	데이터베이스의 이름입니다.
dbid	숫자	UNIFIED_AUDIT_TRAIL 의 DBID 열	데이터베이스의 숫자 식별자입니다. 샘플 값은 1559204751 입니다.
dbProtocol	string	해당 사항 없음	데이터베이스 프로토콜. 이 베타에서 값은 oracle입니다.
dbUserName	string	UNIFIED_AUDIT_TRAIL 의 DBUSERNAME 열	작업이 감사된 데이터베이스 사용자의 이름입니다. 샘플 값은 RDSADMIN입니다.

필드	데이터 형식	소스	설명
endTime	string	해당 사항 없음	이 필드는 RDS for Oracle에 사용되지 않으며 항상 null입니다.

필드	데이터 형식	소스	설명
engineNativeAuditFields	객체	UNIFIED_AUDIT_TRAIL	기본적으로 이 객체는 비어 있습니다. --engine-native-audit-fields-included 옵션을 사용하여 활동 스트림을 시작하면 이 객체에는 다음과 같은 열과 값이 포함됩니다. ADDITIONAL_INFO APPLICATION_CONTEXTS AUDIT_OPTION AUTHENTICATION_TYPE CLIENT_IDENTIFIER CURRENT_USER DBLINK_INFO DBPROXY_USERNAME DIRECT_PATH_NUM_COLUMNS_LOADED DP_BOOLEAN_PARAMETERS1 DP_TEXT_PARAMETERS1 DV_ACTION_CODE DV_ACTION_NAME DV_ACTION_OBJECT_NAME DV_COMMENT DV_EXTENDED_ACTION_CODE DV_FACTOR_CONTEXT DV_GRANTEE DV_OBJECT_STATUS

필드	데이터 형식	소스	설명
			DV_RETURN_CODE DV_RULE_SET_NAME ENTRY_ID EXCLUDED_OBJECT EXCLUDED_SCHEMA EXCLUDED_USER EXECUTION_ID EXTERNAL_USERID FGA_POLICY_NAME GLOBAL_USERID INSTANCE_ID KSACL_SER VICE_NAME KSACL_SOURCE_LOCA TION KSACL_USER_NAME NEW_NAME NEW_SCHEMA OBJECT_EDITION OBJECT_PRIVILEGES OLS_GRANTEE OLS_LABEL_COM PONENT_NAME OLS_LABEL_COMPO NENT_TYPE OLS_MAX_READ_LABEL OLS_MAX_WRITE_LA BEL OLS_MIN_WRITE_ LABEL OLS_NEW_VALUE OLS_OLD_VALUE OLS_PARENT_ GROUP_NAME OLS_POLICY_NAME OLS_PRIVILEGES_GRA NTED

필드	데이터 형식	소스	설명
			OLS_PRIVILEGES_USED OLS_PROGRAM_UNIT_NAME OLS_STRING_LABEL OS_USERNAME PROTOCOL_ACTION_NAME PROTOCOL_MESSAGE PROTOCOL_RETURN_CODE PROTOCOL_SESSION_ID PROTOCOL_USERHOST PROXY_SESSIONID RLS_INFO RMAN_DEVICE_TYPE RMAN_OBJECT_TYPE RMAN_OPERATION RMAN_SESSION_RECID RMAN_SESSION_STAMP ROLE SCN SYSTEM_PRIVILEGE SYSTEM_PRIVILEGE_USED TARGET_USER TERMINAL UNIFIED_AUDIT_POLICIES USERHOST XS_CALLBACK_EVENT_TYPE XS_COOKIE XS_DATASEC_POLICY_NAME XS_ENABLED_ROLE

필드	데이터 형식	소스	설명
			XS_ENTITY_TYPE XS_INACTIVITY_TIMEOUT XS_NS_ATTRIBUTE XS_NS_ATTRIBUTE_NEW_VAL XS_NS_ATTRIBUTE_OLD_VAL XS_NS_NAME XS_PACKAGE_NAME XS_PROCEDURE_NAME XS_PROXY_USER_NAME XS_SCHEMA_NAME XS_SESSIONID XS_TARGET_PRINCIPAL_NAME XS_USER_NAME 자세한 내용은 Oracle 데이터베이스 문서의 UNIFIED_AUDIT_TRAIL을 참조하세요.
errorMessage	string	해당 사항 없음	이 필드는 RDS for Oracle에 사용되지 않으며 항상 null입니다.
exitCode	숫자	UNIFIED_AUDIT_TRAIL 의 RETURN_CODE 열	작업으로 인해 생성된 Oracle 데이터베이스 오류 코드입니다. 작업이 성공한 경우 값은 0입니다.

필드	데이터 형식	소스	설명
logTime	string	UNIFIED_AUDIT_TRAIL 의 EVENT_TIMESTAMP_UTC 열	감사 추적 항목 작성에 대한 타임스탬프입니다. 샘플 값은 2020-11-27 06:56:14.981404 입니다.
netProtocol	string	UNIFIED_AUDIT_TRAIL 의 AUTHENTICATION_TYPE 열	네트워크 통신 프로토콜. 샘플 값은 TCP입니다.
objectName	string	UNIFIED_AUDIT_TRAIL 의 OBJECT_NAME 열	작업의 영향을 받는 객체의 이름입니다. 샘플 값은 employees 입니다.
objectType	string	UNIFIED_AUDIT_TRAIL 의 OBJECT_SCHEMA 열	작업의 영향을 받는 객체의 스키마 이름입니다. 샘플 값은 hr입니다.
paramList	목록	UNIFIED_AUDIT_TRAIL 의 SQL_BINDS 열	해당되는 경우 SQL_TEXT와 관련된 경우 바인딩 변수 목록입니다. 샘플 값은 parameter_1,parameter_2 입니다.
pid	숫자	UNIFIED_AUDIT_TRAIL 의 OS_PROCESS 열	Oracle 데이터베이스 프로세스의 운영 체제 프로세스 식별자입니다. 샘플 값은 22396입니다.

필드	데이터 형식	소스	설명
remoteHost	string	UNIFIED_AUDIT_TRAIL 의 AUTHENTICATION_TYPE 열	세션을 생성한 호스트의 클라이언트 IP 주소 또는 클라이언트 이름입니다. 샘플 값은 123.456.789.123 입니다.
remotePort	string	UNIFIED_AUDIT_TRAIL 의 AUTHENTICATION_TYPE 열	클라이언트 포트 번호. Oracle 데이터베이스 환경의 일반적인 값은 1521입니다.
rowCount	숫자	해당 사항 없음	이 필드는 RDS for Oracle에 사용되지 않으며 항상 null입니다.
serverHost	string	데이터베이스 호스트	데이터베이스 서버 호스트 IP 주소. 샘플 값은 123.456.789.123 입니다.
serverType	string	해당 사항 없음	데이터베이스 서버 유형입니다. 이 값은 항상 ORACLE입니다.
serverVersion	string	데이터베이스 호스트	Amazon RDS for Oracle 버전, RU(릴리스 업데이트) 및 RUR(릴리스 업데이트 버전) 샘플 값은 19.0.0.0.ru-2020-01.rur-2020-01.r1.EE.3 입니다.

필드	데이터 형식	소스	설명
serviceName	string	데이터베이스 호스트	서비스의 이름입니다. 샘플 값은 oracle-ee 입니다.
sessionId	숫자	UNIFIED_AUDIT_TRAIL 의 SESSIONID 열	감사의 세션 식별자입니다. 예를 들면, 1894327130 입니다.
startTime	string	해당 사항 없음	이 필드는 RDS for Oracle에 사용되지 않으며 항상 null입니다.
statementId	숫자	UNIFIED_AUDIT_TRAIL 의 STATEMENT_ID 열	각 문 실행에 대한 숫자 ID입니다. 문은 많은 작업을 일으킬 수 있습니다. 샘플 값은 142197입니다.
substatementId	해당 사항 없음	해당 사항 없음	이 필드는 RDS for Oracle에 사용되지 않으며 항상 null입니다.
transactionId	string	UNIFIED_AUDIT_TRAIL 의 TRANSACTION_ID 열	객체가 수정된 트랜잭션의 식별자입니다. 샘플 값은 02000800D5030000 입니다.

Amazon RDS for SQL Server에 대한 databaseActivityEventList 필드

Amazon RDS for SQL Server에 대한 databaseActivityEventList 필드는 다음과 같습니다.

필드	데이터 형식	소스	설명
class	문자열	sys.fn_get_audit_file.class_type 이 sys.dm_audit_class_type_map.class_type_desc 에 매핑됨	활동 이벤트의 클래스입니다. 자세한 내용은 Microsoft 설명서의 SQL Server Audit(데이터베이스 엔진)를 참조하세요.
clientApplication	문자열	sys.fn_get_audit_file.application_name	클라이언트가 보고한 대로 클라이언트가 연결하는 애플리케이션(SQL Server 버전 14 이상). SQL Server 버전 13에서는 이 필드가 null입니다.
command	문자열	sys.fn_get_audit_file.action_id 이 sys.dm_audit_actions.name 에 매핑됨	SQL 문의 일반 범주입니다. 이 필드의 값은 클래스의 값에 따라 달라집니다.
commandText	문자열	sys.fn_get_audit_file.statement	이 필드는 SQL 문을 나타냅니다.
databaseName	문자열	sys.fn_get_audit_file.database_name	데이터베이스 이름
dbProtocol	문자열	해당 사항 없음	데이터베이스 프로토콜. 이 값은 SQLSERVER 입니다.
dbUserName	문자열	sys.fn_get_audit_file.server_principal_name	클라이언트 인증을 위한 데이터베이스 사용자.
endTime	문자열	해당 사항 없음	이 필드는 Amazon RDS for SQL Server에서 사용되지 않으며 값은 null입니다.
engineNativeAuditFields	객체	sys.fn_get_audit_file 에서 이 열에 나열되지 않은 각 필드.	기본적으로 이 객체는 비어 있습니다. --engine-native-audit-fields-included 옵

필드	데이터 형식	소스	설명
			션을 사용하여 활동 스트림을 시작하면 이 객체에는 이 JSON 맵에서 반환되지 않는 다른 네이티브 엔진 감사 필드가 포함됩니다.
errorMessage	문자열	해당 사항 없음	이 필드는 Amazon RDS for SQL Server에서 사용되지 않으며 값은 null입니다.
exitCode	정수	sys.fn_get_audit_file.succeeded	이벤트를 시작한 작업의 성공 여부를 나타냅니다. 이 필드는 null일 수 없습니다. 로그인 이벤트를 제외한 모든 이벤트의 경우 이 필드는 권한 검사의 성공 또는 실패 여부를 보고하지만 작업의 성공 또는 실패 여부는 보고하지 않습니다. 값은 다음과 같습니다. • 0 - 실패 • 1 - 성공
logTime	문자열	sys.fn_get_audit_file.event_time	SQL Server에서 기록되는 이벤트 타임스탬프입니다.
netProtocol	문자열	해당 사항 없음	이 필드는 Amazon RDS for SQL Server에서 사용되지 않으며 값은 null입니다.
objectName	문자열	sys.fn_get_audit_file.object_name	SQL 문이 하나의 객체에서 작동하는 경우 데이터베이스 객체의 이름.

필드	데이터 형식	소스	설명
objectType	문자열	sys.fn_get_audit_file.class_type 이 sys.dm_audit_class_type_map.class_type_desc 에 매핑됨	SQL 문이 하나의 객체 유형에서 작동하는 경우 데이터베이스 객체의 유형.
paramList	문자열	해당 사항 없음	이 필드는 Amazon RDS for SQL Server에서 사용되지 않으며 값은 null입니다.
pid	정수	N/A	이 필드는 Amazon RDS for SQL Server에서 사용되지 않으며 값은 null입니다.
remoteHost	문자열	sys.fn_get_audit_file.client_ip	SQL 문을 실행한 클라이언트의 IP 주소 또는 호스트 이름(SQL Server 버전 14 이상). SQL Server 버전 13에서는 이 필드가 null입니다.
remotePort	정수	N/A	이 필드는 Amazon RDS for SQL Server에서 사용되지 않으며 값은 null입니다.
rowCount	정수	sys.fn_get_audit_file.affected_rows	SQL 문에 의해 영향을 받는 테이블 행 수(SQL Server 버전 14 이상). 이 필드는 SQL Server 버전 13에 있습니다.
serverHost	문자열	데이터베이스 호스트	호스트 데이터베이스 서버의 IP 주소.
serverType	문자열	해당 사항 없음	데이터베이스 서버 유형입니다. 이 값은 SQLSERVER 입니다.

필드	데이터 형식	소스	설명
serverVersion	문자열	데이터베이스 호스트	데이터베이스 서버 버전(예: SQL Server 2017의 경우 15.00.4073.23.v1.R1)
serviceName	문자열	데이터베이스 호스트	서비스의 이름입니다. 예시 값은 sqlserver-ee 입니다.
sessionId	정수	sys.fn_get_audit_file.session_id	세션의 고유 식별자.
startTime	문자열	해당 사항 없음	이 필드는 Amazon RDS for SQL Server에서 사용되지 않으며 값은 null입니다.
statementId	문자열	sys.fn_get_audit_file.sequence_group_id	클라이언트의 SQL 문에 대한 고유 식별자. 생성되는 이벤트마다 식별자가 다릅니다. 샘플 값은 0x38eaf4156267184094bb82071aaab644 입니다.
substatementId	정수	sys.fn_get_audit_file.sequence_number	문의 시퀀스 번호를 결정하는 식별자. 이 식별자는 대용량 레코드를 여러 레코드로 분할할 때 유용합니다.
transactionId	정수	sys.fn_get_audit_file.transaction_id	트랜잭션의 식별자. 활성 트랜잭션이 없으면 값은 0입니다.
type	문자열	데이터베이스 활동 스트림 생성됨	이벤트의 유형입니다. 값은 record 또는 heartbeat 입니다.

AWS SDK를 사용하여 데이터베이스 활동 스트림 처리

AWS SDK를 사용하여 프로그래밍 방식으로 활동 스트림을 처리할 수 있습니다. 다음은 제대로 작동하는 Java 및 Python 예시로, 인스턴스 기반 활성화를 위해 데이터베이스 활동 스트림을 사용하는 방법을 보여줍니다.

Java

```
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.Security;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.UUID;
import java.util.zip.GZIPInputStream;

import javax.crypto.Cipher;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.SecretKeySpec;

import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.encryptionsdk.AwsCrypto;
import com.amazonaws.encryptionsdk.CryptoInputStream;
import com.amazonaws.encryptionsdk.jce.JceMasterKey;
import
 com.amazonaws.services.kinesis.clientlibrary.exceptions.InvalidStateException;
import com.amazonaws.services.kinesis.clientlibrary.exceptions.ShutdownException;
import com.amazonaws.services.kinesis.clientlibrary.exceptions.ThrottlingException;
import com.amazonaws.services.kinesis.clientlibrary.interfaces.IRecordProcessor;
import
 com.amazonaws.services.kinesis.clientlibrary.interfaces.IRecordProcessorCheckpointer;
import
 com.amazonaws.services.kinesis.clientlibrary.interfaces.IRecordProcessorFactory;
import
 com.amazonaws.services.kinesis.clientlibrary.lib.worker.InitialPositionInStream;
```

```
import
 com.amazonaws.services.kinesis.clientlibrary.lib.worker.KinesisClientLibConfiguration;
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.ShutdownReason;
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker;
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker.Builder;
import com.amazonaws.services.kinesis.model.Record;
import com.amazonaws.services.kms.AWSKMS;
import com.amazonaws.services.kms.AWSKMSClientBuilder;
import com.amazonaws.services.kms.model.DecryptRequest;
import com.amazonaws.services.kms.model.DecryptResult;
import com.amazonaws.util.Base64;
import com.amazonaws.util.IOUtils;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.annotations.SerializedName;
import org.bouncycastle.jce.provider.BouncyCastleProvider;

public class DemoConsumer {

    private static final String STREAM_NAME = "aws-rds-das-[instance-external-
resource-id]"; // aws-rds-das-db-ABCD123456
    private static final String APPLICATION_NAME = "AnyApplication"; //unique
 application name for dynamo table generation that holds kinesis shard tracking
    private static final String AWS_ACCESS_KEY =
 "[AWS_ACCESS_KEY_TO_ACCESS_KINESIS]";
    private static final String AWS_SECRET_KEY =
 "[AWS_SECRET_KEY_TO_ACCESS_KINESIS]";
    private static final String RESOURCE_ID = "[external-resource-id]"; // db-
ABCD123456
    private static final String REGION_NAME = "[region-name]"; //us-east-1, us-
east-2...
    private static final BasicAWSCredentials CREDENTIALS = new
 BasicAWSCredentials(AWS_ACCESS_KEY, AWS_SECRET_KEY);
    private static final AWSStaticCredentialsProvider CREDENTIALS_PROVIDER = new
 AWSStaticCredentialsProvider(CREDENTIALS);

    private static final AwsCrypto CRYPTO = new AwsCrypto();
    private static final AWSKMS KMS = AWSKMSClientBuilder.standard()
            .withRegion(REGION_NAME)
            .withCredentials(CREDENTIALS_PROVIDER).build();

    class Activity {
        String type;
        String version;
```

```
        String databaseActivityEvents;
        String key;
    }

    class ActivityEvent {
        @SerializedName("class") String _class;
        String clientApplication;
        String command;
        String commandText;
        String databaseName;
        String dbProtocol;
        String dbUserName;
        String endTime;
        String errorMessage;
        String exitCode;
        String logTime;
        String netProtocol;
        String objectName;
        String objectType;
        List<String> paramList;
        String pid;
        String remoteHost;
        String remotePort;
        String rowCount;
        String serverHost;
        String serverType;
        String serverVersion;
        String serviceName;
        String sessionId;
        String startTime;
        String statementId;
        String substatementId;
        String transactionId;
        String type;
    }

    class ActivityRecords {
        String type;
        String clusterId; // note that clusterId will contain an empty string on RDS
Oracle and RDS SQL Server
        String instanceId;
        List<ActivityEvent> databaseActivityEventList;
    }
```

```
   static class RecordProcessorFactory implements IRecordProcessorFactory {
       @Override
       public IRecordProcessor createProcessor() {
           return new RecordProcessor();
       }
   }

   static class RecordProcessor implements IRecordProcessor {

       private static final long BACKOFF_TIME_IN_MILLIS = 3000L;
       private static final int PROCESSING_RETRIES_MAX = 10;
       private static final long CHECKPOINT_INTERVAL_MILLIS = 60000L;
       private static final Gson GSON = new
GsonBuilder().serializeNulls().create();

       private static final Cipher CIPHER;
       static {
           Security.insertProviderAt(new BouncyCastleProvider(), 1);
           try {
               CIPHER = Cipher.getInstance("AES/GCM/NoPadding", "BC");
           } catch (NoSuchAlgorithmException | NoSuchPaddingException |
NoSuchProviderException e) {
               throw new ExceptionInInitializerError(e);
           }
       }

       private long nextCheckpointTimeInMillis;

       @Override
       public void initialize(String shardId) {
       }

       @Override
       public void processRecords(final List<Record> records, final
IRecordProcessorCheckpointer checkpointer) {
           for (final Record record : records) {
               processSingleBlob(record.getData());
           }

           if (System.currentTimeMillis() > nextCheckpointTimeInMillis) {
               checkpoint(checkpointer);
               nextCheckpointTimeInMillis = System.currentTimeMillis() +
CHECKPOINT_INTERVAL_MILLIS;
           }
```

```
        }

        @Override
        public void shutdown(IRecordProcessorCheckpointer checkpointer,
ShutdownReason reason) {
            if (reason == ShutdownReason.TERMINATE) {
                checkpoint(checkpointer);
            }
        }

        private void processSingleBlob(final ByteBuffer bytes) {
            try {
                // JSON $Activity
                final Activity activity = GSON.fromJson(new String(bytes.array(),
StandardCharsets.UTF_8), Activity.class);

                // Base64.Decode
                final byte[] decoded =
Base64.decode(activity.databaseActivityEvents);
                final byte[] decodedDataKey = Base64.decode(activity.key);

                Map<String, String> context = new HashMap<>();
                context.put("aws:rds:db-id", RESOURCE_ID);

                // Decrypt
                final DecryptRequest decryptRequest = new DecryptRequest()

.withCiphertextBlob(ByteBuffer.wrap(decodedDataKey)).withEncryptionContext(context);
                final DecryptResult decryptResult = KMS.decrypt(decryptRequest);
                final byte[] decrypted = decrypt(decoded,
getByteArray(decryptResult.getPlaintext()));

                // GZip Decompress
                final byte[] decompressed = decompress(decrypted);
                // JSON $ActivityRecords
                final ActivityRecords activityRecords = GSON.fromJson(new
String(decompressed, StandardCharsets.UTF_8), ActivityRecords.class);

                // Iterate throught $ActivityEvents
                for (final ActivityEvent event :
activityRecords.databaseActivityEventList) {
                    System.out.println(GSON.toJson(event));
                }
            } catch (Exception e) {
```

```
                // Handle error.
                e.printStackTrace();
            }
        }

        private static byte[] decompress(final byte[] src) throws IOException {
            ByteArrayInputStream byteArrayInputStream = new
ByteArrayInputStream(src);
            GZIPInputStream gzipInputStream = new
GZIPInputStream(byteArrayInputStream);
            return IOUtils.toByteArray(gzipInputStream);
        }

        private void checkpoint(IRecordProcessorCheckpointer checkpointer) {
            for (int i = 0; i < PROCESSING_RETRIES_MAX; i++) {
                try {
                    checkpointer.checkpoint();
                    break;
                } catch (ShutdownException se) {
                    // Ignore checkpoint if the processor instance has been shutdown
(fail over).
                    System.out.println("Caught shutdown exception, skipping
checkpoint." + se);
                    break;
                } catch (ThrottlingException e) {
                    // Backoff and re-attempt checkpoint upon transient failures
                    if (i >= (PROCESSING_RETRIES_MAX - 1)) {
                        System.out.println("Checkpoint failed after " + (i + 1) +
"attempts." + e);
                        break;
                    } else {
                        System.out.println("Transient issue when checkpointing -
attempt " + (i + 1) + " of " + PROCESSING_RETRIES_MAX + e);
                    }
                } catch (InvalidStateException e) {
                    // This indicates an issue with the DynamoDB table (check for
table, provisioned IOPS).
                    System.out.println("Cannot save checkpoint to the DynamoDB table
used by the Amazon Kinesis Client Library." + e);
                    break;
                }
                try {
                    Thread.sleep(BACKOFF_TIME_IN_MILLIS);
                } catch (InterruptedException e) {
```

```
                    System.out.println("Interrupted sleep" + e);
                }
            }
        }
    }

    private static byte[] decrypt(final byte[] decoded, final byte[] decodedDataKey)
throws IOException {
        // Create a JCE master key provider using the random key and an AES-GCM
encryption algorithm
        final JceMasterKey masterKey = JceMasterKey.getInstance(new
SecretKeySpec(decodedDataKey, "AES"),
                "BC", "DataKey", "AES/GCM/NoPadding");
        try (final CryptoInputStream<JceMasterKey> decryptingStream =
CRYPTO.createDecryptingStream(masterKey, new ByteArrayInputStream(decoded));
             final ByteArrayOutputStream out = new ByteArrayOutputStream()) {
            IOUtils.copy(decryptingStream, out);
            return out.toByteArray();
        }
    }

    public static void main(String[] args) throws Exception {
        final String workerId = InetAddress.getLocalHost().getCanonicalHostName() +
":" + UUID.randomUUID();
        final KinesisClientLibConfiguration kinesisClientLibConfiguration =
                new KinesisClientLibConfiguration(APPLICATION_NAME, STREAM_NAME,
CREDENTIALS_PROVIDER, workerId);

kinesisClientLibConfiguration.withInitialPositionInStream(InitialPositionInStream.LATEST);
        kinesisClientLibConfiguration.withRegionName(REGION_NAME);
        final Worker worker = new Builder()
                .recordProcessorFactory(new RecordProcessorFactory())
                .config(kinesisClientLibConfiguration)
                .build();

        System.out.printf("Running %s to process stream %s as worker %s...\n",
APPLICATION_NAME, STREAM_NAME, workerId);

        try {
            worker.run();
        } catch (Throwable t) {
            System.err.println("Caught throwable while processing data.");
            t.printStackTrace();
            System.exit(1);
```

```
        }
        System.exit(0);
    }

    private static byte[] getByteArray(final ByteBuffer b) {
        byte[] byteArray = new byte[b.remaining()];
        b.get(byteArray);
        return byteArray;
    }
}
```

Python

```
import base64
import json
import zlib
import aws_encryption_sdk
from aws_encryption_sdk import CommitmentPolicy
from aws_encryption_sdk.internal.crypto import WrappingKey
from aws_encryption_sdk.key_providers.raw import RawMasterKeyProvider
from aws_encryption_sdk.identifiers import WrappingAlgorithm, EncryptionKeyType
import boto3

REGION_NAME = '<region>'                    # us-east-1
RESOURCE_ID = '<external-resource-id>'      # db-ABCD123456
STREAM_NAME = 'aws-rds-das-' + RESOURCE_ID  # aws-rds-das-db-ABCD123456

enc_client =
 aws_encryption_sdk.EncryptionSDKClient(commitment_policy=CommitmentPolicy.FORBID_ENCRYPT_AL

class MyRawMasterKeyProvider(RawMasterKeyProvider):
    provider_id = "BC"

    def __new__(cls, *args, **kwargs):
        obj = super(RawMasterKeyProvider, cls).__new__(cls)
        return obj

    def __init__(self, plain_key):
        RawMasterKeyProvider.__init__(self)
        self.wrapping_key =
 WrappingKey(wrapping_algorithm=WrappingAlgorithm.AES_256_GCM_IV12_TAG16_NO_PADDING,
                                        wrapping_key=plain_key,
 wrapping_key_type=EncryptionKeyType.SYMMETRIC)
```

```
    def _get_raw_key(self, key_id):
        return self.wrapping_key


def decrypt_payload(payload, data_key):
    my_key_provider = MyRawMasterKeyProvider(data_key)
    my_key_provider.add_master_key("DataKey")
    decrypted_plaintext, header = enc_client.decrypt(
        source=payload,

 materials_manager=aws_encryption_sdk.materials_managers.default.DefaultCryptoMaterialsManag
    return decrypted_plaintext


def decrypt_decompress(payload, key):
    decrypted = decrypt_payload(payload, key)
    return zlib.decompress(decrypted, zlib.MAX_WBITS + 16)


def main():
    session = boto3.session.Session()
    kms = session.client('kms', region_name=REGION_NAME)
    kinesis = session.client('kinesis', region_name=REGION_NAME)

    response = kinesis.describe_stream(StreamName=STREAM_NAME)
    shard_iters = []
    for shard in response['StreamDescription']['Shards']:
        shard_iter_response = kinesis.get_shard_iterator(StreamName=STREAM_NAME,
 ShardId=shard['ShardId'],

 ShardIteratorType='LATEST')
        shard_iters.append(shard_iter_response['ShardIterator'])

    while len(shard_iters) > 0:
        next_shard_iters = []
        for shard_iter in shard_iters:
            response = kinesis.get_records(ShardIterator=shard_iter, Limit=10000)
            for record in response['Records']:
                record_data = record['Data']
                record_data = json.loads(record_data)
                payload_decoded =
 base64.b64decode(record_data['databaseActivityEvents'])
                data_key_decoded = base64.b64decode(record_data['key'])
```

```
                data_key_decrypt_result =
 kms.decrypt(CiphertextBlob=data_key_decoded,

 EncryptionContext={'aws:rds:db-id': RESOURCE_ID})
                print (decrypt_decompress(payload_decoded,
 data_key_decrypt_result['Plaintext']))
            if 'NextShardIterator' in response:
                next_shard_iters.append(response['NextShardIterator'])
        shard_iters = next_shard_iters


if __name__ == '__main__':
    main()
```

데이터베이스 활동 스트림의 IAM 정책 예제

데이터베이스 활동 스트림에 대한 적절한 AWS Identity and Access Management(IAM) 역할 권한이 있는 사용자는 DB인스턴스에 대한 활동 스트림 설정을 작성, 시작, 중지하고 수정할 수 있습니다. 이러한 작업은 스트림의 감사 로그에 포함됩니다. 규정을 준수하는 최선의 방법은 DBA에 이러한 권한을 제공하지 않는 것입니다.

IAM 정책을 사용하여 데이터베이스 활동 스트림에 대한 액세스를 설정합니다. Amazon RDS 인증에 대한 자세한 내용은 Amazon RDS의 자격 증명 및 액세스 관리 섹션을 참조하세요. IAM 정책 생성에 대한 자세한 내용은 IAM 데이터베이스 액세스를 위한 IAM 정책 생성 및 사용 단원을 참조하십시오.

Example 데이터베이스 활동 스트림 구성을 허용하는 정책

사용자에게 활동 스트림을 수정할 수 있는 세분화된 액세스를 제공하려면 IAM 정책에서 서비스별 작업 컨텍스트 키 `rds:StartActivityStream` 및 `rds:StopActivityStream`을 사용하세요. 다음 IAM 정책 예제는 사용자 또는 역할이 활동 스트림을 구성할 수 있도록 허용합니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
        {
            "Sid":"ConfigureActivityStreams",
            "Effect":"Allow",
            "Action": [
                "rds:StartActivityStream",
                "rds:StopActivityStream"
            ],
```

```
            "Resource":"*",
          }
     ]
}
```

Example 데이터베이스 활동 스트림 시작을 허용하는 정책

다음 IAM 정책 예제는 사용자 또는 역할이 활동 스트림을 시작할 수 있도록 허용합니다.

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"AllowStartActivityStreams",
            "Effect":"Allow",
            "Action":"rds:StartActivityStream",
            "Resource":"*"
        }
    ]
}
```

Example 데이터베이스 활동 스트림 중지를 허용하는 정책

다음 IAM 정책 예제는 사용자 또는 역할이 활동 스트림을 중지할 수 있도록 허용합니다.

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"AllowStopActivityStreams",
            "Effect":"Allow",
            "Action":"rds:StopActivityStream",
            "Resource":"*"
        }
     ]
}
```

Example 데이터베이스 활동 스트림 시작을 거부하는 정책

다음 IAM 정책 예제는 사용자 또는 역할이 활동 스트림을 시작하지 못하게 합니다.

```
{
```

```
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"DenyStartActivityStreams",
            "Effect":"Deny",
            "Action":"rds:StartActivityStream",
            "Resource":"*"
        }
     ]
}
```

Example 데이터베이스 활동 스트림 중지를 거부하는 정책

다음 IAM 정책 예제는 사용자 또는 역할이 활동 스트림을 중지하지 못하게 합니다.

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"DenyStopActivityStreams",
            "Effect":"Deny",
            "Action":"rds:StopActivityStream",
            "Resource":"*"
        }
    ]
}
```

Amazon GuardDuty RDS Protection을 이용한 위협 모니터링

Amazon GuardDuty는 AWS 환경 내 계정, 컨테이너, 워크로드 및 데이터를 보호하는 데 도움이 되는 위협 감지 서비스입니다. GuardDuty는 기계 학습(ML) 모델, 이상 및 위협 감지 기능을 통해 다양한 로그 소스와 런타임 활동을 지속적으로 모니터링하여 사용자 환경의 잠재적 보안 위험과 악의적 활동을 식별하고 우선순위를 지정합니다.

GuardDuty RDS Protection은 Amazon RDS 데이터베이스에 대한 잠재적 액세스 위협 관련 로그인 이벤트를 분석하고 프로파일링합니다. RDS Protection을 켜면 GuardDuty는 RDS 데이터베이스에서 RDS 로그인 이벤트를 사용합니다. RDS Protection은 이러한 이벤트를 모니터링하고 잠재적인 내부자 위협 또는 외부 행위자가 있는지 프로파일링합니다.

GuardDuty RDS Protection 활성화에 대한 자세한 내용은 Amazon GuardDuty 사용 설명서의 GuardDuty RDS Protection을 참조하세요.

RDS Protection이 성공하거나 실패한 로그인 시도에서의 비정상적인 패턴과 같은 잠재적 위협을 탐지하면, GuardDuty는 손상되었을지도 모르는 데이터베이스 관련 세부 정보가 포함된 새 탐지 결과를 생성합니다. Amazon GuardDuty 콘솔의 결과 요약 섹션에서 결과 세부 정보를 볼 수 있습니다. 결과 세부 정보는 결과 유형에 따라 달라집니다. 기본 세부 정보, 리소스 유형 및 리소스 역할에 따라 결과에 사용할 수 있는 정보의 종류가 결정됩니다. 일반적으로 사용할 수 있는 검색 결과 세부 정보 및 검색 유형에 대한 자세한 내용은 Amazon GuardDuty 사용 설명서에서 Finding details(결과 세부 정보) 및 GuardDuty RDS Protection finding types(GuardDuty RDS Protection 결과 유형)를 각각 참조하세요.

RDS Protection 기능은 이 기능을 사용할 수 있는 모든 AWS 리전의 AWS 계정에 대해 켜거나 해제할 수 있습니다. RDS Protection이 활성화되지 않은 경우 GuardDuty는 손상되었을지도 모르는 RDS 데이터베이스를 탐지하거나 보안 침해 관련 세부 정보를 제공하지 않습니다.

기존 GuardDuty 계정은 30일 평가판 기간으로 RDS Protection을 활성화할 수 있습니다. 새 GuardDuty 계정의 경우 RDS Protection이 이미 활성화되어 있으며 30일 무료 평가판 기간에 포함되어 있습니다. 자세한 내용은 Amazon GuardDuty 사용 설명서의 Estimating GuardDuty cost(GuardDuty 비용 추정)를 참조하세요.

GuardDuty가 아직 RDS Protection을 지원하지 않는 AWS 리전에 대한 자세한 내용은 Amazon GuardDuty 사용 설명서의 Region-specific feature availability를 참조하세요.

GuardDuty RDS 보호가 지원하는 RDS 데이터베이스 버전에 대한 자세한 내용은 Amazon GuardDuty 사용 설명서에 나와 있는 Supported Amazon Aurora, Amazon RDS, and Aurora Limitless databases를 참조하시기 바랍니다.

Amazon RDS Custom

Amazon RDS Custom은 데이터베이스 관리 작업 및 운영을 자동화합니다. RDS Custom은 데이터베이스 관리자가 데이터베이스 환경 및 운영 체제에 액세스하고 사용자 지정할 수 있도록 합니다. RDS Custom을 사용하면 레거시, 커스텀 및 패키지 애플리케이션의 요구 사항에 맞게 커스터마이징할 수 있습니다.

RDS Custom에 대한 최신 웹 세미나 및 블로그는 Amazon RDS Custom 리소스를 참조하세요.

주제

- 데이터베이스 커스터마이징의 문제 해결
- Amazon RDS Custom을 위한 관리 모델 및 이점
- Amazon RDS Custom 아키텍처
- Amazon RDS Custom의 보안
- RDS Custom for Oracle 작업
- RDS Custom for SQL Server 작업

데이터베이스 커스터마이징의 문제 해결

Amazon RDS Custom은 타사 애플리케이션에 필요한 커스텀으로 인해 완전관리형 서비스로 쉽게 이동할 수 없는 시장에 Amazon RDS의 이점을 제공합니다. Amazon RDS Custom은 관리 시간을 절약하고 내구성이 뛰어나며 비즈니스에 맞게 확장할 수 있습니다.

전체 데이터베이스 및 운영 체제를 AWS로 완전 관리해야 하는 경우 Amazon RDS를 사용하는 것이 좋습니다. 종속 애플리케이션을 사용할 수 있도록 데이터베이스 및 기본 운영 체제에 대한 관리 권한이 필요한 경우 Amazon RDS Custom 이 더 나은 선택입니다. 완전한 관리 책임을 원하고 관리형 컴퓨팅 서비스가 필요한 경우 Amazon EC2 상용 데이터베이스를 자체 관리하는 것이 가장 좋습니다.

관리형 서비스 환경을 제공하기 위해 Amazon RDS는 기본 호스트에 액세스할 수 없습니다. 또한 Amazon RDS는 고급 권한이 필요한 특정 시스템 프로시저와 테이블에 대한 액세스를 제한합니다. 그러나 일부 애플리케이션의 경우 권한이 있는 운영 시스템(OS) 사용자로 작업을 수행해야 할 수 있습니다.

예를 들어, 다음 작업 중 일부를 수행해야 할 수 있습니다.

- 사용자 지정 데이터베이스 및 OS 패치 및 패키지를 설치합니다.

- 특정 데이터베이스 설정을 구성합니다.
- 애플리케이션과 직접 파일을 공유하도록 파일 시스템을 구성합니다.

이전에는 애플리케이션을 커스터마이징해야 하는 경우 온프레미스 또는 Amazon EC2 데이터베이스를 배포해야 했습니다. 이 경우 다음 테이블에 요약된 대로 데이터베이스 관리에 대한 책임의 대부분 또는 전부를 부담합니다.

Feature	온프레미스 책임	Amazon EC2 책임	Amazon RDS 책임
애플리케이션 최적화	고객	고객	고객
확장성	고객	고객	AWS
높은 가용성	고객	고객	AWS
데이터베이스 백업	고객	고객	AWS
데이터베이스 소프트웨어 패치	고객	고객	AWS
데이터베이스 소프트웨어 설치	고객	고객	AWS
OS 패치	고객	고객	AWS
OS 설치	고객	고객	AWS
서버 유지 관리	고객	AWS	AWS
하드웨어 수명	고객	AWS	AWS
전력, 네트워크 및 냉각	고객	AWS	AWS

데이터베이스 소프트웨어를 직접 관리하면 더 많은 제어 권한을 얻을 수 있지만 사용자 오류도 발생하기 쉽습니다. 예를 들어 수동으로 변경하면 실수로 애플리리케이션 다운타임이 발생할 수 있습니다. 모든 변경 사항을 확인하여 문제를 파악하고 수정하려면 몇 시간이 걸리기도 합니다. 일반적인 DBA 작업

을 자동화하고 데이터베이스 및 기본 운영 체제에 대한 권한 있는 액세스도 지원하는 관리형 데이터베이스 서비스가 필요할 것입니다.

Amazon RDS Custom을 위한 관리 모델 및 이점

Amazon RDS Custom은 기본 운영 체제 및 데이터베이스 환경에 액세스해야 하는 레거시, 커스텀 및 패키지 애플리케이션을 위한 관리형 데이터베이스 서비스입니다. RDS Custom은 데이터베이스 및 기본 운영 체제에 대한 액세스 권한을 부여하는 동안 AWS 클라우드의 데이터베이스 설정, 운영 및 확장을 자동화합니다. 이 액세스를 통해 설정을 구성하고 패치를 설치하며 종속 애플리케이션의 요구 사항을 충족하도록 기본 기능을 활성화할 수 있습니다. RDS Custom을 사용하면 AWS Management Console 또는 AWS CLI를 사용하여 데이터베이스 워크로드를 실행할 수 있습니다.

RDS Custom은 Oracle 데이터베이스 및 Microsoft SQL Server DB 엔진만 지원합니다.

주제

- RDS Custom의 공동 책임 모델
- RDS Custom에서 지원 범위 및 지원하지 않는 구성
- RDS Custom의 주요 이점

RDS Custom의 공동 책임 모델

RDS Custom을 사용하면 Amazon RDS의 관리형 기능을 사용하지만 Amazon EC2에서처럼 호스트를 관리하고 OS를 사용자 지정할 수 있습니다. Amazon RDS에서 수행하는 작업 외에 추가로 데이터베이스 관리 책임을 부여받게 됩니다. 따라서 Amazon RDS에서보다 데이터베이스 및 DB 인스턴스 관리를 더 잘 제어하면서도 RDS 자동화의 혜택을 누릴 수 있습니다.

공동 책임은 다음을 의미합니다.

1. RDS Custom 기능을 사용할 때 프로세스의 일부를 소유합니다.

   예를 들어 RDS Custom for Oracle에서는 어떤 Oracle 데이터베이스 패치를 사용하고 언제 DB 인스턴스에 적용할지 제어할 수 있습니다.
2. 사용자는 RDS Custom 기능 관련 모든 사용자 지정 기능이 제대로 작동하는지 직접 확인해야 합니다.

   잘못된 사용자 지정을 방지하도록 지원하고자 RDS Custom은 DB 인스턴스 이외에서 실행되는 자동화 소프트웨어를 갖추고 있습니다. 기본 Amazon EC2 인스턴스가 손상되면 RDS Custom은 EC2

인스턴스를 재부팅하거나 교체하여 이러한 문제를 자동 해결하기 위해 시도합니다. 사용자가 인식할 수 있는 유일한 변경 사항은 새 IP 주소입니다. 자세한 내용은 Amazon RDS Custom 호스트 교체 섹션을 참조하세요.

다음 테이블은 RDS Custom 공동 책임 모델의 다양한 기능을 자세히 보여줍니다.

Feature	Amazon EC2 책임	Amazon RDS 책임	RDS Custom for Oracle 책임	RDS Custom for SQL Server 책임
애플리케이션 최적화	고객	고객	고객	고객
확장성	고객	AWS	공유	공유
높은 가용성	고객	AWS	AWS	AWS
데이터베이스 백업	고객	AWS	공유	AWS
데이터베이스 소프트웨어 패치	고객	AWS	공유	RPEV의 경우 AWS, CEV[1]의 경우 고객
데이터베이스 소프트웨어 설치	고객	AWS	공유	RPEV의 경우 AWS, CEV[1]의 경우 고객
OS 패치	고객	AWS	Customer	RPEV의 경우 AWS, CEV[1]의 경우 고객
OS 설치	고객	AWS	공유	AWS
서버 유지 관리	AWS	AWS	AWS	AWS
하드웨어 수명	AWS	AWS	AWS	AWS
전력, 네트워크 및 쿨링	AWS	AWS	AWS	AWS

[1] 커스텀 엔진 버전(CEV)은 데이터베이스 버전과 Amazon Machine Image(AMI)의 바이너리 볼륨 스냅샷입니다. RDS에서 제공하는 엔진 버전(RPEV)은 기본 Amazon Machine Image(AMI)와 Microsoft SQL Server 설치 환경입니다.

Microsoft SQL Server를 사용하여 RDS Custom DB 인스턴스를 만들 수 있습니다. 이 경우

- 라이선스 포함(LI) 및 기존 보유 미디어 사용(BYOM)이라는 두 가지 라이선스 모델 중에서 선택할 수 있습니다.
- LI의 경우 SQL Server 라이선스를 별도로 구매할 필요가 없습니다. AWS는 SQL Server 데이터베이스 소프트웨어에 대한 라이선스를 보유합니다.
- BYOM의 경우 고유한 Microsoft SQL Server 바이너리와 라이선스를 제공하고 설치할 수 있습니다.

Oracle 데이터베이스를 사용하여 RDS 커스텀 DB 인스턴스를 생성할 수 있습니다. 이 경우 다음과 같이 합니다.

- 자체 미디어를 관리하세요.

  RDS Custom을 사용하는 경우 자체 데이터베이스 설치 파일 및 패치를 업로드합니다. 이러한 파일에서 커스텀 엔진 버전(CEV)을 생성합니다. 그런 다음 이 CEV를 사용하여 RDS 커스텀 DB 인스턴스를 생성할 수 있습니다.
- 자체 라이선스를 관리합니다.

  자체 Oracle 데이터베이스 라이센스를 가져오고 라이센스를 직접 관리할 수 있습니다.

RDS Custom에서 지원 범위 및 지원하지 않는 구성

RDS Custom은 지원 범위라는 모니터링 기능을 제공합니다. 이 기능을 사용하면 호스트 및 데이터베이스 환경을 올바르게 구성할 수 있습니다. DB 인스턴스가 지원 범위를 벗어나도록 변경하는 경우 RDS Custom은 구성 문제가 수동으로 해결될 때까지 인스턴스 상태를 `unsupported-configuration`으로 변경합니다. 자세한 내용은 RDS Custom 지원 범위 섹션을 참조하세요.

RDS Custom의 주요 이점

RDS Custom을 사용하여 다음 작업을 할 수 있습니다.

- 다음을 포함하는 Amazon RDS와 동일한 많은 수의 관리 작업을 자동화합니다.
  - 데이터베이스의 수명 주기 관리

  - 자동 백업 및 특정 시점 복구(PITR)
  - RDS Custom DB 인스턴스의 상태 모니터링 및 인프라, 운영 체제 및 데이터베이스 프로세스의 변경 사항 관찰
  - DB 인스턴스 중단에 따른 문제 해결을 위한 알림 또는 조치
- 서드 파티 애플리케이션을 설치합니다.

  소프트웨어를 설치하여 커스텀 애플리케이션 및 에이전트를 실행할 수 있습니다. 호스트에 대한 액세스 권한이 있으므로 레거시 애플리케이션을 지원하도록 파일 시스템을 수정할 수 있습니다.
- 커스텀 패치를 설치합니다.

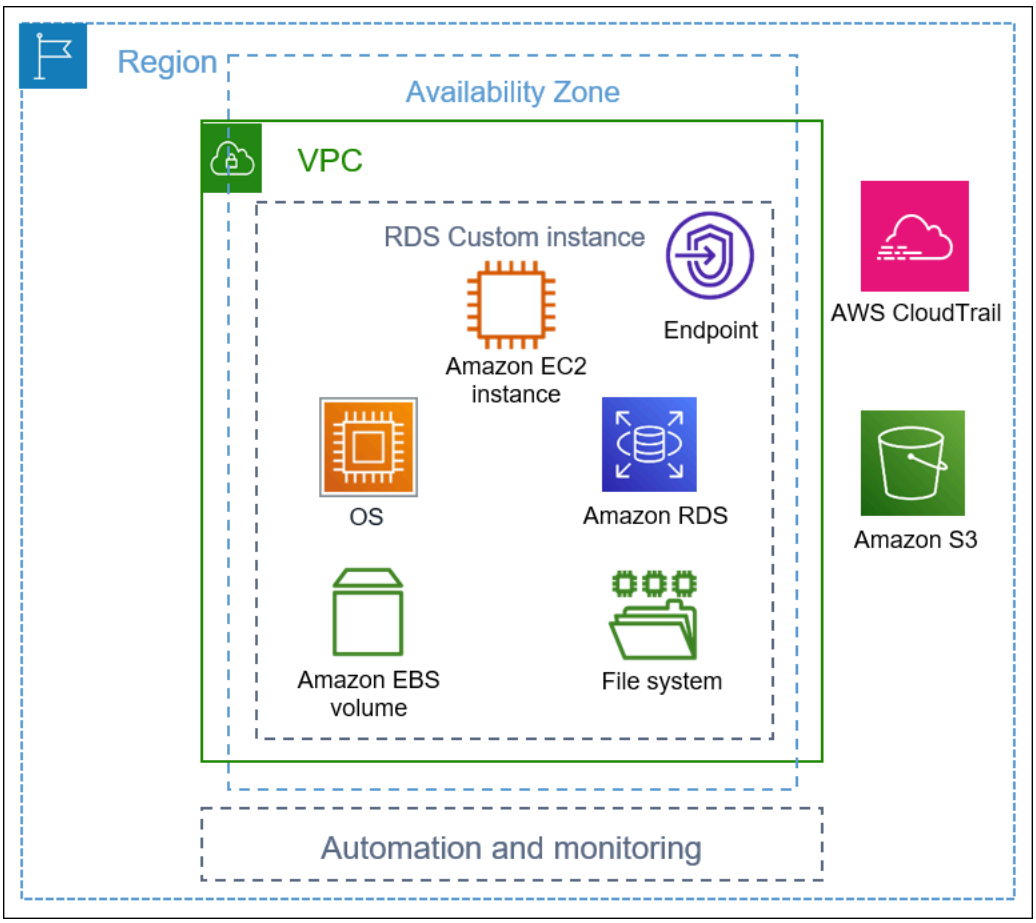
  RDS Custom DB 인스턴스에서 커스텀 데이터베이스 패치를 적용하거나 OS 패키지를 수정할 수 있습니다.
- 온프레미스 데이터베이스를 완전 관리형 서비스로 이동하기 전에 스테이징합니다.

  자체 온프레미스 데이터베이스를 관리하는 경우 데이터베이스를 있는 그대로 RDS Custom으로 스테이징할 수 있습니다. 클라우드 환경에 익숙해지면 데이터베이스를 완전관리형 Amazon RDS DB 인스턴스로 마이그레이션할 수 있습니다.
- 자체 자동화를 생성합니다.

  보고, 관리 또는 진단 도구를 위한 사용자 지정 자동화 스크립트를 생성, 예약 및 실행할 수 있습니다.

Amazon RDS Custom 아키텍처

Amazon RDS Custom 아키텍처는 중요한 차이점이 있는 Amazon RDS를 기반으로 합니다. 다음 다이어그램은 RDS Custom 아키텍처의 주요 구성 요소를 보여줍니다.



주제

- VPC
- RDS Custom 자동화 및 모니터링
- Amazon S3
- AWS CloudTrail

VPC

Amazon RDS와 마찬가지로, RDS Custom DB 인스턴스는 Virtual Private Cloud(VPC)에 있습니다.



RDS Custom DB 인스턴스의 주요 구성 요소는 다음과 같습니다.

- Amazon EC2 인스턴스
- 인스턴스 엔드포인트
- Amazon EC2 인스턴스에 설치된 운영 체제
- 추가 파일 시스템을 포함하는 Amazon EBS 스토리지

RDS Custom 자동화 및 모니터링

RDS Custom에는 DB 인스턴스 외부에서 실행되는 자동화 소프트웨어가 있습니다. 이 소프트웨어는 DB 인스턴스의 에이전트 및 전체 RDS Custom 환경 내의 다른 구성 요소와 커뮤니케이션합니다.

RDS Custom 모니터링 및 복구 기능은 Amazon RDS와 유사한 기능을 제공합니다. 기본적으로 RDS Custom은 전체 자동화 모드에 있습니다. 자동화 소프트웨어에는 다음과 같은 주요 책임이 있습니다.

- 지표 수집 및 알림 전송
- 자동 인스턴스 복구 수행

RDS Custom 자동화의 중요한 책임은 Amazon EC2 인스턴스의 문제에 대응하는 것입니다. 여러 가지 이유로 호스트가 손상되거나 도달하지 못할 수 있습니다. RDS Custom은 Amazon EC2 인스턴스를 재부팅하거나 교체하여 이러한 문제를 해결합니다.

주제

- Amazon RDS Custom 호스트 교체

- RDS Custom 지원 범위

Amazon RDS Custom 호스트 교체

Amazon EC2 호스트가 손상되면 RDS Custom이 호스트를 재부팅하려고 시도합니다. 이 작업이 실패하면 RDS Custom은 Amazon EC2에 포함되어 있는 동일한 중지 및 시작 기능을 사용합니다. 호스트를 교체할 때 고객이 볼 수 있는 유일한 변경 사항은 새로운 공용 IP 주소입니다.

주제

- 호스트 중지 및 시작
- 호스트 교체의 영향
- Amazon EC2 호스트 모범 사례

호스트 중지 및 시작

RDS Custom은 사용자 개입 없이 다음 단계를 자동으로 수행합니다.

1. Amazon EC2 호스트를 중지합니다.

   EC2 인스턴스는 정상적인 종료를 수행하고 실행을 중지합니다. 모든 Amazon EBS 볼륨이 인스턴스에 연결된 상태로 유지되고 해당 데이터도 남습니다. 호스트 컴퓨터의 인스턴스 스토리지 볼륨(RDS Custom에서 지원되지 않음) 또는 RAM에 저장된 모든 데이터가 사라집니다.

   자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 중지 및 시작을 참조하세요.
2. Amazon EC2 호스트를 시작합니다.

   EC2 인스턴스는 새로운 기본 호스트 하드웨어로 마이그레이션됩니다. 경우에 따라 RDS Custom DB 인스턴스가 원래 호스트에 남아 있기도 합니다.

호스트 교체의 영향

RDS Custom에서는 루트 디바이스 볼륨 및 Amazon EBS 스토리지 볼륨을 완벽하게 제어할 수 있습니다. 루트 볼륨에는 유실되어서는 안 될 중요한 데이터 및 구성이 포함될 수 있습니다.

Oracle용 RDS Custom은 루트 볼륨 데이터를 포함한 모든 데이터베이스 및 고객 데이터를 작업 후에도 유지합니다. 따라서 사용자가 별다른 조치를 취할 필요가 없습니다. SQL Server용 RDS Custom에서 데이터베이스 데이터는 유지되지만, 운영 체제 및 고객 데이터를 포함한 `C:` 드라이브의 모든 데이터는 손실됩니다.

호스트 교체 프로세스가 끝나면 Amazon EC2 호스트가 새로운 퍼블릭 IP 주소를 갖게 됩니다. 새 호스트에는 다음 표에 나온 데이터와 메타데이터가 유지됩니다.

메타데이터 또는 데이터	RDS Custom for Oracle 보존	RDS Custom for SQL Server 보존
EC2 인스턴스	예	아니요
EC2 인스턴스 메타데이터	예	예
데이터 스토리지 볼륨 데이터	예	예
루트 볼륨 데이터	예	아니요
프라이빗 IP 주소	예	예
탄력적 IP 주소	예	예

Amazon EC2 호스트 모범 사례

Amazon EC2 호스트 교체 기능은 Amazon EC2가 손상되는 대부분의 상황에서 사용할 수 있습니다. 다음 모범 사례를 따르는 것이 좋습니다.

- 구성 또는 운영 체제를 변경하기 전에 데이터를 백업합니다. 루트 볼륨 또는 운영 체제가 손상되면 호스트 교체로 복구할 수 없습니다. 이 경우 유일한 옵션은 DB 스냅샷 또는 특정 시점으로 복구에서 복원하는 것입니다.
- 물리적 Amazon EC2 호스트를 수동으로 중지하거나 종료하지 마세요. 이렇게 하면 인스턴스가 RDS Custom 지원 경계 밖에 배치됩니다.
- (RDS Custom for SQL Server) 추가 볼륨을 Amazon EC2 호스트에 연결하는 경우 재시작할 때 다시 탑재하도록 구성합니다. 호스트가 손상된 경우 RDS Custom이 호스트를 자동으로 중지하고 시작할 수 있습니다.

RDS Custom 지원 범위

RDS Custom은 지원 범위라는 모니터링 기능을 제공합니다. 이 추가적인 모니터링은 RDS Custom DB 인스턴스가 지원되는 AWS 인프라, 운영 체제 및 데이터베이스를 사용하도록 보장합니다.

지원 범위에서는 DB 인스턴스가 RDS Custom for Oracle에서 지원되지 않는 구성 문제 해결 및 RDS Custom for SQL Server에서 지원되지 않는 구성 문제 해결에 나열된 요구 사항을 준수하는지 확인합니다. RDS Custom은 이러한 요구 사항 중 하나라도 충족되지 않으면 DB 인스턴스가 지원 경계를 벗어난 것으로 간주합니다.

주제

- RDS Custom에서 지원되지 않는 구성
- 지원되지 않는 구성 문제 해결

RDS Custom에서 지원되지 않는 구성

DB 인스턴스가 지원 경계를 벗어나면 RDS Custom은 DB 인스턴스 상태를 `unsupported-configuration`으로 변경하고 이벤트 알림을 전송합니다. 구성 문제가 해결되면 RDS Custom에서는 DB 인스턴스 상태를 `available`로 다시 변경합니다.

DB 인스턴스가 `unsupported-configuration` 상태인 동안에는 다음과 같은 상황이 발생할 수 있습니다.

- 데이터베이스가 연결 가능한 상태가 됩니다. DB 인스턴스가 `unsupported-configuration` 상태이면 데이터베이스가 예기치 않게 종료되기 때문에 예외 상황도 있습니다.
- DB 인스턴스를 수정할 수 없습니다.
- DB 스냅샷을 생성할 수 없습니다.
- 자동 백업이 생성되지 않습니다.
- RDS Custom for SQL Server DB 인스턴스 전용의 경우, RDS Custom은 기본 Amazon EC2 인스턴스가 손상된 경우 이를 교체하지 않습니다. 호스트 교체에 관한 자세한 내용은 Amazon RDS Custom 호스트 교체 단원을 참조하세요.
- DB 인스턴스를 삭제할 수는 있지만 대부분의 다른 RDS Custom API 작업은 사용할 수 없습니다.
- RDS Custom은 다시 실행 로그 파일을 보관하고 Amazon S3에 이를 업로드하여 시점 복구(PITR)를 계속 지원합니다. `unsupported-configuration` 상태에 있는 PITR은 다음의 부분에서 차이가 있습니다.
  - PITR이 새로운 RDS Custom DB 인스턴스로 완전히 복원하는 데 시간이 오래 걸릴 수 있는데, 이러한 상황은 인스턴스가 `unsupported-configuration` 상태인 동안에는 자동 스냅샷이나 수동 스냅샷을 생성할 수 없기 때문에 발생합니다.
  - PITR은 인스턴스가 `unsupported-configuration` 상태로 전환되기 전에 가장 최근에 생성된 스냅샷부터 시작해서 다시 실행 로그를 더 많이 재생해야 합니다.

- 보관된 다시 실행 파일을 업로드하지 못하도록 변경했기 때문에 DB 인스턴스가 `unsupported-configuration` 상태에 있는 경우도 있습니다. EC2 인스턴스 중지, RDS Custom 에이전트 중지, EBS 볼륨 분리 등을 예로 들 수 있습니다. 이 경우 PITR을 통해 DB 인스턴스를 복원 가능한 가장 빠른 시간으로 복원하지 못할 수 있습니다.

지원되지 않는 구성 문제 해결

RDS Custom은 `unsupported-configuration` 상태 관련 문제 해결 지침을 제공합니다. RDS Custom for Oracle 및 RDS Custom for SQL Server에 적용되는 일부 지침이 있긴 하지만, DB 엔진에 따라 달라지는 지침도 있습니다. 엔진별 문제 해결 정보는 다음 주제를 참조하세요.

- RDS Custom for Oracle에서 지원되지 않는 구성 문제 해결
- RDS Custom for SQL Server에서 지원되지 않는 구성 문제 해결

Amazon S3

Oracle용 RDS Custom을 사용하는 경우 설치 미디어를 사용자가 생성한 Amazon S3 버킷에 업로드합니다. Oracle용 RDS Custom은 이 버킷의 미디어를 사용하여 사용자 지정 엔진 버전(CEV)을 생성합니다. CEV는 데이터베이스 버전과 Amazon Machine Image(AMI)의 이진 볼륨 스냅샷입니다. CEV에서 RDS Custom DB 인스턴스를 생성할 수 있습니다. 자세한 정보는 사용자 지정 엔진 버전의 Amazon RDS Custom for Oracle으로 작업을 참조하세요.

RDS Custom for Oracle 및 RDS Custom for SQL Server 모두에 대해 RDS Custom은 문자열 `do-not-delete-rds-custom-` 접두사가 붙은 Amazon S3 버킷을 자동으로 생성합니다. RDS Custom은 `do-not-delete-rds-custom-` S3 버킷을 사용하여 다음 유형의 파일을 저장합니다.

- RDS Custom에서 생성한 추적에 대한 AWS CloudTrail 로그
- 주변 아티팩트 지원(RDS Custom 지원 범위 참조)
- 데이터베이스 다시 실행 로그 파일(RDS Custom for Oracle만 해당)
- 트랜잭션 로그(RDS Custom for SQL Server만 해당)
- Custom 엔진 버전 아티팩트(RDS Custom for Oracle만 해당)

RDS Custom은 다음 리소스 중 하나를 생성할 때 `do-not-delete-rds-custom-` S3 버킷을 생성합니다.

- RDS Custom for Oracle을 위한 첫 번째 CEV

- RDS Custom for SQL Server의 첫 번째 DB 인스턴스

RDS Custom은 다음의 조합마다 하나의 버킷을 생성합니다.

- AWS 계정 ID
- 엔진 유형(RDS Custom for Oracle 또는 RDS Custom for SQL Server)
- AWS 리전

예를 들어 단일 AWS 리전에서 Oracle CEV용 RDS Custom을 생성하는 경우 하나의 `do-not-delete-rds-custom-` 버킷이 존재합니다. 여러 RDS Custom for SQL Server 인스턴스를 생성하고 다른 AWS 리전에 상주하는 경우 각 AWS 리전에 하나의 `do-not-delete-rds-custom-` 버킷이 있습니다. 단일 AWS 리전에 RDS Custom for Oracle 인스턴스 1개와 RDS Custom for SQL Server 인스턴스 2개를 생성하는 경우 `do-not-delete-rds-custom-` 버킷이 2개 존재합니다.

AWS CloudTrail

RDS Custom은 이름이 `do-not-delete-rds-custom-`으로 시작하는 AWS CloudTrail 추적을 자동으로 생성합니다. RDS Custom 지원 경계는 CloudTrail의 이벤트에 따라 작업이 RDS Custom 자동화에 영향을 미치는지 여부를 결정합니다. 자세한 내용은 지원되지 않는 구성 문제 해결 섹션을 참조하세요.

RDS Custom은 첫 번째 DB 인스턴스를 생성할 때 추적을 생성합니다. RDS Custom은 다음의 조합마다 하나의 추적을 생성합니다.

- AWS 계정 ID
- 엔진 유형(RDS Custom for Oracle 또는 RDS Custom for SQL Server)
- AWS 리전

RDS Custom DB 인스턴스를 삭제해도 이 인스턴스의 CloudTrail은 자동으로 제거되지 않습니다. 이 경우, 삭제되지 않은 CloudTrail에 대해 AWS 계정에 계속 요금이 청구됩니다. RDS Custom은 이 리소스의 삭제에 대해 책임을 지지 않습니다. CloudTrail을 수동으로 제거하는 방법을 알아보려면 AWS CloudTrail 사용 설명서에서 추적 삭제를 참조하세요.

Amazon RDS Custom의 보안

RDS Custom의 보안 고려 사항을 숙지하세요.

RDS Custom의 보안에 대한 자세한 내용은 다음 주제를 참조하세요.

- Amazon S3 버킷 보안으로 혼동된 대리자 문제 방지
- 규정 준수 프로그램을 위한 RDS Custom for Oracle 자격 증명

RDS Custom이 사용자를 대신하여 작업을 안전하게 관리하는 방법

RDS Custom은 다음과 같은 도구와 기법을 사용하여 사용자를 대신하여 안전하게 작업을 실행합니다.

AWSServiceRoleForRDSCustom 서비스 연결 역할

서비스 연결 역할은 서비스에서 사전 정의하며, 사용자를 대신하여 다른 AWS 서비스를 자동으로 호출하는 데 필요한 모든 권한을 포함합니다. RDS Custom의 경우 `AWSServiceRoleForRDSCustom`이 최소 권한의 원칙에 따라 정의된 서비스 연결 역할입니다. RDS Custom은 이 역할에 연결된 정책인 `AmazonRDSCustomServiceRolePolicy`의 권한을 사용하여 대부분의 프로비저닝과 모든 오프 호스트 관리 작업을 수행합니다. 자세한 내용은 AmazonRDSCustomServiceRolePolicy를 참조하세요.

호스트에 작업을 수행할 때 RDS Custom 자동화는 서비스 연결 역할의 보안 인증 정보를 사용하여 AWS Systems Manager로 명령을 실행합니다. Systems Manager 명령 기록 및 AWS CloudTrail을 통해 명령 기록을 감사할 수 있습니다. Systems Manager는 네트워킹 설정을 사용하여 RDS Custom DB 인스턴스에 연결합니다. 자세한 내용은 4단계: RDS Custom for Oracle의 IAM 구성 섹션을 참조하세요.

임시 IAM 보안 인증 정보

리소스를 프로비저닝하거나 삭제할 때 RDS Custom은 호출하는 IAM 보안 주체의 보안 인증 정보에서 파생된 임시 보안 인증 정보를 사용하는 경우가 있습니다. 이러한 IAM 보안 인증 정보는 해당 보안 주체에 연결된 IAM 정책에 의해 제한되며 작업이 완료된 후 만료됩니다. RDS Custom을 사용하는 IAM 보안 주체에 필요한 권한에 대해 알아보려면 5단계: IAM 사용자 또는 역할에 필요한 권한 부여 섹션을 참조하세요.

Amazon EC2 인스턴스 프로파일

EC2 인스턴스 프로파일이란 IAM 역할을 위한 컨테이너로, EC2 인스턴스에 역할 정보를 전달하는 데 사용됩니다. EC2 인스턴스는 RDS Custom DB 인스턴스의 기반이 됩니다. RDS Custom DB 인스턴스를 만들 때 인스턴스 프로필을 제공합니다. RDS Custom은 백업과 같은 호스트 기반 관리

작업을 수행할 때 EC2 인스턴스 프로파일 보안 인증 정보를 사용합니다. 자세한 내용은 수동으로 IAM 역할과 인스턴스 프로파일 생성 섹션을 참조하세요.

SSH 키 페어

RDS Custom은 DB 인스턴스의 기반이 되는 EC2 인스턴스를 만들 때 사용자를 대신하여 SSH 키 페어를 생성합니다. 키는 이름 지정 접두사 `do-not-delete-rds-custom-ssh-privatekey-db-` 또는 `rds-custom!oracle-do-not-delete-`*`db_resource_id-uuid`*`-ssh-privatekey`를 사용하며, AWS Secrets Manager는 SSH 프라이빗 키를 AWS 계정에 보안 암호로 저장합니다. Amazon RDS는 이러한 보안 인증 정보를 저장, 액세스 또는 사용하지 않습니다. 자세한 내용은 Amazon EC2 키 페어 및 Linux 인스턴스를 참조하세요.

SSL 인증서

RDS Custom DB 인스턴스는 관리형 SSL 인증서를 지원하지 않습니다. SSL을 배포하려는 경우 고유 Wallet에서 SSL 인증서를 자체 관리하고 SSL 리스너를 생성하여 클라이언트 데이터베이스 간 연결을 보호하거나 데이터베이스 복제를 수행할 수 있습니다. 자세한 내용은 Oracle Database 설명서에서 전송 계층 보안 인증 구성을 참조하세요.

Amazon S3 버킷 보안으로 혼동된 대리자 문제 방지

Amazon RDS Custom for Oracle 사용자 지정 엔진 버전(CEV) 또는 RDS Custom for SQL Server DB 인스턴스를 생성하면 RDS Custom이 Amazon S3 버킷을 생성합니다. S3 버킷은 CEV 아티팩트, 다시 실행(트랜잭션) 로그, 지원 경계에 대한 구성 항목 및 AWS CloudTrail 로그와 같은 파일을 저장합니다.

혼란스러운 대리인 문제를 방지하기 위해 전역 조건 컨텍스트 키를 사용하여 이러한 S3 버킷을 보다 안전하게 만들 수 있습니다. 자세한 내용은 교차 서비스 혼동된 대리자 문제 방지을 참조하세요.

다음 RDS Custom for Oracle 예는 S3 버킷 정책에서 `aws:SourceArn` 및 `aws:SourceAccount` 전역 조건 컨텍스트 키의 사용을 보여줍니다. RDS Custom for Oracle의 경우 CEV 및 DB 인스턴스에 대한 Amazon 리소스 이름(ARN)을 포함해야 합니다. RDS Custom for SQL Server의 경우 DB 인스턴스 ARN을 포함해야 합니다.

```
...
{
  "Sid": "AWSRDSCustomForOracleInstancesObjectLevelAccess",
  "Effect": "Allow",
  "Principal": {
     "Service": "custom.rds.amazonaws.com"
  },
```

```
  "Action": [
     "s3:GetObject",
     "s3:GetObjectVersion",
     "s3:DeleteObject",
     "s3:DeleteObjectVersion",
     "s3:GetObjectRetention",
     "s3:BypassGovernanceRetention"
  ],
  "Resource": "arn:aws:s3:::do-not-delete-rds-custom-123456789012-us-east-2-c8a6f7/
RDSCustomForOracle/Instances/*",
  "Condition": {
     "ArnLike": {
        "aws:SourceArn": [
            "arn:aws:rds:us-east-2:123456789012:db:*",
            "arn:aws:rds:us-east-2:123456789012:cev:*/*"
        ]
     },
     "StringEquals": {
        "aws:SourceAccount": "123456789012"
    }
  }
},
...
```

규정 준수 프로그램을 위한 RDS Custom for Oracle 자격 증명

일부 규정 준수 프로그램에서는 데이터베이스 사용자 자격 증명을 주기적으로(예: 90일마다) 변경해야 합니다. RDS Custom for Oracle은 미리 정의된 일부 데이터베이스 사용자의 자격 증명을 자동으로 교체합니다.

주제

- 미리 정의된 사용자의 자격 증명 자동 교체
- 사용자 자격 증명 교체에 대한 지침
- 사용자 자격 증명을 수동으로 교체

미리 정의된 사용자의 자격 증명 자동 교체

RDS Custom for Oracle DB 인스턴스가 Amazon RDS에서 호스팅되는 경우, 다음과 같은 미리 정의된 Oracle 사용자의 자격 증명이 30일마다 자동으로 교체됩니다. 이전 사용자의 자격 증명은 AWS Secrets Manager에 있습니다.

데이터베이스 사용자	생성한 사람	지원되는 엔진 버전	Notes
SYS	Oracle	custom-oracle-ee custom-oracle-ee-cdb custom-oracle-se2 custom-oracle-se2-cdb	
SYSTEM	Oracle	custom-oracle-ee custom-oracle-ee-cdb custom-oracle-se2 custom-oracle-se2-cdb	
RDSADMIN	RDS	custom-oracle-ee custom-oracle-se2	
C##RDSADMIN	RDS	custom-oracle-ee-cdb custom-oracle-se2-cdb	C## 접두사가 있는 사용자 이름은 CDB에만 존재합니다. CDB에 대한 자세한 내용은 Amazon RDS Custom for Oracle 아키텍처 개요를 참조하세요.
RDS_DATAGUARD	RDS	custom-oracle-ee	이 사용자는 읽기 전용 복제본, 읽기 전용 복제본의 소스 데이터베이스, Oracle Data Guard를 사용하여 RDS Custom에 물리적으로 마이그레이션된 데이터베이스에만 존재합니다.
C##RDS_DATAGUARD	RDS	custom-oracle-ee-cdb	이 사용자는 읽기 전용 복제본, 읽기 전용 복제본의 소스 데이터베이스, Oracle Data Guard를 사용하여 RDS Custom에 물리적으로

데이터베이스 사용자	생성한 사람	지원되는 엔진 버전	Notes
			마이그레이션된 데이터베이스에만 존재합니다. C## 접두사가 있는 사용자 이름은 CDB에만 존재합니다. CDB에 대한 자세한 내용은 Amazon RDS Custom for Oracle 아키텍처 개요를 참조하세요.

수동 방식을 통해 대기 데이터베이스로 구성된 RDS Custom for Oracle DB 인스턴스는 자동 자격 증명 교체의 예외입니다. RDS는 `create-db-instance-read-replica` CLI 명령 또는`CreateDBInstanceReadReplica` API를 사용하여 생성한 읽기 전용 복제본의 자격 증명만 교체합니다.

사용자 자격 증명 교체에 대한 지침

규정 준수 프로그램에 따라 자격 증명을 교체하려면 다음 지침을 참고하세요.

- DB 인스턴스에서 자격 증명을 자동으로 교체할 경우, 미리 정의된 Oracle 사용자에 나열된 사용자의 암호, 암호 파일 또는 암호를 수동으로 변경하거나 삭제하지 마세요. 그렇지 않으면 RDS Custom이 DB 인스턴스를 지원 경계 밖에 배치하여 자동 교체가 일시 중단될 수 있습니다.
- RDS 마스터 사용자는 미리 정의되어 있지 않으므로 암호를 수동으로 변경하거나, Secrets Manager에서 자동 교체를 설정해야 합니다. 자세한 내용을 알아보려면 AWS Secrets Manager 보안 암호 교체를 참조하세요.

사용자 자격 증명을 수동으로 교체

다음과 같은 데이터베이스 범주의 경우 RDS는 미리 정의된 Oracle 사용자에 나열된 사용자의 자격 증명을 자동으로 교체하지 않습니다.

- 대기 데이터베이스로 작동하도록 수동으로 구성한 데이터베이스.
- 온프레미스 데이터베이스.
- 지원 경계 밖에 있거나 RDS Custom 자동화를 실행할 수 없는 상태의 DB 인스턴스. 이 경우 RDS Custom은 키 교체도 수행하지 않습니다.

데이터베이스가 위의 범주 중 하나에 속할 경우 사용자 자격 증명을 수동으로 교체해야 합니다.

DB 인스턴스의 사용자 자격 증명을 수동으로 교체하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 데이터베이스에서 RDS가 현재 DB 인스턴스 백업 또는 고가용성 구성 같은 작업을 수행하고 있지 않은지 확인합니다.
3. 데이터베이스 세부 정보 페이지에서 구성을 선택하고 DB 인스턴스의 리소스 ID를 기록해 둡니다. AWS CLI 명령인 `describe-db-instances`를 사용할 수 있습니다.
4. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
5. 검색 상자에 데이터베이스의 리소스 ID를 입력하고 다음 이름 지정 규칙 중 하나를 사용하여 보안 암호를 검색합니다.

   ```
   do-not-delete-rds-custom-resource_id-uuid
   rds-custom!oracle-do-not-delete-resource_id-uuid
   ```

   이 암호는 RDSADMIN, SYS, SYSTEM의 암호를 저장합니다. 아래의 샘플 키는 리소스 ID가 db-ABCDEFG12HIJKLNMNOPQRS3TUVWX, UUID가 123456인 DB 인스턴스에 사용되는 키입니다.

   ```
   do-not-delete-rds-custom-db-ABCDEFG12HIJKLNMNOPQRS3TUVWX-123456
   rds-custom!oracle-do-not-delete-db-ABCDEFG12HIJKLNMNOPQRS3TUVWX-123456
   ```

   Important

   DB 인스턴스가 읽기 전용 복제본이고 `custom-oracle-ee-cdb` 엔진을 사용할 경우 접미사 *db-resource_id-uuid*이 붙은 두 개의 암호가 존재합니다. 하나는 마스터 사용자용이고, 다른 하나는 RDSADMIN, SYS, SYSTEM용입니다. 올바른 암호를 찾으려면 호스트에서 다음 명령을 실행합니다.

   ```
   cat /opt/aws/rdscustomagent/config/database_metadata.json | python3 -c
    "import sys,json; print(json.load(sys.stdin)['dbMonitoringUserPassword'])"
   ```

   `dbMonitoringUserPassword` 속성은 RDSADMIN, SYS, SYSTEM의 암호를 나타냅니다.

6. DB 인스턴스가 Oracle Data Guard 구성에 있는 경우 다음 이름 지정 규칙 중 하나를 사용하여 보안 암호를 검색합니다.

```
do-not-delete-rds-custom-resource_id-uuid-dg
rds-custom!oracle-do-not-delete-resource_id-uuid-dg
```

이 암호는 RDS_DATAGUARD의 암호를 저장합니다. 아래의 샘플 키는 DB 리소스 ID가 db-ABCDEFG12HIJKLNMNOPQRS3TUVWX, UUID가 789012인 DB 인스턴스에 사용되는 키입니다.

```
do-not-delete-rds-custom-db-ABCDEFG12HIJKLNMNOPQRS3TUVWX-789012-dg
rds-custom!oracle-do-not-delete-db-ABCDEFG12HIJKLNMNOPQRS3TUVWX-789012-dg
```

7. 미리 정의된 Oracle 사용자에 나열된 모든 데이터베이스 사용자의 경우 AWS Secrets Manager 암호 수정의 지침에 따라 암호를 업데이트하세요.
8. 데이터베이스가 독립 실행형 데이터베이스이거나 Oracle Data Guard 구성의 소스 데이터베이스인 경우:

   a. Oracle SQL 클라이언트를 시작하고 SYS로 로그인합니다.

   b. 미리 정의된 Oracle 사용자에 나열된 각 데이터베이스 사용자에 대해 다음 형식으로 된 SQL 문을 실행합니다.

   ```
   ALTER USER user-name IDENTIFIED BY pwd-from-secrets-manager ACCOUNT UNLOCK;
   ```

   예를 들어 Secrets Manager에 저장된 RDSADMIN의 새 암호가 pwd-123인 경우 다음 문을 실행합니다.

   ```
   ALTER USER RDSADMIN IDENTIFIED BY pwd-123 ACCOUNT UNLOCK;
   ```

9. Oracle Database 12c 릴리스 1(12.1)을 실행하고 Oracle Data Guard에서 관리하는 DB 인스턴스의 경우, 기본 DB 인스턴스의 암호 파일(orapw)을 각 대기 DB 인스턴스에 수동으로 복사하세요.

   Amazon RDS에서 DB 인스턴스를 호스팅하는 경우 암호 파일 위치는 /rdsdbdata/config/orapw입니다. Amazon RDS에서 호스팅되지 않는 데이터베이스의 경우 기본 위치는 Linux 및 UNIX는 $ORACLE_HOME/dbs/orapw$ORACLE_SID이고, Windows는 %ORACLE_HOME%\database\PWD%ORACLE_SID%.ora입니다.

RDS Custom for Oracle 작업

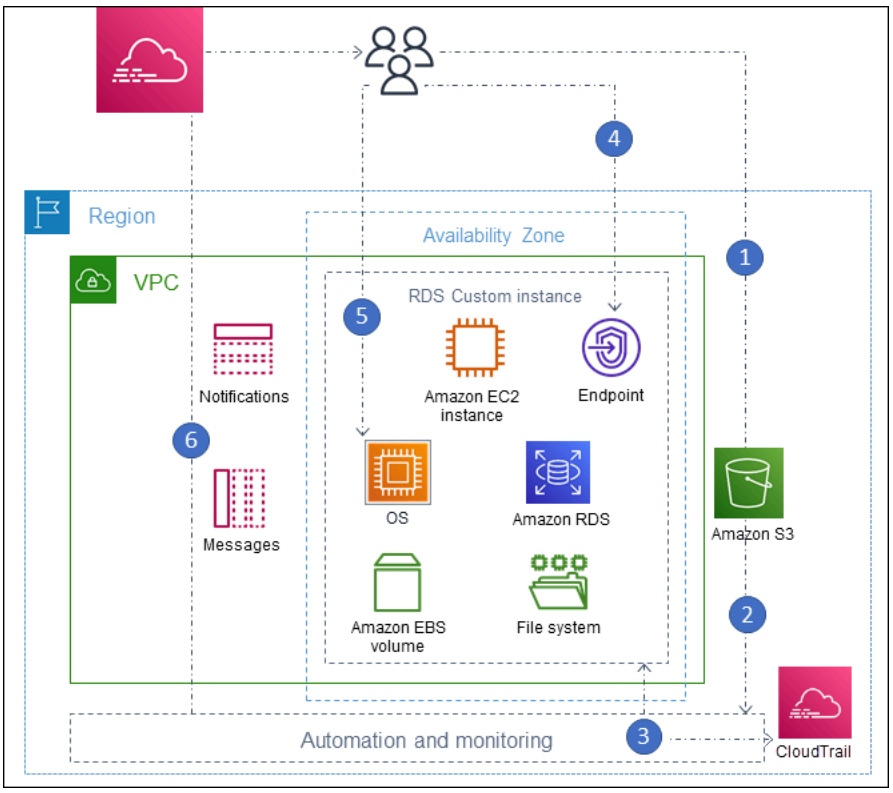
아래에서는 RDS Custom for Oracle DB 인스턴스의 관리 및 유지 관리에 대한 지침을 확인할 수 있습니다.

주제

- RDS Custom for Oracle 워크플로
- Amazon RDS Custom for Oracle 데이터베이스 아키텍처
- RDS Custom for Oracle에 대한 기능 가용성 및 지원
- RDS Custom for Oracle 요구 사항 및 제한
- Amazon RDS Custom for Oracle을 위한 환경 설정
- 사용자 지정 엔진 버전의 Amazon RDS Custom for Oracle으로 작업
- Amazon RDS Custom for Oracle용 DB 인스턴스 구성
- Amazon RDS Custom for Oracle DB 인스턴스 관리
- RDS Custom for Oracle에 대한 다중 AZ 배포 관리
- RDS Custom for Oracle의 Oracle 복제본으로 작업
- Amazon RDS Custom for Oracle DB 인스턴스 백업 및 복원
- RDS Custom for Oracle에서 옵션 그룹을 사용한 작업
- 온프레미스 데이터베이스를 RDS Custom for Oracle로 마이그레이션
- Oracle용 Amazon RDS Custom DB 인스턴스 업그레이드
- Amazon RDS Custom for Oracle의 DB 문제 해결
- Amazon RDS Custom for Oracle의 알려진 문제

RDS Custom for Oracle 워크플로

다음 다이어그램은 RDS Custom for Oracle의 일반적인 워크플로를 보여줍니다.



단계는 다음과 같습니다.

1. Amazon S3 버킷에 데이터베이스 소프트웨어를 업로드합니다.

   자세한 내용은 3단계: Amazon S3에 설치 파일 업로드 섹션을 참조하세요.

2. 미디어에서 RDS Custom for Oracle 사용자 지정 엔진 버전(CEV)을 만듭니다.

   CDB 아키텍처 또는 기존의 비CDB 아키텍처를 선택합니다. 자세한 내용은 CEV 생성 섹션을 참조하세요.

3. CEV에서 RDS Custom for Oracle DB 인스턴스를 생성합니다.

   자세한 내용은 RDS Custom for Oracle DB 인스턴스 생성 섹션을 참조하세요.

4. 애플리케이션을 DB 인스턴스 엔드포인트에 연결합니다.

   자세한 내용은 SSH를 사용하여 RDS Custom DB 인스턴스에 연결 및 세션 관리자를 사용하여 RDS Custom DB 인스턴스에 연결 섹션을 참조하세요.

5. (선택 사항) 호스트에 액세스하여 소프트웨어를 커스터마이징합니다.

6. RDS Custom 자동화로 생성된 알림 및 메시지를 모니터링합니다.

데이터베이스 설치 파일

미디어에 대한 책임은 Amazon RDS와 RDS Custom 간의 주요 차이점입니다. 완전관리형 서비스인 Amazon RDS는 Amazon Machine Image(AMI) 및 데이터베이스 소프트웨어를 제공합니다. Amazon RDS 데이터베이스 소프트웨어가 사전 설치되어 있으므로 데이터베이스 엔진과 버전을 선택하고 데이터베이스를 생성하기만 하면 됩니다.

RDS Custom의 경우 자체 미디어를 제공합니다. 커스텀 엔진 버전을 만들면 RDS Custom은 사용자가 제공한 미디어를 설치합니다. RDS Custom 미디어에는 데이터베이스 설치 파일 및 패치가 포함되어 있습니다. 이 서비스 모델은 자체 미디어 가져오기(BYOM)입니다.

RDS Custom for Oracle의 사용자 지정 엔진 버전

RDS Custom for Oracle 사용자 지정 엔진 버전(CEV)은 데이터베이스 버전 및 AMI의 바이너리 볼륨 스냅샷입니다. 기본적으로 RDS Custom for Oracle은 Amazon EC2에서 제공하는 가장 최신 AMI를 사용합니다. 기존 AMI를 재사용하는 방법도 있습니다.

CEV 매니페스트

Oracle에서 Oracle 데이터베이스 설치 파일을 다운로드한 후 Amazon S3 버킷에 업로드합니다. CEV를 만들 때 JSON 문서에 CEV 매니페스트로 파일 이름을 지정합니다. RDS Custom for Oracle은 이 설치 파일과 AMI를 사용하여 CEV를 생성합니다.

RDS Custom for Oracle은 지원되는 각 Oracle 데이터베이스 릴리스에 대한 권장 .zip 파일이 포함된 JSON 매니페스트 템플릿을 제공합니다. 예를 들어, 다음 템플릿은 19.17.0.0.0 RU용입니다.

```
{
    "mediaImportTemplateVersion": "2020-08-14",
    "databaseInstallationFileNames": [
        "V982063-01.zip"
    ],
    "opatchFileNames": [
        "p6880880_190000_Linux-x86-64.zip"
    ],
    "psuRuPatchFileNames": [
        "p34419443_190000_Linux-x86-64.zip",
        "p34411846_190000_Linux-x86-64.zip"
    ],
    "otherPatchFileNames": [
        "p28852325_190000_Linux-x86-64.zip",
        "p29997937_190000_Linux-x86-64.zip",
```

```
        "p31335037_190000_Linux-x86-64.zip",
        "p32327201_190000_Linux-x86-64.zip",
        "p33613829_190000_Linux-x86-64.zip",
        "p34006614_190000_Linux-x86-64.zip",
        "p34533061_190000_Linux-x86-64.zip",
        "p34533150_190000_Generic.zip",
        "p28730253_190000_Linux-x86-64.zip",
        "p29213893_1917000DBRU_Generic.zip",
        "p33125873_1917000DBRU_Linux-x86-64.zip",
        "p34446152_1917000DBRU_Linux-x86-64.zip"
    ]
}
```

JSON 매니페스트에서 설치 파라미터를 지정할 수도 있습니다. 예를 들어 Oracle base, Oracle home, UNIX/Linux 사용자 및 그룹의 ID 및 이름에 대해 기본값이 아닌 값을 설정할 수 있습니다. 자세한 내용은 CEV 매니페스트 내의 JSON 섹션을 참조하세요.

CEV 이름 지정 형식

고객 지정 문자열을 사용하여 RDS Custom for Oracle CEV의 이름을 지정합니다. 이름 형식은 Oracle Database 릴리스에 따라 다음과 같습니다.

- `19.customized_string`
- `18.customized_string`
- `12.2.customized_string`
- `12.1.customized_string`

1~50개의 영숫자, 밑줄, 대시 및 마침표를 사용할 수 있습니다. 예를 들어 CEV 이름을 `19.my_cev1`으로 지정할 수 있습니다.

RDS Custom for Oracle의 Oracle 멀티테넌트 아키텍처

Oracle 멀티테넌트 아키텍처에서 Oracle 데이터베이스는 컨테이너 데이터베이스(CDB) 기능을 할 수 있습니다. CDB에는 0개, 1개 또는 여러 개의 고객 생성 플러그형 데이터베이스(PDB)가 포함됩니다. PDB는 애플리케이션에 기존 비CDB로 나타나는 스키마와 객체의 이동식 모음입니다.

RDS Custom for Oracle CEV를 생성할 때 CDB 또는 비CDB 아키텍처를 지정합니다. RDS Custom for Oracle CDB는 생성에 사용한 CEV가 Oracle 멀티테넌트 아키텍처를 사용하는 경우에만 생성할 수 있습니다. 자세한 내용은 사용자 지정 엔진 버전의 Amazon RDS Custom for Oracle으로 작업 섹션을 참조하세요.

RDS Custom for Oracle DB 인스턴스 생성

CEV는 생성 후 사용 가능한 상태가 됩니다. 여러 CEV를 생성하고, CEV에서 RDS Custom for Oracle DB 인스턴스를 여러 개 생성할 수 있습니다. CEV의 상태를 변경하여 사용 가능 또는 비활성 상태로 만들 수도 있습니다.

Oracle 멀티테넌트 아키텍처(`custom-oracle-ee-cdb` 또는 `custom-oracle-se2-cdb` 엔진 유형) 또는 기존의 비CDB 아키텍처(`custom-oracle-ee` 또는 `custom-oracle-se2` 엔진 유형)를 사용하여 RDS Custom for Oracle DB 인스턴스를 생성할 수 있습니다. 생성한 컨테이너 데이터베이스(CDB)에는 플러그형 데이터베이스(PDB) 1개와 PDB 시드 1개가 포함됩니다. Oracle SQL을 사용하여 수동으로 추가 PDB를 생성할 수 있습니다.

RDS Custom for Oracle DB 인스턴스를 생성하려면 `create-db-instance` 명령을 사용하세요. 이 명령에서 사용할 CEV를 지정합니다. 생성 절차는 Amazon RDS DB 인스턴스 생성과 유사합니다. 하지만 일부 파라미터는 다릅니다. 자세한 내용은 Amazon RDS Custom for Oracle용 DB 인스턴스 구성 섹션을 참조하세요.

데이터베이스 연결

Amazon RDS DB 인스턴스와 마찬가지로 RDS Custom DB 인스턴스는 Virtual Private Cloud(VPC)에 있습니다. 애플리케이션은 Oracle 리스너를 사용하여 Oracle 데이터베이스에 연결합니다.

데이터베이스가 CDB인 경우 리스너 `L_RDSCDB_001`을 사용하여 CDB 루트와 PDB에 연결할 수 있습니다. 비 CDB를 CDB에 연결하는 경우 마이그레이션된 애플리케이션이 동일한 설정을 유지하도록 `USE_SID_AS_SERVICE_LISTENER = ON`을 설정해야 합니다.

비 CDB에 연결하는 경우 마스터 사용자는 비 CDB의 사용자입니다. CDB에 연결하는 경우 마스터 사용자는 PDB의 사용자입니다. CDB 루트에 연결하려면 호스트에 로그인하고 SQL 클라이언트를 시작한 다음 SQL 명령을 사용하여 관리자를 생성합니다.

RDS Custom 커스터마이징

RDS Custom 호스트에 액세스하여 소프트웨어를 설치하거나 커스터마이징할 수 있습니다. 변경 사항과 RDS Custom 자동화 간의 충돌을 방지하려면 지정된 기간 동안 자동화를 일시 중지하면 됩니다. 이 기간 동안 RDS Custom은 모니터링 또는 인스턴스 복구를 수행하지 않습니다. 기간이 만료되면 RDS Custom은 전체 자동화를 재개합니다. 자세한 내용은 RDS Custom DB 인스턴스 일시 중지 및 재개 섹션을 참조하세요.

Amazon RDS Custom for Oracle 데이터베이스 아키텍처

RDS Custom for Oracle은 Oracle 멀티테넌트 아키텍처와 비멀티테넌트 아키텍처를 모두 지원합니다.

주제

- 지원되는 Oracle 데이터베이스 아키텍처
- 지원되는 엔진 유형
- Oracle 멀티테넌트 아키텍처에서 지원되는 특성

지원되는 Oracle 데이터베이스 아키텍처

Oracle 멀티테넌트 아키텍처(CDB 아키텍처)는 Oracle 데이터베이스가 컨테이너 데이터베이스(CDB) 기능을 하도록 합니다. CDB에는 플러그형 데이터베이스(PDB)가 포함되어 있습니다. PDB는 애플리케이션에 기존 Oracle 데이터베이스로 나타나는 스키마와 객체의 모음입니다. 자세한 내용은 Oracle Multitenant 관리자 안내서의 멀티테넌트 아키텍처 소개를 참조하세요.

CDB 아키텍처와 비CDB 아키텍처는 상호 배타적입니다. Oracle 데이터베이스가 CDB가 아닌 경우 비CDB이므로 PDB를 포함할 수 없습니다. RDS Custom for Oracle에서는 Oracle Database 19c만 CDB 아키텍처를 지원합니다. 따라서 이전 Oracle 데이터베이스 릴리스를 사용하여 DB 인스턴스를 생성할 경우 비CDB만 생성할 수 있습니다. 자세한 내용은 멀티테넌트 아키텍처 고려 사항 단원을 참조하십시오.

지원되는 엔진 유형

Amazon RDS Custom for Oracle CEV 또는 DB 인스턴스를 생성할 때 다음 CDB 엔진 유형 또는 비CDB 엔진 유형을 선택합니다.

- `custom-oracle-ee-cdb` 및 `custom-oracle-se2-cdb`

  이러한 엔진 유형은 Oracle 멀티테넌트 아키텍처를 지정합니다. 이 옵션은 Oracle Database 19c에서만 지원됩니다. 멀티테넌트 아키텍처를 사용하여 RDS for Oracle DB 인스턴스를 생성하는 경우 CDB에는 다음과 같은 컨테이너가 포함됩니다.

  - CDB 루트(`CDB$ROOT`)
  - PDB 시드(`PDB$SEED`)
  - 초기 PDB

  Oracle SQL 명령 `CREATE PLUGGABLE DATABASE`를 사용하여 더 많은 PDB를 생성할 수 있습니다. RDS API를 사용하여 PDB를 만들거나 삭제할 수 없습니다.

- `custom-oracle-ee` 및 `custom-oracle-se2`

  이러한 엔진 유형은 기존의 비CDB 아키텍처를 지정합니다. 비 CDB에는 플러그형 데이터베이스(PDB)가 포함될 수 없습니다.

자세한 내용은 멀티테넌트 아키텍처 고려 사항 단원을 참조하십시오.

Oracle 멀티테넌트 아키텍처에서 지원되는 특성

RDS Custom for Oracle CDB 인스턴스는 다음 기능을 지원합니다.

- 백업
- 백업에서 복원 및 시점 복구(PITR)
- 읽기 전용 복제본
- 마이너 버전 업그레이드

RDS Custom for Oracle에 대한 기능 가용성 및 지원

이 주제에서는 빠른 참조를 위해 RDS Custom for Oracle 기능 가용성 및 지원 요약을 찾을 수 있습니다.

주제

- RDS Custom for Oracle에 대한 AWS 리전 및 데이터베이스 버전 지원
- RDS Custom for Oracle에 대한 데이터베이스 버전 지원
- RDS Custom for Oracle에 대한 에디션 및 라이선스 지원
- RDS Custom for Oracle에 대한 DB 인스턴스 클래스 지원
- RDS Custom for Oracle에 대한 옵션 그룹 지원

RDS Custom for Oracle에 대한 AWS 리전 및 데이터베이스 버전 지원

기능 가용성 및 해당 지원은 각 데이터베이스 엔진의 특정 버전 및 AWS 리전에 따라 다릅니다. RDS Custom for Oracle의 버전 및 리전 가용성에 대한 자세한 내용은 RDS Custom을 지원하는 리전 및 DB 엔진 섹션을 참조하세요.

RDS Custom for Oracle에 대한 데이터베이스 버전 지원

RDS Custom for Oracle은 다음과 같은 Oracle 데이터베이스 버전을 지원합니다.

- Oracle Database 19c
- Oracle Database 18c
- Oracle Database 12c 릴리스 2(12.2)
- Oracle Database 12c 릴리스 1(12.1)

RDS Custom for Oracle에 대한 에디션 및 라이선스 지원

RDS Custom for Oracle은 BYOL 모델에서 Enterprise Edition(EE) 및 Standard Edition 2(SE2)를 지원합니다.

Standard Edition 2의 경우 다음 제한 사항이 있으니 참조하세요.

- Oracle Data Guard가 지원되지 않습니다. 따라서 Oracle 읽기 전용 복제본을 만들 수 없습니다.
- vCPU가 16개 이하(최대 4배)인 DB 인스턴스 클래스만 사용할 수 있습니다.
- Standard Edition 2의 CDB 인스턴스는 최대 3개의 테넌트 데이터베이스를 지원합니다.

- Enterprise Edition 및 Standard Edition 2 간에는 데이터를 마이그레이션할 수 없습니다.

RDS Custom for Oracle에 대한 DB 인스턴스 클래스 지원

RDS Custom for Oracle은 다음 DB 인스턴스 클래스를 지원합니다. Standard Edition 2에서 DB 인스턴스를 생성하는 경우 vCPU가 16개 이하(최대 4배)인 인스턴스 클래스만 사용할 수 있습니다.

유형	크기
db.r7i	db.r7i.large |db.r7i.xlarge |db.r7i.2xlarge |db.r7i.4xlarge |db.r7i.8xlarge |db.r7i.12xlarge |db.r7i.16xlarge |db.r7i.24xlarge |db.r7i.48xlarge
db.r6i	db.r6i.large |db.r6i.xlarge |db.r6i.2xlarge |db.r6i.4xlarge |db.r6i.8xlarge |db.r6i.12xlarge |db.r6i.16xlarge |db.r6i.24xlarge |db.r6i.32xlarge
db.r5b	db.r5b.large |db.r5b.xlarge |db.r5b.2xlarge |db.r5b.4xlarge |db.r5b.8xlarge |db.r5b.12xlarge |db.r5b.16xlarge |db.r5b.24xlarge
db.r5	db.r5.large |db.r5.xlarge |db.r5.2xlarge |db.r5.4xlarge | db.r5.8xlarge |db.r5.12xlarge |db.r5.16xlarge |db.r5.24xlarge
db.x2iec	db.x2iedn.xlarge |db.x2iedn.2xlarge |db.x2iedn.4xlarge | db.x2iedn.8xlarge |db.x2iedn.16xlarge |db.x2iedn.24xlarge | db.x2iedn.32xlarge
db.x2iez	db.x2iezn.2xlarge |db.x2iezn.4xlarge |db.x2iezn.6xlarge | db.x2iezn.8xlarge |db.x2iezn.12xlarge
db.m7i	db.m7i.large |db.m7i.xlarge |db.m7i.2xlarge |db.m7i.4xlarge |db.m7i.8xlarge |db.m7i.12xlarge |db.m7i.16xlarge |db.m7i.24xlarge |db.m7i.48xlarge
db.m6i	db.m6i.large |db.m6i.xlarge |db.m6i.2xlarge |db.m6i.4xlarge |db.m6i.8xlarge |db.m6i.12xlarge |db.m6i.16xlarge |db.m6i.24xlarge |db.m6i.32xlarge

유형	크기
db.m5	db.m5.large | db.m5.xlarge | db.m5.2xlarge | db.m5.4xlarge | db.m5.8xlarge | db.m5.12xlarge | db.m5.16xlarge | db.m5.24xlarge
db.t3	db.t3.medium | db.t3.large | db.t3.xlarge | db.t3.2xlarge

RDS Custom for Oracle에 대한 옵션 그룹 지원

RDS Custom for Oracle DB 인스턴스를 생성하거나 수정할 때 옵션 그룹을 지정할 수 있습니다. 자세한 내용은 RDS Custom for Oracle에서 옵션 그룹을 사용한 작업 섹션을 참조하세요.

RDS Custom for Oracle 요구 사항 및 제한

이 주제에서는 빠른 참조를 위해 Amazon RDS Custom for Oracle 가용성 및 요구 사항의 요약을 찾을 수 있습니다.

주제

- RDS Custom for Oracle 일반 요구 사항
- RDS Custom for Oracle의 일반적 제한 사항
- RDS Custom for Oracle의 CEV 및 AMI 제한 사항
- 워크플로 생성 및 수정에 지원되지 않는 설정
- AWS 계정의 DB 인스턴스 할당량

RDS Custom for Oracle 일반 요구 사항

Amazon RDS Custom for Oracle에 대한 다음 요구 사항을 준수해야 합니다.

- My Oracle Support 및 Oracle Software Delivery Cloud에 액세스하여 RDS Custom for Oracle의 지원되는 설치 파일 및 패치 목록을 다운로드할 수 있습니다. 알 수 없는 패치를 사용하는 경우 사용자 지정 엔진 버전(CEV) 생성에 실패합니다. 이 경우 RDS Custom 지원팀에 문의하여 누락된 패치를 추가해달라고 요청하세요. 자세한 내용은 2단계: Oracle Software Delivery Cloud에서 데이터베이스 설치 파일 및 패치 다운로드 섹션을 참조하세요.
- Amazon S3에 액세스할 수 있습니다. 다음과 같은 이유로 이 서비스가 필요합니다.
  - Oracle 설치 파일을 S3 버킷에 업로드합니다. 업로드된 설치 파일은 RDS Custom CEV를 생성할 때 사용합니다.
  - RDS Custom for Oracle은 내부적으로 정의된 S3 버킷에서 다운로드한 스크립트를 사용하여 DB 인스턴스에서 작업을 수행합니다. 이러한 스크립트는 온보딩 및 RDS Custom 자동화에 필요합니다.
  - RDS Custom for Oracle은 특정 파일을 고객 계정에 있는 S3 버킷에 업로드합니다. 이러한 버킷은 `do-not-delete-rds-custom-`*`account_id`*`-`*`region`*`-`*`uuid`*라는 이름 지정 형식을 사용합니다. 예를 들어, `do-not-delete-rds-custom-123456789012-us-east-1-12a3b4`라는 이름이 지정된 버킷이 있는 경우를 가정해 보겠습니다.

  자세한 내용은 3단계: Amazon S3에 설치 파일 업로드 및 CEV 생성(을)를 참조하세요.
- RDS Custom for Oracle에 대한 DB 인스턴스 클래스 지원에 나열된 DB 인스턴스 클래스를 사용하여 RDS Custom for Oracle DB 인스턴스를 생성할 수 있습니다.

- RDS Custom for Oracle DB 인스턴스는 Oracle Linux 8(권장) 또는 Oracle Linux 7을 실행합니다. Oracle Linux 7이 필요한 경우 지원에 문의하세요. 자세한 내용은 RDS Custom for Oracle 데이터베이스 업그레이드에 대한 고려 사항 섹션을 참조하세요.
- Amazon EBS 스토리지에 대한 gp2, gp3 또는 io1 솔리드 스테이트 드라이브를 지정합니다. 최대 스토리지 크기는 64TiB입니다.
- RDS Custom for Oracle DB 인스턴스를 생성할 수 있는 AWS KMS 키가 있습니다. 자세한 내용은 1단계: 대칭 암호화 AWS KMS 키 생성 또는 재사용 섹션을 참조하세요.
- RDS Custom for Oracle DB 인스턴스 생성에 필요한 AWS Identity and Access Management(IAM) 역할 및 인스턴스 프로파일이 있습니다. 자세한 내용은 4단계: RDS Custom for Oracle의 IAM 구성 섹션을 참조하세요.
- CEV 또는 RDS Custom DB 인스턴스를 생성하는 AWS Identity and Access Management(IAM) 사용자는 IAM, CloudTrail 및 Amazon S3에 대한 필수 권한을 보유합니다.

  자세한 내용은 5단계: IAM 사용자 또는 역할에 필요한 권한 부여 섹션을 참조하세요.
- 자체 Virtual Private Cloud(VPC) 및 보안 그룹 구성을 제공합니다. 자세한 내용은 6단계: RDS Custom for Oracle용 VPC 구성 섹션을 참조하세요.
- RDS Custom for Oracle이 다른 AWS 서비스에 액세스하는 데 사용할 수 있는 네트워킹 구성을 제공합니다. 특정 요구 사항은 4단계: RDS Custom for Oracle의 IAM 구성 단원을 참조하세요.

RDS Custom for Oracle의 일반적 제한 사항

RDS Custom for Oracle에는 다음과 같은 제한 사항이 적용됩니다.

- 기존 RDS Custom for Oracle DB 인스턴스의 DB 인스턴스 식별자는 수정할 수 없습니다.
- Oracle Database 19c 이외의 릴리스에는 Oracle 멀티테넌트 아키텍처를 지정할 수 없습니다.
- RDS Custom for Oracle 인스턴스에서는 여러 Oracle 데이터베이스를 생성할 수 없습니다.
- RDS Custom for Oracle DB 인스턴스 또는 기본 Amazon EC2 인스턴스는 중지할 수 없습니다. RDS Custom for Oracle DB 인스턴스의 요금 청구는 중지할 수 없습니다.
- RDS Custom for Oracle은 자동 메모리 관리만 지원하므로 자동 공유 메모리 관리는 사용할 수 없습니다. 자세한 내용은 Oracle 데이터베이스 관리자 안내서의 자동 메모리 관리를 참조하세요.
- 프라이머리 DB 인스턴스의 `DB_UNIQUE_NAME`을 변경하지 않도록 주의하세요. 이름을 변경하면 모든 복원 작업이 중단됩니다.
- 같은 리전에서 동시에 20개 이상의 스냅샷 복사본을 만들 수 없습니다.

- RDS Custom for Oracle DB 인스턴스에는 `describe-reserved-db-instances` API를 사용할 수 없습니다.

RDS Custom for Oracle DB 인스턴스 수정 관련 제한 사항은 RDS Custom for Oracle DB 인스턴스 수정을 참조하세요. 복제 제한은 RDS Custom for Oracle 복제본의 일반적 제한 사항을 참조하세요.

RDS Custom for Oracle의 CEV 및 AMI 제한 사항

RDS Custom for Oracle CEV 및 AMI에는 다음과 같은 제한 사항이 적용됩니다.

- RDS Custom for Oracle CEV에서 사용할 자체 AMI를 제공할 수 없습니다. Oracle Linux 8을 사용하는 기본 AMI 또는 이전에 RDS Custom for Oracle CEV에서 사용한 AMI를 지정할 수 있습니다.

  > **Note**
  >
  > RDS Custom for Oracle은 일반적인 취약성 및 노출이 발견되면 새로운 기본 AMI를 릴리스합니다. 일정이 정해져 있거나 보장되지 않습니다. RDS Custom for Oracle은 보통 30일마다 새로운 기본 AMI를 게시합니다.

- CEV를 수정하여 다른 AMI를 사용할 수는 없습니다.
- `custom-oracle-ee` 또는 `custom-oracle-se2` 엔진 유형을 사용하는 CEV에서 CDB 인스턴스를 만들 수 없습니다. CEV에는 `custom-oracle-ee-cdb` 또는 `custom-oracle-se2-cdb`를 사용해야 합니다.
- RDS Custom for Oracle에서는 현재 RDS API 호출을 사용하여 RDS Custom for Oracle DB 인스턴스의 OS를 업그레이드할 수 없습니다. 차선책으로서 다음 `sudo yum update --security` 명령을 사용하여 OS를 수동으로 업데이트할 수 있습니다.

워크플로 생성 및 수정에 지원되지 않는 설정

RDS Custom for Oracle DB 인스턴스를 생성하거나 수정할 때는 다음과 같은 작업을 수행할 수 없습니다.

- DB 인스턴스 클래스의 코어당 CPU 코어 및 스레드 수를 변경합니다.
- 스토리지 자동 조정을 활성화합니다.
- 백업 보존을 `0`으로 설정합니다.
- Kerberos 인증을 구성합니다.
- 고유한 DB 파라미터 그룹 또는 옵션 그룹을 지정합니다.

- 성능 개선 도우미를 활성화합니다.
- 마이너 버전 자동 업그레이드를 활성화합니다.

AWS 계정의 DB 인스턴스 할당량

RDS Custom과 Amazon RDS DB 인스턴스의 합산 수가 할당량 한도를 초과하지 않아야 합니다. 예를 들어, Amazon RDS에 대한 할당량이 40개의 DB 인스턴스인 경우 RDS Custom for Oracle DB 인스턴스 20개와 Amazon RDS DB 인스턴스 20개를 보유할 수 있습니다.

Amazon RDS Custom for Oracle을 위한 환경 설정

Amazon RDS Custom for Oracle DB 인스턴스를 생성하기 전에 다음 태스크를 수행해야 합니다.

주제

- 1단계: 대칭 암호화 AWS KMS 키 생성 또는 재사용
- 2단계: AWS CLI 다운로드 및 설치
- 3단계: RDS Custom for Oracle용 CloudFormation 템플릿 추출
- 4단계: RDS Custom for Oracle의 IAM 구성
- 5단계: IAM 사용자 또는 역할에 필요한 권한 부여
- 6단계: RDS Custom for Oracle용 VPC 구성

1단계: 대칭 암호화 AWS KMS 키 생성 또는 재사용

고객 관리형 키는 사용자가 생성, 소유 및 관리하는 AWS 계정의 AWS KMS keys입니다. 고객 관리형 대칭 암호화 KMS 키는 RDS Custom에 필수입니다. RDS Custom for Oracle DB 인스턴스를 생성할 때 KMS 키 식별자를 제공해야 합니다. 자세한 내용은 Amazon RDS Custom for Oracle용 DB 인스턴스 구성 단원을 참조하세요.

다음과 같은 옵션이 있습니다:

- 기존에 AWS 계정에 고객 관리형 KMS 키가 있는 경우 RDS Custom과 함께 사용할 수 있습니다. 별도로 조치를 취할 필요가 없습니다.
- 다른 RDS Custom 엔진에 대해 고객 관리형 대칭 암호화 KMS 키를 이미 생성한 경우 동일한 KMS 키를 재사용할 수 있습니다. 별도로 조치를 취할 필요가 없습니다.
- 계정에 사용 중이던 고객 관리형 대칭 암호화 KMS 키가 없는 경우 AWS Key Management Service 개발자 가이드의 키 생성 지침에 따라 KMS 키를 생성합니다.
- CEV 또는 RDS Custom DB 인스턴스를 생성하고 있으며 KMS 키가 다른 AWS 계정에 있는 경우 반드시 AWS CLI를 사용해야 합니다. 교차 계정 KMS 키에는 AWS 콘솔을 사용할 수 없습니다.

> **Important**
>
> RDS Custom은 AWS 관리형 KMS 키를 지원하지 않습니다.

사용하는 대칭 암호화 키는 `kms:Decrypt` 및 `kms:GenerateDataKey` 작업에 액세스할 수 있는 권한을 IAM 인스턴스 프로파일의 AWS Identity and Access Management(IAM) 역할에 제공해야 합니다. 계정에 대칭 암호화 키를 새로 사용하는 경우에는 변경할 필요가 없습니다. 아니면 대칭 암호화 키의 정책이 이러한 작업에 액세스할 권한을 부여할 수 있는지 확인하세요.

자세한 내용은 4단계: RDS Custom for Oracle의 IAM 구성 섹션을 참조하세요.

RDS Custom for Oracle의 IAM 구성에 대한 자세한 내용은 4단계: RDS Custom for Oracle의 IAM 구성 섹션을 참조하세요.

2단계: AWS CLI 다운로드 및 설치

AWS에서는 RDS Custom 기능을 사용할 수 있는 명령줄 인터페이스를 제공합니다. AWS CLI의 버전 1 또는 버전 2를 사용할 수 있습니다.

AWS CLI 다운로드 및 설치에 대한 내용은 AWS CLI의 최신 버전 설치 또는 업데이트를 참조하세요.

다음 중 하나에 해당하는 경우 이 단계를 건너뜁니다.

- AWS Management Console에서만 RDS Custom에 액세스하려는 경우.
- Amazon RDS용 AWS CLI 또는 다른 RDS Custom DB 엔진을 이미 다운로드한 경우.

3단계: RDS Custom for Oracle용 CloudFormation 템플릿 추출

설정을 간소화하려면 AWS CloudFormation 템플릿을 사용하여 CloudFormation 스택을 만드는 것이 좋습니다. IAM과 VPC를 수동으로 구성하려면 이 단계를 건너뜁니다.

주제

- 3a단계: CloudFormation 템플릿 파일 다운로드
- 3b단계: custom-oracle-iam.json 추출
- 3c단계: custom-vpc.json 추출

3a단계: CloudFormation 템플릿 파일 다운로드

CloudFormation 템플릿이란 스택을 구성하는 AWS 리소스의 선언입니다. 템플릿은 JSON 파일로 저장됩니다.

CloudFormation 템플릿 파일을 다운로드하려면

1. custom-oracle-iam.zip 링크의 컨텍스트 메뉴(마우스 오른쪽 버튼 클릭)를 열고 다른 이름으로 링크 저장을 선택합니다.
2. 파일을 컴퓨터에 저장합니다.
3. custom-vpc.zip 링크에 대해 이전 단계를 반복합니다.

   RDS Custom의 VPC를 이미 구성한 경우 이 단계를 건너뜁니다.

3b단계: custom-oracle-iam.json 추출

다운로드한 `custom-oracle-iam.zip` 파일을 연 다음 `custom-oracle-iam.json` 파일을 추출합니다. 파일의 시작 부분은 다음과 같습니다.

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters": {
    "EncryptionKey": {
      "Type": "String",
      "Default": "*",
      "Description": "KMS Key ARN for encryption of data managed by RDS Custom and by
 DB Instances."
    }
  },
  "Resources": {
    "RDSCustomInstanceServiceRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "RoleName": { "Fn::Sub": "AWSRDSCustomInstanceRole-${AWS::Region}" },
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Action": "sts:AssumeRole",
              "Effect": "Allow",
              "Principal": {
                "Service": "ec2.amazonaws.com"
              }
            }
          ]
        },...
```

3c단계: custom-vpc.json 추출

> **Note**
>
> 기존 RDS Custom for Oracle용 VPC를 이미 구성한 경우 이 단계를 건너뜁니다. 자세한 내용은 수동으로 RDS Custom for Oracle용 VPC 구성 섹션을 참조하세요.

다운로드한 `custom-vpc.zip` 파일을 연 다음 `custom-vpc.json` 파일을 추출합니다. 파일의 시작 부분은 다음과 같습니다.

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters": {
    "PrivateVpc": {
      "Type": "AWS::EC2::VPC::Id",
      "Description": "Private VPC Id to use for RDS Custom DB Instances"
    },
    "PrivateSubnets": {
      "Type": "List<AWS::EC2::Subnet::Id>",
      "Description": "Private Subnets to use for RDS Custom DB Instances"
    },
    "RouteTable": {
      "Type": "String",
      "Description": "Route Table that must be associated with the PrivateSubnets and
 used by S3 VPC Endpoint",
      "AllowedPattern": "rtb-[0-9a-z]+"
    }
  },
  "Resources": {
    "DBSubnetGroup": {
      "Type": "AWS::RDS::DBSubnetGroup",
      "Properties": {
        "DBSubnetGroupName": "rds-custom-private",
        "DBSubnetGroupDescription": "RDS Custom Private Network",
        "SubnetIds": {
          "Ref": "PrivateSubnets"
        }
      }
    },...
```

4단계: RDS Custom for Oracle의 IAM 구성

IAM 역할 또는 IAM 사용자(IAM 엔터티라고 함)를 사용하여 콘솔 또는 AWS CLI를 사용하는 RDS Custom DB 인스턴스를 생성할 수 있습니다. 이 IAM 엔터티에는 인스턴스를 생성하는 데 필요한 권한이 있어야 합니다.

CloudFormation 또는 수동 단계를 사용하여 IAM을 구성할 수 있습니다.

> **Important**
>
> AWS CloudFormation을 사용하여 RDS Custom for Oracle 환경을 구성하는 것이 좋습니다. 이 방법은 가장 쉽고 오류가 가장 적습니다.

주제

- CloudFormation을 사용하여 IAM 구성
- 수동으로 IAM 역할과 인스턴스 프로파일 생성

CloudFormation을 사용하여 IAM 구성

IAM에 CloudFormation 템플릿을 사용할 경우, 다음과 같은 필수 리소스가 생성됩니다.

- 이름이 `AWSRDSCustomInstanceProfile-`*`region`*인 인스턴스 프로파일
- `AWSRDSCustomInstanceRole-`*`region`*이라는 서비스 역할
- 서비스 역할에 연결된 `AWSRDSCustomIamRolePolicy`라는 이름의 액세스 정책

CloudFormation을 사용하여 IAM을 구성하는 방법

1. https://console.aws.amazon.com/cloudformation에서 CloudFormation 콘솔을 엽니다.
2. 스택 생성 마법사를 시작하고 Create Stack(스택 생성)을 선택합니다.
3. Create stack(스택 생성) 페이지에서 다음을 수행합니다.
   a. Prepare template(템플릿 준비)에서 Template is ready(템플릿 준비가 완료되었습니다)를 선택합니다.
   b. Template source(템플릿 소스)로 Upload a template file(템플릿 파일 업로드)을 선택합니다.
   c. 파일 선택의 경우 custom-oracle-iam.json을 찾아 선택합니다.

d. 다음을 선택합니다.

4. 스택 세부 정보 지정 페이지에서 다음 작업을 수행합니다.

   a. Stack name(스택 이름)에 **custom-oracle-iam**을 입력합니다.

   b. Next(다음)를 선택합니다.

5. 스택 옵션 구성 페이지에서 다음을 선택합니다.

6. custom-oracle-iam 검토 페이지에서 다음을 수행합니다.

   a. 사용자 지정 이름을 사용하여 AWS CloudFormation이 IAM 리소스를 생성할 수 있음에 동의합니다 확인란을 선택합니다.

   b. Submit(제출)을 선택합니다.

   CloudFormation은 RDS Custom for Oracle에 필요한 IAM 역할을 생성합니다. 왼쪽 패널의 custom-oracle-iam에 CREATE_COMPLETE이 표시되면 다음 단계를 진행합니다.

7. 왼쪽 패널에서 custom-oracle-iam을 선택합니다. 오른쪽 패널에서 다음을 수행합니다.

   a. 스택 정보를 선택합니다. 스택에는 arn:aws:cloudformation:***region***:***account-no***:stack/custom-oracle-iam/***identifier*** 형식으로 된 ID가 있습니다.

   b. 리소스를 선택합니다. 다음과 같은 모양이어야 합니다.

      - AWSRDSCustomInstanceProfile-***region***이라는 이름의 인스턴스 프로파일
      - AWSRDSCustomInstanceRole-***region***이라는 이름의 서비스 역할

      RDS Custom DB 인스턴스를 생성할 경우 인스턴스 프로파일 ID를 제공해야 합니다.

수동으로 IAM 역할과 인스턴스 프로파일 생성

CloudFormation을 사용하는 방법이 가장 쉽습니다. 하지만 IAM을 수동으로도 구성할 수 있습니다. 수동으로 설정하려면 다음을 수행합니다.

- 1단계: IAM 역할 AWSRDSCustomInstanceRoleForRdsCustomInstance 생성.
- 2단계: AWSRDSCustomInstanceRoleForRdsCustomInstance에 액세스 정책 추가.
- 2단계: AWSRDSCustomInstanceRoleForRdsCustomInstance에 액세스 정책 추가.
- 4단계: AWSRDSCustomInstanceRoleForRdsCustomInstance를 AWSRDSCustomInstanceProfile에 추가.

1단계: IAM 역할 AWSRDSCustomInstanceRoleForRdsCustomInstance 생성

이 단계에서는 이름 지정 형식을 사용하여 역할을 생성합니다AWSRDSCustomInstanceRole-*region*. Amazon EC2에서는 신뢰 정책을 사용하여 역할을 위임할 수 있습니다. 다음 예시에서는 DB 인스턴스를 생성하려는 환경 변수 $REGION을 AWS 리전으로 설정했다고 가정합니다.

```
aws iam create-role \
  --role-name AWSRDSCustomInstanceRole-$REGION \
  --assume-role-policy-document '{
    "Version": "2012-10-17",
      "Statement": [
        {
          "Action": "sts:AssumeRole",
          "Effect": "Allow",
          "Principal": {
              "Service": "ec2.amazonaws.com"
          }
        }
      ]
    }'
```

2단계: AWSRDSCustomInstanceRoleForRdsCustomInstance에 액세스 정책 추가

인라인 정책을 IAM 역할에 포함시켰을 때 인라인 정책은 역할의 액세스(권한) 정책의 일환으로 사용됩니다. 사용자는 Amazon EC2가 메시지를 보내고 받으며 다양한 작업을 수행할 수 있도록 허용하는 AWSRDSCustomIamRolePolicy 정책을 생성하게 됩니다.

다음 예제에서는 이름이 AWSRDSCustomIamRolePolicy인 액세스 정책을 생성하여 AWSRDSCustomInstanceRole-*region* IAM 역할에 추가합니다. 이 예제에서는 다음과 같은 환경 변수를 설정했다고 가정합니다.

$REGION

이 변수를 DB 인스턴스를 생성할 AWS 리전으로 설정합니다.

$ACCOUNT_ID

이 변수를 사용자의 AWS 계정 번호로 설정합니다.

$KMS_KEY

이 변수를 RDS Custom DB 인스턴스에 사용할 AWS KMS key의 Amazon 리소스 이름(ARN)으로 설정합니다. 두 개 이상의 KMS 키를 지정하려면 문 ID(시드) 11의 Resources 섹션에 추가합니다.

```
aws iam put-role-policy \
  --role-name AWSRDSCustomInstanceRole-$REGION \
  --policy-name AWSRDSCustomIamRolePolicy \
  --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "1",
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeAssociation",
                "ssm:GetDeployablePatchSnapshotForInstance",
                "ssm:GetDocument",
                "ssm:DescribeDocument",
                "ssm:GetManifest",
                "ssm:GetParameter",
                "ssm:GetParameters",
                "ssm:ListAssociations",
                "ssm:ListInstanceAssociations",
                "ssm:PutInventory",
                "ssm:PutComplianceItems",
                "ssm:PutConfigurePackageResult",
                "ssm:UpdateAssociationStatus",
                "ssm:UpdateInstanceAssociationStatus",
                "ssm:UpdateInstanceInformation",
                "ssm:GetConnectionStatus",
                "ssm:DescribeInstanceInformation",
                "ssmmessages:CreateControlChannel",
                "ssmmessages:CreateDataChannel",
                "ssmmessages:OpenControlChannel",
                "ssmmessages:OpenDataChannel"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "2",
            "Effect": "Allow",
            "Action": [
                "ec2messages:AcknowledgeMessage",
                "ec2messages:DeleteMessage",
                "ec2messages:FailMessage",
```

```
                "ec2messages:GetEndpoint",
                "ec2messages:GetMessages",
                "ec2messages:SendReply"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "3",
            "Effect": "Allow",
            "Action": [
                "logs:PutRetentionPolicy",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams",
                "logs:DescribeLogGroups",
                "logs:CreateLogStream",
                "logs:CreateLogGroup"
            ],
            "Resource": [
                "arn:aws:logs:'$REGION':'$ACCOUNT_ID':log-group:rds-custom-instance*"
            ]
        },
        {
            "Sid": "4",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::do-not-delete-rds-custom-*/*"
            ]
        },
        {
            "Sid": "5",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": [
                "*"
            ],
```

```
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": [
                        "RDSCustomForOracle/Agent"
                    ]
                }
            }
        },
        {
            "Sid": "6",
            "Effect": "Allow",
            "Action": [
                "events:PutEvents"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "7",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret"
            ],
            "Resource": [
                "arn:aws:secretsmanager:'$REGION':'$ACCOUNT_ID':secret:do-not-delete-
rds-custom-*",
                "arn:aws:secretsmanager:'$REGION':'$ACCOUNT_ID':secret:rds-custom!
oracle-do-not-delete-*"
            ]
        },
        {
           "Sid": "8",
           "Effect": "Allow",
           "Action": [
             "s3:ListBucketVersions"
           ],
           "Resource": [
             "arn:aws:s3:::do-not-delete-rds-custom-*"
           ]
         },
         {
            "Sid": "9",
```

```
            "Effect": "Allow",
            "Action": "ec2:CreateSnapshots",
            "Resource": [
                "arn:aws:ec2:*:*:instance/*",
                "arn:aws:ec2:*:*:volume/*"
            ],
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/AWSRDSCustom": "custom-oracle"
                }
            }
        },
        {
            "Sid": "10",
            "Effect": "Allow",
            "Action": "ec2:CreateSnapshots",
            "Resource": [
                "arn:aws:ec2:*::snapshot/*"
            ]
        },
        {
            "Sid": "11",
            "Effect": "Allow",
            "Action": [
              "kms:Decrypt",
              "kms:GenerateDataKey"
            ],
            "Resource": [
              "arn:aws:kms:'$REGION':'$ACCOUNT_ID':key/'$KMS_KEY'"
            ]
        },
        {
            "Sid": "12",
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "ec2:CreateAction": [
                        "CreateSnapshots"
                    ]
                }
            }
        },
```

```
        {
            "Sid": "13",
            "Effect": "Allow",
            "Action": [
                "sqs:SendMessage",
                "sqs:ReceiveMessage",
                "sqs:DeleteMessage",
                "sqs:GetQueueUrl"
            ],
            "Resource": "arn:aws:sqs:'$REGION':'$ACCOUNT_Id':do-not-delete-rds-custom-
*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/AWSRDSCustom": "custom-oracle"
                }
            }
        }
    ]
}'
```

3단계: RDS Custom 인스턴스 프로파일 AWSRDSCustomInstanceProfile 생성

인스턴스 프로파일은 단일 IAM 역할을 포함하는 컨테이너입니다. RDS Custom은 인스턴스 프로파일을 사용하여 인스턴스에 역할을 전달합니다.

CLI를 사용하여 역할을 생성하면 역할과 인스턴스 프로파일이 별개의 작업으로 생성되며 이름은 각각 다를 수 있습니다. 다음과 같이 IAM 인스턴스 프로파일을 생성하고 `AWSRDSCustomInstanceProfile-`*`region`* 형식을 사용하여 이름을 지정합니다. 다음 예시에서는 DB 인스턴스를 생성하려는 환경 변수 `$REGION`을 AWS 리전으로 설정했다고 가정합니다.

```
aws iam create-instance-profile \
    --instance-profile-name AWSRDSCustomInstanceProfile-$REGION
```

4단계: AWSRDSCustomInstanceRoleForRdsCustomInstance를 AWSRDSCustomInstanceProfile에 추가

이전에 생성한 인스턴스 프로파일에 IAM 역할을 추가합니다. 다음 예시에서는 DB 인스턴스를 생성하려는 환경 변수 `$REGION`을 AWS 리전으로 설정했다고 가정합니다.

```
aws iam add-role-to-instance-profile \
    --instance-profile-name AWSRDSCustomInstanceProfile-$REGION \
    --role-name AWSRDSCustomInstanceRole-$REGION
```

5단계: IAM 사용자 또는 역할에 필요한 권한 부여

CEV 또는 RDS Custom DB 인스턴스를 생성하는 IAM 보안 주체(사용자 또는 역할)에는 다음 정책 중 하나가 있어야 합니다.

- `AdministratorAccess` 정책
- Amazon S3 및 AWS KMS, CEV 생성 및 DB 인스턴스 생성에 필요한 권한이 있는 `AmazonRDSFullAccess` 정책

주제

- Amazon S3 및 AWS KMS에 필요한 IAM 권한
- CEV를 생성하는 데 필요한 IAM 권한
- CEV에서 DB 인스턴스를 생성하는 데 필요한 IAM 권한

Amazon S3 및 AWS KMS에 필요한 IAM 권한

CEV 또는 RDS Custom for Oracle DB 인스턴스를 생성하려면 IAM 보안 주체가 Amazon S3 및 AWS KMS에 액세스해야 합니다. 다음 샘플 JSON 정책은 필요한 권한을 부여합니다.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateS3Bucket",
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutBucketPolicy",
                "s3:PutBucketObjectLockConfiguration",
                "s3:PutBucketVersioning"
            ],
            "Resource": "arn:aws:s3:::do-not-delete-rds-custom-*"
        },
        {
            "Sid": "CreateKmsGrant",
            "Effect": "Allow",
            "Action": [
```

```
                "kms:CreateGrant",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        }
    ]
}
```

kms:CreateGrant 권한에 대한 자세한 내용은 AWS KMS key 관리 섹션을 참조하세요.

CEV를 생성하는 데 필요한 IAM 권한

CEV를 생성하려면 IAM 보안 주체는 다음과 같은 추가 권한이 필요합니다.

```
s3:GetObjectAcl
s3:GetObject
s3:GetObjectTagging
s3:ListBucket
mediaimport:CreateDatabaseBinarySnapshot
```

다음 샘플 JSON 정책은 *my-custom-installation-files* 버킷 및 내용에 액세스하는 데 필요한 추가 권한을 부여합니다.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AccessToS3MediaBucket",
            "Effect": "Allow",
            "Action": [
                "s3:GetObjectAcl",
                "s3:GetObject",
                "s3:GetObjectTagging",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::my-custom-installation-files",
                "arn:aws:s3:::my-custom-installation-files/*"
            ]
```

```
        },
        {
            "Sid": "PermissionForByom",
            "Effect": "Allow",
            "Action": [
                "mediaimport:CreateDatabaseBinarySnapshot"
            ],
            "Resource": "*"
        }
    ]
}
```

S3 버킷 정책을 사용하여 발신자 계정에 Amazon S3에 대해 유사한 권한을 부여할 수 있습니다.

CEV에서 DB 인스턴스를 생성하는 데 필요한 IAM 권한

기존 CEV에서 RDS Custom for Oracle DB 인스턴스를 생성하려면 IAM 보안 주체에 다음과 같은 추가 권한이 필요합니다.

```
iam:SimulatePrincipalPolicy
cloudtrail:CreateTrail
cloudtrail:StartLogging
```

다음 샘플 JSON 정책은 IAM 역할 및 로그 정보를 검증하는 데 필요한 권한을 AWS CloudTrail에 부여합니다.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ValidateIamRole",
            "Effect": "Allow",
            "Action": "iam:SimulatePrincipalPolicy",
            "Resource": "*"
        },
        {
            "Sid": "CreateCloudTrail",
            "Effect": "Allow",
            "Action": [
```

```
                "cloudtrail:CreateTrail",
                "cloudtrail:StartLogging"
            ],
            "Resource": "arn:aws:cloudtrail:*:*:trail/do-not-delete-rds-custom-*"
        }
    ]
}
```

6단계: RDS Custom for Oracle용 VPC 구성

RDS Custom DB 인스턴스는 Amazon EC2 인스턴스 또는 Amazon RDS 인스턴스와 마찬가지로 Amazon VPC 서비스를 기반으로 하는 Virtual Private Cloud(VPC)에 있습니다. 자체 VPC를 제공하고 구성해야 RDS Custom for SQL Server와 달리, RDS Custom for Oracle은 액세스 제어 목록이나 보안 그룹을 생성하지 않습니다. 사용자의 자체 보안 그룹, 서브넷, 라우팅 테이블을 연결해야 합니다.

CloudFormation 또는 수동 프로세스를 사용하여 Virtual Private Cloud(VPC)를 구성할 수 있습니다.

> **Important**
>
> AWS CloudFormation을 사용하여 RDS Custom for Oracle 환경을 구성하는 것이 좋습니다. 이 방법은 가장 쉽고 오류가 가장 적습니다.

주제

- CloudFormation을 사용하여 VPC 구성(권장)
- 수동으로 RDS Custom for Oracle용 VPC 구성

CloudFormation을 사용하여 VPC 구성(권장)

다른 RDS Custom 엔진의 VPC를 이미 구성했고 기존 VPC를 재사용하려는 경우 이 단계를 건너뜁니다. 이 섹션에서는 다음과 같이 가정합니다.

- 기존에 CloudFormation을 사용하여 IAM 인스턴스 프로파일과 역할을 생성했습니다.
- 라우팅 테이블 ID를 알고 있습니다.

  DB 인스턴스가 프라이빗 인스턴스가 되려면 프라이빗 서브넷에 있어야 합니다. 서브넷이 프라이빗이 되려면 기본 인터넷 게이트웨이가 있는 라우팅 테이블과 연결되어 있지 않아야 합니다. 자세한 내용은 Amazon VPC 사용 설명서의 라우팅 테이블 구성을 참조하세요.

VPC에 CloudFormation 템플릿을 사용할 경우 다음과 같은 리소스가 생성됩니다.

- 프라이빗 VPC
- 이름이 `rds-custom-private`인 서브넷 그룹
- DB 인스턴스가 종속 AWS 서비스와 통신하는 데 사용하는 다음 VPC 엔드포인트:
  - `com.amazonaws.region.ec2messages`
  - `com.amazonaws.region.events`
  - `com.amazonaws.region.logs`
  - `com.amazonaws.region.monitoring`
  - `com.amazonaws.region.s3`
  - `com.amazonaws.region.secretsmanager`
  - `com.amazonaws.region.ssm`
  - `com.amazonaws.region.ssmmessages`

  다중 AZ 배포를 생성하는 경우:

  - `com.amazonaws.region.sqs`

> **Note**
>
> 기존 계정을 사용하는 복잡한 네트워킹 설정의 경우 액세스 권한이 아직 없다면 종속 서비스에 대한 액세스를 수동으로 구성하는 것이 좋습니다. 자세한 내용은 VPC가 종속 AWS 서비스에 액세스할 수 있는지 확인 섹션을 참조하세요.

CloudFormation을 사용하여 VPC를 구성하는 방법

1. https://console.aws.amazon.com/cloudformation에서 CloudFormation 콘솔을 엽니다.
2. 스택 생성 마법사를 시작하고 스택 생성을 선택한 다음, 새 리소스 사용(표준)을 선택합니다.
3. Create stack(스택 생성) 페이지에서 다음을 수행합니다.
   a. Prepare template(템플릿 준비)에서 Template is ready(템플릿 준비가 완료되었습니다)를 선택합니다.
   b. Template source(템플릿 소스)로 Upload a template file(템플릿 파일 업로드)을 선택합니다.
   c. 파일 선택(Choose file)에서 `custom-vpc.json`을 찾아 선택합니다.
   d. Next(다음)를 선택합니다.

4. 스택 세부 정보 지정 페이지에서 다음 작업을 수행합니다.

   a. Stack name(스택 이름)에 **custom-vpc**를 입력합니다.

   b. 파라미터(Parameters)로 RDS Custom DB 인스턴스에 사용할 프라이빗 서브넷을 선택합니다.

   c. RDS Custom DB 인스턴스에 사용할 프라이빗 VPC ID를 선택합니다.

   d. 프라이빗 서브넷과 연결된 라우팅 테이블을 입력합니다.

   e. Next(다음)를 선택합니다.

5. Configure stack options(스택 옵션 구성) 페이지에서 Next(다음)를 선택합니다.

6. custom-vpc 검토 페이지에서 제출을 선택합니다.

   CloudFormation은 프라이빗 VPC를 구성합니다. 왼쪽 패널의 custom-vpc에 CREATE_COMPLETE이 표시되면 다음 단계를 진행합니다.

7. (선택 사항) VPC 세부 정보를 검토합니다. 스택 창에서 custom-vpc를 선택합니다. 오른쪽 창에서 다음을 수행합니다.

   a. 스택 정보를 선택합니다. 스택에는 arn:aws:cloudformation:***region***:***account-no***:stack/custom-vpc/***identifier*** 형식으로 된 ID가 있습니다.

   b. 리소스를 선택합니다. rds-custom-private이라는 이름의 서브넷 그룹, 그리고 vpce-***string***이라는 이름 지정 형식을 사용하는 몇몇 VPC 엔드포인트가 표시되어야 합니다. 각 엔드포인트는 RDS Custom이 통신을 수행해야 하는 AWS 서비스에 해당합니다. 자세한 내용은 VPC가 종속 AWS 서비스에 액세스할 수 있는지 확인 섹션을 참조하세요.

   c. 파라미터를 선택합니다. 스택을 생성했을 때 지정한 프라이빗 서브넷, 프라이빗 VPC, 라우팅 테이블이 표시되어야 합니다. DB 인스턴스를 생성할 경우, VPC ID 및 서브넷 그룹을 제공해야 합니다.

수동으로 RDS Custom for Oracle용 VPC 구성

AWS CloudFormation을 사용하여 VPC 생성을 자동화하지 않고 VPC를 수동으로 구성할 수 있습니다. 이 옵션은 기존 리소스를 사용하는 복잡한 네트워킹 설정이 있는 경우에 가장 적합할 수 있습니다.

주제

- VPC가 종속 AWS 서비스에 액세스할 수 있는지 확인
- 인스턴스 메타데이터 서비스 구성

VPC가 종속 AWS 서비스에 액세스할 수 있는지 확인

RDS Custom은 DB 인스턴스에서 다른 AWS 서비스로 통신을 전송합니다. RDS Custom DB 인스턴스를 생성하는 서브넷에서 다음 서비스에 액세스할 수 있는지 확인합니다.

- Amazon CloudWatch(`com.amazonaws.region.monitoring`)
- Amazon CloudWatch Logs(`com.amazonaws.region.logs`)
- Amazon CloudWatch Events(`com.amazonaws.region.events`)
- Amazon EC2(`com.amazonaws.region.ec2` 및 `com.amazonaws.region.ec2messages`)
- Amazon EventBridge(`com.amazonaws.region.events`)
- Amazon S3(`com.amazonaws.region.s3`)
- AWS Secrets Manager (`com.amazonaws.region.secretsmanager`)
- AWS Systems Manager (`com.amazonaws.region.ssm` 및 `com.amazonaws.region.ssmmessages`)

다중 AZ 배포를 생성하는 경우

- Amazon Simple Queue Service(`com.amazonaws.region.sqs`)

필요한 서비스와 RDS Custom이 통신할 수 없는 경우 다음 이벤트가 게시됩니다.

```
Database instance in incompatible-network. SSM Agent connection not available. Amazon
 RDS can't connect to the dependent AWS services.
```

```
Database instance in incompatible-network. Amazon RDS can't connect to dependent AWS
 services. Make sure port 443 (HTTPS) allows outbound connections, and try again.
 "Failed to connect to the following services: s3 events"
```

`incompatible-network` 오류를 방지하려면 RDS Custom DB 인스턴스와 AWS 서비스 간의 통신과 관련된 VPC 구성 요소가 다음 요구 사항을 충족하는지 확인합니다.

- DB 인스턴스는 포트 443에서 다른 AWS 서비스로 아웃바운드 연결을 생성할 수 있습니다.
- VPC는 RDS Custom DB 인스턴스에서 시작된 요청에 대한 수신 응답을 허용합니다.
- RDS Custom는 각 AWS 서비스에 대한 엔드포인트의 도메인 이름을 정확하게 확인할 수 있습니다.

다른 RDS Custom DB 엔진의 VPC를 이미 구성한 경우 해당 VPC를 재사용하고 이 프로세스를 건너뛸 수 있습니다.

인스턴스 메타데이터 서비스 구성

인스턴스가 다음을 수행할 수 있는지 확인하세요.

- 인스턴스 메타데이터 서비스 버전 2(IMDSv2)를 사용하여 인스턴스 메타데이터 서비스에 액세스합니다.
- 포트 80(HTTP)을 통한 IMDS 링크 IP 주소로의 아웃바운드 통신을 허용합니다.
- IMDSv2 링크인 `http://169.254.169.254`에서 인스턴스 메타데이터를 요청합니다.

자세한 내용은 https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/configuring-instance-metadata-service.htmlAmazon EC2 사용 설명서의 IMDSv2 사용을 참조하세요.

RDS Custom for Oracle 자동화는 기본 Amazon EC2 인스턴스에서 `HttpTokens=enabled`를 설정하여 기본적으로 IMDSv2를 사용합니다. 하지만 필요할 경우 IMDSv1을 사용할 수 있습니다. 자세한 내용을 알아보려면 Amazon EC2 사용 설명서의 인스턴스 메타데이터 옵션 구성을 참조하세요.

사용자 지정 엔진 버전의 Amazon RDS Custom for Oracle으로 작업

Amazon RDS Custom for Oracle의 사용자 지정 엔진 버전(CEV)은 데이터베이스 엔진 및 특정 Amazon Machine Image(AMI)의 바이너리 볼륨 스냅샷입니다. RDS Custom for Oracle에서는 기본적으로 RDS Custom에서 관리되는 최신 AMI를 사용하지만, 사용자가 이전 CEV에서 사용된 AMI를 지정하는 것도 가능합니다. 데이터베이스 설치 파일은 Amazon S3에 저장되며, RDS Custom은 이 설치 파일과 AMI를 사용하여 CEV를 생성합니다.

주제

- CEV 생성 준비
- CEV 생성
- CEV 상태 수정
- Amazon RDS Custom for Oracle의 CEV 세부 정보 보기
- CEV 삭제

CEV 생성 준비

CEV를 생성하려면 다음 릴리스 중 하나에 대해 Amazon S3 버킷에 저장된 설치 파일 및 패치를 사용해야 합니다.

- Oracle Database 19c
- Oracle Database 18c
- Oracle Database 12c 릴리스 2(12.2)
- Oracle Database 12c 릴리스 1(12.1)

예를 들어, 2021년 4월 RU/RUR for Oracle Database 19c 또는 설치 파일과 패치의 유효한 조합을 사용할 수 있습니다. RDS Custom for Oracle이 지원하는 버전 및 리전에 대한 자세한 내용은 RDS Oracle을 지원하는 RDS Custom을 참조하세요.

주제

- 1단계(선택 사항): 매니페스트 템플릿 다운로드
- 2단계: Oracle Software Delivery Cloud에서 데이터베이스 설치 파일 및 패치 다운로드
- 3단계: Amazon S3에 설치 파일 업로드
- 4단계(선택 사항): 여러 AWS 계정에 걸쳐 S3의 설치 미디어 공유

- 5단계: CEV 매니페스트 준비
- 6단계(선택 사항): CEV 매니페스트 검증
- 7단계: 필요한 IAM 권한 추가

1단계(선택 사항): 매니페스트 템플릿 다운로드

CEV 매니페스트는 CEV의 데이터베이스 설치 .zip 파일 목록이 포함된 JSON 문서입니다. CEV를 생성하려면 다음을 수행합니다.

1. CEV에 포함하려는 Oracle 데이터베이스 설치 파일을 식별합니다.
2. 설치 파일을 다운로드합니다.
3. 설치 파일을 나열하는 JSON 매니페스트를 생성합니다.

RDS Custom for Oracle은 지원되는 각 Oracle 데이터베이스 릴리스에 대한 권장 .zip 파일이 포함된 JSON 매니페스트 템플릿을 제공합니다. 예를 들어, 다음 템플릿은 19.17.0.0.0 RU용입니다.

```
{
    "mediaImportTemplateVersion": "2020-08-14",
    "databaseInstallationFileNames": [
        "V982063-01.zip"
    ],
    "opatchFileNames": [
        "p6880880_190000_Linux-x86-64.zip"
    ],
    "psuRuPatchFileNames": [
        "p34419443_190000_Linux-x86-64.zip",
        "p34411846_190000_Linux-x86-64.zip"
    ],
    "otherPatchFileNames": [
        "p28852325_190000_Linux-x86-64.zip",
        "p29997937_190000_Linux-x86-64.zip",
        "p31335037_190000_Linux-x86-64.zip",
        "p32327201_190000_Linux-x86-64.zip",
        "p33613829_190000_Linux-x86-64.zip",
        "p34006614_190000_Linux-x86-64.zip",
        "p34533061_190000_Linux-x86-64.zip",
        "p34533150_190000_Generic.zip",
        "p28730253_190000_Linux-x86-64.zip",
        "p29213893_1917000DBRU_Generic.zip",
        "p33125873_1917000DBRU_Linux-x86-64.zip",
```

```
        "p34446152_1917000DBRU_Linux-x86-64.zip"
    ]
}
```

각 템플릿에는 패치 다운로드 지침, .zip 파일의 URL, 파일 체크섬이 포함된 관련 Readme 파일이 있습니다. 이러한 템플릿을 그대로 사용하거나 자체 패치를 사용하여 수정할 수 있습니다. 템플릿을 검토하려면 로컬 디스크에 custom-oracle-manifest.zip을 다운로드한 후 파일 보관 애플리케이션을 사용하여 파일을 엽니다. 자세한 내용은 5단계: CEV 매니페스트 준비 단원을 참조하십시오.

2단계: Oracle Software Delivery Cloud에서 데이터베이스 설치 파일 및 패치 다운로드

CEV에 사용할 설치 파일을 확인했으면 해당 파일을 로컬 시스템에 다운로드하세요. Oracle Database 설치 파일 및 패치는 Oracle Software Delivery Cloud에서 호스팅됩니다. 각 CEV에는 Oracle Database 19c 또는 Oracle Database 12c 릴리스 2(12.2), 그리고 패치 목록(선택 사항)이 필요합니다.

Oracle Database용 데이터베이스 설치 파일을 다운로드하려면

1. https://edelivery.oracle.com/으로 이동하여 로그인합니다.
2. 검색 상자에 **`Oracle Database Enterprise Edition`** 또는 **`Oracle Database Standard Edition 2`**를 입력하고 검색을 선택합니다.
3. 다음 기본 릴리스 중 하나를 선택합니다.

데이터베이스 버전	Enterprise Edition	Standard Edition 2
Oracle Database 19c	DLP: Oracle Database 19c Enterprise Edition 19.3.0.0.0(Oracle Database Enterprise Edition)	DLP: Oracle Database 19c Standard Edition 2 19.3.0.0.0(Oracle Database Standard Edition 2)
Oracle Database 18c	DLP: Oracle Database 18c Enterprise Edition 18.0.0.0.0(Oracle Database Enterprise Edition)	DLP: Oracle Database Standard Edition 2 18.0.0.0.0(Oracle Database Standard Edition 2)
Oracle Database 12c 릴리스 2(12.2.0.1)	DLP: Oracle Database 12c Enterprise Edition 12.2.0.1.0(Oracle Database Enterprise Edition)	DLP: Oracle Database Standard Edition 2 12.2.0.1.0(Oracle Database Standard Edition 2)

데이터베이스 버전	Enterprise Edition	Standard Edition 2
Oracle Database 12c 릴리스 1(12.1.0.2)	DLP: Oracle Database 12c Enterprise Edition 12.1.0.2.0(Oracle Database Enterprise Edition)	DLP: Oracle Database Standard Edition 2 12.1.0.2.0(Oracle Database Standard Edition 2)

4. Continue(계속)를 선택합니다.
5. Download Queue(대기열 다운로드) 확인란을 선택 취소합니다.
6. 기본 릴리스에 해당하는 옵션을 선택합니다.
   - Oracle Database 19.3.0.0.0 - 장기 지원 릴리스.
   - Oracle Database 18.0.0.0.0
   - Oracle Database 12.2.0.1.0.
   - Oracle Database 12.1.0.2.0.
7. Platform/Languages(플랫폼/언어)에서 Linux x86-64를 선택합니다.
8. 계속을 선택한 다음 Oracle 라이선스 계약에 서명합니다.
9. 데이터베이스 릴리스에 해당하는 .zip 파일을 선택합니다.

데이터베이스 릴리스 및 에디션	Zip 파일	SHA-256 해시
19c EE 및 SE2	V982063-01.zip	BA8329C757133DA313ED3B6D7F86C5AC42CD9970A28BF2E6233F3235233AA8D8
18c EE 및 SE2	V978967-01.zip	C96A4FD768787AF98272008833FE10B172691CF84E42816B138C12D4DE63AB96
12.2.0.1 EE 및 SE2	V839960-01.zip	96ED97D21F15C1AC0CCE3749DA6C3DAC7059BB60672D76B008103FC754D22DDE
12.1.0.2 EE	V46095-01_1of2.zip	V46095-01_1of2.zip용 31FDC2AF41687B4E547A3A18F796424D8C1AF36406D2160F65B0AF6A9CD47355

데이터베이스 릴리스 및 에디션	Zip 파일	SHA-256 해시
	V46095-01_2of2.zip	V46095-01_2of2.zip용 03DA14F5E875304B28F0F3BB02AF0EC33227885B99C9865DF70749D1E220ACCD
12.1.0.2 SE2	V77388-01_1of2.zip V77388-01_2of2.zip	73873369753230F5A0921F95ACEADB591388CB06ED72A7F3AEA7BCBCEA2403BC (V77388-01_1of2.zip용) 2492E1BE1E3E3531DA83D0843C09C08E435AC8CEFD9A00C0DF56BE4F15CEEBF3 (V77388-01_2of2.zip용)

10. 원하는 Oracle 패치를 updates.oracle.com 또는 support.oracle.com에서 로컬 시스템으로 다운로드합니다. 다음 위치에서 패치의 URL을 찾을 수 있습니다.

    - 1단계(선택 사항): 매니페스트 템플릿 다운로드에서 다운로드한 .zip 파일의 Readme 파일
    - Release notes for Amazon Relational Database Service (Amazon RDS) for Oracle(Amazon Relational Database Service(RDS) for Oracle에 대한 릴리스 노트)의 Release Update(RU)에 나열된 패치

3단계: Amazon S3에 설치 파일 업로드

AWS CLI를 사용하여 Oracle 설치 및 패치 파일을 Amazon S3 업로드할 수 있습니다. 단, 설치 파일이 포함된 S3 버킷이 CEV와 동일한 AWS 리전에 있어야 합니다.

이 섹션의 예제에서는 다음 자리 표시자를 사용합니다.

- *install-or-patch-file.zip* - Oracle 설치 미디어 파일입니다. 예를 들어, p32126828_190000_Linux-x86-64.zip은 패치입니다.
- *amzn-s3-demo-destination-bucket* - 업로드된 설치 파일에 지정된 Amazon S3 버킷입니다.
- *123456789012/cev1* - Amazon S3 버킷의 접두사(선택 사항)입니다.
- *amzn-s3-demo-source-bucket* - 필요에 따라 파일을 단계화할 수 있는 Amazon S3 버킷입니다.

주제

- 3a단계: S3 버킷이 올바른 AWS 리전에 있는지 확인
- 3b단계: S3 버킷 정책에 올바른 권한이 있는지 확인
- 3c단계: cp 또는 sync 명령을 사용하여 파일 업로드
- 3d단계: S3 버킷의 파일 나열

3a단계: S3 버킷이 올바른 AWS 리전에 있는지 확인

S3 버킷이 `create-custom-db-engine-version` 명령을 실행하려는 AWS 리전에 있는지 확인합니다.

```
aws s3api get-bucket-location --bucket amzn-s3-demo-destination-bucket
```

3b단계: S3 버킷 정책에 올바른 권한이 있는지 확인

CEV는 처음부터 생성하거나 소스 CEV에서 생성할 수 있습니다. 소스 CEV에서 새 CEV를 생성하려는 경우 S3 버킷 정책에 올바른 권한이 있는지 확인하세요.

1. RDS Custom에서 예약한 S3 버킷을 식별합니다. 버킷 이름의 형식은 `do-not-delete-rds-custom-account-region-string`입니다. 예를 들어, 버킷의 이름은 `do-not-delete-rds-custom-123456789012-us-east-1-abc123EXAMPLE`일 수 있습니다.
2. S3 버킷 정책에 다음 권한이 추가되었는지 확인하세요. `do-not-delete-rds-custom-123456789012-us-east-1-abc123EXAMPLE`을 사용 중인 버킷의 이름으로 바꿉니다.

```
{
    "Sid": "AWSRDSCustomForOracleCustomEngineVersionGetObject",
    "Effect": "Allow",
    "Principal": {
        "Service": "custom.rds.amazonaws.com"
    },
    "Action": [
        "s3:GetObject",
        "s3:GetObjectTagging"
    ],
    "Resource": "arn:aws:s3:::do-not-delete-rds-custom-123456789012-us-
east-1-abc123EXAMPLE/CustomEngineVersions/*"
}, ...
```

3c단계: cp 또는 sync 명령을 사용하여 파일 업로드

다음 옵션 중 하나를 선택합니다.

- `aws s3 cp`를 사용하여 .zip 파일 하나만 업로드합니다.

  각 설치 .zip 파일을 별도로 업로드합니다. .zip 파일을 하나의 .zip 파일로 결합하지 마세요.
- `aws s3 sync`를 사용하여 디렉터리를 업로드합니다.

Example

다음 예제에서는 RDS Custom Amazon S3 버킷에 있는 *123456789012/cev1* 폴더에 *install-or-patch-file.zip*을 업로드합니다. 업로드하려는 각 .zip 파일에 대해 `aws s3` 명령을 별도로 실행합니다.

대상 LinuxmacOS, 또는Unix:

```
aws s3 cp install-or-patch-file.zip \
    s3://amzn-s3-demo-destination-bucket/123456789012/cev1/
```

Windows의 경우:

```
aws s3 cp install-or-patch-file.zip ^
    s3://amzn-s3-demo-destination-bucket/123456789012/cev1/
```

Example

다음 예제에서는 로컬 *cev1* 폴더의 파일을 Amazon S3 버킷의 *123456789012/cev1* 폴더로 업로드합니다.

대상 LinuxmacOS, 또는Unix:

```
aws s3 sync cev1 \
    s3://amzn-s3-demo-destination-bucket/123456789012/cev1/
```

Windows의 경우:

```
aws s3 sync cev1 ^
    s3://amzn-s3-demo-destination-bucket/123456789012/cev1/
```

Example

다음 예제에서는 *amzn-s3-demo-source-bucket*에 있는 모든 파일을 Amazon S3 버킷의 *123456789012/cev1* 폴더로 업로드합니다.

대상 LinuxmacOS, 또는Unix:

```
aws s3 sync s3://amzn-s3-demo-source-bucket/ \
    s3://amzn-s3-demo-destination-bucket/123456789012/cev1/
```

Windows의 경우:

```
aws s3 sync s3://amzn-s3-demo-source-bucket/ ^
    s3://amzn-s3-demo-destination-bucket/123456789012/cev1/
```

3d단계: S3 버킷의 파일 나열

다음 예시에서는 `s3 ls` 명령을 사용하여 RDS Custom Amazon S3 버킷에 있는 파일을 나열합니다.

```
aws s3 ls \
    s3://amzn-s3-demo-destination-bucket/123456789012/cev1/
```

4단계(선택 사항): 여러 AWS 계정에 걸쳐 S3의 설치 미디어 공유

이 섹션에서는 업로드한 Oracle 설치 파일이 포함된 Amazon S3 버킷이 미디어 버킷입니다. 조직은 한 AWS 계정에서 여러 AWS 리전을 사용할 수 있습니다. 그렇다면 하나를 사용하고 싶을 것입니다.AWS 계정미디어 버킷 및 다른 미디어 버킷 채우기AWS 계정CEV를 만들 수 있습니다. 미디어 버킷에 폴더를 공유하고 싶지 않은 경우 다음 섹션으로 건너뜁니다.

이 섹션에서는 다음과 같이 가정합니다.

- 미디어 버킷을 생성한 계정과 CEV를 생성하려는 다른 계정에 액세스할 수 있습니다.
- AWS 리전에서 CEV를 하나만 만들려고 합니다. 여러 리전을 사용하려는 경우 각 리전에 미디어 버킷을 생성하세요.
- CLI를 사용 중입니다. Amazon S3 콘솔을 사용 중인 경우 다음 단계를 조정합니다.

AWS 계정에서 공유할 미디어 버킷을 구성하려면

1. 설치 미디어를 업로드한 S3 버킷이 포함된 AWS 계정에 로그인합니다.

2. 먼저 빈 JSON 정책 템플릿이나 적용할 수 있는 기존 정책으로 시작하세요.

   다음 명령은 기존 정책을 검색하여 *my-policy.json*으로 저장합니다. 이 예제에서 설치 파일이 포함된 S3 버킷의 이름은 *amzn-s3-demo-bucket*입니다.

```
aws s3api get-bucket-policy \
    --bucket amzn-s3-demo-bucket \
    --query Policy \
    --output text > my-policy.json
```

3. 미디어 버킷 권한을 다음과 같이 편집합니다.

   - 템플릿의 `Resource` 요소에서 Oracle Database 설치 파일을 업로드한 S3 버킷을 지정합니다.
   - `Principal` 요소에서 CEV를 생성하는 데 사용할 모든 AWS 계정에 대한 ARN을 지정합니다. S3 버킷 허용 목록에 루트, 사용자 또는 역할을 추가할 수 있습니다. 자세한 내용은 AWS Identity and Access Management 사용 설명서에서 IAM 식별자를 참조하세요.

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "GrantAccountsAccess",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::account-1:root",
                    "arn:aws:iam::account-2:user/user-name-with-path",
                    "arn:aws:iam::account-3:role/role-name-with-path",
                    ...
                ]
            },
            "Action": [
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectTagging",
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
```

```
            ]
        }
    ]
}
```

4. 미디어 버킷에 다음 정책을 연결합니다.

   다음 예제에서 *amzn-s3-demo-bucket*은 설치 파일이 포함된 S3 버킷의 이름이고 *my-policy.json*은 JSON 파일의 이름입니다.

   ```
   aws s3api put-bucket-policy \
       --bucket amzn-s3-demo-bucket \
       --policy file://my-policy.json
   ```

5. CEV를 만들려는 AWS 계정에 로그인합니다.
6. 이 계정이 생성한 AWS 계정의 미디어 버킷에 액세스할 수 있는지 확인합니다.

   ```
   aws s3 ls --query "Buckets[].Name"
   ```

   자세한 내용은 AWS CLI 명령 참조에서 aws s3 ls를 참조합니다.
7. CEV 생성의 단계에 따라 CEV를 생성합니다.

5단계: CEV 매니페스트 준비

CEV 매니페스트는 다음이 포함된 JSON 문서입니다.

- (필수) Amazon S3에 업로드한 설치 .zip 파일 목록. RDS Custom은 매니페스트에 나열된 순서대로 패치를 적용합니다.
- (선택 사항) Oracle base, Oracle home, UNIX/Linux 사용자 및 그룹의 ID 및 이름에 기본값이 아닌 값을 설정하는 설치 파라미터. 기존 CEV 또는 기존 DB 인스턴스의 설치 파라미터는 수정할 수 없다는 점에 유의하세요. 또한 설치 파라미터의 설정이 다른 경우 한 CEV에서 다른 CEV로 업그레이드할 수 없습니다.

샘플 CEV 매니페스트는 1단계(선택 사항): 매니페스트 템플릿 다운로드에서 다운로드한 JSON 템플릿을 참조하세요. CEV 매니페스트 예에서 샘플을 검토할 수도 있습니다.

주제

- CEV 매니페스트 내의 JSON

- CEV 매니페스트 생성
- CEV 매니페스트 예

CEV 매니페스트 내의 JSON

다음 표에서는 매니페스트의 JSON 필드를 설명합니다.

JSON 필드	설명
`MediaImportTemplateVersion`	CEV 매니페스트 버전으로, 날짜 형식은 `YYYY-MM-DD` 입니다.
`databaseInstallationFileNames`	데이터베이스의 설치 파일이 나열된 목록입니다.
`opatchFileNames`	Oracle DB 엔진에 사용된 OPatch 설치 관리자가 나열된 목록으로, 하나의 값만 유효합니다. `opatchFileNames` 값은 `p6880880_` 으로 시작해야 합니다.
`psuRuPatchFileNames`	이 데이터베이스에 대한 PSU 및 RU 패치입니다. **Important** `psuRuPatchFileNames` 가 포함되어 있는 경우 `opatchFileNames` 는 필수입니다. `opatchFileNames` 값은 `p6880880_` 으로 시작해야 합니다.
`OtherPatchFileNames`	PSU 및 RU 패치 목록에 없는 패치입니다. RDS Custom은 PSU 및 RU 패치를 적용한 후에 이러한 패치를 적용합니다. **Important** `OtherPatchFileNames` 가 포함되어 있는 경우 `opatchFileNames` 는 필수입니다. `opatchFileNames` 값은 `p6880880_` 으로 시작해야 합니다.

JSON 필드	설명
`installationParameters`	Oracle base, Oracle home, UNIX/Linux 사용자 및 그룹의 ID 및 이름의 기본이 아닌 설정입니다. 다음과 같은 파라미터를 설정할 수 있습니다. oracleBase Oracle 바이너리가 설치되는 디렉터리입니다. 이 디렉터리는 파일을 저장하는 바이너리 볼륨의 탑재 지점입니다. Oracle base 디렉터리에는 여러 Oracle home이 포함될 수 있습니다. 예를 들어 `/home/oracle/oracle.19.0.0.0.ru-2020-04.rur-2020-04.r1.EE.1` 이 Oracle home 디렉터리 중 하나인 경우 `/home/oracle` 이 Oracle base 디렉터리입니다. 사용자 지정 Oracle base 디렉터리는 심볼 링크가 아닙니다. Oracle base를 지정하지 않을 경우 기본 디렉터리는 `/rdsdbbin` 입니다. oracleHome Oracle 데이터베이스 바이너리가 설치되는 디렉터리입니다. 예를 들어 `/home/oracle/` 을 Oracle base로 지정하는 경우 `/home/oracle/oracle.19.0.0.0.ru-2020-04.rur-2020-04.r1.EE.1/` 을 Oracle home으로 지정할 수 있습니다. 사용자 지정 Oracle home 디렉터리는 심볼 링크가 아닙니다. Oracle home 값은`$ORACLE_HOME` 환경 변수에서 참조됩니다. Oracle home을 지정하지 않을 경우 기본 이름 지정 형식은 `/rdsdbbin/oracle.` *major-engine-version* `.custom.r1.` *engine-edition* `.1`입니다. unixUname Oracle 소프트웨어를 소유한 UNIX 사용자의 이름입니다. RDS Custom은 로컬 데이터베이스 명령을 실행할 때 이 사용자를 가정합니다. `unixUid` 및 `unixUname` 을 모두 지정하는 경우

JSON 필드	설명
	RDS Custom은 사용자가 없으면 사용자를 생성한 다음 초기 UID와 동일하지 않을 경우 사용자에게 UID를 할당합니다. 기본 사용자 이름은 `rdsdb`입니다. unixUid Oracle 소프트웨어를 소유한 UNIX 사용자의 ID(UID)입니다. `unixUid` 및 `unixUname` 을 모두 지정하는 경우 RDS Custom은 사용자가 없으면 사용자를 생성한 다음 초기 UID와 동일하지 않을 경우 사용자에게 UID를 할당합니다. 기본 UID는 `61001`입니다. 이것은 사용자 `rdsdb`의 UID입니다. unixGroupName UNIX 그룹 이름입니다. Oracle 소프트웨어를 소유한 UNIX 사용자는 이 그룹에 속합니다. 기본 그룹 이름은 `rdsdb`입니다. unixGroupId UNIX 사용자가 속한 UNIX 그룹의 ID입니다. 기본 그룹 ID는 `1000`입니다. 이것은 그룹 `rdsdb`의 ID입니다.

각 Oracle Database 릴리스에는 지원되는 설치 파일 목록이 다릅니다. CEV 매니페스트를 생성할 때 Oracle용 RDS Custom에서 지원하는 파일만 지정해야 합니다. 그렇지 않으면 오류가 발생하여 CEV 생성이 실패합니다. Release notes for Amazon Relational Database Service (Amazon RDS) for Oracle(Amazon Relational Database Service(RDS) for Oracle에 대한 릴리스 노트)에 나열된 모든 패치가 지원됩니다.

CEV 매니페스트 생성

CEV 매니페스트를 생성하려면

1. 적용할 모든 설치 파일을 나열할 순서대로 나열합니다.
2. 설치 파일을 CEV 매니페스트 내의 JSON에 설명된 JSON 필드와 상호 연관시킵니다.
3. 다음 중 하나를 수행하세요.

- CEV 매니페스트를 JSON 텍스트 파일로 생성합니다.
- 콘솔에서 CEV를 만들 때 CEV 매니페스트 템플릿을 편집합니다. 자세한 내용은 CEV 생성을 참조하세요.

CEV 매니페스트 예

다음 예는 다양한 Oracle 데이터베이스 릴리스에 대한 CEV 매니페스트 파일을 보여줍니다. 매니페스트 형태로, JSON 필드를 포함할 경우 비워둘 것이 아닌지 확인하세요. 예를 들어, 다음 매니페스트는 `otherPatchFileNames`가 비어 있으므로 유효하지 않습니다.

```
{
    "mediaImportTemplateVersion": "2020-08-14",
    "databaseInstallationFileNames": [
        "V982063-01.zip"
    ],
    "opatchFileNames": [
        "p6880880_190000_Linux-x86-64.zip"
    ],
    "psuRuPatchFileNames": [
        "p32126828_190000_Linux-x86-64.zip"
    ],
    "otherPatchFileNames": [
    ]
}
```

주제

- Sample CEV manifest for Oracle Database 12c Release 1 (12.1)
- Sample CEV manifest for Oracle Database 12c Release 2 (12.2)
- Sample CEV manifest for Oracle Database 18c
- Sample CEV manifest for Oracle Database 19c

Example Oracle Database 12c Release 1(12.1) 샘플 CEV 매니페스트

Oracle Database 12c 릴리스 1(12.1)용 2021년 7월 PSU의 다음 예에서 RDS Custom은 지정된 순서대로 패치를 적용합니다. 따라서 RDS Custom은 p32768233, p32876425, p18759211 순으로 적용합니다. 이 예제에서는 UNIX/Linux 사용자 및 그룹, Oracle home, Oracle base의 새 값을 설정합니다.

```
{
    "mediaImportTemplateVersion":"2020-08-14",
    "databaseInstallationFileNames":[
        "V46095-01_1of2.zip",
        "V46095-01_2of2.zip"
    ],
    "opatchFileNames":[
        "p6880880_121010_Linux-x86-64.zip"
    ],
    "psuRuPatchFileNames":[
        "p32768233_121020_Linux-x86-64.zip"
    ],
    "otherPatchFileNames":[
        "p32876425_121020_Linux-x86-64.zip",
        "p18759211_121020_Linux-x86-64.zip",
        "p19396455_121020_Linux-x86-64.zip",
        "p20875898_121020_Linux-x86-64.zip",
        "p22037014_121020_Linux-x86-64.zip",
        "p22873635_121020_Linux-x86-64.zip",
        "p23614158_121020_Linux-x86-64.zip",
        "p24701840_121020_Linux-x86-64.zip",
        "p25881255_121020_Linux-x86-64.zip",
        "p27015449_121020_Linux-x86-64.zip",
        "p28125601_121020_Linux-x86-64.zip",
        "p28852325_121020_Linux-x86-64.zip",
        "p29997937_121020_Linux-x86-64.zip",
        "p31335037_121020_Linux-x86-64.zip",
        "p32327201_121020_Linux-x86-64.zip",
        "p32327208_121020_Generic.zip",
        "p17969866_12102210119_Linux-x86-64.zip",
        "p20394750_12102210119_Linux-x86-64.zip",
        "p24835919_121020_Linux-x86-64.zip",
        "p23262847_12102201020_Linux-x86-64.zip",
        "p21171382_12102201020_Generic.zip",
        "p21091901_12102210720_Linux-x86-64.zip",
        "p33013352_12102210720_Linux-x86-64.zip",
        "p25031502_12102210720_Linux-x86-64.zip",
        "p23711335_12102191015_Generic.zip",
        "p19504946_121020_Linux-x86-64.zip"
    ],
    "installationParameters": {
        "unixGroupName": "dba",
        "unixGroupId": 12345,
```

```
        "unixUname": "oracle",
        "unixUid": 12345,
        "oracleHome": "/home/oracle/oracle.12.1.0.2",
        "oracleBase": "/home/oracle"
    }
}
```

Example Oracle Database 12c Release 2(12.2) 샘플 CEV 매니페스트

Oracle Database 12c 릴리스 2(12.2)용 2021년 10월 PSU의 다음 예에서 RDS Custom은 p33261817, p33192662, p29213893 등을 적용합니다. 이 예제에서는 UNIX/Linux 사용자 및 그룹, Oracle home, Oracle base의 새 값을 설정합니다.

```
{
    "mediaImportTemplateVersion":"2020-08-14",
    "databaseInstallationFileNames":[
        "V839960-01.zip"
    ],
    "opatchFileNames":[
        "p6880880_122010_Linux-x86-64.zip"
    ],
    "psuRuPatchFileNames":[
        "p33261817_122010_Linux-x86-64.zip"
    ],
    "otherPatchFileNames":[
        "p33192662_122010_Linux-x86-64.zip",
        "p29213893_122010_Generic.zip",
        "p28730253_122010_Linux-x86-64.zip",
        "p26352615_12201211019DBOCT2021RU_Linux-x86-64.zip",
        "p23614158_122010_Linux-x86-64.zip",
        "p24701840_122010_Linux-x86-64.zip",
        "p25173124_122010_Linux-x86-64.zip",
        "p25881255_122010_Linux-x86-64.zip",
        "p27015449_122010_Linux-x86-64.zip",
        "p28125601_122010_Linux-x86-64.zip",
        "p28852325_122010_Linux-x86-64.zip",
        "p29997937_122010_Linux-x86-64.zip",
        "p31335037_122010_Linux-x86-64.zip",
        "p32327201_122010_Linux-x86-64.zip",
        "p32327208_122010_Generic.zip"
    ],
    "installationParameters": {
        "unixGroupName": "dba",
```

```
        "unixGroupId": 12345,
        "unixUname": "oracle",
        "unixUid": 12345,
        "oracleHome": "/home/oracle/oracle.12.2.0.1",
        "oracleBase": "/home/oracle"
    }
}
```

Example Oracle Database 18c 샘플 CEV 매니페스트

Oracle Database 18c용 2021년 10월 PSU의 다음 예에서 RDS Custom은 p33261817, p33192662, p29213893 등을 적용합니다. 이 예제에서는 UNIX/Linux 사용자 및 그룹, Oracle home, Oracle base의 새 값을 설정합니다.

```
{
    "mediaImportTemplateVersion":"2020-08-14",
    "databaseInstallationFileNames":[
        "V978967-01.zip"
    ],
    "opatchFileNames":[
        "p6880880_180000_Linux-x86-64.zip"
    ],
    "psuRuPatchFileNames":[
        "p32126855_180000_Linux-x86-64.zip"
    ],
    "otherPatchFileNames":[
        "p28730253_180000_Linux-x86-64.zip",
        "p27539475_1813000DBRU_Linux-x86-64.zip",
        "p29213893_180000_Generic.zip",
        "p29374604_1813000DBRU_Linux-x86-64.zip",
        "p29782284_180000_Generic.zip",
        "p28125601_180000_Linux-x86-64.zip",
        "p28852325_180000_Linux-x86-64.zip",
        "p29997937_180000_Linux-x86-64.zip",
        "p31335037_180000_Linux-x86-64.zip",
        "p31335142_180000_Generic.zip"
    ]
    "installationParameters": {
        "unixGroupName": "dba",
        "unixGroupId": 12345,
        "unixUname": "oracle",
        "unixUid": 12345,
        "oracleHome": "/home/oracle/18.0.0.0.ru-2020-10.rur-2020-10.r1",
```

```
        "oracleBase": "/home/oracle/"
    }
}
```

Example Oracle Database 19c 샘플 CEV 매니페스트

다음 Oracle Database 19c 예제에서 RDS Custom은 p32126828을 적용한 다음 p29213893, p29782284 순으로 적용합니다. 이 예제에서는 UNIX/Linux 사용자 및 그룹, Oracle home, Oracle base의 새 값을 설정합니다.

```
{
    "mediaImportTemplateVersion": "2020-08-14",
    "databaseInstallationFileNames": [
        "V982063-01.zip"
    ],
    "opatchFileNames": [
        "p6880880_190000_Linux-x86-64.zip"
    ],
    "psuRuPatchFileNames": [
        "p32126828_190000_Linux-x86-64.zip"
    ],
    "otherPatchFileNames": [
        "p29213893_1910000DBRU_Generic.zip",
        "p29782284_1910000DBRU_Generic.zip",
        "p28730253_190000_Linux-x86-64.zip",
        "p29374604_1910000DBRU_Linux-x86-64.zip",
        "p28852325_190000_Linux-x86-64.zip",
        "p29997937_190000_Linux-x86-64.zip",
        "p31335037_190000_Linux-x86-64.zip",
        "p31335142_190000_Generic.zip"
    ],
    "installationParameters": {
        "unixGroupName": "dba",
        "unixGroupId": 12345,
        "unixUname": "oracle",
        "unixUid": 12345,
        "oracleHome": "/home/oracle/oracle.19.0.0.0.ru-2020-04.rur-2020-04.r1.EE.1",
        "oracleBase": "/home/oracle"
    }
}
```

6단계(선택 사항): CEV 매니페스트 검증

필요에 따라 `json.tool` Python 스크립트를 실행하여 매니페스트가 유효한 JSON 파일인지 확인할 수 있습니다. 예를 들어, `manifest.json`이라는 이름의 CEV 매니페스트를 포함하는 디렉터리로 변경하는 경우 다음 명령을 실행하면 됩니다.

```
python -m json.tool < manifest.json
```

7단계: 필요한 IAM 권한 추가

CEV를 생성하는 IAM 보안 주체에는 5단계: IAM 사용자 또는 역할에 필요한 권한 부여에 설명된 필수 정책이 있는지 확인합니다.

CEV 생성

AWS Management Console 또는 AWS CLI를 사용하여 CEV를 생성할 수 있습니다. 멀티테넌트 또는 비 멀티테넌트 아키텍처를 지정합니다. 자세한 내용은 멀티테넌트 아키텍처 고려 사항 단원을 참조하십시오.

일반적으로 CEV 생성에는 약 2시간이 소요됩니다. CEV를 만든 후 이를 사용하여 RDS Custom DB 인스턴스를 만들 수 있습니다. 자세한 내용은 RDS Custom for Oracle DB 인스턴스 생성 및 RDS Custom for Oracle DB 인스턴스 업그레이드 단원을 참조하세요.

> **Note**
>
> DB 인스턴스가 현재 Oracle Linux 7.9를 사용하는 경우 Oracle Linux 8을 사용하는 최신 AMI를 사용하는 새 CEV를 만듭니다. 그런 다음 새 CEV를 사용하도록 인스턴스를 수정합니다.

CEV 생성에 대해 다음과 같은 요구 사항 및 제한 사항에 유의합니다.

- 단, 설치 파일이 포함된 Amazon S3 버킷이 CEV와 동일한 AWS 리전에 있어야 하며, 그렇지 않으면 생성 프로세스가 실패합니다.
- CEV 이름은 *major-engine-version*.*customized_string*과 같은 `19.cdb_cev1` 형식이어야 합니다.
- CEV 이름은 1~50개의 영숫자, 밑줄, 대시 또는 마침표를 포함해야 합니다.
- CEV 이름에는 `19..cdb_cev1`에서와 같이 연속된 마침표를 포함할 수 없습니다.

콘솔

CEV를 생성하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 사용자 지정 엔진 버전(Custom engine versions)을 선택합니다.

   사용자 지정 엔진 버전(Custom engine versions) 페이지에는 현재 존재하는 모든 CEV가 표시됩니다. CEV를 생성한 적이 없으면 페이지가 비어 있습니다.
3. 사용자 지정 엔진 버전 생성(Create custom engine version)을 선택합니다.
4. 엔진 옵션에서 다음을 수행합니다.

   a. 엔진 유형(Engine type)으로 Oracle을 선택합니다.

   b. 아키텍처 설정에서 필요에 따라 멀티테넌트 아키텍처를 선택하여 DB 엔진 `custom-oracle-ee-cdb` 또는 `custom-oracle-se2-cdb`를 사용하는 Oracle 멀티테넌트 CEV를 생성합니다. 멀티테넌트 CEV가 있는 RDS Custom for Oracle CDB만 생성할 수 있습니다. 이 옵션을 선택하지 않는 경우 CEV는 엔진 `custom-oracle-ee` 또는 `custom-oracle-se2`를 사용하는 비CDB입니다.

   > **Note**
   >
   > 선택한 아키텍처는 CEV의 영구적인 특성입니다. 나중에 다른 아키텍처를 사용하도록 CEV를 수정할 수 없습니다.

   c. 다음 옵션 중 하나를 선택합니다.

   - 새 CEV 생성 - 처음부터 CEV를 생성합니다. 이 경우 데이터베이스 바이너리를 지정하는 JSON 매니페스트를 지정해야 합니다.
   - 소스에서 CEV 생성 - 복사할 CEV 지정에서 소스 CEV로 사용할 기존 CEV를 선택합니다. 이 경우 새 Amazon Machine Image(AMI)를 지정할 수 있지만 다른 데이터베이스 바이너리를 지정할 수는 없습니다.

   d. 엔진 버전에서 메이저 엔진 버전을 선택합니다.
5. 버전 세부 정보에서 다음을 수행합니다.

   a. 사용자 지정 엔진 버전 이름에 유효한 이름을 입력합니다. 예를 들어, **`19.cdb_cev1`**을 이름으로 입력할 수 있습니다.

b. (선택 사항) CEV에 대한 설명을 입력합니다.

6. 설치 미디어에서 다음을 수행합니다.

a. (선택 사항) AMI ID의 경우 필드를 비워 최신 서비스 제공 AMI를 사용하거나 이전에 CEV를 생성할 때 사용한 AMI를 입력합니다. 유효한 AMI ID를 가져오려면 다음 방법 중 하나를 사용합니다.

- 콘솔의 왼쪽 탐색 창에서 사용자 지정 엔진 버전을 선택하고 CEV 이름을 선택합니다. CEV에서 사용하는 AMI ID는 구성 탭에 표시됩니다.
- AWS CLI에서 `describe-db-engine-versions` 명령을 사용합니다. `ImageID`에 대한 출력을 검색합니다.

b. 매니페스트 파일의 S3 위치(S3 location of manifest files)에서 3단계: Amazon S3에 설치 파일 업로드에 지정한 대로 Amazon S3 버킷의 위치를 입력합니다. 예를 들면 **`s3://my-custom-installation-files/123456789012/cev1/`**를(을) 입력합니다.

> **Note**
>
> CEV를 생성하는 AWS 리전은 S3 버킷과 동일한 리전이어야 합니다.

c. (새 CEV 생성의 경우만 해당) CEV 매니페스트의 경우 CEV 매니페스트 생성에서 생성한 JSON 매니페스트를 입력합니다.

7. KMS 키 섹션에서 키 ARN 입력을 선택하여 사용 가능한 AWS KMS 키를 나열합니다. 그런 다음 목록에서 KMS 키를 선택합니다.

   AWS KMS 키는 RDS Custom에 필수입니다. 자세한 내용은 1단계: 대칭 암호화 AWS KMS 키 생성 또는 재사용 단원을 참조하십시오.

8. (선택 사항) 새 태그 추가를 선택하여 CEV의 키-값 페어를 생성합니다.

9. Create custom engine version(사용자 지정 엔진 버전 생성)을 선택합니다.

   JSON 매니페스트의 형식이 잘못되면 콘솔에 CEV 매니페스트를 검증하는 중 오류 발생 메시지가 표시됩니다. 문제를 해결한 후 다시 시도해 주세요.

사용자 지정 엔진 버전(Custom engine versions) 페이지가 표시됩니다. CEV가 상태가 생성 중(Creating)으로 표시됩니다. CEV를 생성하는 데 약 2시간이 소요됩니다.

AWS CLI

AWS CLI를 사용하여 CEV를 생성하려면 create-custom-db-engine-version 명령을 실행합니다.

다음 옵션이 필요합니다.

- `--engine` - 엔진 유형을 지정합니다. CDB에서 `custom-oracle-ee-cdb` 또는 `custom-oracle-se2-cdb`를 지정합니다. 비CDB에서 `custom-oracle-ee` 또는 `custom-oracle-se2`를 지정합니다. CDB는 `custom-oracle-ee-cdb` 또는 `custom-oracle-se2-cdb`로 생성한 CEV에서만 생성할 수 있습니다. 비CDB는 `custom-oracle-ee` 또는 `custom-oracle-se2`로 생성한 CEV에서만 생성할 수 있습니다.
- `--engine-version` - 엔진 버전을 지정합니다. 형식은 *major-engine-version.customized_string*입니다. CEV 이름은 1~50개의 영숫자, 밑줄, 대시 또는 마침표를 포함해야 합니다. CEV 이름에는 `19..cdb_cev1`에서와 같이 연속된 마침표를 포함할 수 없습니다.
- `--kms-key-id` - AWS KMS key을 지정합니다.
- `--manifest` - *manifest_json_string* 또는 `--manifest file:`*file_name*을 지정합니다. *manifest_json_string*에서 줄 바꿈 문자는 허용되지 않습니다. 백슬래시(\)를 접두사에 부여하여 JSON 코드에서 큰 따옴표(")를 이스케이프해야 합니다.

  다음 예제에서는 5단계: CEV 매니페스트 준비의 19c에 대한 *manifest_json_string*을 보여줍니다. 이 예제에는 Oracle base, Oracle home, UNIX/Linux 사용자 및 그룹의 ID와 이름의 새 값을 설정합니다. 이 문자열을 복사하는 경우 명령에 붙여넣기 전에 모든 줄 바꿈 문자를 삭제하세요.

  ```
  "{\"mediaImportTemplateVersion\": \"2020-08-14\",
  \"databaseInstallationFileNames\": [\"V982063-01.zip\"],
  \"opatchFileNames\": [\"p6880880_190000_Linux-x86-64.zip\"],
  \"psuRuPatchFileNames\": [\"p32126828_190000_Linux-x86-64.zip\"],
  \"otherPatchFileNames\": [\"p29213893_1910000DBRU_Generic.zip\",
  \"p29782284_1910000DBRU_Generic.zip\",\"p28730253_190000_Linux-
  x86-64.zip\",\"p29374604_1910000DBRU_Linux-x86-64.zip\",
  \"p28852325_190000_Linux-x86-64.zip\",\"p29997937_190000_Linux-x86-64.zip
  \",\"p31335037_190000_Linux-x86-64.zip\",\"p31335142_190000_Generic.zip
  \"]\"installationParameters\":{ \"unixGroupName\":\"dba\",
  \ \"unixUname\":\"oracle\", \ \"oracleHome\":\"/home/oracle/
  oracle.19.0.0.0.ru-2020-04.rur-2020-04.r1.EE.1\", \ \"oracleBase\":\"/
  home/oracle/\"}}"
  ```

- `--database-installation-files-s3-bucket-name` - 3단계: Amazon S3에 설치 파일 업로드에서 지정한 것과 동일한 버킷 이름을 지정합니다. `create-custom-db-engine-version`을 실행하는 AWS 리전은 Amazon S3 버킷과 동일한 리전이어야 합니다.

다음 옵션도 지정할 수 있습니다.

- `--description` - CEV에 대한 설명을 지정합니다.
- `--database-installation-files-s3-prefix` - 3단계: Amazon S3에 설치 파일 업로드에서 지정한 폴더 이름을 지정합니다.
- `--image-id` - 재사용할 AMI ID를 지정합니다. 유효한 ID를 찾으려면 `describe-db-engine-versions` 명령을 실행한 다음 `ImageID`의 출력을 검색하세요. 기본적으로 RDS Custom for Oracle은 가장 최신 AMI를 사용합니다.

다음 예시에서는 `19.cdb_cev1`이라는 Oracle 멀티테넌트 CEV를 생성합니다. 예제에서는 사용 가능한 최신 AMI를 사용하는 대신 기존 AMI를 재사용합니다. CEV의 이름이 주요 엔진 버전 번호로 시작하는지 확인하세요.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-custom-db-engine-version \
    --engine custom-oracle-se2-cdb \
    --engine-version 19.cdb_cev1 \
    --database-installation-files-s3-bucket-name us-east-1-123456789012-custom-
installation-files \
    --database-installation-files-s3-prefix 123456789012/cev1 \
    --kms-key-id my-kms-key \
    --description "test cev" \
    --manifest manifest_string \
    --image-id ami-012a345678901bcde
```

Windows의 경우:

```
aws rds create-custom-db-engine-version ^
    --engine custom-oracle-se2-cdb ^
    --engine-version 19.cdb_cev1 ^
    --database-installation-files-s3-bucket-name us-east-1-123456789012-custom-
installation-files ^
```

```
    --database-installation-files-s3-prefix 123456789012/cev1 ^
    --kms-key-id my-kms-key ^
    --description "test cev" ^
    --manifest manifest_string ^
    --image-id ami-012a345678901bcde
```

Example

describe-db-engine-versions 명령을 사용하여 CEV에 대한 세부 정보를 확인합니다.

```
aws rds describe-db-engine-versions \
    --engine custom-oracle-se2-cdb \
    --include-all
```

다음 부분 샘플 출력은 엔진, 파라미터 그룹, 매니페스트 및 기타 정보를 보여줍니다.

```
{
    "DBEngineVersions": [
        {
            "Engine": "custom-oracle-se2-cdb",
            "EngineVersion": "19.cdb_cev1",
            "DBParameterGroupFamily": "custom-oracle-se2-cdb-19",
            "DBEngineDescription": "Containerized Database for Oracle Custom SE2",
            "DBEngineVersionDescription": "test cev",
            "Image": {
                "ImageId": "ami-012a345678901bcde",
                "Status": "active"
            },
            "ValidUpgradeTarget": [],
            "SupportsLogExportsToCloudwatchLogs": false,
            "SupportsReadReplica": true,
            "SupportedFeatureNames": [],
            "Status": "available",
            "SupportsParallelQuery": false,
            "SupportsGlobalDatabases": false,
            "MajorEngineVersion": "19",
            "DatabaseInstallationFilesS3BucketName": "us-east-1-123456789012-custom-
installation-files",
            "DatabaseInstallationFilesS3Prefix": "123456789012/cev1",
            "DBEngineVersionArn": "arn:aws:rds:us-east-1:123456789012:cev:custom-
oracle-se2-cdb/19.cdb_cev1/abcd12e3-4f5g-67h8-i9j0-k1234l56m789",
            "KMSKeyId": "arn:aws:kms:us-
east-1:732027699161:key/1ab2345c-6d78-9ef0-1gh2-3456i7j89k01",
```

```
            "CreateTime": "2023-03-07T19:47:58.131000+00:00",
            "TagList": [],
            "SupportsBabelfish": false,
...
```

CEV 생성 실패

CEV 생성이 실패하면 RDS Custom에서 `Creation failed for custom engine version` *`major-engine-version.cev_name`* 메시지와 함께 `RDS-EVENT-0198` 오류가 발생하며, 실패 관련 세부 정보가 표시됩니다. 예를 들어, 이벤트는 누락된 파일을 인쇄합니다.

실패한 CEV는 수정할 수 없습니다. 삭제하고 나서 실패 원인을 수정한 후에 CEV를 다시 만들 수 있습니다. CEV 생성에 실패한 이유 문제 해결에 대한 자세한 내용은 Oracle용 RDS Custom에 대한 사용자 지정 엔진 버전 생성 문제 해결 섹션을 참조하세요.

CEV 상태 수정

AWS Management Console 또는 AWS CLI를 사용하여 CEV를 수정하고, CEV 설명 또는 가용 상태를 수정할 수 있습니다. CEV는 다음 상태 값 중 하나를 가집니다.

- `available` - 이 CEV를 사용하여 새로운 RDS Custom DB 인스턴스를 생성하거나 DB 인스턴스를 업그레이드할 수 있습니다. 이 상태는 새로 생성된 CEV의 기본 상태입니다.
- `inactive` - 이 CEV에서는 RDS Custom 인스턴스를 생성하거나 업그레이드할 수 없습니다. 이 CEV에서는 DB 스냅샷을 복원하여 새로운 RDS Custom DB 인스턴스를 생성할 수 없습니다.

CEV를 지원되는 상태에서 다른 지원 상태로 변경할 수 있습니다. 상태를 변경하여 실수로 CEV를 사용하지 못하도록 하거나 중단된 CEV를 다시 사용하도록 할 수 있습니다. 예를 들어 CEV의 상태를 `available`에서 `inactive`로, `inactive`에서 `available`로 변경할 수 있습니다.

콘솔

CEV를 수정하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 사용자 지정 엔진 버전(Custom engine versions)을 선택합니다.
3. 수정하려는 설명 또는 상태가 지정된 CEV를 선택합니다.
4. 작업에서 수정을 선택합니다.
5. 다음을 원하는 대로 변경합니다.

- CEV 상태 설정(CEV status settings)에서 새로운 가용 상태를 선택합니다.
- 버전 설명(Version description)에 새로운 설명을 입력합니다.

6. CEV 수정(Modify CEV)을 선택합니다.

   CEV가 사용 중인 경우 콘솔에 CEV 상태를 수정할 수 없습니다(You can't modify the CEV status)가 표시됩니다. 문제를 해결한 후 다시 시도해 주세요.

사용자 지정 엔진 버전(Custom engine versions) 페이지가 표시됩니다.

AWS CLI

AWS CLI를 사용하여 CEV를 수정하려면 modify-custom-db-engine-version 명령을 실행합니다. describe-db-engine-versions 명령을 실행하여 수정할 CEV를 찾을 수 있습니다.

다음 옵션이 필요합니다.

- `--engine` *`engine-type`*(*`## ##`*은 `custom-oracle-ee`, `custom-oracle-se2`, `custom-oracle-ee-cdb`, `custom-oracle-se2-cdb`)
- `--engine-version` *`cev`*(*`cev`*는 수정하려는 사용자 지정 엔진 버전의 이름)
- `--status` *`status`*(*`status`*는 CEV에 할당하려는 가용 상태)

다음 예제에서는 이름이 `19.my_cev1`로 지정된 CEV를 현재 상태에서 `inactive`로 변경합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-custom-db-engine-version \
    --engine custom-oracle-se2 \
    --engine-version 19.my_cev1 \
    --status inactive
```

Windows의 경우:

```
aws rds modify-custom-db-engine-version ^
    --engine custom-oracle-se2 ^
    --engine-version 19.my_cev1 ^
    --status inactive
```

Amazon RDS Custom for Oracle의 CEV 세부 정보 보기

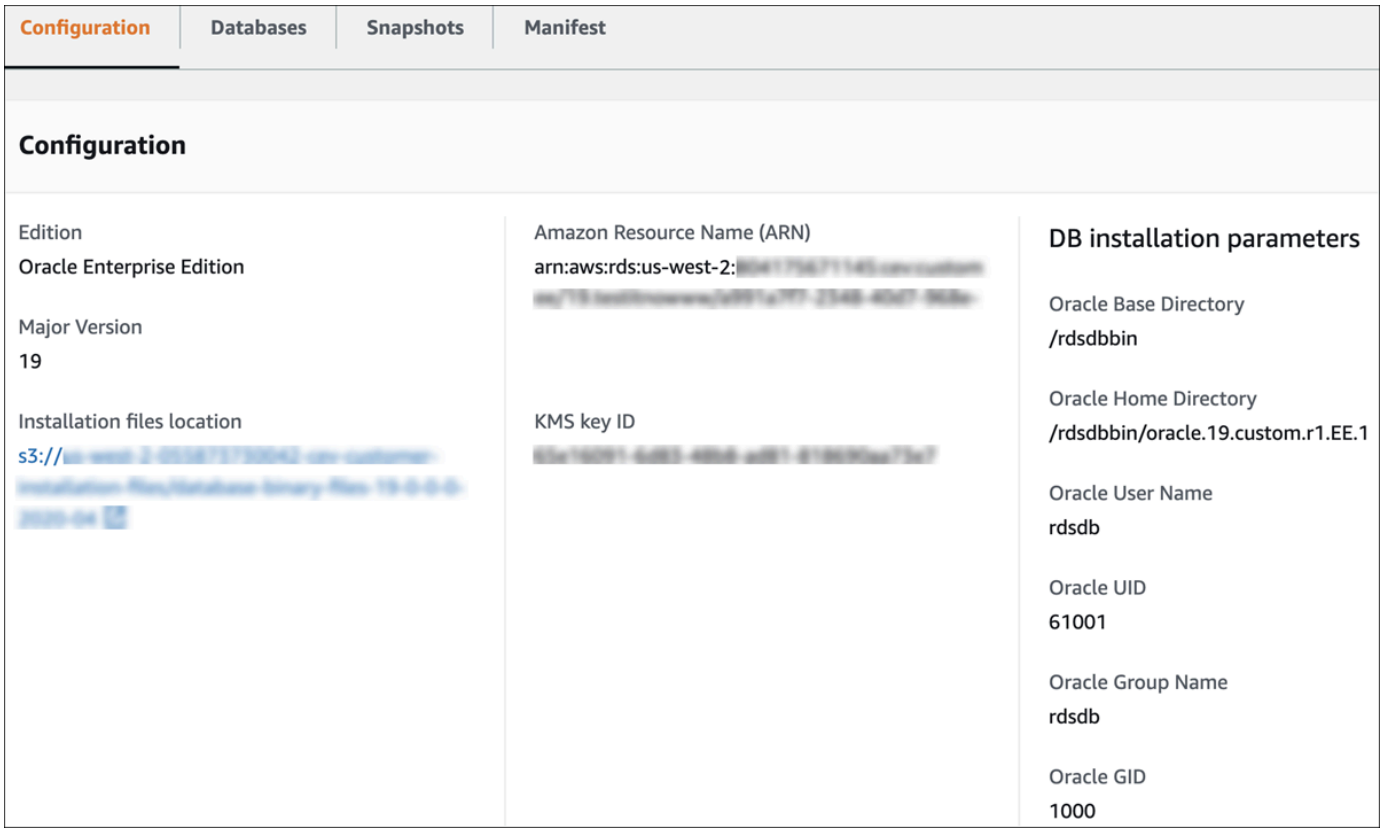
AWS Management Console 또는 AWS CLI를 사용하여 CEV 매니페스트 세부 정보와 CEV를 만드는 데 사용한 명령 관련 세부 정보를 볼 수 있습니다.

콘솔

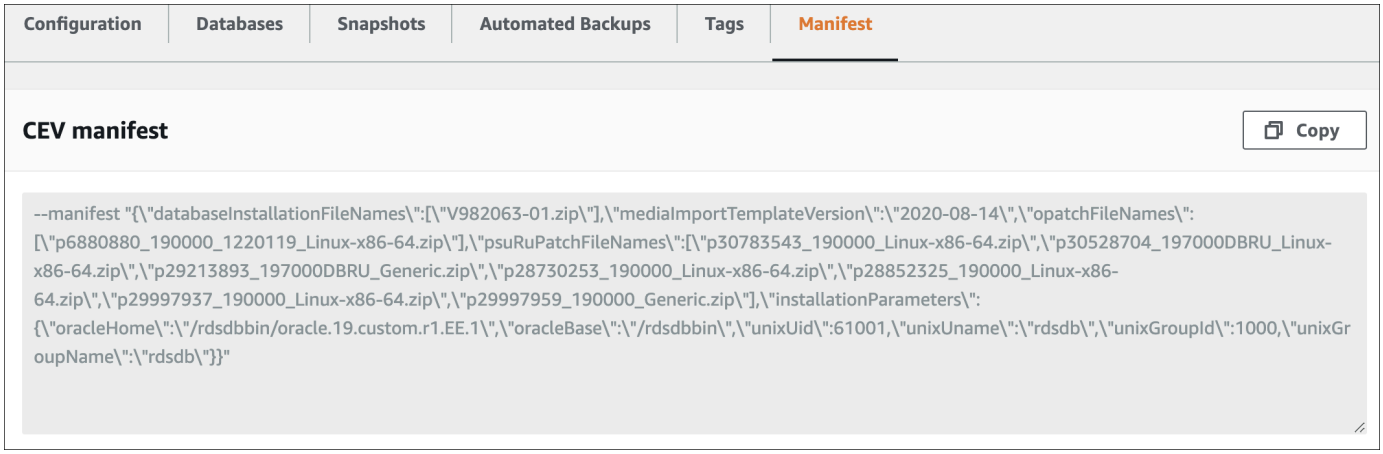
CEV 세부 정보를 보려면

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 사용자 지정 엔진 버전(Custom engine versions)을 선택합니다.

   사용자 지정 엔진 버전(Custom engine versions) 페이지에는 현재 존재하는 모든 CEV가 표시됩니다. CEV를 생성한 적이 없으면 페이지가 비어 있습니다.
3. 확인할 CEV의 이름을 선택합니다.
4. Configuration(구성)을 선택하면 매니페스트에 지정된 설치 파라미터를 볼 수 있습니다.



5. `create-custom-db-engine-version` 명령의`--manifest` 옵션에 지정된 설치 파라미터를 보려면 Manifest(매니페스트)를 선택합니다. 이 텍스트를 복사하고, 필요한 경우 값을 바꾸고, 새 명령에서 사용할 수 있습니다.



AWS CLI

AWS CLI를 사용하여 CEV에 대한 세부 정보를 보려면 describe-db-engine-versions 명령을 실행합니다.

다음 옵션이 필요합니다.

- `--engine` *`engine-type`*(*`## ##`*은 `custom-oracle-ee`, `custom-oracle-se2`, `custom-oracle-ee-cdb`, `custom-oracle-se2-cdb`)
- `--engine-version` *`major-engine-version.customized_string`*

다음 예제에서는 Enterprise Edition을 사용하는 비CDB CEV를 생성합니다. CEV의 `19.my_cev1` 이름이 메이저 엔진 버전 번호로 시작하며, 이 번호는 필수 항목입니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-engine-versions \
    --engine custom-oracle-ee \
    --engine-version 19.my_cev1
```

Windows의 경우:

```
aws rds describe-db-engine-versions ^
    --engine custom-oracle-ee ^
    --engine-version 19.my_cev1
```

다음 부분 샘플 출력은 엔진, 파라미터 그룹, 매니페스트 및 기타 정보를 보여줍니다.

```
"DBEngineVersions": [
    {
        "Engine": "custom-oracle-ee",
        "MajorEngineVersion": "19",
        "EngineVersion": "19.my_cev1",
        "DatabaseInstallationFilesS3BucketName": "us-east-1-123456789012-cev-customer-
installation-files",
        "DatabaseInstallationFilesS3Prefix": "123456789012/cev1",
        "CustomDBEngineVersionManifest": "{\n\"mediaImportTemplateVersion\":
 \"2020-08-14\",\n\"databaseInstallationFileNames\": [\n\"V982063-01.zip\"\n],
\n\"installationParameters\": {\n\"oracleBase\":\"/tmp\",\n\"oracleHome\":\"/
tmp/Oracle\"\n},\n\"opatchFileNames\": [\n\"p6880880_190000_Linux-x86-64.zip
\"\n],\n\"psuRuPatchFileNames\": [\n\"p32126828_190000_Linux-x86-64.zip
\"\n],\n\"otherPatchFileNames\": [\n\"p29213893_1910000DBRU_Generic.zip\",\n
\"p29782284_1910000DBRU_Generic.zip\",\n\"p28730253_190000_Linux-x86-64.zip\",\n
\"p29374604_1910000DBRU_Linux-x86-64.zip\",\n\"p28852325_190000_Linux-x86-64.zip\",
\n\"p29997937_190000_Linux-x86-64.zip\",\n\"p31335037_190000_Linux-x86-64.zip\",\n
\"p31335142_190000_Generic.zip\"\n]\n}\n",
        "DBParameterGroupFamily": "custom-oracle-ee-19",
        "DBEngineDescription": "Oracle Database server EE for RDS Custom",
        "DBEngineVersionArn": "arn:aws:rds:us-west-2:123456789012:cev:custom-oracle-
ee/19.my_cev1/0a123b45-6c78-901d-23e4-5678f901fg23",
        "DBEngineVersionDescription": "test",
        "KMSKeyId": "arn:aws:kms:us-east-1:123456789012:key/ab1c2de3-f4g5-6789-h012-
h3ijk4567l89",
        "CreateTime": "2022-11-18T09:17:07.693000+00:00",
        "ValidUpgradeTarget": [
        {
            "Engine": "custom-oracle-ee",
            "EngineVersion": "19.cev.2021-01.09",
            "Description": "test",
            "AutoUpgrade": false,
            "IsMajorVersionUpgrade": false
        }
]
```

CEV 삭제

AWS Management Console 또는 AWS CLI를 사용하여 CEV를 삭제할 수 있습니다. 일반적으로 삭제하는 데는 몇 분 정도 걸립니다.

CEV를 삭제하면 다음 중 어디에서도 CEV를 사용할 수 없습니다.

- RDS Custom DB 인스턴스
- RDS Custom DB 인스턴스의 스냅샷
- RDS Custom DB 인스턴스의 자동 백업

콘솔

CEV를 삭제하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 사용자 지정 엔진 버전(Custom engine versions)을 선택합니다.
3. 삭제하려는 설명 또는 상태가 지정된 CEV를 선택합니다.
4. 작업에 대해 삭제를 선택합니다.

   cev_name을 삭제하시겠습니까?(Delete cev_name?) 대화 상자가 표시됩니다.
5. **delete me**를 입력한 다음 삭제를 선택합니다.

   사용자 지정 엔진 버전(Custom engine versions) 페이지의 배너에서 CEV가 삭제 중임을 표시합니다.

AWS CLI

AWS CLI를 사용하여 CEV를 삭제하려면 delete-custom-db-engine-version 명령을 실행합니다.

다음 옵션이 필요합니다.

- `--engine` *`engine-type`*(*`## ##`*은 `custom-oracle-ee`, `custom-oracle-se2`, `custom-oracle-ee-cdb`, `custom-oracle-se2-cdb`)
- `--engine-version` *`cev`*(*`cev`*는 삭제할 사용자 지정 엔진 버전의 이름)

다음 예제에서는 이름이 `19.my_cev1`인 CEV를 삭제합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds delete-custom-db-engine-version \
    --engine custom-oracle-ee \
    --engine-version 19.my_cev1
```

Windows의 경우:

```
aws rds delete-custom-db-engine-version ^
    --engine custom-oracle-ee ^
    --engine-version 19.my_cev1
```

Amazon RDS Custom for Oracle용 DB 인스턴스 구성

RDS Custom DB 인스턴스를 생성한 다음 Secure Shell(SSH) 또는 AWS Systems Manager를 사용하여 연결할 수 있습니다.

RDS Custom for Oracle DB 인스턴스에 연결하고 로그인하는 방법에 대한 자세한 내용은 다음 주제를 참조하세요.

- 세션 관리자를 사용하여 RDS Custom DB 인스턴스에 연결
- SSH를 사용하여 RDS Custom DB 인스턴스에 연결
- RDS Custom for Oracle 데이터베이스에 SYS로 로그인

RDS Custom for Oracle DB 인스턴스 생성

AWS Management Console 또는 AWS CLI를 사용하여 Oracle DB 인스턴스용 Amazon RDS Custom을 생성할 수 있습니다. 생성 절차는 Amazon RDS DB 인스턴스를 생성하는 절차와 유사합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.

CEV 매니페스트에 설치 파라미터를 포함시킨 경우 DB 인스턴스는 지정한 Oracle base, Oracle home, UNIX/Linux 사용자 및 그룹의 ID와 이름을 사용합니다. 설치 중에 Oracle Database에서 생성되는 `oratab` 파일은 심볼 링크가 아닌 실제 설치 위치를 가리킵니다. RDS Custom for Oracle은 명령을 실행할 때 기본 사용자 `rdsdb`가 아닌 구성된 OS 사용자로 실행합니다. 자세한 내용은 5단계: CEV 매니페스트 준비 섹션을 참조하세요.

RDS Custom DB 인스턴스를 생성하거나 RDS Custom DB 인스턴스에 연결하려면 먼저 Amazon RDS Custom for Oracle을 위한 환경 설정 섹션에 나와 있는 작업을 완료하세요.

콘솔

RDS Custom for Oracle DB 인스턴스를 생성하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
3. 데이터베이스 생성을 선택합니다.
4. 데이터베이스 생성 방법 선택(Choose a database creation method)에서 표준 생성(Standard Create)을 선택합니다.
5. 엔진 옵션) 섹션에서 다음과 같이 실행합니다.

a. 엔진 유형(Engine type)으로 Oracle을 선택합니다.

b. 데이터베이스 관리 유형(Database management type)에서 Amazon RDS Custom을 선택합니다.

c. 아키텍처 설정에서 다음 중 하나를 수행합니다.

- 멀티테넌트 아키텍처를 선택하여 컨테이너 데이터베이스(CDB)를 생성합니다. 생성 시 CDB에는 PDB 시드와 초기 PDB가 하나씩 포함됩니다.

> **Note**
>
> 멀티테넌트 아키텍처 설정은 Oracle Database 19c에만 지원됩니다.

- 비 CDB를 생성하려면 멀티테넌트 아키텍처를 지웁니다. 비 CDB에는 PDB가 포함될 수 없습니다.

d. 에디션에서 Oracle Enterprise Edition 또는 Oracle Standard Edition 2를 선택합니다.

e. 사용자 지정 엔진 버전의 경우 RDS Custom의 기존 사용자 지정 엔진 버전(CEV)을 선택합니다. CEV의 형식은 *major-engine-version*.*customized_string*입니다. 예제 식별자는 `19.cdb_cev1`입니다.

이전 단계에서 멀티테넌트 아키텍처를 선택한 경우 `custom-oracle-ee-cdb` 또는 `custom-oracle-se2-cdb` 엔진 유형을 사용하는 CEV만 지정할 수 있습니다. 콘솔은 다른 엔진 유형으로 생성된 CEV를 필터링합니다.

6. 템플릿(Templates)에서 프로덕션(Production)을 선택합니다.

7. 설정 섹션에서 다음을 수행합니다.

a. DB 인스턴스 식별자에서 DB 인스턴스의 이름을 입력합니다.

b. 마스터 사용자 이름에 사용자 이름을 입력합니다. 나중에 콘솔에서 이 값을 검색할 수 있습니다.

비 CDB에 연결하는 경우 마스터 사용자는 비 CDB의 사용자입니다. CDB에 연결하는 경우 마스터 사용자는 PDB의 사용자입니다. CDB 루트에 연결하려면 호스트에 로그인하고 SQL 클라이언트를 시작한 다음 SQL 명령을 사용하여 관리자를 생성합니다.

c. 암호 자동 생성을 지웁니다.

8. DB 인스턴스 클래스를 선택합니다.

지원되는 클래스는 RDS Custom for Oracle에 대한 DB 인스턴스 클래스 지원 섹션을 참조하세요.

9. 스토리지(Storage) 섹션에서 다음을 수행합니다.

   a. 스토리지 유형에서 SSD 유형(io1, gp2 또는 gp3)을 선택합니다. 다음과 같은 추가 옵션이 있습니다.

      - io1 또는 gp3의 경우 프로비저닝된 IOPS의 속도를 선택합니다. 기본값은 io1은 1000이고 gp3는 12000입니다.
      - gp3에서 스토리지 처리량 속도를 선택합니다. 기본값은 500MiBps입니다.

   b. 할당된 스토리지에서 스토리지 크기를 선택합니다. 기본값은 40GiB입니다.

10. 연결에서 Virtual Private Cloud(VPC), DB 서브넷 그룹, VPC 보안 그룹(방화벽)을 지정합니다.

11. RDS Custom 보안을 위해서는 다음을 수행합니다.

    a. IAM 인스턴스 프로파일(IAM instance profile)에서 RDS Custom for Oracle DB 인스턴스의 인스턴스 프로파일을 선택합니다.

       IAM 인스턴스 프로파일은 `AWSRDSCustom`으로 시작해야 합니다(예: *`AWSRDSCustomInstanceProfileForRdsCustomInstance`*).

    b. 암호화(Encryption)의 경우 키 ARN 입력(Enter a key ARN)을 선택하여 사용 가능한 AWS KMS 키를 나열합니다. 그런 다음 목록에서 키를 선택합니다.

       AWS KMS 키는 RDS Custom에 필수입니다. 자세한 내용은 1단계: 대칭 암호화 AWS KMS 키 생성 또는 재사용 섹션을 참조하세요.

12. 데이터베이스 옵션에서 다음을 수행합니다.

    a. (선택 사항) 시스템 ID(SID)에 Oracle SID의 값을 입력합니다. 이 값은 CDB의 이름이기도 합니다. SID는 데이터베이스 파일을 관리하는 Oracle 데이터베이스 인스턴스의 이름입니다. 이 맥락에서 'Oracle 데이터베이스 인스턴스'라는 용어는 SGA(시스템 글로벌 영역) 및 Oracle 백그라운드 프로세스만을 지칭합니다. SID를 지정하지 않으면 기본값 **`RDSCDB`**가 사용됩니다.

    b. (선택 사항) 초기 데이터베이스 이름에 이름을 입력합니다. 기본값은 **`ORCL`**입니다. 멀티테넌트 아키텍처에서 초기 데이터베이스 이름은 PDB 이름입니다.

       > **Note**
       >
       > SID와 PDB 이름은 달라야 합니다.

    c. 옵션 그룹의 경우 옵션 그룹을 선택하거나 기본값을 그대로 사용합니다.

> **Note**
>
> RDS Custom for Oracle에 지원되는 유일한 옵션은 `Timezone`입니다. 자세한 내용은 Oracle 시간대 섹션을 참조하세요.

d. 백업 보존 기간에서 값일을 선택합니다. 0일은 선택할 수 없습니다.

e. 나머지 섹션에서 원하는 대로 RDS Custom DB 인스턴스 설정을 지정합니다. 각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하세요. 다음 설정은 콘솔에 표시되지 않으며 지원되지 않습니다.

- 프로세서 기능
- Storage autoscaling(스토리지 Autoscaling)
- 데이터베이스 인증(Database authentication)의 암호 및 Kerberos 인증(Password and Kerberos authentication) 옵션(암호 인증(Password authentication)만 지원)
- 성능 개선 도우미
- 로그 내보내기
- 마이너 버전 자동 업그레이드 활성화
- 삭제 방지

13. Create database(데이터베이스 생성)를 선택합니다.

> **Important**
>
> RDS Custom for Oracle DB 인스턴스를 생성할 때 다음 오류가 표시될 수 있습니다. 서비스 연결 역할이 생성되고 있습니다. 나중에 다시 시도해 주세요. 이 경우에는 몇 분 정도 기다렸다가 다시 DB 인스턴스를 생성하면 됩니다.

View credential details(보안 인증 세부 정보 보기) 버튼이 Databases(데이터베이스) 페이지에 표시됩니다.

RDS Custom DB 인스턴스의 마스터 사용자 이름 및 암호를 보려면 자격 증명 세부 정보 보기(View credential details)를 선택합니다.

DB 인스턴스를 마스터 사용자로 연결하려면 화면에 나타난 사용자 이름과 암호를 사용합니다.

> **Important**
>
> 콘솔에서 마스터 사용자 암호를 다시 볼 수 없습니다. 따라서 기록을 해두지 않으면 이를 변경해야 합니다. RDS Custom DB 인스턴스를 사용할 수 있게 된 후에 마스터 사용자 암호를 변경하려면 데이터베이스에 로그인하여 `ALTER USER` 명령을 실행합니다. 콘솔에서 수정 옵션을 사용하여 암호를 재설정할 수 없습니다.

14. 데이터베이스(Databases)를 선택하여 RDS Custom DB 인스턴스 목록을 확인합니다.
15. 방금 생성한 RDS Custom DB 인스턴스를 선택합니다.

    RDS 콘솔에 새로운 RDS Custom DB 인스턴스의 세부 정보가 표시됩니다.

    - RDS Custom DB 인스턴스를 만들고 사용할 준비가 될 때까지 DB 인스턴스의 상태는 생성 중(creating)입니다. 상태가 available로 변경되면 DB 인스턴스에 연결할 수 있습니다. 할당된 인스턴스 클래스 및 스토리지에 따라 새 DB 인스턴스를 사용할 수 있게 되기까지 몇 분 정도 걸릴 수 있습니다.
    - 역할(Role)에는 인스턴스(RDS Custom)(Instance (RDS Custom)) 값이 있습니다.
    - RDS Custom 자동화 모드(RDS Custom automation mode)에는 완전 자동화(Full automation) 값이 있습니다. 이 설정은 DB 인스턴스가 자동 모니터링 및 인스턴스 복구를 제공함을 의미합니다.

AWS CLI

create-db-instance AWS CLI 명령을 사용하여 RDS Custom DB 인스턴스를 생성합니다.

다음 옵션이 필요합니다.

- `--db-instance-identifier`
- `--db-instance-class`(지원되는 인스턴스 클래스 목록은 RDS Custom for Oracle에 대한 DB 인스턴스 클래스 지원 섹션 참조)
- `--engine` *`engine-type`*(*`## ##`*은 `custom-oracle-ee`, `custom-oracle-se2`, `custom-oracle-ee-cdb`, `custom-oracle-se2-cdb`)
- `--engine-version` *`cev`*(여기서 *`cev`*는 CEV 생성에서 지정한 사용자 지정 엔진 버전의 이름)
- `--kms-key-id` *`my-kms-key`*
- `--backup-retention-period` *`days`*(여기서 *`days`*는 `0`보다 큰 값)

- `--no-auto-minor-version-upgrade`
- `--custom-iam-instance-profile AWSRDSCustomInstanceProfile-`*`us-east-1`*(여기서 *`region`*은 DB 인스턴스를 생성하는 AWS 리전)

다음 예제에서는 `my-cfo-cdb-instance`라는 RDS Custom DB 인스턴스를 생성합니다. 데이터베이스는 기본값이 아닌 이름 *`MYCDB`*가 지정된 CDB입니다. 기본값이 아닌 PDB 이름은 *`MYPDB`*입니다. 백업 보존 기간은 3일로 설정합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance \
    --engine custom-oracle-ee-cdb \
    --db-instance-identifier my-cfo-cdb-instance \
    --engine-version 19.cdb_cev1 \
    --db-name MYPDB \
    --db-system-id MYCDB \
    --allocated-storage 250 \
    --db-instance-class db.m5.xlarge \
    --db-subnet-group mydbsubnetgroup \
    --master-username myuser \
    --master-user-password mypassword \
    --backup-retention-period 3 \
    --port 8200 \
    --kms-key-id my-kms-key \
    --no-auto-minor-version-upgrade \
    --custom-iam-instance-profile AWSRDSCustomInstanceProfile-us-east-1
```

Windows의 경우:

```
aws rds create-db-instance ^
    --engine custom-oracle-ee-cdb ^
    --db-instance-identifier my-cfo-cdb-instance ^
    --engine-version 19.cdb_cev1 ^
    --db-name MYPDB ^
    --db-system-id MYCDB ^
    --allocated-storage 250 ^
    --db-instance-class db.m5.xlarge ^
    --db-subnet-group mydbsubnetgroup ^
    --master-username myuser ^
```

```
    --master-user-password mypassword ^
    --backup-retention-period 3 ^
    --port 8200 ^
    --kms-key-id my-kms-key ^
    --no-auto-minor-version-upgrade ^
    --custom-iam-instance-profile AWSRDSCustomInstanceProfile-us-east-1
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

`describe-db-instances` 명령을 사용하여 인스턴스에 대한 세부 정보를 가져옵니다.

Example

```
aws rds describe-db-instances --db-instance-identifier my-cfo-cdb-instance
```

다음 부분 출력은 엔진, 파라미터 그룹 및 기타 정보를 보여줍니다.

```
        {
            "DBInstanceIdentifier": "my-cfo-cdb-instance",
            "DBInstanceClass": "db.m5.xlarge",
            "Engine": "custom-oracle-ee-cdb",
            "DBInstanceStatus": "available",
            "MasterUsername": "admin",
            "DBName": "MYPDB",
            "DBSystemID": "MYCDB",
            "Endpoint": {
                "Address": "my-cfo-cdb-instance.abcdefghijkl.us-
east-1.rds.amazonaws.com",
                "Port": 1521,
                "HostedZoneId": "A1B2CDEFGH34IJ"
            },
            "AllocatedStorage": 100,
            "InstanceCreateTime": "2023-04-12T18:52:16.353000+00:00",
            "PreferredBackupWindow": "08:46-09:16",
            "BackupRetentionPeriod": 7,
            "DBSecurityGroups": [],
            "VpcSecurityGroups": [
                {
                    "VpcSecurityGroupId": "sg-0a1bcd2e",
```

```
                    "Status": "active"
                }
            ],
            "DBParameterGroups": [
                {
                    "DBParameterGroupName": "default.custom-oracle-ee-cdb-19",
                    "ParameterApplyStatus": "in-sync"
                }
            ],
...
```

멀티테넌트 아키텍처 고려 사항

Oracle 멀티테넌트 아키텍처(`custom-oracle-ee-cdb` 또는 `custom-oracle-se2-cdb` 엔진 유형)를 사용하여 Amazon RDS Custom for Oracle DB 인스턴스를 생성하는 경우 데이터베이스는 컨테이너 데이터베이스(CDB)입니다. Oracle 멀티테넌트 아키텍처를 지정하지 않는 경우 데이터베이스는 `custom-oracle-ee` 또는 `custom-oracle-se2` 엔진 유형을 사용하는 기존의 비CDB입니다. 비CDB에는 플러그형 데이터베이스(PDB)가 포함될 수 없습니다. 자세한 내용은 Amazon RDS Custom for Oracle 데이터베이스 아키텍처 섹션을 참조하세요.

RDS Custom for Oracle CDB 인스턴스를 생성할 때는 다음 사항을 고려하세요.

- Oracle Database 19c CEV에서만 멀티테넌트 데이터베이스를 생성할 수 있습니다.
- CEV가 `custom-oracle-ee-cdb` 또는 `custom-oracle-se2-cdb` 엔진 유형을 사용하는 경우에만 CDB 인스턴스를 만들 수 있습니다.
- Standard Edition 2를 사용하여 CDB 인스턴스를 생성하는 경우 CDB에는 최대 3개의 PDB를 포함할 수 있습니다.
- 기본적으로 CDB의 이름은 `RDSCDB`이며, 이는 Oracle 시스템 ID(Oracle SID)의 이름이기도 합니다. 다른 이름으로 변경할 수 있습니다.
- CDB에는 초기 PDB가 하나만 있습니다. PDB 이름의 기본값은 `ORCL`입니다. 첫 PDB에 다른 이름을 선택할 수도 있지만, Oracle SID와 PDB 이름은 같으면 안 됩니다.
- RDS Custom for Oracle은 PDB용 API를 제공하지 않습니다. 추가 PDB를 생성하려면 Oracle SQL 명령 `CREATE PLUGGABLE DATABASE`를 사용합니다. RDS Custom for Oracle은 생성할 수 있는 PDB 수를 제한하지 않습니다. 일반적으로 온프레미스 배포에서와 마찬가지로 PDB를 생성하고 관리하는 것은 사용자의 책임입니다.
- RDS API를 사용하여 PDB를 생성, 수정 및 삭제할 수 없습니다. Oracle SQL 문을 사용해야 합니다. Oracle SQL로 PDB를 생성할 때 시점 복구(PITR)를 수행해야 하는 경우에 대비하여 나중에 수동 스냅샷을 생성하는 것이 좋습니다.

- 기존 PDB의 이름은 Amazon RDS API를 사용하여 변경할 수 없습니다. `modify-db-instance` 명령을 사용하여 CDB의 이름을 바꿀 수도 없습니다.
- CDB 루트의 열기 모드는 기본 데이터베이스에서는 `READ WRITE`이고 탑재된 대기 데이터베이스에서는 `MOUNTED`입니다. RDS Custom for Oracle은 CDB를 열 때 모든 PDB를 열려고 시도합니다. RDS Custom for Oracle이 일부 PDB를 열 수 없는 경우 `tenant database shutdown` 이벤트가 발생합니다.

RDS Custom 서비스 연결 역할

서비스 연결 역할을 사용하여 AWS 계정의 리소스에 대한 Amazon RDS Custom 액세스 권한을 부여합니다. 필요한 권한을 수동으로 추가할 필요가 없어 RDS Custom을 보다 쉽게 사용할 수 있습니다. RDS Custom은 서비스 연결 역할의 권한을 정의하므로, 달리 정의하지 않으면 RDS Custom만 해당 역할을 맡을 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며, 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

RDS Custom DB 인스턴스를 생성하면 Amazon RDS 및 RDS Custom 서비스 연결 역할이 모두 생성되어(존재하지 않았던 경우) 사용됩니다. 자세한 내용은 Amazon RDS에 서비스 연결 역할 사용 섹션을 참조하세요.

RDS Custom for Oracle DB 인스턴스를 처음 생성할 때 다음 오류가 표시될 수 있습니다. 서비스 연결 역할이 생성되고 있습니다. 나중에 다시 시도해 주세요. 이 경우에는 몇 분 정도 기다렸다가 다시 DB 인스턴스를 생성하면 됩니다.

RDS Custom for Oracle DB 인스턴스에 추가 소프트웨어 구성 요소 설치

새로 만든 DB 인스턴스의 데이터베이스 환경에는 Oracle 바이너리, 데이터베이스 및 데이터베이스 리스너가 포함됩니다. DB 인스턴스의 호스트 운영 체제에 추가 소프트웨어를 설치해야 할 수 있습니다. 예를 들어 Oracle Application Express(APEX), Oracle Enterprise Manager(OEM) 에이전트 또는 Guardium S-TAP 에이전트를 설치할 수 있습니다. 가이드 및 개괄적인 지침이 자세히 나와 있는 AWS 블로그 게시물 Amazon RDS Custom for Oracle에 추가 소프트웨어 구성 요소 설치를 참조하세요.

세션 관리자를 사용하여 RDS Custom DB 인스턴스에 연결

RDS Custom DB 인스턴스를 생성한 후 AWS Systems Manager Session Manager를 사용하여 연결할 수 있습니다. 이는 DB 인스턴스에 공개적으로 액세스할 수 없을 때 우선적으로 사용되는 기술입니다.

Session Manager를 사용하면 브라우저 기반 쉘 또는 AWS CLI를 통해 Amazon EC2 인스턴스에 액세스할 수 있습니다. 자세한 내용은 AWS Systems Manager 세션 관리자를 참조하세요.

콘솔

세션 관리자를 사용하여 DB 인스턴스에 연결하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스(Databases)를 선택한 후 연결하려는 RDS Custom DB 인스턴스를 선택합니다.
3. Configuration(구성)을 선택합니다.
4. DB 인스턴스의 리소스 ID(Resource ID)를 기록해 둡니다. 예를 들어, 리소스 ID는 `db-ABCDEFGHIJKLMNOPQRS0123456`일 수 있습니다.
5. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
6. 탐색 창에서 Instances(인스턴스)를 선택합니다.
7. EC2 인스턴스의 이름을 찾은 다음 연결된 인스턴스 ID를 클릭합니다. 예를 들어, 인스턴스 ID는 `i-abcdefghijklm01234`일 수 있습니다.
8. 연결을 선택합니다.
9. 세션 관리자(Session Manager)를 선택합니다.
10. Connect(연결)를 선택합니다.

    세션에 대한 창이 열립니다.

AWS CLI

AWS CLI를 사용하여 RDS Custom DB 인스턴스에 연결할 수 있습니다. 이 기술을 사용하려면 AWS CLI용 세션 관리자 플러그인이 필요합니다. 플러그인 설치 방법은 AWS CLI용 세션 관리자 플러그인 설치를 참조하세요.

RDS Custom DB 인스턴스의 DB 리소스 ID를 찾으려면 `aws rds describe-db-instances`를 사용합니다.

```
aws rds describe-db-instances \
    --query 'DBInstances[*].[DBInstanceIdentifier,DbiResourceId]' \
    --output text
```

다음 샘플 출력은 RDS Custom 인스턴스의 리소스 ID를 보여줍니다. 접두사는 db-입니다.

```
db-ABCDEFGHIJKLMNOPQRS0123456
```

DB 인스턴스의 EC2 인스턴스 ID를 찾으려면 `aws ec2 describe-instances`를 사용합니다. 다음 예제에는 리소스 ID로 `db-ABCDEFGHIJKLMNOPQRS0123456`이 사용됩니다.

```
aws ec2 describe-instances \
    --filters "Name=tag:Name,Values=db-ABCDEFGHIJKLMNOPQRS0123456" \
    --output text \
    --query 'Reservations[*].Instances[*].InstanceId'
```

다음 샘플 출력에는 EC2 인스턴스 ID가 나와 있습니다.

```
i-abcdefghijklm01234
```

`--target` 파라미터의 EC2 인스턴스 ID를 제공하는 `aws ssm start-session` 명령을 사용합니다.

```
aws ssm start-session --target "i-abcdefghijklm01234"
```

성공적으로 연결되면 다음과 같이 표시됩니다.

```
Starting session with SessionId: yourid-abcdefghijklm1234
[ssm-user@ip-123-45-67-89 bin]$
```

SSH를 사용하여 RDS Custom DB 인스턴스에 연결

SSH(Secure Shell Protocol)는 보안이 되지 않은 네트워크를 통한 암호화된 통신을 지원하는 네트워크 프로토콜입니다. RDS Custom DB 인스턴스를 생성한 후 ssh 클라이언트를 사용하여 이 인스턴스에 연결할 수 있습니다. 자세한 내용은 SSH를 사용하여 Linux 인스턴스에 연결을 참조하세요.

SSH 연결 기술은 DB 인스턴스가 프라이빗 인스턴스인지에 따라 좌우됩니다. 즉, 이 기술은 퍼블릭 인터넷 연결을 허용하지 않습니다. 이 경우 SSH 터널링을 사용하여 ssh 유틸리티를 인스턴스에 연결해야 합니다. 이 기술은 기존 SSH 세션 내에서 전용 데이터 스트림(터널)을 사용하여 데이터를 전송합니다. AWS Systems Manager를 사용하여 SSH 터널링을 구성할 수 있습니다.

> **Note**
>
> 프라이빗 인스턴스에 액세스할 수 있는 다양한 전략이 지원됩니다. Bastion 호스트를 사용하여 ssh 클라이언트를 프라이빗 인스턴스에 연결하는 방법을 알아보려면 Linux Bastion Hosts on AWS을 참조하세요. 포트 전달을 구성하는 방법을 알아보려면 AWS Systems Manager Session Manager를 사용하여 포트 전달을 참조하세요.

DB 인스턴스가 퍼블릭 서브넷에 있고 공개적으로 사용 가능한 설정이 있는 경우 SSH 터널링이 필요하지 않습니다. 퍼블릭 Amazon EC2 인스턴스와 마찬가지로 SSH를 사용하여 연결할 수 있습니다.

ssh 클라이언트를 DB 인스턴스에 연결하려면 다음 단계를 완료하세요.

1. 1단계: SSH 연결을 허용하도록 DB 인스턴스 구성
2. 2단계: SSH 비밀 키 및 EC2 인스턴스 ID 검색
3. 3단계: ssh 유틸리티를 사용하여 EC2 인스턴스에 연결

1단계: SSH 연결을 허용하도록 DB 인스턴스 구성

DB 인스턴스가 SSH 연결을 허용할 수 있는지 확인하려면 다음을 수행합니다.

- DB 인스턴스 보안 그룹이 TCP용 포트 22에서 인바운드 연결을 허용하는지 확인해야 합니다.

  DB 인스턴스의 보안 그룹을 구성하는 방법을 알아보려면 보안 그룹을 통한 액세스 제어 섹션을 참조하세요.
- SSH 터널링을 사용할 계획이 없는 경우 DB 인스턴스가 퍼블릭 서브넷에 있고 퍼블릭 액세스가 가능한지 확인합니다.

  콘솔의 관련 필드는 데이터베이스 세부 정보 페이지의 연결 및 보안 탭에 있는 퍼블릭 액세스 가능입니다. CLI에서 설정을 확인하려면 다음 명령을 실행합니다.

```
aws rds describe-db-instances \
--query 'DBInstances[*].
{DBInstanceIdentifier:DBInstanceIdentifier,PubliclyAccessible:PubliclyAccessible}' \
--output table
```

  DB 인스턴스의 접근성 설정을 변경하려면 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

2단계: SSH 비밀 키 및 EC2 인스턴스 ID 검색

SSH를 사용하여 DB 인스턴스에 연결하려면 인스턴스와 연결된 SSH 키 페어가 필요합니다. RDS Custom은 SSH 키 페어를 자동으로 만드는데, 이때 이름 지정 규칙 `do-not-delete-rds-custom-ssh-privatekey-`*`resource_id`*`-`*`uuid`* 또는 `rds-custom!oracle-do-not-delete-`*`resource_id`*`-`*`uuid`*`-ssh-privatekey`를 사용하여 이름을 지정합니다. AWS Secrets Manager는 SSH 프라이빗 키를 보안 암호로 저장합니다.

AWS Management Console 또는 AWS CLI 중 하나를 사용하여 SSH 비밀 키를 검색할 수 있습니다. 인스턴스에 퍼블릭 DNS가 있고 SSH 터널링을 사용하지 않으려는 경우 DNS 이름도 검색하세요. 퍼블릭 연결의 DNS 이름을 지정합니다.

콘솔

SSH 비밀 키를 검색하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스(Databases)를 선택한 후 연결하려는 RDS Custom DB 인스턴스를 선택합니다.
3. Configuration(구성)을 선택합니다.
4. 리소스 ID(Resource ID) 값을 기록해 둡니다. 예를 들어 DB 인스턴스 리소스 ID는 `db-ABCDEFGHIJKLMNOPQRS0123456`일 수 있습니다.
5. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
6. 탐색 창에서 인스턴스를 선택합니다.
7. EC2 인스턴스의 이름을 찾고 연결된 인스턴스 ID를 선택합니다. 예를 들어, EC2 인스턴스 ID는 `i-abcdefghijklm01234`일 수 있습니다.
8. 세부 정보(Details)에서 키 페어 이름(Key pair name)을 찾습니다. 페어 이름에는 DB 인스턴스 리소스 ID가 포함됩니다. 예를 들어, 페어 이름은 `do-not-delete-rds-custom-ssh-privatekey-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c` 또는 `rds-custom!oracle-do-not-delete-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c-ssh-privatekey`일 수 있습니다.
9. EC2 인스턴스가 퍼블릭인 경우 퍼블릭 IPv4 DNS를 기록해 둡니다. 예를 들어, 퍼블릭 도메인 이름 시스템(DNS) 주소는 `ec2-12-345-678-901.us-east-2.compute.amazonaws.com`일 수 있습니다.
10. https://console.aws.amazon.com/secretsmanager/에서 AWS Secrets Manager 콘솔을 엽니다.
11. 키 페어와 이름이 같은 비밀 키를 선택합니다.
12. Retrieve secret value(보안 암호 값 검색)를 선택합니다.
13. SSH 프라이빗 키를 텍스트 파일로 복사한 다음 `.pem` 확장자로 파일을 저장합니다. 예를 들어, 파일을 `/tmp/do-not-delete-rds-custom-ssh-privatekey-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c.pem` 또는 `/tmp/rds-custom!oracle-do-not-delete-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c-ssh-privatekey.pem`으로 저장합니다.

AWS CLI

SSH 프라이빗 키를 검색하고 .pem 파일에 저장하려면 AWS CLI를 사용하면 됩니다.

1. `aws rds describe-db-instances`를 사용하여 RDS Custom DB 인스턴스의 DB 리소스 ID를 찾습니다.

```
aws rds describe-db-instances \
    --query 'DBInstances[*].[DBInstanceIdentifier,DbiResourceId]' \
    --output text
```

다음 샘플 출력은 RDS Custom 인스턴스의 리소스 ID를 보여줍니다. 접두사는 db-입니다.

```
db-ABCDEFGHIJKLMNOPQRS0123456
```

2. `aws ec2 describe-instances`를 사용하여 DB 인스턴스의 EC2 인스턴스 ID를 찾습니다. 다음 예제에는 리소스 ID로 db-ABCDEFGHIJKLMNOPQRS0123456이 사용됩니다.

```
aws ec2 describe-instances \
    --filters "Name=tag:Name,Values=db-ABCDEFGHIJKLMNOPQRS0123456" \
    --output text \
    --query 'Reservations[*].Instances[*].InstanceId'
```

다음 샘플 출력에는 EC2 인스턴스 ID가 나와 있습니다.

```
i-abcdefghijklm01234
```

3. 키 이름을 찾으려면 EC2 인스턴스 ID를 지정합니다. 다음 예제에서는 EC2 인스턴스 i-0bdc4219e66944afa에 대해 설명합니다.

```
aws ec2 describe-instances \
    --instance-ids i-0bdc4219e66944afa \
    --output text \
    --query 'Reservations[*].Instances[*].KeyName'
```

다음 샘플 출력은 이름 지정 형식 do-not-delete-rds-custom-ssh-privatekey-*resource_id*-*uuid* 또는 rds-custom!oracle-do-not-delete-*resource_id*-*uuid*-ssh-privatekey를 사용하는 키 이름을 보여줍니다.

```
do-not-delete-rds-custom-ssh-privatekey-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c
```

```
rds-custom!oracle-do-not-delete-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c-ssh-privatekey
```

4. aws secretsmanager를 사용하여 키의 이름을 따라 명명된 .pem 파일로 프라이빗 키를 저장합니다.

   다음 예제에서는 /tmp 디렉터리의 파일에 do-not-delete-rds-custom-ssh-privatekey-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c 키를 저장합니다.

   ```
   aws secretsmanager get-secret-value \
       --secret-id do-not-delete-rds-custom-ssh-privatekey-db-
   ABCDEFGHIJKLMNOPQRS0123456-0d726c \
       --query SecretString \
       --output text >/tmp/do-not-delete-rds-custom-ssh-privatekey-db-
   ABCDEFGHIJKLMNOPQRS0123456-0d726c.pem
   ```

   다음 예제에서는 /tmp 디렉터리의 파일에 rds-custom!oracle-do-not-delete-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c-ssh-privatekey 키를 저장합니다.

   ```
   aws secretsmanager get-secret-value \
       --secret-id rds-custom!oracle-do-not-delete-db-
   ABCDEFGHIJKLMNOPQRS0123456-0d726c-ssh-privatekey \
       --query SecretString \
       --output text >/tmp/rds-custom!oracle-do-not-delete-db-
   ABCDEFGHIJKLMNOPQRS0123456-0d726c-ssh-privatekey.pem
   ```

3단계: ssh 유틸리티를 사용하여 EC2 인스턴스에 연결

연결 방법은 프라이빗 DB 인스턴스에 연결하는지 또는 퍼블릭 인스턴스에 연결하는지에 따라 달라집니다. 프라이빗 연결을 위해서는 AWS Systems Manager를 통해 SSH 터널링을 구성해야 합니다.

ssh 유틸리티를 사용하여 EC2 인스턴스에 연결하려면

1. 프라이빗 연결의 경우 SSH 구성 파일을 수정하여 명령을 AWS Systems Manager Session Manager로 프록시합니다. 퍼블릭 연결의 경우 2단계로 건너뜁니다.

   다음 행을 ~/.ssh/config에 추가합니다. 이러한 행은 이름이 i- 또는 mi-로 시작하는 호스트에 대한 SSH 명령을 프록시합니다.

   ```
   Host i-* mi-*
   ```

```
    ProxyCommand sh -c "aws ssm start-session --target %h --document-name AWS-
StartSSHSession --parameters 'portNumber=%p'"
```

2. .pem 파일이 들어 있는 디렉터리로 변경하고, chmod를 사용하여 권한을 400으로 설정합니다.

   다음 예제에서는 /tmp 디렉터리를 변경하고 .pem 파일 do-not-delete-rds-custom-ssh-privatekey-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c.pem에 대한 권한을 설정합니다.

```
cd /tmp
chmod 400 do-not-delete-rds-custom-ssh-privatekey-db-
ABCDEFGHIJKLMNOPQRS0123456-0d726c.pem
```

   다음 예제에서는 /tmp 디렉터리를 변경하고 .pem 파일 rds-custom!oracle-do-not-delete-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c-ssh-privatekey.pem에 대한 권한을 설정합니다.

```
cd /tmp
chmod 400 rds-custom!oracle-do-not-delete-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c-ssh-
privatekey.pem
```

3. ssh 유틸리티를 실행하여.pem 파일과 퍼블릭 DNS 이름(퍼블릭 연결용) 또는 EC2 인스턴스 ID(프라이빗 연결용)를 지정합니다. ec2-user 사용자로 로그인합니다.

   다음 예제에서는 DNS 이름 ec2-12-345-678-901.us-east-2.compute.amazonaws.com을 사용하여 퍼블릭 인스턴스에 연결합니다.

```
# .pem file using naming prefix do-not-delete
ssh -i \
  "do-not-delete-rds-custom-ssh-privatekey-db-
ABCDEFGHIJKLMNOPQRS0123456-0d726c.pem" \
  ec2-user@ec2-12-345-678-901.us-east-2.compute.amazonaws.com

# .pem file using naming prefix rds-custom!oracle-do-not-delete
ssh -i \
  "rds-custom!oracle-do-not-delete-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c-ssh-
privatekey.pem" \
  ec2-user@ec2-12-345-678-901.us-east-2.compute.amazonaws.com
```

   다음 예제에서는 EC2 인스턴스 ID i-0bdc4219e66944afa를 사용하여 프라이빗 인스턴스에 연결합니다.

```
# .pem file using naming prefix do-not-delete
ssh -i \
  "do-not-delete-rds-custom-ssh-privatekey-db-
ABCDEFGHIJKLMNOPQRS0123456-0d726c.pem" \
  ec2-user@i-0bdc4219e66944afa

# .pem file using naming prefix rds-custom!oracle-do-not-delete
ssh -i \
  "rds-custom!oracle-do-not-delete-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c-ssh-
privatekey.pem" \
  ec2-user@i-0bdc4219e66944afa
```

RDS Custom for Oracle 데이터베이스에 SYS로 로그인

RDS Custom DB 인스턴스를 생성한 후 Oracle 데이터베이스에 SYS 사용자로 로그인할 수 있으며, 이 경우 SYSDBA 권한이 부여됩니다. 로그인 옵션은 다음과 같습니다.

- Secrets Manager에서 SYS 암호를 가져오고, SQL 클라이언트에서 이 암호를 지정합니다.
- OS 인증을 사용하여 데이터베이스에 로그인합니다. 이 경우 암호가 필요하지 않습니다.

RDS Custom for Oracle 데이터베이스의 SYS 암호 찾기

Oracle 데이터베이스에 SYS 또는 SYSTEM로 로그인하거나, API 호출에 마스터 사용자 이름을 지정하여 로그인할 수 있습니다. SYS 및 SYSTEM의 암호는 Secrets Manager에 저장됩니다.

보안 암호는 이름 지정 형식 `do-not-delete-rds-custom-`*`resource_id`*`-`*`uuid`* 또는 `rds-custom!oracle-do-not-delete-`*`resource_id`*`-`*`uuid`*를 사용합니다. AWS Management Console을 사용하여 암호를 찾을 수 있습니다.

콘솔

Secrets Manager에서 데이터베이스의 SYS 암호를 찾으려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 콘솔에서 다음 단계를 완료합니다.
   a. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
   b. RDS Custom for Oracle DB 인스턴스의 이름을 선택합니다.

c. 구성을 선택합니다.

d. 리소스 ID 아래의 값을 복사합니다. 예를 들어, 리소스 ID는 db-ABC12CDE3FGH4I5JKLMNO6PQR7일 수 있습니다.

3. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
4. Secrets Manager 콘솔에서 다음 단계를 완료합니다.

   a. 왼쪽 탐색 창에서 보안 암호를 선택합니다.

   b. 2.d단계에서 복사한 리소스 ID를 기준으로 보안 암호를 필터링합니다.

   c. 이름 지정 형식 do-not-delete-rds-custom-***resource_id-uuid*** 또는 rds-custom!oracle-do-not-delete-***resource_id-uuid***를 사용하는 보안 암호를 선택합니다. *resource_id*는 2.d 단계에서 복사한 리소스 ID입니다.

      예를 들어 리소스 ID가 db-ABC12CDE3FGH4I5JKLMNO6PQR7이고 UUID가 1234ab인 경우 보안 암호의 이름은 do-not-delete-rds-custom-db-ABC12CDE3FGH4I5JKLMNO6PQR7-1234ab 또는 rds-custom!oracle-do-not-delete-db-ABC12CDE3FGH4I5JKLMNO6PQR7-1234ab입니다.

   d. 보안 암호 값 섹션에서 보안 암호 값 검색을 선택합니다.

   e. 키/값에서 암호 값을 복사합니다.

5. DB 인스턴스에 SQL*Plus를 설치하고 데이터베이스에 SYS로 로그인합니다. 자세한 내용은 3단계: SQL 클라이언트를 Oracle DB 인스턴스에 연결 섹션을 참조하세요.

OS 인증을 사용하여 RDS Custom for Oracle 데이터베이스에 로그인

OS 사용자 `rdsdb`는 Oracle 데이터베이스 바이너리를 소유합니다. `rdsdb` 사용자로 전환하여 암호 없이 RDS Custom for Oracle 데이터베이스에 로그인할 수 있습니다.

1. AWS Systems Manager를 사용하여 DB 인스턴스에 연결합니다. 자세한 내용은 세션 관리자를 사용하여 RDS Custom DB 인스턴스에 연결 섹션을 참조하세요.
2. 웹 브라우저에서 https://www.oracle.com/database/technologies/instant-client/linux-x86-64-downloads.html로 이동합니다.
3. 웹 페이지에 표시되는 최신 데이터베이스 버전을 보려면 Instant Client Basic Package 및 SQL*Plus Package의 .rpm 링크(.zip 링크 아님)를 복사하세요. 예를 들어, 다음 링크는 Oracle 데이터베이스 버전 21.9의 링크입니다.

   - https://download.oracle.com/otn_software/linux/instantclient/219000/oracle-instantclient-basic-21.9.0.0.0-1.el8.x86_64.rpm

- https://download.oracle.com/otn_software/linux/instantclient/219000/oracle-instantclient-sqlplus-21.9.0.0.0-1.el8.x86_64.rpm

4. SSH 세션에서 wget 명령을 실행하여 이전 단계에서 가져온 링크를 통해 .rpm 파일을 다운로드합니다. 다음 예제에서는 Oracle 데이터베이스 버전 21.9의 .rpm 파일을 다운로드합니다.

```
wget https://download.oracle.com/otn_software/linux/instantclient/219000/oracle-instantclient-basic-21.9.0.0.0-1.el8.x86_64.rpm
wget https://download.oracle.com/otn_software/linux/instantclient/219000/oracle-instantclient-sqlplus-21.9.0.0.0-1.el8.x86_64.rpm
```

5. 다음과 같이 yum 명령을 실행하여 패키지를 설치합니다.

```
sudo yum install oracle-instantclient-*.rpm
```

6. rdsdb 사용자로 전환합니다.

```
sudo su - rdsdb
```

7. OS 인증을 사용하여 데이터베이스에 로그인합니다.

```
$ sqlplus / as sysdba

SQL*Plus: Release 21.0.0.0.0 - Production on Wed Apr 12 20:11:08 2023
Version 21.9.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.10.0.0.0
```

Amazon RDS Custom for Oracle DB 인스턴스 관리

Amazon RDS Custom은 Amazon RDS DB 인스턴스에 대한 일반적인 관리 작업의 하위 집합을 지원합니다. 그런 다음 AWS Management Console 및 AWS CLI를 사용하여 지원되는 RDS Custom for Oracle 관리 작업에 대한 지침을 확인할 수 있습니다.

주제

- RDS Custom for Oracle에서 컨테이너 데이터베이스(CDB) 작업
- RDS Custom for Oracle의 고가용성 기능을 통한 작업
- RDS Custom 환경 사용자 지정
- RDS Custom for Oracle DB 인스턴스 수정
- RDS Custom for Oracle DB 인스턴스의 문자 집합 변경
- RDS Custom for Oracle의 NLS_LANG 값 설정
- 투명한 데이터 암호화 지원
- RDS Custom for Oracle 리소스 태깅
- RDS Custom for Oracle DB 인스턴스 삭제

RDS Custom for Oracle에서 컨테이너 데이터베이스(CDB) 작업

Oracle 멀티테넌트 아키텍처(`custom-oracle-ee-cdb` 또는 `custom-oracle-se2-cdb` 엔진 유형) 또는 기존의 비CDB 아키텍처(`custom-oracle-ee` 또는 `custom-oracle-se2` 엔진 유형)를 사용하여 RDS Custom for Oracle DB 인스턴스를 생성할 수 있습니다. 생성한 컨테이너 데이터베이스(CDB)에는 플러그형 데이터베이스(PDB) 1개와 PDB 시드 1개가 포함됩니다. Oracle SQL을 사용하여 수동으로 추가 PDB를 생성할 수 있습니다.

PDB 및 CDB 이름

RDS Custom for Oracle CDB 인스턴스를 생성할 경우, 초기 PDB의 이름을 지정합니다. 기본적으로 초기 PDB의 이름은 `ORCL`으로 지정됩니다. 다른 이름으로 변경할 수 있습니다.

기본적으로 CDB의 이름은 `RDSCDB`로 지정됩니다. 다른 이름으로 변경할 수 있습니다. CDB 이름은 CDB를 관리하는 메모리와 프로세스를 고유하게 식별하는 Oracle 시스템 식별자(SID)의 이름이기도 합니다. Oracle SID에 대한 자세한 내용은 Oracle Database Concepts에 나온 Oracle System Identifier(SID)를 참조하세요.

기존 PDB의 이름은 Amazon RDS API를 사용하여 변경할 수 없습니다. `modify-db-instance` 명령을 사용하여 CDB의 이름을 바꿀 수도 없습니다.

PDB 관리

RDS Custom for Oracle 공동 책임 모델에서 PDB를 관리하고 추가 PDB를 생성할 책임은 사용자에게 있습니다. RDS Custom은 PDB 수를 제한하지 않습니다. CDB 루트에 연결하고 SQL 문을 실행하여 PDB를 수동으로 생성, 수정 및 삭제할 수 있습니다. Amazon EBS 데이터 볼륨에 PDB를 생성하여 DB 인스턴스가 지원 경계를 벗어나는 것을 방지합니다.

CDB 또는 PDB를 수정하려면 다음 단계를 완료합니다.

1. 자동화를 일시 중지하여 RDS Custom 작업과의 간섭을 방지하세요.
2. CDB 또는 PDB를 수정합니다.
3. 수정된 모든 PDB를 백업합니다.
4. RDS Custom 자동화를 다시 시작합니다.

CDB 루트 자동 복구

RDS Custom은 비 CDB 루트를 열린 상태로 유지하는 것과 동일한 방식으로 CDB 루트를 열린 상태로 유지합니다. CDB 루트 상태가 변경되면 모니터링 및 복구 자동화가 CDB 루트를 원하는 상태로 복구하려고 시도합니다. 비 CDB 아키텍처와 마찬가지로 루트 CDB가 종료(`RDS-EVENT-0004`)되거나 재시작(`RDS-EVENT-0006`)될 때 RDS 이벤트 알림을 받습니다. RDS Custom은 DB 인스턴스 시작 시 모든 PDB를 `READ WRITE` 모드에서 열려고 시도합니다. 일부 PDB를 열 수 없는 경우 RDS Custom은 `tenant database shutdown` 이벤트를 게시합니다.

RDS Custom for Oracle의 고가용성 기능을 통한 작업

RDS Custom for Oracle은 다중 AZ 배포를 통해 내장된 고가용성을 제공합니다. 또는 사용 사례에 따라 Oracle Data Guard를 고객 관리형 옵션으로 사용할 수 있습니다.

다중 AZ 배포(완전관리형)

RDS Custom for Oracle에 대한 다중 AZ 배포에서 Amazon RDS는 자동으로 서로 다른 가용 영역(AZ)에 동기식 대기 복제본을 프로비저닝하고 유지 관리합니다. 기본 DB 인스턴스는 데이터 중복성을 위해 여러 AZ에 대기 복제본으로 동기식으로 복제됩니다. 다중 AZ 배포는 Enterprise Edition과 Standard Edition 2 모두에서 지원됩니다. 세부 정보는 the section called “RDS Custom for Oracle에 대한 다중 AZ 배포 관리” 섹션을 참조하세요.

Oracle Data Guard(고객 관리형)

또는 RDS Custom for Oracle DB 인스턴스 간에 데이터를 복제하도록 Oracle Data Guard를 수동으로 구성하여 고가용성을 달성할 수 있습니다. 프라이머리 DB 인스턴스는 데이터를 대기 인스턴스와 자동으로 동기화합니다. Oracle Data Guard는 Enterprise Edition에서만 지원됩니다.

다음과 같은 방법으로 고가용성 환경을 구성할 수 있습니다.

- AZ 장애에 대처하도록 서로 다른 AZ에서 대기 인스턴스를 구성합니다.
- 대기 데이터베이스를 탑재된 모드 또는 읽기 전용 모드로 전환합니다. 읽기 전용 모드에는 Oracle Active Data Guard 라이선스가 필요합니다.
- 데이터 손실 없이 프라이머리 데이터베이스에서 대기 데이터베이스로 장애 조치하거나 전환합니다.
- 온프레미스 인스턴스에 대해 고가용성을 구성한 다음, RDS Custom for Oracle 대기 데이터베이스로 장애 조치하거나 전환하여 데이터를 마이그레이션합니다.

고가용성을 위해 Oracle Data Guard를 구성하는 방법은 AWS Blog의 Build high availability for RDS Custom for Oracle using read replicas를 참조하세요. 다음 작업을 수행할 수 있습니다.

- 가상 사설 네트워크(VPN) 터널을 사용하여 고가용성 인스턴스에 대해 전송 중인 데이터를 암호화합니다. 전송 중 암호화는 RDS Custom for Oracle에서 자동으로 구성되지 않습니다.
- 고가용성 인스턴스를 모니터링하도록 Oracle Fast-Failover Observer(FSFO)를 구성합니다.
- 필요한 조건이 충족되면 관찰자가 자동 장애 조치를 수행할 수 있도록 합니다.

RDS Custom 환경 사용자 지정

RDS Custom for Oracle에는 자동화를 일시 중지하지 않고도 DB 인스턴스 환경을 사용자 지정할 수 있는 내장 기능이 포함되어 있습니다. 예를 들어 RDS API를 사용하면 다음과 같이 환경을 사용자 지정할 수 있습니다.

- DB 스냅샷을 생성하고 복원하여 클론 환경을 생성합니다.
- 읽기 전용 복제본을 생성합니다.
- 스토리지 설정을 수정합니다.
- CEV를 변경하여 릴리스 업데이트 적용

문자 집합을 변경하는 등의 일부 사용자 지정 작업에는 RDS API를 사용할 수 없습니다. 이러한 경우 루트 사용자로 Amazon EC2 인스턴스에 액세스하거나 Oracle 데이터베이스에 SYSDBA로 로그인하여 환경을 수동으로 변경해야 합니다.

인스턴스를 수동으로 사용자 지정하려면 RDS Custom 자동화를 일시 중지했다가 재개해야 합니다. 이렇게 일시 중지하면 사용자 지정한 인스턴스가 RDS Custom 자동화를 방해하지 않도록 할 수 있습니다. 이 방법을 사용하면 지원 경계가 무너지는 것을 방지할 수 있습니다. 그러면 기본 문제가 해결될 때까지 인스턴스가 `unsupported-configuration` 상태로 유지됩니다. RDS Custom for Oracle DB 인스턴스를 수정할 때 지원되는 자동화 태스크는 자동화 일시 중지 및 재개뿐입니다.

RDS 사용자 지정 환경을 사용자 지정하는 일반적인 단계

RDS Custom DB 인스턴스를 사용자 지정하려면 다음 단계를 완료합니다.

1. 콘솔 또는 CLI를 사용하여 지정된 기간 동안 RDS Custom 자동화를 일시 중지합니다.
2. 기본 Amazon EC2 인스턴스를 확인합니다.
3. SSH 키 또는 AWS Systems Manager를 사용하여 기본 Amazon EC2 인스턴스에 연결합니다.
4. 데이터베이스 또는 운영 체제 계층에서 현재 구성 설정을 확인합니다.

   초기 구성을 변경된 구성과 비교하여 변경 내용을 검증할 수 있습니다. 사용자 지정 유형에 따라 OS 도구 또는 데이터베이스 쿼리를 사용하세요.
5. 필요에 따라 RDS Custom for Oracle DB 인스턴스를 사용자 지정합니다.
6. 필요한 경우 인스턴스 또는 데이터베이스를 재부팅합니다.

> **Note**
>
> 온프레미스 Oracle CDB에서는 내장 명령을 사용하거나 시작 트리거 이후에 PDB에 대해 지정된 열기 모드를 유지할 수 있습니다. 이 메커니즘은 CDB가 다시 시작될 때 PDB를 지정된 상태로 전환합니다. CDB를 열 때 RDS Custom 자동화는 사용자가 지정한 보존 상태를 삭제하고 모든 PDB를 열려고 시도합니다. RDS Custom이 일부 PDB를 열 수 없는 경우 The `following PDBs failed to open: list-of-PDBs` 이벤트가 발생합니다.

7. 새 구성 설정을 이전 설정과 비교하여 확인합니다.
8. 다음 방법 중 하나를 사용하여 RDS Custom 자동화를 재개합니다.
   - 수동으로 자동화를 다시 시작합니다.
   - 일시 중지 기간이 끝날 때까지 기다립니다. 이 경우 RDS Custom은 모니터링 및 인스턴스 복구를 자동으로 재개합니다.

9. RDS Custom 자동화 프레임워크 확인

   이전 단계를 올바르게 수행한 경우 RDS Custom은 자동 백업을 시작합니다. 콘솔의 인스턴스 상태는 사용 가능으로 표시됩니다.

모범 사례 및 단계별 지침은 AWS 블로그 게시물인 Make configuration changes to an Amazon RDS Custom for Oracle instance: Part 1(Amazon RDS Custom for Oracle 인스턴스의 구성 변경: 1부) 및 Recreate an Amazon RDS Custom for Oracle database: Part 2(Amazon RDS Custom for Oracle 데이터베이스 다시 생성: 2부)를 참조하세요.

RDS Custom DB 인스턴스 일시 중지 및 재개

콘솔 또는 CLI를 사용하여 DB 인스턴스의 자동화를 일시 중지하고 재개할 수 있습니다.

콘솔

RDS Custom 자동화를 일시 중지하거나 다시 시작하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스(Databases)를 선택한 다음 변경하려는 RDS Custom DB 인스턴스를 선택합니다.
3. Modify(수정)를 선택합니다. Modify DB instance(DB 인스턴스 수정) 페이지가 나타납니다.
4. RDS Custom 자동화 모드(RDS Custom automation mode)로 다음 옵션 중 하나를 선택합니다.
   - 일시 중지됨(Paused)을 선택하면 RDS Custom DB 인스턴스에 대한 모니터링 및 인스턴스 복구가 일시 중지됩니다. 자동화 모드 지속 시간(Automation mode duration)으로 원하는 일시 중지 기간(분)을 입력합니다. 최소값은 60분(기본값)입니다. 최대값은 1,440분입니다.
   - 완전 자동화(Full automation)를 선택하면 자동화가 재개됩니다.
5. 계속(Continue)을 선택하여 수정 사항을 요약한 내용을 확인합니다.

   RDS Custom에서 변경 사항을 즉시 적용한다는 메시지가 표시됩니다.
6. 변경 내용이 정확할 경우 DB 인스턴스 수정(Modify DB instance)을 선택합니다. 또는 뒤로를 선택하여 변경 내용을 편집하거나 취소를 선택하여 변경 내용을 취소합니다.

   RDS 콘솔에 수정 사항에 대한 세부 정보가 표시됩니다. 자동화를 일시 중지한 경우 RDS Custom DB 인스턴스의 상태(Status)가 자동화 일시 중지됨(Automation paused)으로 표시됩니다.

7. (선택 사항) 탐색 창에서 데이터베이스(Databases)를 선택한 후 RDS Custom DB 인스턴스를 선택합니다.

   요약(Summary) 창에서 RDS Custom 자동화 모드(RDS Custom automation mode)는 자동화 상태를 나타냅니다. 자동화가 일시 중지되면 값이 일시 중지됨(Paused)이 됩니다. ***num***분 후에 자동화가 재개됩니다.

AWS CLI

RDS Custom 자동화를 일시 중지하거나 재개하려면 `modify-db-instance` AWS CLI 명령을 사용합니다. 필수 파라미터 `--db-instance-identifier`를 사용하여 DB 인스턴스를 식별합니다. 다음 파라미터를 사용하여 자동화 모드를 제어합니다.

- `--automation-mode`는 DB 인스턴스의 일시 정지 상태를 지정합니다. 유효한 값은 자동화를 일시 중지하는 `all-paused`와 다시 재개하는 `full`입니다.
- `--resume-full-automation-mode-minutes`는 일시 중지 기간을 지정합니다. 기본값은 60분입니다.

> **Note**
>
> `--no-apply-immediately` 또는 `--apply-immediately`에 관계없이 RDS Custom은 가능한 한 빨리 비동기식으로 수정 사항을 적용합니다.

명령 응답에서 `ResumeFullAutomationModeTime`은 재개 시간을 UTC 타임스탬프로 나타냅니다. 자동화 모드가 `all-paused`인 경우 `modify-db-instance`를 사용하여 자동화 모드를 재개하거나 일시 중지 기간을 연장할 수 있습니다. 다른 `modify-db-instance` 옵션은 지원되지 않습니다.

다음 예제에서는 `my-custom-instance`에 대한 자동화를 90분 동안 일시 중지합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier my-custom-instance \
    --automation-mode all-paused \
    --resume-full-automation-mode-minutes 90
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --automation-mode all-paused ^
    --resume-full-automation-mode-minutes 90
```

다음 예제에서는 일시 중지 기간을 30분 더 연장합니다. 30분이 `ResumeFullAutomationModeTime`에 표시된 원래 시간에 추가됩니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier my-custom-instance \
    --automation-mode all-paused \
    --resume-full-automation-mode-minutes 30
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --automation-mode all-paused ^
    --resume-full-automation-mode-minutes 30
```

다음 예제에서는 `my-custom-instance`에 대한 전체 자동화를 재개합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier my-custom-instance \
    --automation-mode full \
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --automation-mode full
```

다음 부분 샘플 출력에서 보류 중인 AutomationMode 값은 full입니다.

```
{
    "DBInstance": {
        "PubliclyAccessible": true,
        "MasterUsername": "admin",
        "MonitoringInterval": 0,
        "LicenseModel": "bring-your-own-license",
        "VpcSecurityGroups": [
            {
                "Status": "active",
                "VpcSecurityGroupId": "0123456789abcdefg"
            }
        ],
        "InstanceCreateTime": "2020-11-07T19:50:06.193Z",
        "CopyTagsToSnapshot": false,
        "OptionGroupMemberships": [
            {
                "Status": "in-sync",
                "OptionGroupName": "default:custom-oracle-ee-19"
            }
        ],
        "PendingModifiedValues": {
            "AutomationMode": "full"
        },
        "Engine": "custom-oracle-ee",
        "MultiAZ": false,
        "DBSecurityGroups": [],
        "DBParameterGroups": [
            {
                "DBParameterGroupName": "default.custom-oracle-ee-19",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        ...
        "ReadReplicaDBInstanceIdentifiers": [],
        "AllocatedStorage": 250,
        "DBInstanceArn": "arn:aws:rds:us-west-2:012345678912:db:my-custom-instance",
        "BackupRetentionPeriod": 3,
        "DBName": "ORCL",
        "PreferredMaintenanceWindow": "fri:10:56-fri:11:26",
        "Endpoint": {
            "HostedZoneId": "ABCDEFGHIJKLMNO",
            "Port": 8200,
```

```
            "Address": "my-custom-instance.abcdefghijk.us-west-2.rds.amazonaws.com"
        },
        "DBInstanceStatus": "automation-paused",
        "IAMDatabaseAuthenticationEnabled": false,
        "AutomationMode": "all-paused",
        "EngineVersion": "19.my_cev1",
        "DeletionProtection": false,
        "AvailabilityZone": "us-west-2a",
        "DomainMemberships": [],
        "StorageType": "gp2",
        "DbiResourceId": "db-ABCDEFGHIJKLMNOPQRSTUVW",
        "ResumeFullAutomationModeTime": "2020-11-07T20:56:50.565Z",
        "KmsKeyId": "arn:aws:kms:us-west-2:012345678912:key/
aa111a11-111a-11a1-1a11-1111a11a1a1a",
        "StorageEncrypted": false,
        "AssociatedRoles": [],
        "DBInstanceClass": "db.m5.xlarge",
        "DbInstancePort": 0,
        "DBInstanceIdentifier": "my-custom-instance",
        "TagList": []
    }
```


RDS Custom for Oracle DB 인스턴스 수정

RDS Custom for Oracle DB 인스턴스를 수정하는 작업은 Amazon RDS DB 인스턴스를 수정하는 것과 유사합니다. 다음과 같은 설정을 변경할 수 있습니다.

- DB 인스턴스 클래스
- 스토리지 할당 및 유형
- 백업 보관 기간
- 삭제 방지
- 옵션 그룹
- CEV(RDS Custom for Oracle DB 인스턴스 업그레이드 참조)
- Port

주제

- DB 인스턴스 스토리지 수정 시 요구 사항 및 제한
- DB 인스턴스 클래스 수정 시 요구 사항 및 제한

- 인스턴스 클래스 수정 시 RDS Custom이 DB 인스턴스를 생성하는 방법
- RDS Custom for Oracle DB 인스턴스 수정

DB 인스턴스 스토리지 수정 시 요구 사항 및 제한

RDS Custom for Oracle DB 인스턴스를 수정할 때는 다음과 같은 요구 사항과 제한을 고려하세요.

- RDS Custom for Oracle에 대해 할당된 최소 스토리지는 40GiB이며 최대 64TiB입니다.
- Amazon RDS와 마찬가지로 할당된 스토리지를 줄일 수는 없는데, 이것이 Amazon EBS 볼륨의 한계입니다.
- RDS Custom DB 인스턴스에는 스토리지 자동 크기 조정이 지원되지 않습니다.
- RDS Custom DB 인스턴스에 수동으로 연결하는 모든 스토리지 볼륨은 지원 경계를 벗어납니다.

  자세한 내용은 RDS Custom 지원 범위 단원을 참조하십시오.
- 마그네틱(표준) Amazon EBS 스토리지는 RDS Custom에 지원되지 않습니다. io1, gp2 또는 gp3 SSD 스토리지 유형만 선택할 수 있습니다.

Amazon EBS 스토리지에 대한 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 섹션을 참조하세요. 스토리지 수정에 대한 일반적인 정보는 Amazon RDS DB 인스턴스 스토리지 작업 섹션을 참조하세요.

DB 인스턴스 클래스 수정 시 요구 사항 및 제한

RDS Custom for Oracle DB 인스턴스의 인스턴스 클래스를 수정할 때는 다음과 같은 요구 사항과 제한을 고려하세요.

- DB 인스턴스는 `available` 상태여야 합니다.
- DB 인스턴스에는 루트 볼륨, 데이터 볼륨, 바이너리 볼륨에 최소 100MiB의 여유 공간이 있어야 합니다.
- 기본 탄력적 네트워크 인터페이스(ENI)를 사용할 때는 RDS Custom for Oracle DB 인스턴스에 탄력적 IP(EIP) 하나만 할당할 수 있습니다. DB 인스턴스에 여러 ENI를 연결하면 수정 작업이 실패합니다.
- 모든 RDS Custom for Oracle 태그가 있어야 합니다.
- RDS Custom for Oracle 복제를 사용하는 경우 다음과 같은 요구 사항과 제한에 유의하세요.
  - 기본 DB 인스턴스 및 읽기 전용 복제본의 경우 한 번에 하나의 DB 인스턴스만 인스턴스 클래스를 변경할 수 있습니다.

- RDS Custom for Oracle DB 인스턴스에 온프레미스 기본 또는 복제본 데이터베이스가 있는 경우 수정이 완료된 후 온프레미스 DB 인스턴스의 프라이빗 IP 주소를 수동으로 업데이트해야 합니다. 이 작업은 Oracle DataGuard 기능을 유지하는 데 필요합니다. RDS Custom for Oracle은 수정이 성공하면 이벤트를 게시합니다.
- 기본 또는 읽기 전용 복제본 DB 인스턴스에 FSFO(Fast-Start Failover)가 구성된 경우 RDS Custom for Oracle DB 인스턴스 클래스를 수정할 수 없습니다.

인스턴스 클래스 수정 시 RDS Custom이 DB 인스턴스를 생성하는 방법

인스턴스 클래스를 수정하는 경우 RDS Custom은 다음과 같이 DB 인스턴스를 생성합니다.

- Amazon EC2 인스턴스를 생성합니다.
- 최신 DB 스냅샷에서 루트 볼륨을 생성합니다. RDS Custom for Custom은 최신 DB 스냅샷 이후에 루트 볼륨에 추가된 정보를 유지하지 않습니다.
- Amazon CloudWatch 경보를 생성합니다.
- 원래 키 페어를 삭제한 경우 Amazon EC2 SSH 키 페어를 생성합니다. 그렇지 않은 경우 RDS Custom for Custom은 원래 키 페어를 유지합니다.
- 수정을 시작하면 DB 인스턴스에 연결된 태그를 사용하여 새 리소스를 생성합니다. RDS Custom은 태그가 기본 리소스에 직접 연결되면 태그를 새 리소스로 전송하지 않습니다.
- 최신 수정 사항이 포함된 바이너리 및 데이터 볼륨을 새 DB 인스턴스로 전송합니다.
- 탄력적 IP 주소(EIP)를 전송합니다. DB 인스턴스에 공개적으로 액세스할 수 있는 경우 RDS Custom은 EIP를 전송하기 전에 새 DB 인스턴스에 퍼블릭 IP 주소를 임시로 연결합니다. DB 인스턴스에 공개적으로 액세스할 수 없는 경우 RDS Custom은 퍼블릭 IP 주소를 생성하지 않습니다.

RDS Custom for Oracle DB 인스턴스 수정

콘솔, AWS CLI 또는 RDS API를 사용하여 DB 인스턴스 클래스나 스토리지를 수정할 수 있습니다.

콘솔

RDS Custom for Oracle DB 인스턴스를 수정하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 수정하려는 DB 인스턴스를 선택합니다.

4. Modify(수정)를 선택합니다.
5. (선택 사항) 인스턴스 구성에서 DB 인스턴스 클래스의 값을 선택합니다. 지원되는 클래스는 RDS Custom for Oracle에 대한 DB 인스턴스 클래스 지원 섹션을 참조하세요.
6. (선택 사항) 스토리지에서 필요에 따라 다음과 같이 변경합니다.
   a. Allocated storage(할당된 스토리지)에 새로운 값을 입력합니다. 현재 값보다 커야 하며 40GiB~64TiB여야 합니다.
   b. 스토리지 유형 값을 범용 SSD(gp2), 범용 SSD(gp3) 또는 프로비저닝된 IOPS(io1)로 변경합니다.
   c. 프로비저닝된 IOPS(io1) 또는 범용 SSD(gp3)를 사용하는 경우 프로비저닝된 IOPS 값을 변경할 수 있습니다.
7. (선택 사항) 추가 구성에서 필요에 따라 다음과 같이 변경합니다.
   - 옵션 그룹에서 새로운 옵션 그룹을 선택합니다. 자세한 내용은 RDS Custom for Oracle에서 옵션 그룹을 사용한 작업 단원을 참조하십시오.
8. Continue(계속)를 선택합니다.
9. 즉시 적용(Apply immediately) 또는 예약된 다음 유지 관리 기간에 적용(Apply during the next scheduled maintenance window)을 선택합니다.
10. Modify DB instance(DB 인스턴스 수정)를 선택합니다.

AWS CLI

RDS Custom for Oracle DB 인스턴스 스토리지를 수정하려면 modify-db-instance AWS CLI 명령을 사용하면 됩니다. 필요한 경우 다음 파라미터를 설정합니다.

- `--db-instance-class` - 새 인스턴스 클래스입니다. 지원되는 클래스는 RDS Custom for Oracle에 대한 DB 인스턴스 클래스 지원 섹션을 참조하세요.
- `--allocated-storage` - DB 인스턴스에 할당할 스토리지 크기(GiB)입니다. 현재 값보다 커야 하며 40~65,536GiB여야 합니다.
- `--storage-type` - 스토리지 유형: gp2, gp3 또는 io1.
- `--iops` - io1 또는 gp3 스토리지 유형을 사용하는 경우 DB 인스턴스에 대해 프로비저닝된 IOPS입니다.
- `--apply-immediately` – `--apply-immediately`를 사용하여 스토리지 변경 사항을 바로 적용합니다.

그 밖에 다음 유지 관리 기간에 스토리지 변경 사항을 적용하려면 `--no-apply-immediately`(기본값)를 사용합니다.

다음 예제는 my-cfo-instance의 DB 인스턴스 클래스를 db.m5.16xlarge로 변경합니다. 또한 이 명령은 스토리지 크기를 1TiB로, 스토리지 유형을 io1로, 프로비저닝된 IOPS를 3,000으로, 옵션 그룹을 cfo-ee-19-mt로 변경합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier my-cfo-instance \
    --db-instance-class db.m5.16xlarge \
    --storage-type io1 \
    --iops 3000 \
    --allocated-storage 1024 \
    --option-group cfo-ee-19-mt \
    --apply-immediately
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-cfo-instance ^
    --db-instance-class db.m5.16xlarge ^
    --storage-type io1 ^
    --iops 3000 ^
    --allocated-storage 1024 ^
    --option-group cfo-ee-19-mt ^
    --apply-immediately
```

RDS Custom for Oracle DB 인스턴스의 문자 집합 변경

RDS Custom for Oracle 문자 집합의 기본값은 US7ASCII입니다. 언어 또는 멀티바이트 문자 요구 사항을 충족하기 위해 서로 다른 문자 집합을 지정할 수 있습니다. RDS Custom for Oracle을 사용하는 경우 자동화를 일시 중지한 다음 데이터베이스의 문자 집합을 수동으로 변경할 수 있습니다.

RDS Custom for Oracle DB 인스턴스의 문자 집합을 변경하려면 다음 요구 사항을 충족해야 합니다.

- 애플리케이션 데이터가 없는 빈 데이터베이스나 스타터 데이터베이스가 있는 새로 프로비저닝된 RDS Custom 인스턴스의 문자만 변경할 수 있습니다. 다른 모든 경우에는 Database Migration Assistant for Unicode(DMU)를 사용하여 문자 집합을 변경하면 됩니다.
- RDS for Oracle에서 지원하는 문자 집합으로만 변경할 수 있습니다. 자세한 내용은 지원되는 DB 문자 집합 단원을 참조하세요.

RDS Custom for Oracle DB 인스턴스의 문자 집합을 변경하려면

1. RDS Custom 자동화를 일시 중지합니다. 자세한 내용은 RDS Custom DB 인스턴스 일시 중지 및 재개 단원을 참조하세요.
2. SYSDBA 권한을 사용하여 데이터베이스에 사용자로 로그인합니다.
3. 제한 모드에서 데이터베이스를 다시 시작하고 문자 집합을 변경한 다음 일반 모드에서 데이터베이스를 다시 시작합니다.

   SQL 클라이언트에서 다음 스크립트를 실행합니다.

   ```
   SHUTDOWN IMMEDIATE;
   STARTUP RESTRICT;
   ALTER DATABASE CHARACTER SET INTERNAL_CONVERT AL32UTF8;
   SHUTDOWN IMMEDIATE;
   STARTUP;
   SELECT VALUE FROM NLS_DATABASE_PARAMETERS WHERE PARAMETER = 'NLS_CHARACTERSET';
   ```

   출력에 올바른 문자 집합이 표시되는지 확인합니다.

   ```
   VALUE
   --------
   AL32UTF8
   ```
4. RDS Custom 자동화를 다시 시작합니다. 자세한 내용은 RDS Custom DB 인스턴스 일시 중지 및 재개 단원을 참조하십시오.

RDS Custom for Oracle의 NLS_LANG 값 설정

로캘은 지정된 언어와 국가에 해당하는 언어 및 문화적 요구 사항을 해결하는 일련의 정보입니다. Oracle 소프트웨어의 로캘 동작을 지정하려면 클라이언트 호스트에서 NLS_LANG 환경 변수를 설정합니다. 이 변수는 클라이언트 애플리케이션과 데이터베이스 세션에서 사용되는 언어, 지역, 문자를 설정합니다.

RDS Custom for Oracle의 경우 NLS_LANG 변수에서 언어만 설정할 수 있습니다. 지역 및 문자는 기본값을 사용합니다. 언어는 Oracle 데이터베이스 메시지, 데이터 정렬, 요일 이름, 월 이름에 사용됩니다. 지원되는 각 언어는 고유한 이름(예: 미국 영어, 프랑스어 또는 독일어)이 있습니다. 언어를 지정하지 않을 경우 기본값은 미국 영어입니다.

RDS Custom for Oracle 데이터베이스를 생성한 후 클라이언트 호스트에서 NLS_LANG을 영어가 아닌 다른 언어로 설정할 수 있습니다. Oracle Database에서 지원되는 언어 목록을 보려면 RDS Custom for Oracle 데이터베이스에 로그인한 후 다음 쿼리를 실행하세요.

```
SELECT VALUE FROM V$NLS_VALID_VALUES WHERE PARAMETER='LANGUAGE' ORDER BY VALUE;
```

호스트 명령줄에서 NLS_LANG을 설정할 수 있습니다. 다음 예제에서는 Linux에서 Z 쉘을 사용하는 클라이언트 애플리케이션의 언어를 독일어로 설정합니다.

```
export NLS_LANG=German
```

애플리케이션은 시작 시 NLS_LANG 값을 읽은 다음, 연결 시 이를 데이터베이스에 전달합니다.

자세한 내용은 Oracle Database Globalization Support Guide의 Choosing a Locale with the NLS_LANG Environment Variable(NLS_LANG 환경 변수를 사용하여 로캘 선택)을 참조하세요.

투명한 데이터 암호화 지원

RDS Custom은 RDS Custom for Oracle DB 인스턴스에 대해 투명한 데이터 암호화(TDE)를 지원합니다.

그러나 RDS for Oracle에서는 사용자 지정 옵션 그룹의 옵션을 사용하여 TDE를 활성화할 수 없습니다. TDE를 수동으로 켭니다. Oracle의 투명한 데이터 보안 사용에 대한 정보는 Oracle 문서의 투명한 데이터 암호화를 사용하여 저장된 데이터 보안을 참조하세요.

RDS Custom for Oracle 리소스 태깅

Amazon RDS 리소스와 마찬가지로 RDS Custom 리소스에 태그를 지정할 수 있지만, 몇 가지 중요한 차이점이 있습니다.

- RDS Custom 자동화에 필요한 AWSRDSCustom 태그를 생성하거나 수정해서는 안 됩니다. 이렇게 하면 자동화가 중단될 수 있습니다.
- Name 태그는 접두사 값이 rds-custom!oracle-do-not-delete 또는 do-not-delete-rds-custom인 RDS Custom 리소스에 추가됩니다. 고객이 전달한 키 값을 덮어씁니다.

- 생성 중에 RDS Custom DB 인스턴스에 추가된 태그는 다른 모든 관련 RDS Custom 리소스로 전파됩니다.
- DB 인스턴스 생성 후 RDS Custom 리소스에 태그를 추가하면 태그가 전파되지 않습니다.

리소스 태그 지정에 대한 일반적인 정보는 Amazon RDS 리소스에 태그 지정 섹션을 참조하세요.

RDS Custom for Oracle DB 인스턴스 삭제

RDS Custom DB 인스턴스를 삭제하려면 다음을 수행하세요.

- DB 인스턴스 이름을 입력합니다.
- DB 인스턴스의 최종 DB 스냅샷을 생성하는 옵션을 비활성화합니다.
- 자동화된 백업을 유지하는 옵션을 선택하거나 선택 취소합니다.

콘솔이나 CLI를 사용하여 RDS Custom DB 인스턴스를 삭제할 수 있습니다. DB 인스턴스를 삭제하는 데 필요한 시간은 백업 보존 기간(삭제할 백업 수)과 삭제되는 데이터 양에 따라 달라질 수 있습니다.

콘솔

RDS Custom DB 인스턴스를 삭제하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스(Databases)를 선택한 후 삭제하려는 RDS Custom DB 인스턴스를 선택합니다. RDS Custom DB 인스턴스는 역할 인스턴스(RDS Custom)(Instance (RDS Custom))를 보여줍니다.
3. 작업에 대해 삭제를 선택합니다.
4. 자동 백업을 보관하려면 Retain automated backups(자동 백업 보관)를 선택합니다.
5. 상자에 **`delete me`**를 입력합니다.
6. 삭제를 선택합니다.

AWS CLI

delete-db-instance AWS CLI 명령을 사용하여 RDS Custom DB 인스턴스를 삭제할 수 있습니다. 필수 파라미터 `--db-instance-identifier`를 사용하여 DB 인스턴스를 식별합니다. 나머지 파라미터는 Amazon RDS DB 인스턴스와 동일하지만, 다음과 같은 예외가 있습니다.

- `--skip-final-snapshot`은 필수입니다.
- `--no-skip-final-snapshot`는 지원되지 않습니다.
- `--final-db-snapshot-identifier`는 지원되지 않습니다.

다음 예제에서는 이름이 `my-custom-instance`인 RDS Custom DB 인스턴스를 삭제하고 자동화된 백업을 유지합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds delete-db-instance \
    --db-instance-identifier my-custom-instance \
    --skip-final-snapshot \
    --no-delete-automated-backups
```

Windows의 경우:

```
aws rds delete-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --skip-final-snapshot ^
    --no-delete-automated-backups
```

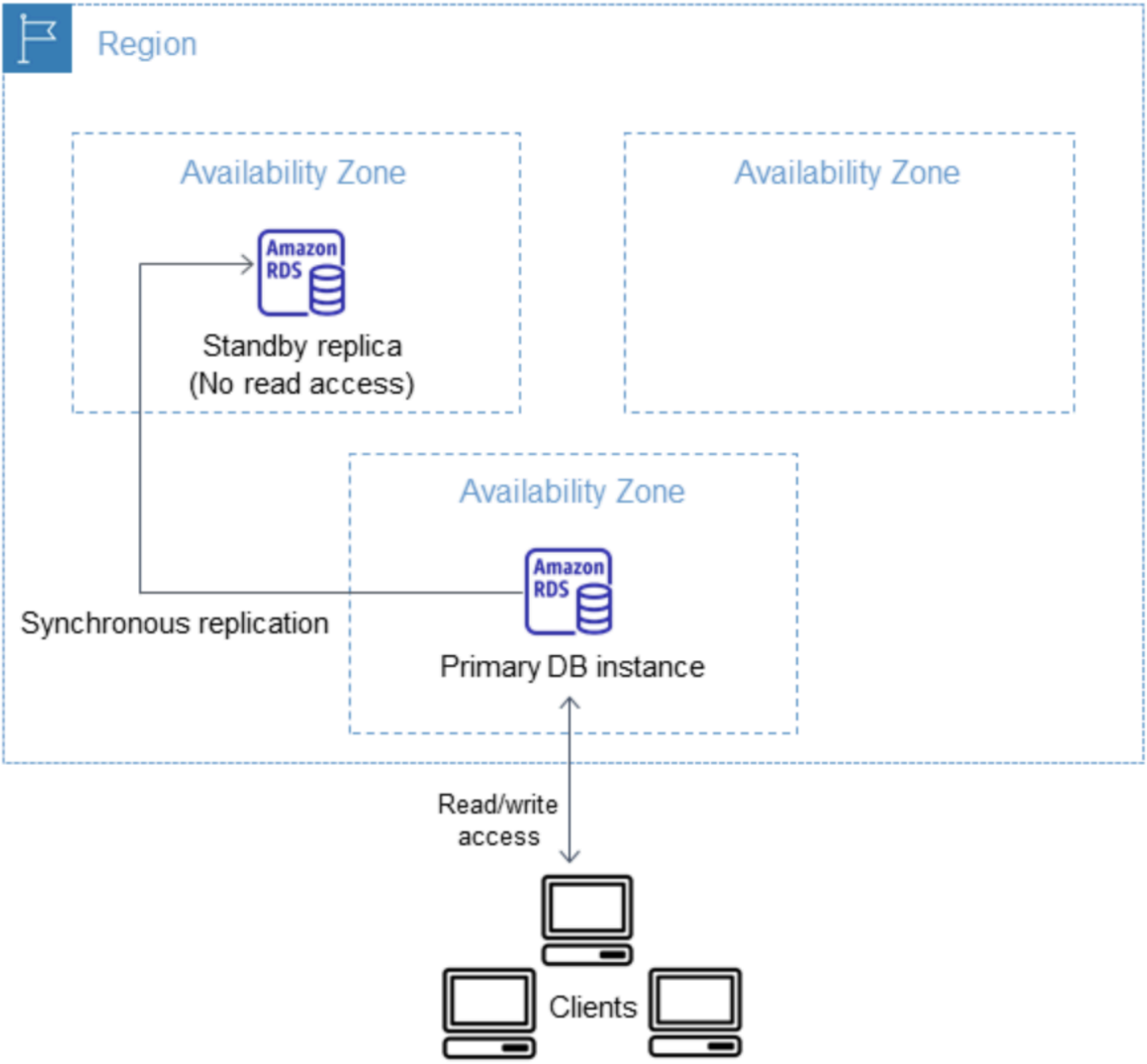
RDS Custom for Oracle에 대한 다중 AZ 배포 관리

RDS Custom for Oracle에 대한 다중 AZ DB 인스턴스 배포에서 Amazon RDS는 자동으로 서로 다른 가용 영역(AZ)에 동기식 대기 복제본을 프로비저닝하고 유지 관리합니다. 기본 DB 인스턴스는 가용 영역 전체에서 대기 복제본으로 동기식으로 복제되어 데이터 중복성을 제공합니다.

> **Important**
>
> RDS Custom for Oracle에 대한 다중 AZ 배포는 RDS for Oracle에 대한 다중 AZ 배포와 다릅니다. RDS for Oracle에 대한 다중 AZ와 달리, 다중 AZ DB 인스턴스를 만들기 전에 RDS Custom for Oracle의 사전 조건을 설정해야 합니다. RDS Custom은 사용자의 계정 내에서 실행되어 권한이 필요하기 때문입니다. 또한 다중 AZ 배포를 지원하는 최신 서비스 제공 AMI를 사용하여 CEV에서 다중 AZ 인스턴스를 생성해야 합니다.
> 사전 조건을 완료하지 않으면 다중 AZ DB 인스턴스가 실행되지 않거나, 단일 AZ DB 인스턴스로 자동으로 되돌아갈 수 있습니다. 사전 조건에 대한 자세한 내용은 the section called “사전 조건” 단원을 참조하세요.

DB 인스턴스를 고가용성으로 실행하면 계획된 시스템 유지 관리 중 가용성을 높일 수 있습니다. 계획된 데이터베이스 유지 관리 또는 예기치 않은 서비스 중단이 발생할 경우, Amazon RDS가 최신 대기 DB 인스턴스로 자동으로 장애 조치를 수행합니다. 이 기능을 통해 수동 개입 없이 데이터베이스 작업을 빠르게 재개할 수 있습니다. 기본 인스턴스 및 대기 인스턴스는 동일한 엔드포인트를 사용합니다. 이 엔드포인트의 물리적 네트워크 주소는 장애 조치 프로세스의 일환으로 대기 복제본으로 전환됩니다. 장애 조치가 발생하는 경우 애플리케이션을 다시 구성할 필요가 없습니다.

다음 다이어그램은 RDS Custom for Oracle의 다중 AZ 아키텍처를 보여줍니다.



RDS Custom DB 인스턴스 생성 시 다중 AZ를 지정하여 RDS Custom for Oracle 다중 AZ 배포를 생성할 수 있습니다. 2025년 6월 30일 이전에 생성된 RDS Custom for Oracle DB 인스턴스는 DB 인스턴스를 수정하고 다중 AZ 옵션을 지정하여 다중 AZ 배포로 변환할 수 없습니다. 다중 AZ 옵션을 지정하여 새 RDS Custom for Oracle DB 인스턴스를 생성해야 합니다. 2025년 6월 30일 후에 사용 가능한 서비스 제공 AMI를 사용하여 생성된 새 DB 인스턴스는 다중 AZ 옵션을 지원합니다. 그런 다음, 기존 RDS Custom for Oracle 데이터베이스를 새로 생성된 DB 인스턴스로 마이그레이션해야 합니다. 자세한 내용은 the section called “2025년 6월 30일 전 CEV의 마이그레이션”을 참조하세요.

Amazon RDS 콘솔에 대기 복제본의 가용 영역(보조 AZ)이 표시됩니다. `describe-db-instances` CLI 명령 또는 `DescribeDBInstances` API 작업을 사용하여 보조 AZ를 찾을 수도 있습니다.

다중 AZ 배포를 사용하는 RDS Custom for Oracle DB 인스턴스는 단일 AZ 배포에 비해 쓰기 및 커밋 지연 시간이 길어질 수 있습니다. 이러한 대기 시간 증가는 DB 인스턴스 간의 동기 데이터 복제로 인해 발생할 수 있습니다. AWS는 가용 영역 간 지연 시간이 짧은 네트워크 연결을 제공하도록 설계되었지만 배포가 예비 복제본으로 장애 조치될 경우 지연 시간이 변경될 수 있습니다.

주제

- 리전 및 버전 사용 가능 여부
- RDS Custom for Oracle을 사용한 다중 AZ 배포의 제한 사항
- 다중 AZ 배포를 사용하는 RDS Custom for Oracle DB 인스턴스 생성
- RDS Custom for Oracle을 사용한 다중 AZ 배포의 사전 조건
- RDS Custom for Oracle 단일 AZ 배포를 다중 AZ 배포로 수정
- RDS Custom for Oracle 다중 AZ 배포를 단일 AZ 배포로 수정
- RDS Custom for Oracle 다중 AZ 배포에 대한 장애 조치 프로세스
- AWS CloudFormation을 사용하여 RDS Custom for Oracle 배포

리전 및 버전 사용 가능 여부

다음과 같은 Oracle Database 에디션의 경우 RDS Custom for Oracle에 대한 다중 AZ 배포가 지원됩니다.

- Oracle Database 12c 릴리스 1(12.1)
- Oracle Database 12c 릴리스 2(12.2)
- Oracle Database 19c
- Enterprise Edition(EE)
- Standard Edition 2(SE2)

> **Note**
>
> RDS Custom for Oracle에 대한 다중 AZ 배포는 Oracle Database 18c에서 지원되지 않습니다.

RDS Custom for Oracle에 대한 다중 AZ 배포는 RDS Custom for Oracle을 사용 가능한 모든 리전에서 사용할 수 있습니다. RDS Custom for Oracle에 대한 다중 AZ 배포의 리전 가용성에 대한 자세한 내용은 the section called “RDS Custom for Oracle을 지원하는 리전 및 DB 엔진” 섹션을 참조하세요.

RDS Custom for Oracle을 사용한 다중 AZ 배포의 제한 사항

RDS Custom for Oracle을 사용한 다중 AZ 배포에는 다음과 같은 제한 사항이 있습니다.

- 리전 간 다중 AZ 배포는 지원되지 않습니다.
- 대기 DB 인스턴스가 데이터베이스 읽기 작업을 허용하도록 구성할 수 없습니다.
- 다중 AZ 배포와 함께 사용자 지정 엔진 버전(CEV)을 사용할 경우 대기 DB 인스턴스도 동일한 CEV를 사용합니다. 대기 DB 인스턴스는 다른 CEV를 사용할 수 없습니다.
- 다중 AZ 배포 인스턴스에서 읽기 전용 복제본을 생성할 수 없으며 다중 AZ 배포를 사용하도록 읽기 전용 복제본의 기본 인스턴스를 수정할 수도 없습니다.

다중 AZ 배포를 사용하는 RDS Custom for Oracle DB 인스턴스 생성

다중 AZ 배포를 사용하는 RDS Custom for Oracle DB 인스턴스를 생성하려면the section called “RDS Custom for Oracle 환경 설정”의 단계에 따라 사전 조건으로 환경을 설정합니다.

> **Important**
>
> 설정을 간소화하려면 네트워크 설정 지침에 제공된 최신 AWS CloudFormation 템플릿 파일을 사용하는 것이 좋습니다. 자세한 내용은 the section called “AWS CloudFormation를 사용하여 배포” 섹션을 참조하세요.

Amazon RDS 콘솔에서 데이터베이스 인스턴스를 생성하는 동안 다중 AZ 옵션을 선택하여 다중 AZ 배포를 사용하는 RDS Custom for Oracle 인스턴스를 생성할 수 있습니다. 아니면 AWS CLI의 Amazon RDS `create-db-instance` 명령에서 `--multi-az` 파라미터를 지정하면 됩니다.

RDS Custom for Oracle을 사용한 다중 AZ 배포의 사전 조건

> **Note**
>
> 2025년 6월 30일 이전에 RDS Custom for Oracle 인스턴스를 생성한 경우 단일 AZ에서 다중 AZ 배포로 수정할 수 없습니다. 기본 사용자 지정 엔진 버전(CEV)은 다중 AZ 배포 지원이 없는 이전 서비스 제공 AMI로 빌드되었습니다.

2025년 6월 30일 후 서비스 제공 AMI로 생성한 CEV를 사용하여 데이터베이스를 새 인스턴스로 마이그레이션해야 합니다.
the section called “2025년 6월 30일 전 CEV의 마이그레이션”을(를) 참조하세요.

RDS Custom for Oracle은 단일 AZ에서 다중 AZ 배포로 변환할 때 특정 사전 조건을 요구합니다. 사전 조건이 불완전하게 충족되면 다중 AZ 설정이 실패합니다. 네트워크 설정 지침에 제공된 수동 설정 또는 최신 AWS CloudFormation 템플릿 파일을 사용하세요. 자세한 내용은 the section called “3단계: RDS Custom for Oracle용 CloudFormation 템플릿 추출” 섹션을 참조하세요.

사전 조건을 수동으로 충족하려면 the section called “단일 AZ에서 다중 AZ로 변경”의 단계를 따르고 다음 사항에 유의하세요.

- RDS Custom for Oracle DB 인스턴스가 2025년 6월 30일 후에 생성된 CEV를 서비스 제공 최신 AMI와 함께 사용해야 합니다.
- 포트 1120을 허용하도록 Amazon RDS 보안 그룹 인바운드 및 아웃바운드 규칙을 업데이트합니다.
- Amazon RDS Custom for Oracle DB 인스턴스가 Amazon SQS와 통신할 수 있도록 새 Amazon SQS VPC 엔드포인트를 생성합니다.
- 인스턴스 프로파일 역할에서 Amazon SQS 권한을 업데이트합니다.

Important

RDS Custom for Oracle 자동화를 일시 중지하지 않고 인스턴스에 로그인하여 다중 AZ 기본 DB 인스턴스를 수동으로 재부팅하지 마세요.

2025년 6월 30일 이전에 생성한 CEV를 사용하는 DB 인스턴스의 마이그레이션 단계

2025년 6월 30일 이전에 RDS Custom for Oracle DB 인스턴스를 생성한 경우 기본 CEV가 다중 AZ 지원이 없는 이전 서비스 제공 AMI로 빌드되었기 때문에 다중 AZ 배포로 직접 수정할 수 없습니다. 2025년 6월 30일 후에 생성한 CEV를 사용하여 데이터베이스를 새 인스턴스로 마이그레이션해야 합니다. 마이그레이션을 위해 다음을 수행합니다.

1. 최신 서비스 제공 AMI를 사용하여 소스에서 CEV 생성(2025년 6월 30일 후)
2. 새 CEV를 사용하여 새 DB 인스턴스 시작
3. 아래 방법 중 하나를 사용하여 다중 AZ 배포를 지원하지 않는 기존 DB 인스턴스에서 새로 생성된 인스턴스로 데이터베이스를 마이그레이션합니다.

- Physical migration of Oracle databases to Amazon RDS Custom using Data Guard
- Physical migration of Oracle databases to Amazon RDS Custom using RMAN duplication

RDS Custom for Oracle 단일 AZ 배포를 다중 AZ 배포로 수정

다중 AZ와 호환되는 기존 RDS Custom for Oracle DB 인스턴스를 단일 AZ 배포에서 다중 AZ 배포로 수정할 수 있습니다. DB 인스턴스를 수정할 경우 Amazon RDS는 여러 가지 작업을 수행합니다.

- 기본 DB 인스턴스의 스냅샷을 생성합니다.
- 스냅샷에서 스탠바이 복제본용 새 볼륨을 생성합니다. 이러한 볼륨은 백그라운드에서 초기화되며 데이터가 완전히 초기화된 후에 최대 볼륨 성능이 달성됩니다.
- 기본 DB 인스턴스와 대기 DB 인스턴스 간의 동기식 블록 수준 복제를 활성화합니다.

> **Important**
>
> 사용량이 가장 많은 기간에는 RDS Custom for Oracle DB 인스턴스를 단일 AZ에서 프로덕션 DB 인스턴스의 다중 AZ 배포로 수정하지 않는 것이 좋습니다.

AWS는 스냅샷을 사용하여 대기 인스턴스를 생성하면 단일 AZ에서 다중 AZ로 변환할 때 가동 중지 시간을 피할 수 있지만, 다중 AZ로 변환하는 동안과 이후에 성능에 영향을 미칠 수 있습니다. 이는 쓰기 대기 시간에 민감한 워크로드에 상당한 영향을 미칠 수 있습니다. 이 기능을 사용하면 스냅샷에서 대용량 볼륨을 신속하게 복원할 수 있지만, 이 경우 동기식 복제로 인해 I/O 작업의 지연 시간이 증가할 수 있습니다. 이러한 지연 시간은 데이터베이스 성능에 영향을 줄 수 있습니다.

CloudFormation을 사용하여 단일 AZ를 다중 AZ 배포로 수정하기 위한 사전 조건 구성

the section called “3단계: RDS Custom for Oracle용 CloudFormation 템플릿 추출” 섹션을 따라 VPC 및 IAM 프로파일을 다시 설정하여 IAM 프로파일에 SQS VPC 엔드포인트 및 SQS 권한을 추가합니다.

단일 AZ를 다중 AZ 배포로 수동으로 수정하기 위한 사전 조건 구성

사전 조건을 수동으로 구성하도록 선택한 경우 다음 태스크를 수행합니다.

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. Endpoint(엔드포인트)를 선택합니다. 엔드포인트 생성페이지가 나타납니다.
3. 서비스 범주에서 AWS 서비스를 선택합니다.

4. 서비스에서 SQS를 검색합니다.
5. VPC에서 RDS Custom for Oracle DB 인스턴스가 배포되는 VPC를 선택합니다.
6. 서브넷에서 RDS Custom for Oracle DB 인스턴스가 배포되는 서브넷을 선택합니다.
7. 보안 그룹에서 RDS Custom for Oracle DB 인스턴스가 배포되는 보안 그룹을 선택합니다.
8. 정책에서 사용자 지정을 선택합니다.
9. 사용자 지정 정책에서 `AWS partition`, `Region`, `accountId` 및 `IAM-Instance-role`을 실제 값으로 바꿉니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/AWSRDSCustom": "custom-oracle"
                }
            },
            "Action": [
                "SQS:SendMessage",
                "SQS:ReceiveMessage",
                "SQS:DeleteMessage",
                "SQS:GetQueueUrl"
            ],
            "Resource": "arn:${AWS::Partition}:sqs:${AWS::Region}:${AWS::AccountId}:do-
not-delete-rds-custom-*",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:${AWS::Partition}:iam::${AWS::AccountId}:role/{IAM-
Instance-role}"
            }
        }
    ]
}
```

Amazon SQS에 액세스할 수 있는 권한으로 인스턴스 프로파일을 업데이트합니다. `AWS partition`, `Region`, `accountId`를 실제 값으로 바꿉니다.

```
{
    "Sid": "13",
```

```
    "Effect": "Allow",
    "Action": [
        "SQS:SendMessage",
        "SQS:ReceiveMessage",
        "SQS:DeleteMessage",
        "SQS:GetQueueUrl"
    ],
    "Resource": [
        {
            "Fn::Sub": "arn:${AWS::Partition}:sqs:${AWS::Region}:${AWS::AccountId}:do-
not-delete-rds-custom-*"
        }
    ],
    "Condition": {
        "StringLike": {
            "aws:ResourceTag/AWSRDSCustom": "custom-oracle"
        }
    }
}
```

포트 1120을 허용하도록 Amazon RDS 보안 그룹 인바운드 및 아웃바운드 규칙을 업데이트합니다.

- 보안 그룹에서 RDS Custom for Oracle DB 인스턴스가 배포되는 그룹을 선택합니다.
- 인바운드 규칙에서 소스 그룹의 포트 1120을 허용하는 사용자 지정 TCP 규칙을 만듭니다.
- 아웃바운드 규칙에서 대상 그룹의 포트 1120을 허용하는 사용자 지정 TCP 규칙을 만듭니다.

RDS 콘솔, AWS CLI 또는 RDS API를 사용하여 수정

사전 조건을 충족한 후 RDS 콘솔, AWS CLI 또는 RDS API를 사용하여 RDS Custom for Oracle DB 인스턴스를 단일 AZ에서 다중 AZ 배포로 수정할 수 있습니다.

콘솔

기존 RDS Custom for Oracle 단일 AZ 배포를 다중 AZ 배포로 수정하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 콘솔에서 데이터베이스를 선택합니다. 데이터베이스 창이 표시됩니다.
3. 수정하려는 RDS Custom for Oracle DB 인스턴스를 선택합니다.
4. 작업에서 다중 AZ 배포로 변환을 선택합니다.

5. 확인 페이지에서 즉시 적용을 선택하여 변경 사항을 즉시 적용합니다. 이 옵션을 선택하면 다운타임이 발생하지 않지만 성능이 영향을 받을 수 있습니다. 다음 유지 관리 기간에 업데이트를 적용하도록 선택할 수도 있습니다. 자세한 내용은 the section called “수정 예약” 섹션을 참조하세요.
6. 확인 페이지에서 다중 AZ로 변환을 선택합니다.

AWS CLI

AWS CLI를 사용하여 다중 AZ DB 인스턴스 배포로 변환하려면 modify-db-instance 명령을 호출하고 `--multi-az` 옵션을 설정합니다. DB 인스턴스 식별자와 수정하려는 기타 옵션 값을 지정합니다. 각 옵션에 대한 자세한 내용은 the section called “사용 가능한 설정” 단원을 참조하세요.

Example 예제

다음 코드는 `--multi-az` 옵션을 포함하여 `mycustomdbinstance`를 수정합니다. 변경 사항은 `--no-apply-immediately`를 사용하여 다음 유지 관리 기간에 적용됩니다. 변경 사항을 바로 적용하려면 `--apply-immediately`를 사용합니다. 자세한 내용은 the section called “수정 예약” 섹션을 참조하세요.

Linux, macOS 또는 Unix의 경우는 다음과 같습니다.

```
aws rds modify-db-instance \
    --db-instance-identifier mycustomdbinstance \
    --multi-az \
    [--no-apply-immediately | --apply-immediately]
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mycustomdbinstance ^
    --multi-az ^
    [--no-apply-immediately | --apply-immediately]
```

RDS API

Amazon RDS API를 사용하여 다중 AZ DB 인스턴스 배포로 변환하려면 ModifyDBInstance 작업을 직접 호출하고 `MultiAZ` 파라미터를 true로 설정합니다.

RDS Custom for Oracle 다중 AZ 배포를 단일 AZ 배포로 수정

기존 RDS Custom for Oracle DB 인스턴스를 다중 AZ 배포에서 단일 AZ 배포로 수정할 수 있습니다.

콘솔

RDS Custom for Oracle DB 인스턴스를 다중 AZ 배포에서 단일 AZ 배포로 수정하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 콘솔에서 데이터베이스를 선택합니다. 데이터베이스 창이 표시됩니다.
3. 수정하려는 RDS Custom for Oracle DB 인스턴스를 선택합니다.
4. 다중 AZ 배포의 경우 No(아니요)를 선택합니다.
5. 확인 페이지에서 즉시 적용을 선택하여 변경 사항을 즉시 적용합니다. 이 옵션을 선택하면 다운타임이 발생하지 않지만 성능이 영향을 받을 수 있습니다. 다음 유지 관리 기간에 업데이트를 적용하도록 선택할 수도 있습니다. 자세한 내용은 the section called “수정 예약” 섹션을 참조하세요.
6. 확인 페이지에서 DB 인스턴스 수정을 선택합니다.

AWS CLI

AWS CLI를 사용하여 다중 AZ 배포를 단일 AZ 배포로 수정하려면 modify-db-instance 명령을 호출하고 `--no-multi-az` 옵션을 포함합니다. DB 인스턴스 식별자와 수정하려는 기타 옵션 값을 지정합니다. 각 옵션에 대한 자세한 내용은 the section called “사용 가능한 설정” 단원을 참조하세요.

Example 예제

다음 코드는 `--no-multi-az` 옵션을 포함하여 `mycustomdbinstance`를 수정합니다. 변경 사항은 `--no-apply-immediately`를 사용하여 다음 유지 관리 기간에 적용됩니다. 변경 사항을 바로 적용하려면 `--apply-immediately`를 사용합니다. 자세한 내용은 the section called “수정 예약” 섹션을 참조하세요.

Linux, macOS 또는 Unix의 경우는 다음과 같습니다.

```
aws rds modify-db-instance \
    --db-instance-identifier mycustomdbinstance \
    --no-multi-az \
    [--no-apply-immediately | --apply-immediately]
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mycustomdbinstance ^
    --no-multi-az ^
```

```
    [--no-apply-immediately | --apply-immediately]
```

RDS API

Amazon RDS API를 사용하여 다중 AZ 배포를 단일 AZ 배포로 수정하려면 ModifyDBInstance 작업을 직접 호출하고 `MultiAZ` 파라미터를 `false`로 설정합니다.

RDS Custom for Oracle 다중 AZ 배포에 대한 장애 조치 프로세스

계획되거나 계획되지 않은 DB 인스턴스의 운영 중단으로 인해 인프라 장애가 발생한 경우, 다중 AZ를 설정하면 Amazon RDS는 자동으로 다른 가용 영역의 대기 복제본으로 전환됩니다. 장애 조치가 완료되는 데 소요되는 시간은 프라이머리 DB 인스턴스를 사용할 수 없게 된 시점의 데이터베이스 활동 및 기타 조건에 따라 달라집니다. 장애 조치에 소요되는 시간은 일반적으로 60~120초입니다. 그러나 트랜잭션의 규모가 크거나 복구 프로세스가 복잡한 경우 장애 조치에 소요되는 시간이 증가할 수 있습니다. 장애 조치가 완료되면 Amazon RDS 콘솔에 새 가용 영역이 표시되는 데 시간이 더 걸릴 수 있습니다.

> **Note**
>
> DB 인스턴스를 사용할 수 있는 동안 기본 EC2 호스트를 중지하고 시작할 때 장애 조치를 수동으로 수행할 수 있습니다.

Amazon RDS는 자동으로 장애 조치를 취하여 관리자의 개입 없이 데이터베이스 작업을 신속하게 재개할 수 있도록 합니다. 다음 표에 설명된 조건 중 하나가 발생하면 기본 DB 인스턴스는 자동으로 예비 복제본으로 전환됩니다. 이 장애 조치 이유는 Amazon RDS 이벤트 로그에서 확인할 수 있습니다.

장애 조치 이유	설명
RDS 데이터베이스 인스턴스 기반의 운영 체제가 오프라인 작업에서 패치되고 있습니다.	OS 패치 또는 보안 업데이트를 위한 유지 관리 기간 동안 장애 조치가 트리거되었습니다. 자세한 내용은 the section called “DB 인스턴스 유지 관리” 섹션을 참조하세요.
RDS 다중 AZ 인스턴스의 기본 호스트가 비정상입니다.	다중 AZ DB 인스턴스 배포에서 손상된 프라이머리 DB 인스턴스를 감지하여 장애 조치를 수행했습니다.
네트워크 연결 손실로 인해 RDS 다중 AZ 인스턴스의 기본 호스트에 연결할 수 없습니다.	RDS 모니터링이 기본 DB 인스턴스에 대한 네트워크 연결 실패를 감지하여 장애 조치를 트리거했습니다.

장애 조치 이유	설명
RDS 인스턴스를 고객이 수정했습니다.	RDS DB 인스턴스 수정 때문에 장애 조치가 트리거되었습니다. 자세한 내용은 the section called “DB 인스턴스 수정” 섹션을 참조하세요.
RDS 다중 AZ 인스턴스의 기본 호스트 기반의 스토리지 볼륨에 오류가 발생했습니다.	다중 AZ DB 인스턴스 배포가 프라이머리 DB 인스턴스에서 스토리지 문제를 감지하여 장애 조치를 수행했습니다.
RDS 다중 AZ 기본 인스턴스가 사용 중이며 응답하지 않습니다.	기본 DB 인스턴스가 응답하지 않습니다. CPU, 메모리 또는 스왑 공간이 과도하게 사용되는지 이벤트 및 CloudWatch 로그에서 조사하는 것이 좋습니다. 자세한 내용은 the section called “Amazon RDS 이벤트 알림 작업” 및 the section called “Amazon RDS 이벤트에서 트리거되는 규칙 생성” 단원을 참조하십시오. 워크로드를 평가하여 적절한 DB 인스턴스 클래스를 사용 중인지 확인합니다. 자세한 내용은 the section called “DB 인스턴스 클래스” 섹션을 참조하세요.

다음 단계에 따라 다중 AZ DB 인스턴스가 장애 조치를 수행했는지 확인할 수 있습니다.

- 장애 조치가 시작되었음을 이메일 또는 SMS로 사용자에게 알리도록 DB 이벤트 구독을 설정합니다. 이벤트에 대한 자세한 내용은 the section called “Amazon RDS 이벤트 알림 작업” 단원을 참조하세요.
- Amazon RDS 콘솔 또는 API 작업을 사용하여 DB 이벤트를 확인합니다.
- Amazon RDS 콘솔, CLI 또는 API 작업을 사용하여 RDS Custom for Oracle 다중 AZ DB 인스턴스 배포의 현재 상태를 확인합니다.

RDS Custom for Oracle 다중 AZ 배포를 사용하는 애플리케이션의 유지 시간(TTL) 설정

장애 조치 메커니즘은 DB 인스턴스의 Domain Name System(DNS) 레코드가 예비 DB 인스턴스를 가리키도록 자동으로 변경합니다. 그 결과 DB 인스턴스의 기존 연결을 모두 재설정해야 합니다. DNS 캐시 TTL(Time-to-Live) 구성 값이 낮은지 확인하고, 애플리케이션이 오랜 시간 동안 DNS를 캐시하지 않는지 확인합니다. TTL 값이 높으면 장애 조치 후 애플리케이션이 DB 인스턴스에 빠르게 다시 연결되지 않을 수 있습니다.

AWS CloudFormation을 사용하여 RDS Custom for Oracle 배포

제공된 AWS CloudFormation 템플릿을 사용하여 RDS Custom for Oracle 배포를 자동화합니다. 리소스를 배포하기 전에 다음 사전 조건을 충족합니다.

사전 조건

필수 Oracle 파일 다운로드

AWS CloudFormation 템플릿을 생성하려면 먼저 특정 Oracle 설치 파일이 필요합니다. 배포하기 전에 이러한 파일을 다운로드하세요.

1. Oracle Database 19c (19.3)로 이동합니다.
2. `LINUX.X64_193000_db_home.zip` 파일을 찾아 다운로드합니다.
3. 파일 이름을 `V982063-01.zip`으로 변경합니다.
4. 플랫폼 또는 언어를 `Linux x86-64`로 선택하여 나머지 패치를 다운로드합니다.

최신 OPatch 유틸리티

Patch 6880880

2023년 1월 PSU 패치

데이터베이스 PSU 및 RU 패치

- Patch 34765931
- Patch 34786990

추가 필수 패치

- Patch 35099667
- Patch 35099674
- Patch 28730253
- Patch 29213893
- Patch 35012866

2023년 4월 PSU 패치

데이터베이스 PSU 및 RU 패치

- Patch 35042068
- Patch 35050341

추가 필수 패치

- Patch 28730253
- Patch 29213893
- Patch 33125873
- Patch 35220732
- Patch 35239280

Amazon S3 버킷 설정

1. AWS 계정에서 Amazon S3 버킷을 생성하거나 기존 버킷을 선택합니다.
2. 아래 예시와 유사한 폴더 구조를 버킷에 생성합니다.

```
<bucket-name>/
### oracle_cev/
    ### V982063-01.zip
    ### p6880880_190000_Linux-x86-64.zip
    ### p34765931_190000_Linux-x86-64.zip
    ### p34786990_190000_Linux-x86-64.zip
    ### p35099667_190000_Linux-x86-64.zip
    ### p35099674_190000_Generic.zip
    ### p28730253_190000_Linux-x86-64.zip
    ### p29213893_1918000DBRU_Generic.zip
    ### p35012866_1918000DBRU_Linux-x86-64.zip
    ### p35042068_190000_Linux-x86-64.zip
    ### p35050341_190000_Linux-x86-64.zip
    ### p29213893_1919000DBRU_Generic.zip
    ### p33125873_1919000DBRU_Linux-x86-64.zip
    ### p35220732_190000_Linux-x86-64.zip
    ### p35239280_190000_Generic.zip
```

3. 이전에 다운로드한 모든 Oracle 파일을 적절한 폴더에 업로드합니다.

AWS CloudFormation을 사용하여 RDS Custom for Oracle 배포

1단계: AWS CloudFormation 템플릿 준비

RDS Custom for Oracle을 배포하려면 먼저 필요한 사전 조건을 생성하는 AWS CloudFormation 템플릿을 다운로드하고 구성해야 합니다.

템플릿 복사 및 저장

1. Deploying RDS Custom for Oracle with single and multiple Availability Zones으로 이동합니다.
2. 원하는 형식(YAML 또는 JSON)으로 템플릿을 복사합니다.
3. 파일을 YAML 또는 JSON 형식으로 저장합니다. 예: `rds-custom-oracle-prereqs.yaml`

AWS Console에서 스택 시작

1. AWS Console을 열고 AWS CloudFormation으로 이동합니다.
2. 스택 생성 > 새 리소스 사용(표준)을 선택합니다.
3. 기존 템플릿 선택을 선택합니다.
4. 템플릿 파일 업로드 > 파일 선택을 선택합니다.
5. 이전에 다운로드한 템플릿 파일을 선택합니다.
6. 기본 파라미터 값을 유지합니다.
7. 다음을 선택하여 스택을 생성합니다.

대안: AWS CLI 사용

콘솔을 사용하는 대신 AWS CLI를 사용하여 스택을 생성할 수 있습니다.

```
aws cloudformation create-stack \
  --stack-name rds-custom-oracle \
  --template-body file://rds-custom-oracle-prereqs.yaml \
  --capabilities CAPABILITY_NAMED_IAM
```

2단계: 사용자 지정 엔진 버전(CEV) 및 Amazon RDS 인스턴스 생성

템플릿 복사 및 저장

1. Deploying RDS Custom for Oracle with single and multiple Availability Zones으로 이동합니다.

2. 원하는 형식(YAML 또는 JSON)으로 템플릿을 복사합니다.
3. 필요한 경우 템플릿에서 다음 파라미터를 업데이트합니다.
   - `BucketName`
   - `CEVS3Prefix`
   - 데이터베이스 마스터 암호(********** 대체)
4. 파일을 YAML 또는 JSON 형식으로 저장합니다.

3단계: AWS Console을 사용하여 배포

1. AWS Console을 열고 AWS CloudFormation으로 이동합니다.
2. 스택 생성 > 새 리소스 사용(표준)을 선택합니다.
3. 기존 템플릿 선택을 선택합니다.
4. 템플릿 파일 업로드 > 파일 선택을 선택합니다.
5. 이전에 다운로드한 템플릿 파일을 선택합니다.
6. 파라미터를 기본값으로 둡니다.
7. 다음과 같이 파라미터를 입력합니다:

```
BucketName: rds-custom-id
CEVS3Prefix: oracle_cev
CEVCreation: Yes
```

8. 구성을 검토하고 다음을 선택하여 스택을 생성합니다.

선택 사항: AWS CLI를 사용하여 배포

```
aws cloudformation create-stack \
  --stack-name rds-custom-oracle \
  --template-body file://rds-custom-oracle.yaml \
  --parameters \
    ParameterKey=BucketName,ParameterValue=rds-custom-id \
    ParameterKey=CEVS3Prefix,ParameterValue=oracle_cev \
    ParameterKey=CEVCreation,ParameterValue=Yes \
  --capabilities CAPABILITY_NAMED_IAM
```

배포 리소스 생성됨

템플릿은 다음 리소스를 만듭니다.

- 퍼블릭 및 프라이빗 서브넷이 있는 Amazon VPC
- 보안 그룹
- Amazon VPC 엔드포인트
- IAM 역할 및 정책
- 암호화를 위한 AWS KMS 키
- 사용자 지정 엔진 버전(CEV)
- 단일 AZ 및 다중 AZ 구성 모두에 대한 RDS Custom for Oracle 인스턴스

배포 진행 상황 모니터링

AWS CloudFormation 스택을 생성한 후 진행 상황을 모니터링하여 배포가 성공했는지 확인합니다. 배포 프로세스에는 사용자 지정 엔진 버전(CEV) 및 RDS 인스턴스 생성이 포함됩니다.

배포 진행 상황을 모니터링하는 방법:

1. AWS CloudFormation 콘솔을 엽니다.
2. 스택 이름을 선택합니다.
3. 이벤트 탭을 선택하여 진행 상황을 보고 오류를 식별합니다.

> **Note**
>
> CEV 생성에는 일반적으로 2~3시간이 소요됩니다. CEV 생성이 성공적으로 완료되면 Amazon RDS는 Amazon RDS 인스턴스 생성을 자동으로 시작합니다.

배포 후

스택 생성 프로세스가 완료되면 다음 배포 후 확인 및 구성 단계를 수행합니다.

1. Amazon RDS 콘솔 페이지에서 사용자 지정 엔진 버전으로 이동하여 CEV 생성을 확인합니다.
2. Amazon RDS 인스턴스가 생성되고 사용 가능한지 확인합니다.
3. Amazon RDS 인스턴스에 대한 연결을 테스트합니다.
4. 필요에 따라 모니터링 및 백업 전략을 설정합니다.

정리

모든 리소스를 제거하려면 다음 AWS CLI 명령을 실행합니다.

```
aws cloudformation delete-stack --stack-name rds-custom-oracle
```

문제 해결

배포 중에 문제가 발생하면 다음 솔루션을 사용하여 일반적인 문제를 해결하세요.

CEV 생성 실패

- 필요한 모든 패치가 Amazon S3에 업로드되었는지 확인합니다.
- IAM 권한을 확인합니다.
- 패치 버전이 올바른지 확인합니다. 필요한 패치 목록은 the section called “사전 조건” 섹션을 참조하세요.

Amazon RDS 인스턴스 생성 실패

- VPC/서브넷 구성을 확인합니다.
- 보안 그룹 규칙을 확인합니다.
- CEV를 사용할 수 있는지 확인합니다.

RDS Custom for Oracle의 Oracle 복제본으로 작업

Oracle Enterprise Edition이 실행되는 RDS Custom for Oracle DB 인스턴스에서 Oracle 복제본을 생성할 수 있습니다. 컨테이너 데이터베이스(CDB)와 비 CDB가 모두 지원됩니다. Standard Edition 2는 Oracle Data Guard를 지원하지 않습니다.

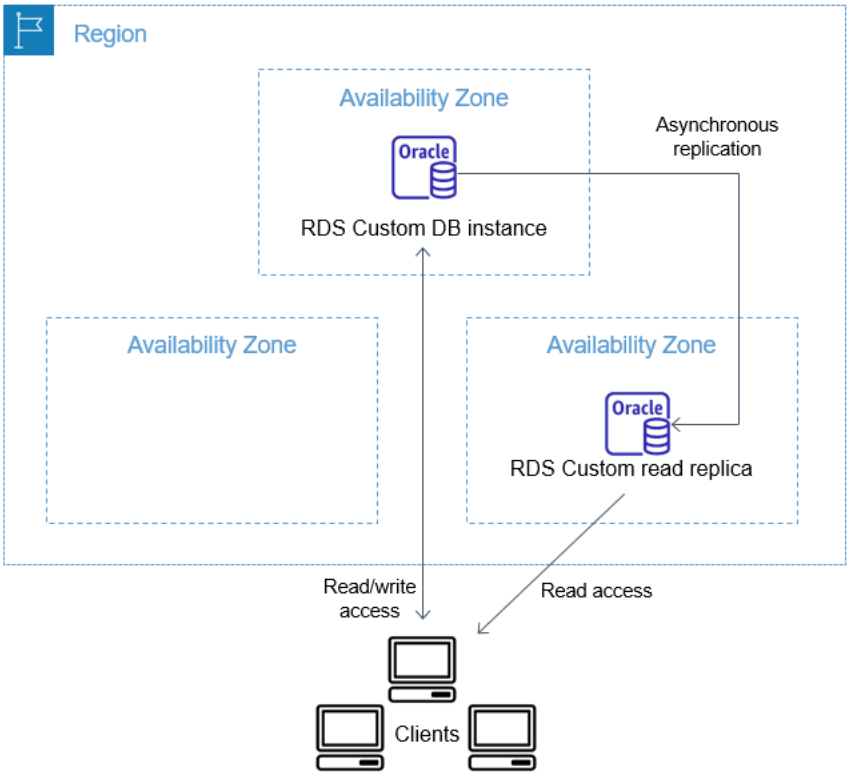
RDS Custom for Oracle 복제본을 생성하는 방법은 Oracle 복제본을 생성하는 방법과 비슷하지만, 몇 가지 중요한 차이점이 있습니다. Oracle 복제본 생성 및 관리에 대한 일반적인 정보는 DB 인스턴스 읽기 전용 복제본 작업 및 Amazon RDS의 Oracle의 읽기 전용 복제본 작업 섹션을 참조하세요.

주제

- RDS Custom for Oracle 복제 개요
- RDS Custom for Oracle 복제본의 지침 및 제한 사항
- RDS Custom for Oracle 복제본을 독립 실행형 DB 인스턴스로 승격
- RDS Custom for Oracle 기본 인스턴스와 복제본 인스턴스 간에 VPN 터널 구성

RDS Custom for Oracle 복제 개요

Oracle 복제를 위한 RDS 커스텀 아키텍처는 오라클용 RDS 복제와 유사합니다. 기본 DB 인스턴스는 하나 이상의 Oracle 복제본에 비동기적으로 복제합니다.



최대 복제본 수

Oracle용 RDS와 마찬가지로 RDS Custom for Oracle 기본 DB 인스턴스의 관리형 Oracle 복제본을 최대 5개까지 생성할 수 있습니다. 사용자 고유의 수동 구성 (외부) Oracle 복제본을 생성할 수도 있습니다. 외부 복제본은 DB 인스턴스 제한에 포함되지 않습니다. 또한 RDS Custom 지원 경계 밖에 배치됩니다. 지원 범위에 대한 자세한 내용은 RDS Custom 지원 범위 섹션을 참조하세요.

복제본 명명 규칙

Oracle 복제본 이름은 데이터베이스 고유 이름을 기반으로 합니다. 이러한 *DB_UNIQUE_NAME_X* 형식으로, 문자가 순차적으로 추가됩니다. 예를 들어, 데이터베이스 고유 이름이 다음과 같은 경우`ORCL`, 처음 두 개의 복제본에는 이름이 `ORCL_A`과`ORCL_B` 이렇게 들어갑니다. 처음 6개의 문자 A~F는 RDS Custom 전용입니다. 데이터베이스 파라미터는 프라이머리 DB 인스턴스에서 복제본으로 복사됩니다. 자세한 내용은 Oracle 문서의 DB_UNIQUE_NAME을 참조하세요.

복제본 백업 보존

기본적으로 RDS Custom Oracle 복제본은 기본 DB 인스턴스와 동일한 백업 보존 기간을 사용합니다. 백업 보존 기간을 1~35일로 수정할 수 있습니다. RDS Custom은 백업, 복원 및 시점 복구(PITR)를 지원합니다. RDS Custom DB 인스턴스 백업 및 복원에 대한 자세한 내용은 Amazon RDS Custom for Oracle DB 인스턴스 백업 및 복원 단원을 참조하세요.

> **Note**
>
> RDS Custom에서 Oracle 복제본을 생성하는 동안 RDS Custom 자동화는 다시 실행 로그 정리 작업을 일시적으로 중지합니다. Oracle 복제본을 생성하는 동안 RDS Custom은 다시 실행 로그 파일 정리를 일시적으로 일시 중지합니다.

복제본 승격

콘솔, `promote-read-replica` AWS CLI 명령 또는 `PromoteReadReplica` API를 사용하여 RDS Custom for Oracle의 관리형 Oracle 복제본을 승격할 수 있습니다. 기본 DB 인스턴스를 삭제하고 모든 복제본이 정상이면 RDS Custom for Oracle은 관리형 복제본을 독립 실행형 인스턴스로 자동 승격합니다. 복제본이 자동화를 일시 중지했거나 지원 경계 밖에 있는 경우 RDS Custom이 자동으로 승격할 수 있으려면 먼저 복제본을 수정해야 합니다. 외부 Oracle 복제본은 수동으로만 프로모션할 수 있습니다.

RDS Custom for Oracle 복제본의 지침 및 제한 사항

RDS Custom for Oracle 복제본을 생성할 때 모든 RDS Oracle 복제본 옵션이 지원되는 것은 아닙니다.

주제

- RDS Custom for Oracle 복제본의 일반적 지침
- RDS Custom for Oracle 복제본의 일반적 제한 사항
- RDS Custom for Oracle 복제본의 네트워킹 요구 사항 및 제한 사항
- RDS Custom for Oracle의 외부 복제본 제한 사항

RDS Custom for Oracle 복제본의 일반적 지침

RDS Custom for Oracle로 작업을 수행할 경우 다음 지침을 따르세요.

- RDS Custom for Oracle 복제본을 Oracle Enterprise Edition에서만 사용할 수 있습니다. Standard Edition 2는 지원되지 않습니다.
- VPN 터널을 구현하여 기본 인스턴스와 대기 인스턴스 간의 통신을 암호화하는 것이 좋습니다. 자세한 내용은 RDS Custom for Oracle 기본 인스턴스와 복제본 인스턴스 간에 VPN 터널 구성 섹션을 참조하세요.
- `RDS_DATAGUARD` 사용자를 수정하지 마세요. RDS Custom for Oracle 자동화를 위한 사용자입니다. 이 사용자를 수정하면 RDS Custom for Oracle DB 인스턴스에 대한 Oracle 복제본을 생성할 수 없는 것과 같은 원하지 않는 결과가 발생할 수 있습니다.
- 복제본 사용자 암호를 변경하지 마세요. 이 암호는 RDS Custom 호스트에서 Oracle Data Guard 구성을 관리하는 데 필요합니다. 암호를 변경하면 RDS Custom for Oracle이 Oracle 복제본을 지원 경계 외부에 둘 수 있습니다. 자세한 내용은 RDS Custom 지원 범위 섹션을 참조하세요.

  암호는 DB 리소스 ID로 태그가 지정된 AWS Secrets Manager에 저장됩니다. 각 읽기 전용 복제본은 Secrets Manager에 고유한 암호가 있습니다. 보안 암호는 다음 이름 지정 형식 중 하나를 사용합니다.

  ```
  do-not-delete-rds-custom-db-DB_resource_id-uuid-dg
  rds-custom!oracle-do-not-delete-DB_resource_id-uuid-dg
  ```

- 기본 DB 인스턴스의 `DB_UNIQUE_NAME`을 변경하지 마세요. 이름을 변경하면 모든 복원 작업이 중단됩니다.
- RDS Custom CDB의 `CREATE PLUGGABLE DATABASE` 명령에 `STANDBYS=NONE` 절을 지정하지 마세요. 이렇게 하면 장애 조치가 발생할 경우 대기 CDB에 모든 PDB가 포함됩니다.

RDS Custom for Oracle 복제본의 일반적 제한 사항

RDS Custom for Oracle 복제본에는 다음과 같은 제한 사항이 있습니다.

- 읽기 전용 모드에서는 RDS Custom for Oracle 복제본을 생성할 수 없습니다. 그러나 마운트된 복제본의 모드를 읽기 전용으로, 읽기 전용에서 마운트된 모드로 수동으로 변경할 수 있습니다. 자세한 내용은 create-db-instance-read-replica AWS CLI 명령에 대한 문서를 참조하세요.
- Oracle 복제본의 경우 리전 간 RDS Custom for Oracle 복제본을 생성할 수 없습니다.
- Oracle Data Guard `CommunicationTimeout` 파라미터의 값은 변경할 수 없습니다. RDS Custom for Oracle DB 인스턴스의 경우 이 파라미터는 15초로 설정됩니다.

RDS Custom for Oracle 복제본의 네트워킹 요구 사항 및 제한 사항

네트워크 구성이 RDS Custom for Oracle 복제본을 지원하는지 확인해야 합니다. 다음을 고려하세요.

- 기본 DB 인스턴스와 모든 복제본의 Virtual Private Cloud(VPC) 내 인바운드 및 아웃바운드 통신 모두에 대해 포트 1140을 활성화해야 합니다. 이는 읽기 전용 복제본 간의 Oracle Data Guard 통신에 필요합니다.
- RDS Custom for Oracle은 Oracle 복제본을 생성하는 동안 네트워크의 유효성을 검사합니다. 기본 DB 인스턴스와 새 복제본이 네트워크를 통해 연결할 수 없는 경우 RDS Custom for Oracle은 복제본을 생성하지 않고 `INCOMPATIBLE_NETWORK` 상태에 배치합니다.
- Amazon EC2 또는 온프레미스에서 생성하는 복제본과 같은 외부 Oracle 복제본의 경우 Oracle Data Guard 복제에 대해 다른 포트와 리스너를 사용합니다. 포트 1140을 사용하려고 하면 RDS Custom 자동화와 충돌이 발생할 수 있습니다.
- `/rdsdbdata/config/tnsnames.ora` 파일에는 리스너 프로토콜 주소에 매핑된 네트워크 서비스 이름이 포함되어 있습니다. 다음 요구 사항 및 권장 사항을 참고하세요.
  - Oracle 복제본 작업을 처리할 때 `tnsnames.ora` 접두사가 붙은 `rds_custom_`의 항목은 RDS Custom 전용입니다.

    `tnsnames.ora`에서 수동으로 항목을 생성할 때는 이 접두사를 사용하지 마세요.
  - 경우에 따라 수동으로 전환하거나 장애 조치하거나, 또는 고속 장애 조치(FSFO)와 같은 장애 조치 기법을 사용할 수도 있습니다. 그럴 경우 프라이머리 DB 인스턴스의 `tnsnames.ora` 항목을 모든 대기 인스턴스로 수동으로 동기화해야 합니다. 이 권장 사항은 RDS Custom에서 관리하는 Oracle 복제본과 외부 Oracle 복제본에 모두 적용됩니다.

    RDS Custom 자동화는 프라이머리 DB 인스턴스에서만 `tnsnames.ora` 항목을 업데이트합니다. Oracle 복제본을 추가하거나 제거할 때도 동기화해야 합니다.

`tnsnames.ora` 파일을 동기화하고 수동으로 전환하거나 장애 조치하지 않으면 프라이머리 DB 인스턴스의 Oracle Data Guard가 Oracle 복제본과 통신하지 못할 수 있습니다.

RDS Custom for Oracle의 외부 복제본 제한 사항

온프레미스 복제본을 포함하는 RDS Custom for Oracle 외부 복제본에는 다음과 같은 제한 사항이 있습니다.

- RDS Custom for Oracle은 FSFO와 같은 수동 장애 조치 시 외부 Oracle 복제본에 대한 인스턴스 역할 변경을 감지하지 않습니다.

  RDS Custom for Oracle은 관리형 복제본의 변경 사항을 감지합니다. 역할 변경은 이벤트 로그에 기록됩니다. describe-db-instances AWS CLI 명령을 사용하여 새로운 상태를 확인할 수도 있습니다.
- RDS Custom for Oracle은 외부 Oracle 복제본에 대한 복제본 지연을 감지하지 않습니다.

  RDS Custom for Oracle은 관리형 복제본의 지연을 감지합니다. 복제 지연률이 높으면 `Replication has stopped` 이벤트가 발생합니다. describe-db-instances AWS CLI 명령을 사용하여 복제 상태를 볼 수도 있지만, 업데이트되는 데 지연이 있을 수 있습니다.
- RDS Custom for Oracle은 기본 DB 인스턴스를 삭제한 경우 외부 Oracle 복제본을 자동으로 승격시키지 않습니다.

  자동 프로모션 기능은 관리형 Oracle 복제본에만 사용할 수 있습니다. Oracle 복제본을 수동으로 승격하는 방법에 대한 내용은 Amazon RDS Custom for Oracle의 Data Guard를 사용하여 고가용성 활성화 백서를 참조하세요

RDS Custom for Oracle 복제본을 독립 실행형 DB 인스턴스로 승격

RDS for Oracle,와 마찬가지로 RDS Custom for Oracle 복제본을 독립형 DB 인스턴스로 승격할 수 있습니다. Oracle 복제본을 승격하면 RDS Custom for Oracle의 DB 인스턴스가 먼저 재부팅된 후에 사용할 수 있습니다. Oracle 복제본 승격에 대한 자세한 내용은 읽기 전용 복제본을 독립 DB 인스턴스로 승격 섹션을 참조하세요.

복제본을 승격할 경우 다음 지침을 참고하세요.

- RDS Custom for Oracle이 복제본을 승격하는 동안 장애 조치를 시작하지 마세요. 그렇지 않으면 프로모션 워크플로가 중단될 수 있습니다.

- RDS Custom for Oracle이 Oracle 복제본을 승격하는 동안에는 기본 DB 인스턴스를 전환하지 마세요. 그렇지 않으면 프로모션 워크플로가 중단될 수 있습니다.
- RDS Custom for Oracle이 Oracle 복제본을 승격하는 동안에는 기본 DB 인스턴스를 종료하지 마세요. 그렇지 않으면 프로모션 워크플로가 중단될 수 있습니다.
- 새로 승격된 DB 인스턴스를 대상으로 삼아 복제를 다시 시작하지 마세요. RDS Custom for Oracle이 Oracle 복제본을 승격시킨 후에는 독립형 DB 인스턴스가 되며 더 이상 복제본 역할이 없습니다.

RDS Custom for Oracle 복제본 승격의 다음과 같은 제한 사항을 참고하세요.

- RDS Custom for Oracle이 백업하는 동안에는 복제본을 프로모션할 수 없습니다.
- Oracle 복제본을 승격할 때 백업 보존 기간을 `0`으로 변경할 수 없습니다.
- 정상 상태가 아닌 복제본은 승격시킬 수 없습니다.

  기본 DB 인스턴스에서 `delete-db-instance`를 발행하는 경우 RDS Custom for Oracle은 각 관리형 Oracle 복제본이 정상이고 승격에 사용할 수 있는지 확인합니다. 자동화가 일시 중지되었거나 지원 경계 밖에 있기 때문에 복제본은 승격에 적합하지 않을 수 있습니다. 이러한 경우 RDS Custom for Oracle은 Oracle 복제본을 수동으로 복구할 수 있도록 문제를 설명하는 이벤트를 게시합니다.

다음 단계는 Oracle 복제본을 DB 인스턴스로 승격하기 위한 일반적인 프로세스입니다.

1. 기본 DB 인스턴스에 대한 트랜잭션 쓰기를 중단합니다.
2. RDS Custom for Oracle이 모든 업데이트를 Oracle 복제본에 적용할 때까지 기다리십시오.
3. Amazon RDS 콘솔의 승격 옵션, AWS CLI 명령 `promote-read-replica` 또는 `PromoteReadReplica` Amazon RDS API 작업을 사용하여 Oracle 복제본을 승격합니다.

Oracle 복제본의 승격 작업은 완료할 때까지 몇 분 걸립니다. 프로세스 중에 RDS Custom for Oracle은 복제를 중지하고 복제본을 재부팅합니다. 재부팅이 완료되면 Oracle 복제본을 새 DB 인스턴스로 사용할 수 있습니다. 복제본 승격의 문제 해결에 대한 자세한 내용은 RDS Custom for Oracle 복제본 승격 문제 해결 섹션을 참조하세요.

콘솔

RDS Custom for Oracle 복제본을 독립 실행형 DB 인스턴스로 승격하려면

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.

2. Amazon RDS 콘솔에서 데이터베이스를 선택합니다.

   데이터베이스 창이 표시됩니다. 각 Oracle 복제본은 역할 열에 복제본이라고 표시됩니다.
3. 승격하려는 RDS Custom for Oracle 복제본을 선택합니다.
4. Actions(작업)에서 Promote(승격)를 선택합니다.
5. 읽기 전용 복제본 승격 페이지에서 새롭게 승격된 DB 인스턴스의 백업 보존 기간과 백업 기간을 입력합니다. 이 값은 0으로 설정할 수 없습니다.
6. 원하는 대로 설정되었으면 Oracle 복제본 승격을 선택합니다.

AWS CLI

RDS Custom for Oracle 복제본을 독립 실행형 DB 인스턴스로 승격하려면 AWS CLI `promote-read-replica` 명령을 사용합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds promote-read-replica \
--db-instance-identifier my-custom-read-replica \
--backup-retention-period 2 \
--preferred-backup-window 23:00-24:00
```

Windows의 경우:

```
aws rds promote-read-replica ^
--db-instance-identifier my-custom-read-replica ^
--backup-retention-period 2 ^
--preferred-backup-window 23:00-24:00
```

RDS API

RDS Custom for Oracle 복제본을 독립형 DB 인스턴스로 승격하려면 필수 파라미터 `DBInstanceIdentifier`를 사용하여 Amazon RDS API `PromoteReadReplica` 작업을 호출합니다.

RDS Custom for Oracle 기본 인스턴스와 복제본 인스턴스 간에 VPN 터널 구성

VPN 터널은 네트워크를 통한 디바이스 두 개 이상 간의 암호화된 연결입니다. RDS Custom for Oracle에서 Oracle Data Guard 인스턴스에 대한 최고 수준의 보안을 보장하려면 VPN 터널을 구현하여 기본

인스턴스와 대기 인스턴스 간의 통신을 암호화하는 것이 좋습니다. 터널은 민감한 데이터가 인스턴스 간에 네트워크를 이동할 때 민감한 데이터를 보호하는 역할을 합니다. 이 구성은 선택 사항이지만 데이터 보안 및 규정 준수를 달성하기 위한 모범 사례로 사용하는 것이 좋습니다.

다음 사전 조건을 충족하는지 확인합니다.

- 기본 및 대기 호스트에 대한 루트 액세스 권한이 있어야 합니다.
- `ipsec` 명령을 실행할 수 있는 기술적 전문 지식이 있어야 합니다.

RDS Custom for Oracle에서 기본 복제본과 복제본 간에 VPN 터널을 구성하는 방법

1. 다음 규칙을 사용하여 기본 인스턴스와 대기 인스턴스 모두에 대한 보안 그룹을 허용 목록에 추가합니다.

```
ACTION FLOW SOURCE PROTO PORT

ALLOW ingress this-SG 50 (ESP) all (N/A)
ALLOW egress this-SG 50 (ESP) all (N/A)

ALLOW ingress this-SG 17 (UDP) 500 (IKE)
ALLOW egress this-SG 17 (UDP) 500 (IKE)
```

2. 루트 사용자로 전환합니다.

```
$ sudo su - root
```

3. 기본 인스턴스와 대기 인스턴스 모두에서 다음 명령을 실행하여 사용자 `root`를 사용하여 네트워크 보안 서비스(NSS) 데이터베이스를 초기화합니다.

```
ipsec initnss --nssdir /etc/ipsec.d
```

4. 다음과 같이 RSA 키를 생성합니다.

   a. 기본 인스턴스에서 OS 버전에 따라 다음 `ipsec` 명령 중 하나를 사용하여 키를 생성합니다.

```
ipsec newhostkey --nssdir /etc/ipsec.d       ## for Oracle Linux Version 8
ipsec newhostkey --output /etc/ipsec.secrets ## for Oracle Linux version 7.9
```

   b. 구성을 만드는 데 필요한 퍼블릭 키를 가져옵니다. 다음 예시에서 기본 인스턴스는 `left`입니다. `ipsec` parlance에서 `left`가 현재 구성 중인 디바이스를 나타내고 `right`는 터널의 다른 쪽 끝에 있는 디바이스를 나타내기 때문입니다.

```
ipsec showhostkey --left --ckaid ckaid-returned-in-last-statement
```

c. 대기 인스턴스에서 대기 인스턴스에 대한 키를 생성합니다.

```
ipsec newhostkey --nssdir /etc/ipsec.d       ## for Oracle Linux Version 8
ipsec newhostkey --output /etc/ipsec.secrets ## for Oracle Linux version 7.9
```

d. 구성을 만드는 데 필요한 대기 인스턴스의 퍼블릭 키를 가져옵니다. 다음 예시에서 대기 인스턴스는 터널의 다른 쪽 끝에 있는 디바이스를 나타내는 `right`입니다.

```
ipsec showhostkey --right --ckaid ckaid-returned-in-last-statement
```

5. 가져온 RSA 키를 기반으로 구성을 생성합니다. 구성은 기본 인스턴스와 대기 인스턴스에서 동일합니다. AWS Console에서 기본 인스턴스 IPv4 주소와 대기 인스턴스 IPv4 주소를 찾을 수 있습니다.

   기본 인스턴스와 대기 인스턴스 모두에서 다음 구성을 `/etc/ipsec.d/custom-fb-tunnel.conf` 파일에 저장합니다.

```
conn custom-db-tunnel
 type=transport
 auto=add
 authby=rsasig
 left=IPV4-for-primary
 leftrsasigkey=RSA-key-generated-on-primary
 right=IPV4-for-standby
 rightrsasigkey=RSA-key-generated-on-standby
```

6. 기본 인스턴스와 대기 인스턴스 둘 다, 두 호스트 모두에서 `ipsec` 대몬을 시작합니다.

```
ipsec setup start
```

7. 기본 인스턴스 또는 대기 인스턴스에서 터널을 시작합니다. 출력은 다음과 유사합니다.

```
[root@ip-172-31-6-81 ~]# ipsec auto --up custom-db-tunnel
181 "custom-db-tunnel" #1: initiating IKEv2 connection
181 "custom-db-tunnel" #1: sent IKE_SA_INIT request to 172.31.32.196:500
182 "custom-db-tunnel" #1: sent IKE_AUTH request {cipher=AES_GCM_16_256 integ=n/a
 prf=HMAC_SHA2_512 group=DH19}
003 "custom-db-tunnel" #1: initiator established IKE SA; authenticated peer '3584-
bit PKCS#1 1.5 RSA with SHA1' signature using preloaded certificate '172.31.32.196'
```

```
004 "custom-db-tunnel" #2: initiator established Child SA using #1; IPsec transport
 [172.31.6.81-172.31.6.81:0-65535 0] -> [172.31.32.196-172.31.32.196:0-65535 0]
 {ESP/ESN=>0xda9c4815 <0xb742ca42 xfrm=AES_GCM_16_256-NONE DPD=passive}
[root@ip-172-31-6-81 ~]#
```

Amazon RDS Custom for Oracle DB 인스턴스 백업 및 복원

Amazon RDS와 마찬가지로 RDS Custom은 DB 인스턴스의 백업 기간 동안 RDS Custom for Oracle DB 인스턴스의 자동 백업을 생성하고 저장합니다. 또한, 수동으로 DB 인스턴스를 백업할 수도 있습니다.

절차는 Amazon RDS DB 인스턴스의 스냅샷을 생성하는 것과 동일합니다. RDS Custom DB 인스턴스의 첫 번째 스냅샷에는 전체 DB 인스턴스의 데이터가 포함되며, 후속 스냅샷은 증분적입니다.

AWS Management Console 또는 AWS CLI를 사용하여 DB 스냅샷을 복원할 수 있습니다.

주제

- RDS Custom for Oracle 스냅샷 생성
- RDS Custom for Oracle DB 스냅샷에서 복원
- RDS Custom for Oracle 인스턴스를 특정 시점으로 복원
- RDS Custom for Oracle 스냅샷 삭제
- RDS Custom for Oracle 자동 백업 삭제

RDS Custom for Oracle 스냅샷 생성

RDS Custom for Oracle은 DB 인스턴스의 스토리지 볼륨 스냅샷을 생성하여 개별 데이터베이스뿐만 아니라 전체 DB 인스턴스를 백업합니다. DB 인스턴스에 컨테이너 데이터베이스(CDB) 가 포함된 경우, 인스턴스의 스냅샷에는 루트 CDB와 모든 PDB가 포함됩니다.

RDS Custom for Oracle 스냅샷을 생성할 때 백업할 RDS Custom DB 인스턴스를 지정하고, 나중에 복원할 수 있도록 스냅샷에 이름을 지정합니다.

스냅샷을 생성할 때 RDS Custom for Oracle은 DB 인스턴스에 연결된 모든 볼륨에 대해 Amazon EBS 스냅샷을 생성합니다. RDS Custom for Oracle은 루트 볼륨의 EBS 스냅샷을 사용하여 새로운 Amazon Machine Image(AMI)를 등록합니다. 스냅샷을 특정 DB 인스턴스에 쉽게 연결할 수 있도록 `DBSnapshotIdentifier`, `DbiResourceId` 및 `VolumeType`으로 태그가 지정됩니다.

DB 스냅샷을 생성하면 I/O가 잠시 중단됩니다. 이러한 일시 중단은 DB 인스턴스의 크기와 클래스에 따라 몇 초에서 몇 분까지 지속될 수 있습니다. 스냅샷 생성에 걸리는 시간은 데이터베이스 크기에 따라 다릅니다. 스냅샷에는 전체 스토리지 볼륨이 포함되므로 임시 파일과 같은 파일의 크기도 스냅샷 생성 시간에 영향을 미칩니다. 스냅샷을 생성하는 방법에 대한 자세한 내용은 Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성 섹션을 참조하세요.

콘솔 또는 AWS CLI를 사용하여 RDS Custom for Oracle 스냅샷을 생성할 수 있습니다.

콘솔

RDS Custom 스냅샷을 생성하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
3. RDS Custom DB 인스턴스 목록에서 스냅샷을 생성할 인스턴스를 선택합니다.
4. 작업에서 스냅샷 만들기를 선택합니다.

   [DB 스냅샷 생성(Take DB Snapshot)] 창이 나타납니다.
5. 스냅샷 이름(Snapshot name)에 스냅샷 이름을 입력합니다.
6. [스냅샷 생성(Take Snapshot)]을 선택합니다.

AWS CLI

create-db-snapshot AWS CLI 명령을 사용하여 RDS Custom DB 인스턴스의 스냅샷을 생성합니다.

다음과 같은 옵션을 지정할 수 있습니다.

- `--db-instance-identifier` - 백업할 RDS Custom DB 인스턴스를 식별합니다.
- `--db-snapshot-identifier` - 나중에 복원할 수 있도록 RDS Custom 스냅샷의 이름을 지정합니다.

이 예제에서는 *my-custom-instance*라는 RDS Custom DB 인스턴스에 대해 *my-custom-snapshot*이라는 DB 스냅샷을 생성합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-snapshot \
    --db-instance-identifier my-custom-instance \
    --db-snapshot-identifier my-custom-snapshot
```

Windows의 경우:

```
aws rds create-db-snapshot ^
    --db-instance-identifier my-custom-instance ^
    --db-snapshot-identifier my-custom-snapshot
```

RDS Custom for Oracle DB 스냅샷에서 복원

RDS Custom for Oracle DB 인스턴스를 복원하는 경우 DB 스냅샷의 이름과 새로운 인스턴스의 이름을 입력합니다. 스냅샷에서 기존 RDS Custom DB 인스턴스로 복원할 수 없습니다. 복원 시 새로운 RDS Custom for Oracle DB 인스턴스가 생성됩니다.

복원 프로세스는 다음과 같은 면에서 Amazon RDS의 복원과 다릅니다.

- 스냅샷을 복원하기 전에 RDS Custom for Oracle은 기존 구성 파일을 백업합니다. 이러한 파일은 디렉터리 `/rdsdbdata/config/backup`의 복원된 인스턴스에서 사용할 수 있습니다. RDS Custom for Oracle은 기본 파라미터로 DB 스냅샷을 복원하고 이전 데이터베이스 구성 파일을 기존 데이터베이스 구성 파일로 덮어씁니다. 따라서 복원된 인스턴스는 사용자 지정 파라미터 및 데이터베이스 구성 파일의 변경 사항을 보존하지 않습니다.
- 복원된 데이터베이스의 이름은 스냅샷과 동일하며, 다른 이름을 지정할 수 없습니다. (RDS Custom for Oracle의 경우 기본값은 `ORCL`입니다.)

콘솔

DB 스냅샷에서 RDS Custom DB 인스턴스를 복원하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.
3. 복원 원본으로 사용할 DB 스냅샷을 선택합니다.
4. 작업에서 스냅샷 복원을 선택합니다.
5. DB 인스턴스 복원(Restore DB instance) 페이지의 DB 인스턴스 식별자(DB instance identifier)에 복원된 RDS Custom DB 인스턴스의 이름을 입력합니다.
6. DB 인스턴스 복원을 선택합니다.

AWS CLI

restore-db-instance-from-db-snapshot AWS CLI 명령을 사용하여 RDS Custom DB 스냅샷을 복원합니다.

복원하려는 스냅샷이 프라이빗 DB 인스턴스용인 경우 db-subnet-group-name 및 no-publicly-accessible을 모두 올바르게 지정해야 합니다. 그렇지 않으면 DB 인스턴스의 기본값에 공개적으로 액세스할 수 있게 됩니다. 다음 옵션이 필요합니다.

- db-snapshot-identifier - 복구할 스냅샷을 식별합니다.
- db-instance-identifier - DB 스냅샷에서 생성할 RDS Custom DB 인스턴스의 이름을 지정합니다.
- custom-iam-instance-profile - RDS Custom for Oracle DB 인스턴스의 기본 Amazon EC2 인스턴스와 연결된 인스턴스 프로필을 지정합니다.

다음 코드는 my-custom-instance에 대한 my-custom-snapshot이라는 스냅샷을 복원합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds restore-db-instance-from-db-snapshot \
  --db-snapshot-identifier my-custom-snapshot \
  --db-instance-identifier my-custom-instance \
  --custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance \
  --no-publicly-accessible
```

Windows의 경우:

```
aws rds restore-db-instance-from-db-snapshot ^
  --db-snapshot-identifier my-custom-snapshot ^
  --db-instance-identifier my-custom-instance ^
  --custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance ^
  --no-publicly-accessible
```

RDS Custom for Oracle 인스턴스를 특정 시점으로 복원

DB 인스턴스를 특정 시점(PITR)으로 복원하여 새로운 DB 인스턴스를 생성할 수 있습니다. PITR을 지원하려면 DB 인스턴스의 백업 보존이 0이 아닌 값으로 설정되어 있어야 합니다.

RDS Custom for Oracle DB 인스턴스의 가장 빠른 복원 가능 시간은 여러 요소에 따라 달라지지만, 일반적으로 현재 시간에서 5분 이내입니다. DB 인스턴스의 최근 복원 가능 시간을 확인하려면 AWS CLIdescribe-db-instances 명령을 사용한 후 DB 인스턴스의 LatestRestorableTime 필드에 반환되

는 값을 살펴봅니다. Amazon RDS 콘솔에서 각 DB 인스턴스의 복원 가능한 최신 시간을 보려면 [자동 백업을 선택합니다.

백업 보존 기간 중 어느 특정 시점으로든 복원할 수 있습니다. 각 DB 인스턴스의 복원 가능한 가장 빠른 시간을 보려면 Amazon RDS 콘솔에서 자동 백업을 선택합니다.

PITR에 대한 일반적인 정보는 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 섹션을 참조하세요.

주제

- RDS Custom for Oracle에 대한 PITR 고려 사항

RDS Custom for Oracle에 대한 PITR 고려 사항

RDS Custom for Oracle에서 PITR은 다음과 같은 중요한 면에서 Amazon RDS의 PITR과 다릅니다.

- 복원된 데이터베이스의 이름은 소스 DB 인스턴스와 동일합니다. 다른 이름을 지정할 수 없습니다. 기본값은 `ORCL`입니다.
- `AWSRDSCustomIamRolePolicy`는 새 권한이 필요합니다. 자세한 내용은 2단계: AWSRDSCustomInstanceRoleForRdsCustomInstance에 액세스 정책 추가 단원을 참조하세요.
- Oracle DB 인스턴스용 모든 RDS Custom에는 백업 보존이 0이 아닌 값으로 설정되어 있어야 합니다.
- 운영 체제 또는 DB 인스턴스 시간대를 변경하면 PITR이 작동하지 않을 수 있습니다. 시간대 변경에 대한 내용은 Oracle 시간대 섹션을 참조하세요.
- 자동화를 `ALL_PAUSED`로 설정한 경우 RDS Custom은 복원 가능한 최근 시간(LRT) 이전에 생성된 로그를 포함하여 아카이브된 다시 실행 로그 파일의 업로드를 일시 중지합니다. 잠시 동안 자동화를 일시 중지하는 것이 좋습니다.

  예를 들어, LRT가 10분 전이라고 가정해 봅니다. 자동화를 일시 중지하면 일시 중지한 동안 RDS Custom은 아카이브된 다시 실행 로그를 업로드하지 않습니다. DB 인스턴스가 충돌하는 경우 일시 중지할 때 존재했던 LRT 이전 시간으로만 복구할 수 있습니다. 자동화를 재개하면 RDS Custom은 로그 업로드를 다시 시작합니다. LRT가 변경되며, 일반적인 PITR 규칙이 적용됩니다.
- RDS Custom에서는 업로드 후 RDS Custom에서 삭제하기 전에 아카이브된 다시 실행 로그를 보존할 시간을 수동으로 지정할 수 있습니다. 다음과 같이 시간을 지정합니다.

  1. `/opt/aws/rdscustomagent/config/redo_logs_custom_configuration.json`라는 이름의 텍스트 파일을 만듭니다.

2. {"archivedLogRetentionHours" : "*num_of_hours*"} 형식의 JSON 객체를 추가합니다. 번호는 1~840 범위의 정수여야 합니다.

- CDB가 아닌 항목을 컨테이너 데이터베이스(CDB)에 PDB로 연결한 다음 PITR을 시도한다고 가정하겠습니다. 이전에 PDB를 백업한 경우에만 작업이 성공합니다. PDB를 만들거나 수정한 후에는 항상 백업하는 것이 좋습니다.
- 데이터베이스 초기화 파라미터를 사용자 지정하지 않는 것이 좋습니다. 예를 들어, 다음 파라미터를 수정하면 PITR에 영향을 줍니다.
  - CONTROL_FILE_RECORD_KEEP_TIME은 로그 업로드 및 삭제 규칙에 영향을 줍니다.
  - LOG_ARCHIVE_DEST_*n*는 여러 대상을 지원하지 않습니다.
  - ARCHIVE_LAG_TARGET은 최신 복원 가능 시간에 영향을 줍니다. Recovery Point Objective(RPO)가 5분이기 때문에 ARCHIVE_LAG_TARGET은 300으로 설정됩니다. 이 목표를 달성하기 위해 RDS는 5분마다 온라인 다시 실행 로그를 전환하여 Amazon S3 버킷에 저장합니다. 로그 전환 빈도로 인해 RDS Custom for Oracle 데이터베이스의 성능 문제가 발생하는 경우, DB 인스턴스와 스토리지를 IOPS와 처리량이 더 높은 곳으로 확장할 수 있습니다. 복구 계획에 필요한 경우 ARCHIVE_LAG_TARGET 초기화 파라미터 설정을 60~7,200 사이의 값으로 조정할 수 있습니다.
- 데이터베이스 초기화 파라미터를 사용자 지정하는 경우 다음 항목만 사용자 지정하는 것이 좋습니다.
  - COMPATIBLE
  - MAX_STRING_SIZE
  - DB_FILES
  - UNDO_TABLESPACE
  - ENABLE_PLUGGABLE_DATABASE
  - CONTROL_FILES
  - AUDIT_TRAIL
  - AUDIT_TRAIL_DEST

  다른 모든 초기화 파라미터의 경우 RDS Custom은 기본값을 복원합니다. 이전 목록에 없는 파라미터를 수정하면 PITR에 악영향을 미치고 예측할 수 없는 결과가 발생할 수 있습니다. 예를 들어, CONTROL_FILE_RECORD_KEEP_TIME은 로그 업로드 및 삭제 규칙에 영향을 줍니다.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 RDS Custom DB 인스턴스를 특정 시점으로 복원할 수 있습니다.

콘솔

RDS Custom DB 인스턴스를 지정된 시간으로 복원하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 자동 백업(Automated backups)을 선택합니다.
3. 복원하려는 RDS Custom DB 인스턴스를 선택합니다.
4. 작업에서 특정 시점으로 복구를 선택합니다.

   특정 시점으로 복구 창이 나타납니다.
5. 최근 복원 가능 시간을 선택하여 가능한 최근 시간으로 복원하거나, 사용자 지정을 선택하여 시간을 선택합니다.

   사용자 지정(Custom)을 선택한 경우 인스턴스를 복원하려는 날짜와 시간을 입력합니다.

   시간은 현지 시간대로 표시됩니다. 즉, 협정 세계시(UTC)에서 오프셋으로 표시됩니다. 예를 들어 UTC-5는 동부 표준시/하절기 중부 표준시입니다.
6. DB 인스턴스 식별자(DB instance identifier)에는 복원된 대상 RDS Custom DB 인스턴스의 이름을 입력합니다. 이름은 고유해야 합니다.
7. 필요에 따라 DB 인스턴스 클래스와 같은 기타 옵션을 선택합니다.
8. 특정 시점으로 복구를 선택합니다.

AWS CLI

새로운 RDS Custom DB 인스턴스를 생성하려면 restore-db-instance-to-point-in-time AWS CLI 명령을 사용하여 DB 인스턴스를 지정된 시간으로 복원합니다.

다음 옵션 중 하나를 사용하여 복원 원본으로 사용할 백업을 지정합니다.

- `--source-db-instance-identifier` *`mysourcedbinstance`*
- `--source-dbi-resource-id` *`dbinstanceresourceID`*
- `--source-db-instance-automated-backups-arn` *`backupARN`*

`custom-iam-instance-profile` 옵션은 필수입니다.

다음 예제는 지정된 시간을 기준으로 `my-custom-db-instance`를 `my-restored-custom-db-instance`라는 새로운 DB 인스턴스로 복원합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds restore-db-instance-to-point-in-time \
    --source-db-instance-identifier my-custom-db-instance\
    --target-db-instance-identifier my-restored-custom-db-instance \
    --custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance \
    --restore-time 2022-10-14T23:45:00.000Z
```

Windows의 경우:

```
aws rds restore-db-instance-to-point-in-time ^
    --source-db-instance-identifier my-custom-db-instance ^
    --target-db-instance-identifier my-restored-custom-db-instance ^
    --custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance ^
    --restore-time 2022-10-14T23:45:00.000Z
```

RDS Custom for Oracle 스냅샷 삭제

RDS Custom for Oracle에서 관리하는 DB 스냅샷이 더 이상 필요하지 않으면 삭제할 수 있습니다. Amazon RDS와 RDS Custom DB 인스턴스의 삭제 절차는 동일합니다.

바이너리 및 루트 볼륨에 대한 Amazon EBS 스냅샷은 계정에서 실행 중인 일부 인스턴스 또는 다른 RDS Custom for Oracle 스냅샷에 연결될 수 있어 계정에 더 오래 남아 있습니다. 이러한 EBS 스냅샷은 기존 RDS Custom for Oracle 리소스(DB 인스턴스 또는 백업)와 더 이상 관련이 없는 경우 자동으로 삭제됩니다.

콘솔

RDS Custom DB 인스턴스의 스냅샷을 삭제하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.
3. 삭제하고 싶은 DB 스냅샷을 선택합니다.
4. 작업(Actions)에서 스냅샷 삭제(Delete snapshot)를 선택합니다.

5. 확인 페이지에서 삭제를 선택합니다.

AWS CLI

RDS Custom 스냅샷을 삭제하려면 AWS CLI 명령 delete-db-snapshot을 사용합니다.

다음 옵션이 필요합니다.

- `--db-snapshot-identifier` - 삭제할 스냅샷입니다.

다음 예제에서는 `my-custom-snapshot` DB 스냅샷을 삭제합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds delete-db-snapshot \
  --db-snapshot-identifier my-custom-snapshot
```

Windows의 경우:

```
aws rds delete-db-snapshot ^
  --db-snapshot-identifier my-custom-snapshot
```

RDS Custom for Oracle 자동 백업 삭제

보관된 RDS Custom for Oracle 자동 백업이 더 이상 필요하지 않으면 삭제할 수 있습니다. 절차는 Amazon RDS 백업을 삭제하는 절차와 동일합니다.

콘솔

보관된 자동 백업을 삭제하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 자동 백업(Automated backups)을 선택합니다.
3. Retained(보관됨)를 선택합니다.
4. 삭제하려는 보관된 자동 백업을 선택합니다.
5. [Actions]에 대해 [Delete]를 선택합니다.

6. 확인 페이지에서 **`delete me`**를 입력하고 삭제를 선택합니다.

AWS CLI

delete-db-instance-automated-backup이라는 AWS CLI 명령을 사용하여 보관된 자동 백업을 삭제할 수 있습니다.

다음 옵션을 사용하여 보관된 자동 백업을 삭제할 수 있습니다.

- `--dbi-resource-id` - 소스 RDS Custom DB 인스턴스의 리소스 식별자입니다.

  보관된 자동 백업의 소스 DB 인스턴스에 대한 리소스 식별자는 AWS CLI 명령 describe-db-instance-automated-backups를 사용하여 찾을 수 있습니다.

다음 예시에서는 소스 DB 인스턴스 리소스 식별자가 `custom-db-123ABCEXAMPLE`인 보관된 자동 백업을 삭제합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds delete-db-instance-automated-backup \
    --dbi-resource-id custom-db-123ABCEXAMPLE
```

Windows의 경우:

```
aws rds delete-db-instance-automated-backup ^
    --dbi-resource-id custom-db-123ABCEXAMPLE
```

RDS Custom for Oracle에서 옵션 그룹을 사용한 작업

RDS Custom은 옵션 그룹을 사용하여 추가 기능을 활성화하고 구성합니다. 옵션 그룹은 RDS Custom for Oracle DB 인스턴스에 사용할 수 있는 옵션이라고 하는 기능을 지정합니다. 옵션은 여러 설정을 통해 옵션의 활용 방식을 지정합니다. RDS Custom for Oracle DB 인스턴스를 옵션 그룹과 연결하면 지정된 옵션 및 옵션 설정이 이 인스턴스에 대해 활성화됩니다. Amazon RDS의 옵션 그룹에 대한 일반적인 정보는 옵션 그룹 작업 섹션을 참조하세요.

주제

- RDS Custom for Oracle 옵션 그룹 개요
- Oracle 시간대

RDS Custom for Oracle 옵션 그룹 개요

Oracle 데이터베이스에 대한 옵션을 활성화하려면 먼저 옵션 그룹에 추가한 다음 옵션 그룹을 DB 인스턴스에 연결해야 합니다. 자세한 내용은 옵션 그룹 작업 단원을 참조하십시오.

주제

- RDS Custom for Oracle 옵션 요약
- RDS Custom for Oracle DB 인스턴스에 옵션을 추가하는 기본 단계
- RDS Custom for Oracle의 옵션 그룹 생성
- RDS Custom for Oracle DB 인스턴스와 옵션 그룹 연결

RDS Custom for Oracle 옵션 요약

RDS Custom for Oracle은 DB 인스턴스에 대해 다음 옵션을 지원합니다.

옵션	옵션 ID	설명
Oracle 시간대	`Timezone`	RDS Custom for Oracle DB 인스턴스에서 사용하는 시간대.

RDS Custom for Oracle DB 인스턴스에 옵션을 추가하는 기본 단계

RDS Custom for Oracle DB 인스턴스에 옵션을 추가하는 일반적인 절차는 다음과 같습니다.

1. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 복사 또는 수정합니다.
2. 옵션을 옵션 그룹에 추가합니다.
3. DB 인스턴스를 만들거나 수정할 때 옵션 그룹을 DB 인스턴스와 연결합니다.

RDS Custom for Oracle의 옵션 그룹 생성

기본 옵션 그룹에서 설정을 가져오는 새 옵션 그룹을 생성할 수 있습니다. 그런 다음, 하나 이상의 옵션을 새 옵션 그룹에 추가합니다. 아니면 이미 기존 옵션 그룹이 있는 경우에는 기존 옵션 그룹과 모든 옵션을 새 옵션 그룹에 복사할 수 있습니다. 옵션 그룹을 복사하는 방법을 알아보려면 옵션 그룹 생성 섹션을 참조하세요.

RDS Custom for Oracle의 기본 옵션 그룹은 다음 내용과 같습니다.

- `default:custom-oracle-ee`
- `default:custom-oracle-se2`
- `default:custom-oracle-ee-cdb`
- `default:custom-oracle-se2-cdb`

옵션 그룹을 생성할 때 설정은 기본 옵션 그룹에서 파생됩니다. `TIME_ZONE` 옵션을 추가한 후에는 옵션 그룹을 DB 인스턴스와 연결할 수 있습니다.

콘솔

옵션 그룹을 생성하는 한 가지 방법은 AWS Management Console을 사용하는 것입니다.

콘솔을 이용하여 새 옵션 그룹을 생성하려면,

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 그룹 생성을 선택합니다.
4. 보안 그룹 생성 창에서 다음과 같이 합니다.
   a. [이름(Name)]에 AWS 계정 내에서 쉽게 식별할 수 있는 옵션 그룹 이름을 입력합니다. 이름은 글자, 숫자 및 하이픈만 사용 가능합니다.
   b. Description에 옵션 그룹에 대한 간략한 설명을 입력합니다. 이 설명은 표시 용도로만 사용됩니다.

c. 엔진의 경우 다음 RDS Custom for Oracle DB 엔진 중 하나를 선택합니다.

- custom-oracle-ee
- custom-oracle-se2
- custom-oracle-ee-cdb
- custom-oracle-se2-cdb

d. 메이저 엔진 버전의 경우 RDS Custom for Oracle에서 지원하는 메이저 엔진 버전을 선택합니다. 자세한 내용은 RDS Custom for Oracle을 지원하는 리전 및 DB 엔진 단원을 참조하십시오.

5. 계속하려면 생성을 선택합니다. 작업을 취소하려면 Cancel을 선택합니다.

AWS CLI

옵션 그룹을 생성하려면 AWS CLI `create-option-group` 명령을 다음 필수 파라미터와 함께 사용합니다.

- `--option-group-name`
- `--engine-name`
- `--major-engine-version`
- `--option-group-description`

Example

다음은 `testoptiongroup`이라는 이름의 옵션 그룹을 생성하는 예제입니다. 이 옵션 그룹은 Oracle Enterprise Edition DB 엔진과 연동됩니다. 설명은 인용 부호로 묶여 있습니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds create-option-group \
    --option-group-name testoptiongroup \
    --engine-name custom-oracle-ee-cdb \
    --major-engine-version 19 \
    --option-group-description "Test option group for a Custom Oracle CDB"
```

Windows의 경우:

```
aws rds create-option-group ^
```

```
    --option-group-name testoptiongroup ^
    --engine-name custom-oracle-ee-cdb ^
    --major-engine-version 19 ^
    --option-group-description "Test option group for a Custom Oracle CDB"
```

RDS API

옵션 그룹을 생성하려면 Amazon RDS API CreateOptionGroup 작업을 호출합니다.

RDS Custom for Oracle DB 인스턴스와 옵션 그룹 연결

옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스와 연결할 수 있습니다.

- 새 DB 인스턴스의 경우 인스턴스를 생성할 때 옵션 그룹을 적용합니다. 자세한 내용은 RDS Custom for Oracle DB 인스턴스 생성 단원을 참조하십시오.
- 기존 DB 인스턴스의 경우, 해당 인스턴스를 수정하고 새 옵션 그룹을 연결하여 옵션 그룹을 적용하십시오. 자세한 내용은 RDS Custom for Oracle DB 인스턴스 수정 단원을 참조하십시오.

Oracle 시간대

RDS Custom for Oracle DB 인스턴스에서 사용하는 시스템 시간대를 변경하려면 시간대 옵션을 사용합니다. 예를 들면 온프레미스 환경 또는 기존 애플리케이션과 시간을 호환하기 위해 DB 인스턴스의 시간대를 변경할 수 있습니다. 시간대 옵션은 호스트 레벨에서 시간대를 변경합니다. 시간대를 변경하면 SYSDATE 및 SYSTIMESTAMP를 비롯한 모든 날짜 열과 값이 영향을 받습니다.

주제

- RDS Custom for Oracle의 시간대 옵션 설정
- RDS Custom for Oracle의 사용 가능한 시간대
- RDS Custom for Oracle의 시간대 설정에 대한 고려 사항
- RDS Custom for Oracle의 시간대 설정에 대한 제한 사항
- 옵션 그룹에 시간대 옵션 추가
- 시간대 옵션 제거

RDS Custom for Oracle의 시간대 옵션 설정

Amazon RDS는 시간대 옵션에 대해 다음 설정을 지원합니다.

옵션 설정	유효한 값	설명
TIME_ZONE	사용 가능한 시간대 중 하나입니다. 전체 목록은 RDS Custom for Oracle의 사용 가능한 시간대 단원을 참조하십시오.	DB 인스턴스에 대한 새 시간대를 선택합니다.

RDS Custom for Oracle의 사용 가능한 시간대

시간대 옵션에 사용할 수 있는 값은 다음과 같습니다.

영역	시간대
아프리카	Africa/Cairo, Africa/Casablanca, Africa/Harare, Africa/Lagos, Africa/Luanda, Africa/Monrovia, Africa/Nairobi, Africa/Tripoli, Africa/Windhoek
아메리카	America/Araguaina, America/Argentina/Buenos_Aires, America/Asuncion, America/Bogota, America/Caracas, America/Chicago, America/Chihuahua, America/Cuiaba, America/Denver, America/Detroit, America/Fortaleza, America/Godthab, America/Guatemala, America/Halifax, America/Lima, America/Los_Angeles, America/Manaus, America/Matamoros, America/Mexico_City, America/Monterrey, America/Montevideo, America/New_York, America/Phoenix, America/Santiago, America/Sao_Paulo, America/Tijuana, America/Toronto
아시아	Asia/Amman, Asia/Ashgabat, Asia/Baghdad, Asia/Baku, Asia/Bangkok, Asia/Beirut, Asia/Calcutta, Asia/Damascus, Asia/Dhaka, Asia/Hong_Kong, Asia/Irkutsk, Asia/Jakarta, Asia/Jerusalem, Asia/Kabul, Asia/Karachi, Asia/Kathmandu, Asia/Kolkata, Asia/Krasnoyarsk, Asia/Magadan, Asia/Manila, Asia/Muscat, Asia/Novosibirsk, Asia/Rangoon, Asia/Riyadh, Asia/Seoul, Asia/Shanghai, Asia/Singapore, Asia/Taipei, Asia/Tehran, Asia/Tokyo, Asia/Ulaanbaatar, Asia/Vladivostok, Asia/Yakutsk, Asia/Yerevan
대서양	Atlantic/Azores, Atlantic/Cape_Verde

영역	시간대
호주	Australia/Adelaide, Australia/Brisbane, Australia/Darwin, Australia/Eucla, Australia/Hobart, Australia/Lord_Howe, Australia/Perth, Australia/Sydney
브라질	Brazil/DeNoronha, Brazil/East
캐나다	Canada/Newfoundland, Canada/Saskatchewan
기타	Etc/GMT-3
유럽	Europe/Amsterdam, Europe/Athens, Europe/Berlin, Europe/Dublin, Europe/Helsinki, Europe/Kaliningrad, Europe/London, Europe/Madrid, Europe/Moscow, Europe/Paris, Europe/Prague, Europe/Rome, Europe/Sarajevo
태평양	Pacific/Apia, Pacific/Auckland, Pacific/Chatham, Pacific/Fiji, Pacific/Guam, Pacific/Honolulu, Pacific/Kiritimati, Pacific/Marquesas, Pacific/Samoa, Pacific/Tongatapu, Pacific/Wake
US	US/Alaska, US/Central, US/East-Indiana, US/Eastern, US/Pacific
UTC	UTC

RDS Custom for Oracle의 시간대 설정에 대한 고려 사항

DB 인스턴스의 시간대를 설정하기로 선택한 경우 다음을 고려하세요.

- 시간대 옵션을 추가하는 경우 DB 인스턴스를 자동으로 다시 시작하는 동안 인스턴스가 잠시 중단됩니다.
- 실수로 표준 시간대를 잘못 설정한 경우 DB 인스턴스를 이전 표준 시간대 설정으로 복구해야 합니다. 따라서 인스턴스에 시간대 옵션을 추가하기 전에 다음 전략 중 하나를 사용하는 것이 좋습니다.
  - RDS Custom for Oracle DB 인스턴스가 기본 옵션 그룹을 사용하는 경우 DB 인스턴스의 스냅샷을 생성합니다. 자세한 내용은 RDS Custom for Oracle 스냅샷 생성 단원을 참조하십시오.
  - DB 인스턴스가 현재 기본값이 아닌 옵션 그룹을 사용하는 경우 DB 인스턴스의 스냅샷을 생성한 다음 시간대 옵션을 사용하여 새 옵션 그룹을 만듭니다.
- `Timezone` 옵션을 적용한 후에는 DB 인스턴스를 수동으로 백업하는 것이 좋습니다.
- 프로덕션 DB 인스턴스에 추가하기 전에 테스트 DB 인스턴스에서 시간대 옵션을 테스트하기를 강력히 권장합니다. 시간대 옵션을 추가하면 시스템 날짜를 이용해 날짜나 시간을 추가하는 테이블에 문

제가 발생할 수 있습니다. 데이터와 애플리케이션을 분석해 표준 시간대 변경에 따른 영향을 평가하는 것이 좋습니다.

RDS Custom for Oracle의 시간대 설정에 대한 제한 사항

다음과 같은 제한 사항이 있습니다.

- 호스트를 지원 범위 밖으로 이동하지 않고는 호스트에서 직접 시간대를 변경할 수 없습니다. 데이터베이스 시간대를 변경하려면 옵션 그룹을 생성해야 합니다.
- 시간대 옵션은 지속되는 옵션이지만, 영구 옵션은 아니므로 다음을 수행할 수 없습니다.
  - 옵션을 추가한 후에는 옵션 그룹에서 이 옵션을 제거합니다.
  - 옵션의 시간대 설정을 다른 시간대로 수정합니다.
- RDS Custom for Oracle DB 인스턴스와 여러 옵션 그룹을 연결할 수 없습니다.
- CDB 내에서 개별 PDB의 시간대를 설정할 수 없습니다.

옵션 그룹에 시간대 옵션 추가

RDS Custom for Oracle의 기본 옵션 그룹은 다음 내용과 같습니다.

- `default:custom-oracle-ee`
- `default:custom-oracle-se2`
- `default:custom-oracle-ee-cdb`
- `default:custom-oracle-se2-cdb`

옵션 그룹을 생성할 때 설정은 기본 옵션 그룹에서 파생됩니다. Amazon RDS의 옵션 그룹에 대한 일반적인 정보는 옵션 그룹 작업 섹션을 참조하세요.

콘솔

시간대 옵션을 옵션 그룹에 추가하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 수정하려는 옵션 그룹을 선택한 다음 옵션 추가를 선택합니다.

4. 옵션 추가 창에서 다음과 같이 합니다.

   a. 시간대를 선택합니다.

   b. 옵션 설정에서 시간대를 선택합니다.

   c. 옵션을 추가하는 즉시 연결된 모든 RDS Custom for Oracle DB 인스턴스에서 옵션을 활성화하려면 즉시 적용에서 예를 선택합니다. 아니요(기본 설정)를 선택하면 다음 유지 관리 기간에 연결된 모든 DB 인스턴스에서 옵션이 활성화됩니다.

   d.

   > **Important**
   >
   > 하나 이상의 DB 인스턴스에 이미 연결되어 있는 기존 옵션 그룹에 시간대 옵션을 추가하면 모든 DB 인스턴스가 자동으로 다시 시작되는 동안 인스턴스가 잠시 중단됩니다.

5. 원하는 대로 설정이 되었으면 옵션 추가를 선택합니다.

6. 시간대가 업데이트된 RDS Custom for Oracle DB 인스턴스를 백업합니다. 자세한 내용은 RDS Custom for Oracle 스냅샷 생성 단원을 참조하십시오.

AWS CLI

다음 예에서는 AWS CLI add-option-to-option-group 명령을 사용하여 `Timezone` 옵션 및 `TIME_ZONE` 옵션 설정을 `testoptiongroup`이라는 옵션 그룹에 추가합니다. 표준 시간대는 `America/Los_Angeles`로 설정되어 있습니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds add-option-to-option-group \
    --option-group-name "testoptiongroup" \
    --options "OptionName=Timezone,OptionSettings=[{Name=TIME_ZONE,Value=America/
Los_Angeles}]" \
    --apply-immediately
```

Windows의 경우:

```
aws rds add-option-to-option-group ^
    --option-group-name "testoptiongroup" ^
    --options "OptionName=Timezone,OptionSettings=[{Name=TIME_ZONE,Value=America/
Los_Angeles}]" ^
    --apply-immediately
```

시간대 옵션 제거

시간대 옵션은 지속적인 옵션이지만, 영구적인 옵션은 아닙니다. 추가한 후에는 옵션 그룹에서 이 옵션을 제거할 수 없습니다. DB 인스턴스에서 이전 옵션 그룹의 연결을 해제하려면 다음을 수행하세요.

1. 업데이트된 `Timezone` 옵션이 있는 새 옵션 그룹을 생성합니다.
2. 인스턴스를 수정할 때 새 옵션 그룹을 DB 인스턴스와 연결합니다.

온프레미스 데이터베이스를 RDS Custom for Oracle로 마이그레이션

온프레미스 Oracle 데이터베이스를 RDS Custom for Oracle로 마이그레이션하기 전에 다음 요소를 고려하세요.

- 애플리케이션이 감당할 수 있는 가동 중지 시간
- 소스 데이터베이스의 크기
- 네트워크 연결
- 대체 계획의 요구 사항
- 소스 및 대상 Oracle 데이터베이스 버전과 DB 인스턴스 OS 유형
- 사용 가능한 복제 도구(예: AWS Database Migration Service, Oracle GoldenGate 또는 서드 파티 복제 도구)

이러한 요소를 기준으로 물리적 마이그레이션, 논리적 마이그레이션을 선택하거나 조합하여 선택할 수 있습니다. 물리적 마이그레이션을 선택하면 다음과 같은 기술을 사용할 수 있습니다.

RMAN 복제

활성 데이터베이스 복제 시 소스 데이터베이스를 백업할 필요가 없습니다. 이 작업은 네트워크를 통해 데이터베이스 파일을 보조 인스턴스로 복사하여 라이브 소스 데이터베이스를 대상 호스트에 복제합니다. RMAN `DUPLICATE` 명령은 필수 파일을 이미지 복사본 또는 백업 세트로 복사합니다. 이 기술에 대해 알아보려면 AWS 블로그 게시물인 Physical migration of Oracle databases to Amazon RDS Custom using RMAN duplication(RMAN 복제를 사용하여 Oracle 데이터베이스를 Amazon RDS Custom에 물리적으로 마이그레이션)을 참조하세요.

Oracle Data Guard

이 기술에서는 기본 온프레미스 데이터베이스를 백업하고, 백업을 Amazon S3 버킷에 복사합니다. 그런 다음, 백업을 RDS Custom for Oracle 대기 DB 인스턴스에 복사합니다. 필요한 구성을 수행한 후에는 기본 데이터베이스를 RDS Custom for Oracle 대기 데이터베이스로 수동으로 전환하면 됩니다. 이 기술에 대해 알아보려면 AWS 블로그 게시물인 Physical migration of Oracle databases to Amazon RDS Custom using Data Guard(Data Guard를 사용하여 Oracle 데이터베이스를 Amazon RDS Custom에 물리적으로 마이그레이션)을 참조하세요.

데이터를 RDS for Oracle에 논리적으로 가져오는 일반적인 방법은 Amazon RDS의 Oracle로 데이터 가져오기 섹션을 참조하세요.

Oracle용 Amazon RDS Custom DB 인스턴스 업그레이드

Amazon RDS Custom DB 인스턴스를 새로운 사용자 지정 엔진 버전(CEV)을 사용하도록 수정하여 업그레이드할 수 있습니다. 업그레이드에 대한 일반적인 정보는 DB 인스턴스 엔진 버전 업그레이드 섹션을 참조하세요.

주제

- RDS Custom for Oracle 업그레이드 개요
- RDS Custom for Oracle 업그레이드에 대한 요구 사항
- RDS Custom for Oracle 데이터베이스 업그레이드에 대한 고려 사항
- RDS Custom for Oracle OS 업그레이드에 대한 고려 사항
- RDS Custom for Oracle DB 인스턴스의 유효한 CEV 업그레이드 대상 보기
- RDS Custom for Oracle DB 인스턴스 업그레이드
- RDS Custom DB 인스턴스의 보류 중인 데이터베이스 업그레이드 보기
- RDS Custom for Oracle DB 인스턴스의 업그레이드 실패 문제 해결

RDS Custom for Oracle 업그레이드 개요

RDS Custom for Oracle을 사용하면 새 CEV를 만든 다음 새 CEV를 사용하도록 인스턴스를 수정하여 Oracle 데이터베이스 또는 DB 인스턴스 운영 체제(OS)에 패치를 적용할 수 있습니다.

주제

- CEV 업그레이드 옵션
- CEV 없이 패치 적용
- CEV로 DB 인스턴스에 패치를 적용하기 위한 일반적인 단계

CEV 업그레이드 옵션

업그레이드를 위해 CEV를 생성할 때 상호 배타적인 다음 옵션을 사용할 수 있습니다.

데이터베이스 전용

현재 DB 인스턴스에서 사용 중인 Amazon Machine Image(AMI)를 재사용하되, 데이터베이스 바이너리는 다르게 지정하세요. RDS Custom은 새 바이너리 볼륨을 할당한 다음 기존 Amazon EC2 인

스턴스에 연결합니다. RDS Custom은 전체 데이터베이스 볼륨을 대상 데이터베이스 버전을 사용하는 새 볼륨으로 교체합니다.

OS 전용

현재 DB 인스턴스에서 사용 중인 데이터베이스 바이너리를 재사용하되 AMI는 다르게 지정하세요. RDS Custom은 새 Amazon EC2 인스턴스를 할당한 다음 기존 바이너리 볼륨을 새 인스턴스에 연결합니다. 기존 데이터베이스 볼륨은 유지됩니다.

OS와 데이터베이스를 모두 업그레이드하려면 CEV를 두 번 업그레이드해야 합니다. OS를 업그레이드한 다음 데이터베이스를 업그레이드하거나 데이터베이스를 업그레이드한 후 OS를 업그레이드할 수 있습니다.

> **Warning**
>
> OS에 패치를 적용하면 루트 볼륨 데이터와 기존 OS 사용자 지정이 손실됩니다. 따라서 설치용 또는 영구 데이터나 파일 저장용 루트 볼륨은 사용하지 않는 것이 가장 바람직합니다. 또한 업그레이드 전에 데이터를 백업하는 것이 좋습니다.

CEV 없이 패치 적용

CEV를 사용하여 RDS Custom for Oracle DB 인스턴스를 업그레이드하는 것이 좋습니다. RDS Custom for Oracle 자동화는 패치 메타데이터를 DB 인스턴스의 데이터베이스 바이너리와 동기화합니다.

특수한 상황에서 RDS Custom은 OPatch 유틸리티를 사용하여 기본 Amazon EC2 인스턴스에 직접 '일회성' 데이터베이스 패치를 적용할 수 있도록 지원합니다. 유효한 사용 사례는 즉시 적용하려는 데이터베이스 패치일 수 있지만 RDS Custom 팀에서 CEV 기능을 업그레이드하고 있어 지연이 발생합니다. 수동으로 데이터베이스 패치를 적용하려면 다음 단계를 수행합니다.

1. RDS Custom 자동화를 일시 중지합니다.
2. Amazon EC2 인스턴스의 데이터베이스 바이너리에 패치를 적용합니다.
3. RDS Custom 자동화를 다시 시작합니다.

이전 기법의 단점은 업그레이드하려는 모든 인스턴스에 데이터베이스 패치를 수동으로 적용해야 한다는 것입니다. 반대로 새 CEV를 만들 때는 동일한 CEV를 사용하여 여러 DB 인스턴스를 만들거나 업그레이드할 수 있습니다.

CEV로 DB 인스턴스에 패치를 적용하기 위한 일반적인 단계

OS 패치와 데이터베이스 패치 모두 다음 기본 단계를 수행합니다.

1. 패치 적용 대상이 데이터베이스인지 OS인지에 따라 다음 중 하나를 포함하는 CEV를 생성합니다.
   - DB 인스턴스에 적용할 Oracle 데이터베이스 RU
   - 다른 AMI(사용 가능한 최신 AMI 또는 사용자가 지정한 AMI) 및 소스로 사용할 기존 CEV

   CEV 생성 단원의 단계를 따르세요.
2. (데이터베이스 패치 적용의 경우 선택 사항) `describe-db-engine-versions`를 실행하여 사용 가능한 엔진 버전 업그레이드를 확인합니다.
3. `modify-db-instance`를 실행하여 패치 프로세스를 시작합니다.

   패치 적용 중인 인스턴스의 상태는 다음과 같이 다릅니다.
   - RDS에서 데이터베이스에 패치를 적용 중인 경우 DB 인스턴스 상태가 업그레이드 중으로 바뀝니다.
   - RDS에서 OS에 패치를 적용 중인 경우 DB 인스턴스 상태가 수정 중으로 바뀝니다.

   DB 인스턴스의 상태가 사용 가능이면 패치 적용이 완료된 것입니다.
4. `describe-db-instances`를 실행하여 DB 인스턴스가 새 CEV를 사용하는지 확인합니다.

RDS Custom for Oracle 업그레이드에 대한 요구 사항

RDS Custom for Oracle DB 인스턴스를 대상 CEV로 업그레이드할 때 다음 요구 사항을 충족하는지 확인해야 합니다.

- 업그레이드하려는 대상 CEV가 존재해야 합니다.
- 한 번의 작업으로 OS 또는 데이터베이스를 업그레이드해야 합니다. 하나의 API 호출로 OS와 데이터베이스를 모두 업그레이드하는 것은 지원되지 않습니다.
- 대상 CEV는 현재 CEV의 매니페스트에 있는 설치 파라미터 설정을 사용해야 합니다. 예를 들어 기본 Oracle home을 사용하는 데이터베이스는 기본이 아닌 Oracle home을 사용하는 CEV로 업그레이드할 수 없습니다.
- 데이터베이스 업그레이드의 경우 대상 CEV는 새 메이저 버전이 아닌 새 마이너 데이터베이스 버전을 사용해야 합니다. 예를 들어 Oracle Database 12c CEV에서 Oracle Database 19c CEV로 업그레이드할 수 없습니다. 하지만 버전 21.0.0.0.ru-2023-04.rur-2023-04.r1에서 버전 21.0.0.0.ru-2023-07.rur-2023-07.r1로 업그레이드할 수 있습니다.

- OS 업그레이드의 경우 대상 CEV는 다른 AMI를 사용하지만 메이저 버전은 같아야 합니다.

RDS Custom for Oracle 데이터베이스 업그레이드에 대한 고려 사항

데이터베이스를 업그레이드할 계획이라면 다음 사항을 고려해야 합니다.

- 현재 지원되는 운영 체제(OS) 버전은 Oracle Linux 8입니다. RDS Custom for Oracle에서 최신 보안 업데이트 및 패치를 계속 받으려면 이 OS를 기반으로 CEV를 지정하여 DB 인스턴스를 Oracle Linux 8로 업그레이드하세요. Oracle Database 12c 릴리스 1(12.1), Oracle Database 릴리스 2(12.2) 및 Oracle Database 19c만 Oracle Linux 8을 지원합니다. 최신 Oracle Linux 8 AMI로 마이그레이션하려면 OS를 최신 AMI로 업그레이드하세요. 자세한 내용은 RDS Custom for Oracle DB 인스턴스 업그레이드 단원을 참조하십시오.

  Oracle Linux 7.9는 2024년 12월 31일에 지원이 종료되었습니다. 지원 종료 후에도 Oracle Linux 7을 계속 실행하려면 Oracle Extended Support 라이선스를 구매합니다. 보안 업데이트에 대한 책임은 사용자에게 있으며, 사용자가 RDS Custom for Oracle 인스턴스를 수동으로 패치해야 합니다. 자세한 내용은 Lifetime Support Policy: Coverage for Oracle Open Source Service Offerings을 참조하세요.
- 기본 DB 인스턴스에서 데이터베이스 바이너리를 업그레이드할 때 RDS Custom for Oracle에서 읽기 전용 복제본을 자동으로 업그레이드합니다. 하지만 OS를 업그레이드할 때는 읽기 전용 복제본을 수동으로 업그레이드해야 합니다.
- 컨테이너 데이터베이스(CDB)를 새 데이터베이스 버전으로 업그레이드할 때 RDS Custom for Oracle은 모든 PDB가 열려 있거나 열릴 수 있는지 검사합니다. 이러한 조건이 충족되지 않는 경우 RDS Custom은 검사를 중지하고 업그레이드 시도 없이 데이터베이스를 원래 상태로 되돌립니다. 조건이 충족되면 RDS Custom은 먼저 CDB 루트를 패치한 다음 다른 모든 PDB(`PDB$SEED` 포함)를 병렬로 패치합니다.

  패치 적용이 완료된 후 RDS Custom은 모든 PDB를 열려고 시도합니다. 열리지 않는 PDB가 있으면 `The following PDBs failed to open: list-of-PDBs` 이벤트가 발생합니다. RDS Custom이 CDB 루트 또는 PDB를 패치하지 못하면 인스턴스가 `PATCH_DB_FAILED` 상태로 전환됩니다.
- 메이저 데이터베이스 버전 업그레이드와 비CDB를 CDB로 변환하는 작업을 동시에 수행하고 싶을 수도 있습니다. 이 경우 다음 작업을 수행하는 것이 좋습니다.

  1. Oracle 멀티테넌트 아키텍처를 사용하는 새 RDS Custom for Oracle DB 인스턴스를 생성합니다.
  2. 비 CDB를 CDB 루트에 연결하여 PDB로 생성합니다. 비 CDB가 CDB와 동일한 메이저 버전인지 확인합니다.
  3. `noncdb_to_pdb.sql` Oracle SQL 스크립트를 실행하여 PDB를 변환합니다.

4. CDB 인스턴스를 검증합니다.
5. CDB 인스턴스를 업그레이드합니다.

RDS Custom for Oracle OS 업그레이드에 대한 고려 사항

OS를 업그레이드할 계획이라면 다음 사항을 고려해야 합니다.

- RDS Custom for Oracle CEV에서 사용할 자체 AMI를 제공할 수 없습니다. Oracle Linux 8을 사용하는 기본 AMI 또는 이전에 RDS Custom for Oracle CEV에서 사용한 AMI를 지정할 수 있습니다.

  > **Note**
  >
  > RDS Custom for Oracle은 일반적인 취약성 및 노출이 발견되면 새로운 기본 AMI를 릴리스합니다. 일정이 정해져 있거나 보장되지 않습니다. RDS Custom for Oracle은 보통 30일마다 새로운 기본 AMI를 게시합니다.

- 기본 DB 인스턴스에서 OS를 업그레이드할 때 연결된 읽기 전용 복제본을 수동으로 업그레이드해야 합니다.
- OS 패치를 시작하기 전에 AZ의 인스턴스 유형에 충분한 Amazon EC2 컴퓨팅 용량을 예약하세요.

  용량 예약을 생성할 때 AZ, 인스턴스 수, 인스턴스 속성(인스턴스 유형 포함)을 지정합니다. 예를 들어 DB 인스턴스에서 기본 EC2 인스턴스 유형 r5.large를 사용하는 경우 AZ에서 r5.large용으로 EC2 용량을 예약하는 것이 좋습니다. OS 패치를 적용하는 동안 RDS Custom은 db.r5.large 유형의 새 호스트 하나를 생성하는데, AZ에 이 인스턴스 유형을 위한 EC2 용량이 부족하면 이 호스트가 중단될 수 있습니다. EC2 용량을 예약하면 용량 제약으로 인한 패치 적용 차단의 위험을 줄일 수 있습니다. 자세한 내용을 알아보려면 Amazon EC2 사용 설명서의 온디맨드 용량 예약을 참조하세요.
- OS를 업그레이드하기 전에 DB 인스턴스를 백업합니다. 업그레이드 시 루트 볼륨 데이터와 기존 OS 사용자 지정이 모두 제거됩니다.
- 공동 책임 모델에서는 OS를 최신 상태로 유지할 책임이 있습니다. RDS Custom for Oracle은 OS에 적용할 패치를 요구하지 않습니다. RDS Custom for Oracle이 작동하는 경우 이 CEV와 연결된 AMI를 무기한으로 사용할 수 있습니다.

RDS Custom for Oracle DB 인스턴스의 유효한 CEV 업그레이드 대상 보기

AWS Management Console의 사용자 지정 엔진 버전 페이지에서 기존 CEV를 볼 수 있습니다.

describe-db-engine-versions AWS CLI 명령을 사용하여 다음 예시와 같이 DB 인스턴스를 업그레이드할 때 사용할 유효한 CEV를 찾을 수도 있습니다. 이 예시에서는 엔진 버전 19.my_cev1을 사용하여 DB 인스턴스를 만들었고 업그레이드 버전 19.my_cev2와 19.my_cev가 존재한다고 가정합니다.

```
aws rds describe-db-engine-versions --engine custom-oracle-ee --engine-version
 19.my_cev1
```

다음과 유사하게 출력됩니다. ImageId 필드에는 AMI ID가 표시됩니다.

```
{
    "DBEngineVersions": [
        {
            "Engine": "custom-oracle-ee",
            "EngineVersion": "19.my_cev1",
            ...
            "Image": {
                "ImageId": "ami-2345",
                "Status": "active"
            },
            "DBEngineVersionArn": "arn:aws:rds:us-west-2:123456789012:cev:custom-
oracle-ee/19.my_cev1/12a34b5c-67d8-90e1-2f34-gh56ijk78lm9"
            "ValidUpgradeTarget": [
                {
                    "Engine": "custom-oracle-ee",
                    "EngineVersion": "19.my_cev2",
                    "Description": "19.my_cev2 description",
                    "AutoUpgrade": false,
                    "IsMajorVersionUpgrade": false
                },
                {
                    "Engine": "custom-oracle-ee",
                    "EngineVersion": "19.my_cev3",
                    "Description": "19.my_cev3 description",
                    "AutoUpgrade": false,
                    "IsMajorVersionUpgrade": false
                }
            ]
            ...
```

RDS Custom for Oracle DB 인스턴스 업그레이드

RDS Custom for Oracle DB 인스턴스를 업그레이드하려면 새로운 CEV를 사용하도록 수정하면 됩니다. 이 CEV에는 새 데이터베이스 바이너리나 새 AMI가 포함될 수 있습니다. 예를 들어 Oracle Linux 7.9 DB 인스턴스를 Oracle Linux 8로 업그레이드하려면 Oracle Linux 8을 사용하는 최신 AMI를 지정합니다. 데이터베이스와 OS를 업그레이드하려면 별도의 업그레이드를 두 번 수행해야 합니다.

> **Note**
>
> 데이터베이스를 업그레이드하는 경우 RDS Custom은 기본 DB 인스턴스를 업그레이드한 후 읽기 전용 복제본을 자동으로 업그레이드합니다. OS를 업그레이드하면 읽기 전용 복제본을 수동으로 업그레이드해야 합니다.

시작하기 전에 RDS Custom for Oracle 업그레이드에 대한 요구 사항 및 RDS Custom for Oracle 데이터베이스 업그레이드에 대한 고려 사항 섹션을 검토하세요.

콘솔

RDS Custom for Oracle DB 인스턴스를 업그레이드하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 후 업그레이드하려는 RDS Custom for Oracle DB 인스턴스를 선택합니다.
3. Modify(수정)를 선택합니다. Modify DB instance(DB 인스턴스 수정) 페이지가 나타납니다.
4. DB 엔진 버전에서 새로운 CEV를 선택합니다. 다음을 수행합니다.
   - 데이터베이스에 패치를 적용하는 경우 CEV가 DB 인스턴스에서 사용하는 것과 다른 데이터베이스 바이너리를 지정하고 DB 인스턴스에서 현재 사용하는 AMI와 다른 AMI를 지정하지 않도록 해야 합니다.
   - OS에 패치를 적용하는 경우 CEV가 DB 인스턴스에서 사용하는 것과 다른 AMI를 지정하고 다른 데이터베이스 바이너리를 지정하지 않도록 해야 합니다.

   > **Warning**
   >
   > OS에 패치를 적용하면 루트 볼륨 데이터와 기존 OS 사용자 지정이 손실됩니다.

5. 계속(Continue)을 선택하여 수정 사항을 요약한 내용을 확인합니다.

   변경 사항을 즉시 적용하려면 즉시 적용을 선택합니다.

6. 변경 내용이 정확할 경우 DB 인스턴스 수정(Modify DB instance)을 선택합니다. 또는 뒤로를 선택하여 변경 내용을 편집하거나 취소를 선택하여 변경 내용을 취소합니다.

AWS CLI

다음 예시는 가능한 업그레이드 시나리오를 보여줍니다. 이 예시에서는 다음 특징을 가진 RDS Custom for Oracle DB 인스턴스를 생성했다고 가정합니다.

- 이름이 `my-custom-instance`인 DB 인스턴스
- 이름이 `19.my_cev1`인 CEV
- 비 CDB 아키텍처를 사용하는 Oracle Database 19c
- AMI `ami-1234`를 사용하는 Oracle Linux 8

최신 서비스 제공 AMI는 `ami-2345` `describe-db-engine-versions` CLI 명령을 실행하여 AMI를 찾을 수 있습니다.

주제

- OS 업그레이드
- 데이터베이스 업그레이드

OS 업그레이드

이 예시에서는 `ami-1234`를 서비스가 제공하는 최신 AMI인 `ami-2345`로 업그레이드하려고 합니다. OS를 업그레이드하고 있으므로 `ami-1234` 및 `ami-2345`의 데이터베이스 바이너리는 동일해야 합니다. `19.my_cev1`을 기반으로 이름이 `19.my_cev2`인 새 CEV를 생성합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-custom-db-engine-version \
    --engine custom-oracle-ee \
    --engine-version 19.my_cev2 \
    --description "Non-CDB CEV based on ami-2345" \
```

```
    --kms-key-id key-name \
    --source-custom-db-engine-version-identifer arn:aws:rds:us-
west-2:123456789012:cev:custom-oracle-ee/19.my_cev1/12345678-ab12-1234-cde1-
abcde123456789 \
    --image-id ami-2345
```

Windows의 경우:

```
aws rds create-custom-db-engine-version ^
    --engine custom-oracle-ee ^
    --engine-version 19.my_cev2 ^
    --description "Non-CDB CEV based on ami-2345" ^
    --kms-key-id key-name ^
    --source-custom-db-engine-version-identifer arn:aws:rds:us-
west-2:123456789012:cev:custom-oracle-ee/19.my_cev1/12345678-ab12-1234-cde1-
abcde123456789 ^
    --image-id ami-2345
```

RDS Custom DB 인스턴스를 업그레이드하려면 modify-db-instance AWS CLI 명령을 다음 파라미터와 함께 사용합니다.

- `--db-instance-identifier` - 업그레이드할 RDS Custom for Oracle DB 인스턴스를 지정합니다.
- `--engine-version` - 새 AMI가 있는 CEV를 지정합니다.
- `--no-apply-immediately` | `--apply-immediately` - 업그레이드를 즉시 수행할지 아니면 예약된 유지 관리 기간까지 기다릴지를 지정합니다.

다음 예제에서는 `my-custom-instance`를 버전 `19.my_cev2`으로 업그레이드합니다. OS만 업그레이드됩니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier my-custom-instance \
    --engine-version 19.my_cev2 \
    --apply-immediately
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --engine-version 19.my_cev2 ^
    --apply-immediately
```

데이터베이스 업그레이드

이 예시에서는 RDS for Oracle DB 인스턴스에 Oracle 패치 p35042068을 적용하려고 합니다. OS 업그레이드에서 OS를 업그레이드했으므로 DB 인스턴스는 현재 ami-2345를 기반으로 19.my_cev2를 사용하고 있습니다. 동일하게 ami-2345를 사용하는 이름이 19.my_cev3인 새 CEV를 만들되 $MANIFEST 환경 변수에 새 JSON 매니페스트를 지정합니다. 따라서 새 CEV와 인스턴스가 현재 사용 중인 CEV에서 데이터베이스 바이너리만 다르게 설정됩니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-custom-db-engine-version \
    --engine custom-oracle-ee \
    --engine-version 19.my_cev3 \
    --description "Non-CDB CEV with p35042068 based on ami-2345" \
    --kms-key-id key-name \
    --image-id ami-2345 \
    --manifest $MANIFEST
```

Windows의 경우:

```
aws rds create-custom-db-engine-version ^
    --engine custom-oracle-ee ^
    --engine-version 19.my_cev3 ^
    --description "Non-CDB CEV with p35042068 based on ami-2345" ^
    --kms-key-id key-name ^
    --image-id ami-2345 ^
    --manifest $MANIFEST
```

다음 예시에서는 my-custom-instance를 엔진 버전 19.my_cev3으로 업그레이드합니다. 데이터베이스만 업그레이드됩니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier my-custom-instance \
    --engine-version 19.my_cev3 \
    --apply-immediately
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --engine-version 19.my_cev3 ^
    --apply-immediately
```

RDS Custom DB 인스턴스의 보류 중인 데이터베이스 업그레이드 보기

describe-db-instances 또는 describe-pending-maintenance-actions AWS CLI 명령을 사용하여 Amazon RDS Custom DB 인스턴스의 보류 중인 데이터베이스 업그레이드를 확인할 수 있습니다.

그러나 `--apply-immediately` 옵션을 사용했거나 업그레이드가 진행 중인 경우에는 이 방법을 사용할 수 없습니다.

다음 `describe-db-instances` 명령은 `my-custom-instance`에 대한 보류 중인 데이터베이스 업그레이드를 보여줍니다.

```
aws rds describe-db-instances --db-instance-identifier my-custom-instance
```

다음과 유사하게 출력됩니다.

```
{
    "DBInstances": [
        {
           "DBInstanceIdentifier": "my-custom-instance",
            "EngineVersion": "19.my_cev1",
            ...
            "PendingModifiedValues": {
                "EngineVersion": "19.my_cev3"
            ...
            }
        }
    ]
}
```

RDS Custom for Oracle DB 인스턴스의 업그레이드 실패 문제 해결

RDS Custom DB 인스턴스 업그레이드가 실패하면 RDS 이벤트가 생성되고 DB 인스턴스 상태가 `upgrade-failed`로 변경됩니다.

다음 예제와 같이 describe-db-instances AWS CLI 명령을 사용하여 이 상태를 확인할 수 있습니다.

```
aws rds describe-db-instances --db-instance-identifier my-custom-instance
```

다음과 유사하게 출력됩니다.

```
{
    "DBInstances": [
        {
           "DBInstanceIdentifier": "my-custom-instance",
            "EngineVersion": "19.my_cev1",
            ...
            "PendingModifiedValues": {
                "EngineVersion": "19.my_cev3"
            ...
            }
            "DBInstanceStatus": "upgrade-failed"
        }
    ]
}
```

업그레이드 실패 후 다음 작업을 수행할 수 있도록 DB 인스턴스를 수정하는 작업을 제외한 모든 데이터베이스 작업이 차단됩니다.

- 동일한 업그레이드 재시도
- RDS Custom 자동화 일시 중지 및 다시 시작
- 특정 시점으로 복구(PITR)
- DB 인스턴스 삭제

> **Note**
>
> RDS Custom DB 인스턴스에 대해 자동화가 일시 중지된 경우 자동화를 재개할 때까지 업그레이드를 다시 시도할 수 없습니다.

프라이머리에 대해서 RDS 관리형 읽기 전용 복제본의 업그레이드 실패에도 이와 동일한 작업이 적용됩니다.

자세한 내용은 RDS Custom for Oracle 업그레이드 문제 해결을 참조하세요.

Amazon RDS Custom for Oracle의 DB 문제 해결

RDS Custom의 공동 책임 모델은 OS 셸 수준 액세스 권한과 데이터베이스 관리자 액세스 권한을 제공합니다. RDS Custom은 시스템 계정에서 리소스를 실행하는 Amazon RDS와 달리 계정에서 리소스를 실행합니다. 접근 권한이 높을수록 책임도 커집니다. Amazon RDS Custom DB 인스턴스의 문제를 해결하는 방법을 알아볼 수 있습니다.

> **Note**
>
> 이 섹션에서는 RDS Custom for Oracle 관련 문제를 해결하는 방법을 설명합니다. RDS Custom for SQL Server의 문제 해결 방법을 알아보려면 Amazon RDS Custom for SQL Server의 DB 문제 해결 섹션을 참조하세요.

주제

- RDS Custom 이벤트 보기
- RDS Custom 이벤트 구독
- Oracle용 RDS Custom에 대한 사용자 지정 엔진 버전 생성 문제 해결
- RDS Custom for Oracle에서 지원되지 않는 구성 문제 해결
- RDS Custom for Oracle 업그레이드 문제 해결
- RDS Custom for Oracle 복제본 승격 문제 해결

RDS Custom 이벤트 보기

RDS Custom 및 Amazon RDS DB 인스턴스의 이벤트 보기 절차는 동일합니다. 자세한 내용은 Amazon RDS 이벤트 보기 섹션을 참조하세요.

`describe-events` 명령을 사용하여 AWS CLI를 통해 RDS Custom 이벤트 알림을 볼 수 있습니다. RDS Custom에서는 몇 가지 새로운 이벤트가 도입되었습니다. 이벤트 카테고리는 Amazon RDS와 동일합니다. 이벤트 목록은 Amazon RDS 이벤트 카테고리 및 이벤트 메시지 섹션을 참조하세요.

다음 예제에서는 지정된 RDS Custom DB 인스턴스에 대해 발생한 이벤트의 세부 정보를 검색합니다.

```
aws rds describe-events \
    --source-identifier my-custom-instance \
    --source-type db-instance
```

RDS Custom 이벤트 구독

RDS Custom 및 Amazon RDS DB 인스턴스의 이벤트 구독 절차는 동일합니다. 자세한 내용은 Amazon RDS 이벤트 알림 구독 섹션을 참조하세요.

CLI를 사용하여 RDS Custom 이벤트 알림을 구독하려면 `create-event-subscription` 명령을 사용하면 되며, 다음 필수 파라미터를 포함합니다.

- `--subscription-name`
- `--sns-topic-arn`

다음 예제에서는 현재 AWS 계정에서 RDS Custom DB 인스턴스의 백업 및 복구 이벤트에 대한 구독을 생성합니다. 알림은 `--sns-topic-arn`에서 지정한 Amazon Simple Notification Service(Amazon SNS) 주제로 전송됩니다.

```
aws rds create-event-subscription \
    --subscription-name my-instance-events \
    --source-type db-instance \
    --event-categories '["backup","recovery"]' \
    --sns-topic-arn arn:aws:sns:us-east-1:123456789012:interesting-events
```

Oracle용 RDS Custom에 대한 사용자 지정 엔진 버전 생성 문제 해결

CEV 생성이 실패하면 RDS Custom에서 `Creation failed for custom engine version` *`major-engine-version.cev_name`* 메시지와 함께 `RDS-EVENT-0198` 오류가 발생하며, 실패 관련 세부 정보가 표시됩니다. 예를 들어, 이벤트는 누락된 파일을 인쇄합니다.

다음 문제로 인해 CEV 생성에 실패할 수 있습니다.

- 설치 파일이 포함된 Amazon S3 버킷이 CEV와 동일한 AWS 리전에 있지 않습니다.
- AWS 리전에서 CEV 생성을 처음으로 요청하면 RDS Custom이 CEV 아티팩트, AWS CloudTrail 로그 및 트랜잭션 로그와 같은 RDS Custom 리소스를 저장하기 위한 S3 버킷을 생성합니다.

  RDS Custom에서 S3 버킷을 생성할 수 없는 경우 CEV 생성에 실패합니다. 발신자에게 5단계: IAM 사용자 또는 역할에 필요한 권한 부여에 설명된 대로 S3 사용 권한이 없거나 S3 버킷 수가 한도에 도달한 것일 수 있습니다.
- 발신자에게 설치 미디어 파일이 포함된 S3 버킷에서 파일을 가져올 수 있는 권한이 없습니다. 이러한 권한은 7단계: 필요한 IAM 권한 추가에 설명되어 있습니다.

- IAM 정책에 `aws:SourceIp` 조건이 있습니다. AWS Identity and Access Management 사용 설명서의 소스 IP를 바탕으로 AWS에 대한 AWS 액세스 거부의 권장 사항을 따라야 합니다. 또한 5단계: IAM 사용자 또는 역할에 필요한 권한 부여에 설명된 S3 권한이 호출자에게 있는지 확인합니다.
- CEV 매니페스트에 나열된 설치 미디어 파일이 S3 버킷에 없습니다.
- RDS Custom에서 설치 파일의 SHA-256 체크섬을 알 수 없습니다.

  제공된 파일의 SHA-256 체크섬이 Oracle 웹 사이트의 SHA-256 체크섬과 일치하는지 확인해야 합니다. 체크섬이 일치하는 경우 AWS 지원팀에 문의하여 실패한 CEV 이름, 파일 이름 및 체크섬을 제공합니다.
- OPatch 버전이 패치 파일과 호환되지 않습니다. 다음과 같은 메시지가 표시될 수 있습니다. `OPatch is lower than minimum required version. Check that the version meets the requirements for all patches, and try again.` Oracle 패치를 적용하려면 호환되는 버전의 OPatch 유틸리티를 사용해야 합니다. 패치의 추가 정보 파일에서 필요한 Opatch 유틸리티의 버전을 찾을 수 있습니다. My Oracle 지원에서 최신 OPatch 유틸리티를 다운로드하고 CEV를 다시 생성해 보세요.
- CEV 매니페스트에 지정된 패치의 순서가 잘못되었습니다.

RDS 이벤트는 RDS 콘솔(탐색 창에서 이벤트(Events) 선택)에서 보거나 `describe-events` AWS CLI 명령을 사용하여 볼 수 있습니다. 기본 지속 시간은 60분입니다. 이벤트가 반환되지 않으면 다음 예제와 같이 기간이 더 길게 지정됩니다.

```
aws rds describe-events --duration 360
```

현재 Amazon S3에서 파일을 가져와 CEV를 생성하는 MediaImport 서비스가 AWS CloudTrail에 통합되어 있지 않습니다. 이로 인해 CloudTrail에서 Amazon RDS에 대한 데이터 로깅을 설정하면 `CreateCustomDbEngineVersion` 이벤트와 같이 MediaImport 서비스에 대한 호출이 로깅되지 않습니다.

단, Amazon S3 버킷에 액세스하는 API 게이트웨이에서의 호출은 확인할 수 있습니다. 이러한 호출은 `CreateCustomDbEngineVersion` 이벤트의 MediaImport 서비스에서 전송됩니다.

RDS Custom for Oracle에서 지원되지 않는 구성 문제 해결

공유 책임 모델에서는 RDS Custom for Oracle DB 인스턴스를 `unsupported-configuration` 상태로 전환시키는 구성 문제는 사용자가 해결해야 합니다. AWS 인프라와 관련된 문제인 경우 콘솔 또는 AWS CLI를 사용하여 해결합니다. 운영 체제 또는 데이터베이스 구성에 문제가 있는 경우 호스트에 로그인하여 해결합니다.

> **Note**
>
> 이 섹션에서는 RDS Custom for Oracle에서 지원되지 않는 구성 문제를 해결하는 방법을 알아봅니다. RDS Custom for SQL Server에 관한 자세한 내용은 RDS Custom for SQL Server에서 지원되지 않는 구성 문제 해결 섹션을 참조하세요.

다음 테이블에는 지원 경계에서 보내는 알림 이벤트와 이를 해결하는 방법에 대한 설명이 나와 있습니다. 이러한 알림과 지원 경계는 변경될 수 있습니다. 지원 경계의 배경은 RDS Custom 지원 범위 섹션을 참조하세요. 이벤트 설명은 Amazon RDS 이벤트 카테고리 및 이벤트 메시지 섹션을 참조하세요.

이벤트 ID	구성	RDS 이벤트 메시지	작업
SP-00000	지원되지 않는 수동 구성	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성]으로 설정된 이유는 다음과 같습니다. *##*.	이 문제를 해결하려면 지원 사례를 생성하세요.

AWS 리소스(인프라)

이벤트 ID	구성	RDS 이벤트 메시지	작업
SP-O1001	Amazon Elastic Block Store(Amazon EBS) 볼륨	*volume_id* EBS 볼륨이 EC2 인스턴스 *ec2_id*에 추가되었습니다. 이 문제를 해결하려면 인스턴스에서 지정된 볼륨을 분리하세요.	RDS Custom은 Amazon Machine Image(AMI)에서 생성된 루트 볼륨 외에 2가지 유형의 EBS 볼륨을 생성하여 EC2 인스턴스에 연결합니다. • 데이터베이스 소프트웨어 바이너리가 있는 바이너리 볼륨 • 데이터베이스 파일이 있는 데이터 볼륨

이벤트 ID	구성	RDS 이벤트 메시지	작업

			DB 인스턴스를 생성할 때 지정한 스토리지 구성이 데이터 볼륨을 구성합니다.
			
			지원 경계는 다음을 모니터링합니다.
			
			• DB 인스턴스와 함께 생성된 초기 EBS 볼륨은 여전히 인스턴스와 연결되어 있습니다.
			• 초기 EBS 볼륨에 스토리지 유형, 크기, 프로비저닝된 IOPS 및 스토리지 처리량 등 초기 설정과 동일한 구성이 그대로 적용되어 있습니다.
			• DB 인스턴스에 연결된 추가 EBS 볼륨이 없습니다.
			
			EBS 볼륨 세부 정보와 RDS Custom for Oracle DB 인스턴스 세부 정보의 볼륨 유형을 비교하려면 다음 CLI 명령을 사용합니다.
			
			`aws rds describe-db-instances \`
			`--db-instance-identifier db-instance-name | grep StorageType`

이벤트 ID	구성	RDS 이벤트 메시지	작업
SP-O1002	Amazon Elastic Block Store(Amazon EBS) 볼륨	EBS 볼륨 *volume_id* 가 EC2 인스턴스 [*ec2_id*]에서 분리되었습니다. 이 인스턴스에서 원본 볼륨을 분리할 수 없습니다. 이 문제를 해결하려면 *volume_id* 를 *ec2_id*에 다시 연결하세요.	RDS Custom은 Amazon Machine Image(AMI)에서 생성된 루트 볼륨 외에 2가지 유형의 EBS 볼륨을 생성하여 EC2 인스턴스에 연결합니다. • 데이터베이스 소프트웨어 바이너리가 있는 바이너리 볼륨 • 데이터베이스 파일이 있는 데이터 볼륨 DB 인스턴스를 생성할 때 지정한 스토리지 구성이 데이터 볼륨을 구성합니다. 지원 경계는 다음을 모니터링합니다. • DB 인스턴스와 함께 생성된 초기 EBS 볼륨은 여전히 인스턴스와 연결되어 있습니다. • 초기 EBS 볼륨에 스토리지 유형, 크기, 프로비저닝된 IOPS 및 스토리지 처리량 등 초기 설정과 동일한 구성이 그대로 적용되어 있습니다. • DB 인스턴스에 연결된 추가 EBS 볼륨이 없습니다. EBS 볼륨 세부 정보와 RDS Custom for Oracle DB 인스턴스 세부 정보의 볼륨 유형을 비교하려면 다음 CLI 명령을 사용합니다. `aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep StorageType`

이벤트 ID	구성	RDS 이벤트 메시지	작업
SP-O1003	Amazon Elastic Block Store(Amazon EBS) 볼륨	EC2 인스턴스 *ec2_id*에 연결된 원래 EBS *volume_id* 가 크기는 [*X*]에서 [*Y*]로, 유형은 [*N*]에서 [*M*]으로, 혹은 IOPS는 [*J*]에서 [*K*]로 수정되었습니다. 이 문제를 해결하려면 수정 내용을 되돌리세요.	RDS Custom은 Amazon Machine Image(AMI)에서 생성된 루트 볼륨 외에 2가지 유형의 EBS 볼륨을 생성하여 EC2 인스턴스에 연결합니다. • 데이터베이스 소프트웨어 바이너리가 있는 바이너리 볼륨 • 데이터베이스 파일이 있는 데이터 볼륨 DB 인스턴스를 생성할 때 지정한 스토리지 구성이 데이터 볼륨을 구성합니다. 지원 경계는 다음을 모니터링합니다. • DB 인스턴스와 함께 생성된 초기 EBS 볼륨은 여전히 인스턴스와 연결되어 있습니다. • 초기 EBS 볼륨에 스토리지 유형, 크기, 프로비저닝된 IOPS 및 스토리지 처리량 등 초기 설정과 동일한 구성이 그대로 적용되어 있습니다. • DB 인스턴스에 연결된 추가 EBS 볼륨이 없습니다. EBS 볼륨 세부 정보와 RDS Custom for Oracle DB 인스턴스 세부 정보의 볼륨 유형을 비교하려면 다음 CLI 명령을 사용합니다. `aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep StorageType`

이벤트 ID	구성	RDS 이벤트 메시지	작업
SP-O1004	Amazon EC2 인스턴스 상태	자동 복구로 인해 EC2 인스턴스 [*ec2_id*]가 손상된 상태가 되었습니다. 이 문제를 해결하려면 인스턴스 복구 실패 문제 해결을 참조하세요.	DB 인스턴스의 상태를 확인하려면 콘솔을 사용하거나 다음 AWS CLI 명령을 실행합니다. `aws rds describe-db-instances \ --db-instance-identifier *db-instance-name* |grep DBInstanceStatus`
SP-O1005	Amazon EC2 인스턴스 속성	EC2 인스턴스 [*ec2_id*]가 [*att1*] 속성이 [*val-old*]에서 [*val-new*]로, [*att2*] 속성이 [*val-old*]에서 [*val-new*]로 수정되었습니다. 이 문제를 해결하려면 원래 값으로 되돌리세요.	

이벤트 ID	구성	RDS 이벤트 메시지	작업
SP-O1006	Amazon EC2 인스턴스 상태	EC2 인스턴스 [*ec2_id*]가 종료되었거나 찾을 수 없습니다. 이 문제를 해결하려면 RDS Custom DB 인스턴스를 삭제하세요.	지원 경계는 EC2 인스턴스 상태 변경 알림을 모니터링합니다. EC2 인스턴스는 항상 실행 중이어야 합니다. DB 인스턴스를 삭제하려면 1. DB 인스턴스의 상태를 확인하려면 콘솔을 사용하거나 다음 AWS CLI 명령을 실행합니다. `aws rds describe-db-instances \ --db-instance-identifier db-instance-name |grep DBInstanceStatus` 2. RDS Custom for Oracle DB 인스턴스를 삭제합니다.
SP-O1007	Amazon EC2 인스턴스 상태	EC2 인스턴스 [*ec2_id*]가 중지되었습니다. 이 문제를 해결하려면 인스턴스를 시작하세요.	지원 경계는 EC2 인스턴스 상태 변경 알림을 모니터링합니다. EC2 인스턴스는 항상 실행 중이어야 합니다. DB 인스턴스를 다시 시작하려면 1. DB 인스턴스의 상태를 확인하려면 콘솔을 사용하거나 다음 AWS CLI 명령을 실행합니다. `aws rds describe-db-instances \ --db-instance-identifier db-instance-name |grep DBInstanceStatus` 2. DB 인스턴스를 시작합니다. 3. 바이너리 및 데이터 볼륨을 다시 탑재합니다.

이벤트 ID	구성	RDS 이벤트 메시지	작업

SP-10	Amazon SQS 권한	Amazon SQS에 대한 권한이 누락되었습니다. IAM 인스턴스 프로파일, VPC 엔드포인트 정책 및 종속 서비스 연결에 대한 권한을 확인한 다음 다시 시도하세요.	호스트와 연결된 IAM 프로파일에 다음 권한이 있는지 확인하여 이 문제를 해결할 수 있습니다.
			`"SQS:SendMessage"`
			`"SQS:ReceiveMessage"`
			`"SQS:DeleteMessage"`
			`"SQS:GetQueueUrl"`

SP-10	Amazon Simple Queue Service(Amazon SQS)	[%s] SQS 대기열이 삭제되어 복구할 수 없습니다. 이 문제를 해결하려면 대기열을 다시 생성하세요.	Amazon SQS 대기열을 다시 생성합니다.

운영 체제

이벤트 ID	구성	RDS 이벤트 메시지	작업
SP-O2001	RDS Custom 에이전트 상태	RDS Custom 에이전트가 EC2 인스턴스 [*ec2_id*]에서 실행되지 않습니다. 에이전트가 [*ec2_id*]에서 실행되고 있는지 확인하세요.	RDS Custom for Oracle에서 DB 인스턴스는 RDS Custom 에이전트가 중지되면 지원 경계를 벗어나게 됩니다. 에이전트는 30초마다 `IamAlive` 지표를 Amazon CloudWatch에 게시합니다. 지표가 30초 동안 게시되지 않은 경우 경보가 트리거됩니다. 또한, 지원 경계는 호스트의 RDS Custom 에이전트 프로세스 상태를 30분마다 모니터링합니다.

이벤트 ID	구성	RDS 이벤트 메시지	작업

			RDS Custom 에이전트를 다시 시작하려면
			
			1. 호스트에 로그인하고 RDS Custom 에이전트가 실행 중인지 확인합니다.
			2. 다음 명령을 실행하여 에이전트 상태를 찾습니다.
			
			`service rdscustomagent status`
			
			3. 다음 명령을 사용하여 에이전트를 시작합니다.
			
			`service rdscustomagent start`
			
			RDS Custom 에이전트가 다시 실행되면 `IamAlive` 지표가 Amazon CloudWatch에 게시되고 경보가 `OK` 상태로 전환됩니다. 이 스위치는 지원 경계에 에이전트가 실행 중임을 알립니다.

SP-02002	AWS Systems Manager 에이전트(SSM 에이전트) 상태	EC2 인스턴스 [*ec2_id*]의 Systems Manager 에이전트에 연결할 수 없습니다. 네트워크, 에이전트 및 IAM 권한을 올바르게 구성했는지 확인하세요.	SSM 에이전트는 항상 실행 중이어야 합니다. RDS Custom 에이전트는 Systems Manager 에이전트가 실행 중인지 확인하는 역할을 합니다. SSM 에이전트가 종료되었다가 다시 시작된 경우 RDS Custom 에이전트는 CloudWatch에 지표를 게시합니다. RDS Custom 에이전트에는 이전 3분 동안 재시작된 경우 경보가 트리거되도록 설정된 지표 경보가 지정되어 있습니다. 또한 지원 경계는 호스트의 SSM 에이전트 프로세스 상태를 30분마다 모니터링합니다. 자세한 내용은 SSM Agent 문제 해결을 참조하세요.

이벤트 ID	구성	RDS 이벤트 메시지	작업
SP-0200[illegible]	AWS Systems Manager 에이전트 (SSM 에이전트) 상태	EC2 인스턴스 [*ec2_id*]의 Systems Manager 에이전트가 여러 번 충돌했습니다. 자세한 내용은 SSM 에이전트 문제 해결 문서를 참조하세요.	자세한 내용은 SSM Agent 문제 해결을 참조하세요.

이벤트 ID	구성	RDS 이벤트 메시지	작업
SP-O2004	OS 시간대	EC2 인스턴스 [*ec2_id*]의 시간대가 변경되었습니다. 이 문제를 해결하려면 시간대를 이전 설정인 [*previous-time-zone*]으로 되돌리세요. 그런 다음 RDS 옵션 그룹을 사용하여 시간대를 변경합니다.	RDS 자동화가 옵션 그룹을 사용하지 않고 호스트의 시간대가 변경되었음을 감지했습니다. 이러한 호스트 수준 변경으로 인해 RDS 자동화가 실패하여 EC2 인스턴스가 `unsupported-configuration` 상태로 전환될 수 있습니다. 시간대 설정을 수정하려면 1. EC2 호스트에 로그인하고 다음과 같이 OS 시간대를 확인합니다. `timedatectl` 2. RDS Custom 자동화를 일시 중지합니다. 자세한 내용은 RDS Custom DB 인스턴스 일시 중지 및 재개 섹션을 참조하세요. 3. DB 인스턴스를 중지합니다. 4. 운영 체제의 시간대 변경을 되돌립니다. 5. DB 인스턴스를 시작합니다. 6. RDS Custom 자동화를 다시 시작합니다. DB 인스턴스를 30분 안에 사용할 수 있게 됩니다. 나중에 범위를 벗어나지 않도록 하려면 옵션 그룹을 통해 시간대를 수정하세요. 자세한 내용은 Oracle 시간대 섹션을 참조하세요.

이벤트 ID	구성	RDS 이벤트 메시지	작업
SP-O200	sudo 구성	EC2 인스턴스 [*ec2_id*]의 sudo 구성에 필요한 권한이 없습니다. 이 문제를 해결하려면 sudo 구성의 최근 변경 사항을 되돌리세요.	지원 경계는 특정 OS 사용자가 호스트에서 특정 명령을 실행할 수 있는지 확인합니다. sudo 구성을 모니터링하고 이를 지원되는 상태와 비교합니다. sudo 구성이 지원되지 않으면 RDS Custom은 해당 구성을 지원되는 이전 상태로 되돌려 덮어쓰려고 시도합니다. 시도가 성공하면 RDS Custom은 다음 알림을 보냅니다. RDS Custom이 구성을 성공적으로 덮어썼습니다. 덮어쓰기에 실패하면 DB 인스턴스가 지원되지 않는 구성 상태로 유지됩니다. 이 문제를 해결하려면 `sudoers.d/` 파일 내에서 변경 사항을 되돌리거나 권한을 수정합니다. **sudo** 구성의 변경 사항을 조사하는 방법 1. 호스트에 로그인합니다. 2. 다음 명령을 실행합니다. `visudo -c -f /etc/sudoers.d/ individual_sudo_files` 3. 필요에 따라 sudo 구성을 수정합니다. 지원 경계에서 지원되는 sudo 구성이 확인되면 30분 이내에 RDS Custom for Oracle DB 인스턴스를 사용할 수 있습니다.

이벤트 ID	구성	RDS 이벤트 메시지	작업
SP-O2006	S3 버킷 접근성	RDS Custom 자동화가 EC2 인스턴스 [*ec2_id*]의 S3 버킷에서 파일을 다운로드할 수 없습니다. 네트워킹 구성을 확인하고 인스턴스가 S3와의 연결을 허용하는지 확인하세요.	
SP-20	고가용성 소프트웨어 솔루션 버전	인스턴스의 HA 솔루션이 예상 버전과 다릅니다. 이 문제를 해결하려면 AWS Support 사례를 생성하세요.	AWS Support 사례를 만듭니다.

데이터베이스

이벤트 ID	구성	RDS 이벤트 메시지	작업
SP-O3001	데이터베이스 보관 지연 대상	EC2 인스턴스 [*ec2_id*]의 ARCHIVE_LAG_TARGET 파라미터가 권장 범위 *value_range* 를 벗어났습니다. 이 문제를 해결	지원 경계는 `ARCHIVE_LAG_TARGET` 데이터베이스 파라미터를 모니터링하여 DB 인스턴스의 최신 복원 가능 시간이 적절한 범위 내에 있는지 확인합니다. 아카이브된 다시 실행 로그의 지연 대상을 변경하려면 1. EC2 호스트에 로그인합니다. 2. RDS Custom for Oracle DB 인스턴스에 연결합니다.

이벤트 ID	구성	RDS 이벤트 메시지	작업
		하려면 파라미터를 value_range 내의 값으로 설정하세요.	3. ARCHIVE_LAG_TARGET 파라미터를 60~7,200 사이의 값으로 변경합니다. 예를 들어, 다음 SQL 문을 사용합니다. `ALTER SYSTEM SET ARCHIVE_LAG_TARGET=300 SCOPE=BOTH;` DB 인스턴스를 30분 안에 사용할 수 있게 됩니다.

이벤트 ID	구성	RDS 이벤트 메시지	작업

SP-O3002	Oracle Data Guard 역할	데이터베이스 역할 [*role_name*]이 EC2 인스턴스 [*ec2_id*]의 Oracle Data Guard에서 지원되지 않습니다. 이 문제를 해결하려면 DATABASE_ROLE 파라미터를 PRIMARY 또는 PHYSICAL STANDBY로 설정하세요.	지원 경계는 15초마다 현재 데이터베이스 역할을 모니터링하고 데이터베이스 역할이 변경된 경우 CloudWatch 알림을 보냅니다. Oracle Data Guard DATABASE_ROLE 파라미터는 PRIMARY 또는 PHYSICAL STANDBY여야 합니다.
			
			Oracle Data Guard 데이터베이스 역할을 지원되는 값으로 복원하려면
			
			1. 아래의 문을 실행하여 Oracle Data Guard 역할을 확인합니다.
			
			`SELECT DATABASE_ROLE FROM V$DATABASE;`
			
			2. DB 인스턴스가 독립 실행형인 경우 다음 문 중 하나를 사용하여 인스턴스를 다시 PRIMARY 역할로 변경합니다.
			
			`ALTER DATABASE COMMIT TO SWITCHOVER PRIMARY;`
			`ALTER DATABASE ACTIVATE STANDBY DATABASE;`
			
			DB 인스턴스가 복제본인 경우 다음 문을 사용하여 해당 인스턴스를 PHYSICAL STANDBY 역할로 다시 변경합니다.
			
			`ALTER DATABASE CONVERT TO PHYSICAL STANDBY;`
			
			지원 경계에서 데이터베이스 역할이 지원됨이 확인되면 15초 이내에 RDS Custom for Oracle DB 인스턴스를 사용할 수 있게 됩니다.

이벤트 ID	구성	RDS 이벤트 메시지	작업
SP-O3003	데이터베이스 상태	Oracle 데이터베이스의 SMON 프로세스가 좀비 상태입니다. 이 문제를 해결하려면 EC2 인스턴스 [*ec2_id*]에서 데이터베이스를 수동으로 복구하고 데이터베이스를 연 다음 즉시 백업하세요. 도움이 더 필요하면 지원에 문의하세요.	지원 경계는 DB 인스턴스 상태를 모니터링하며, 1시간 전과 하루 동안의 재시작 횟수를 모니터링합니다. 인스턴스가 여전히 존재하더라도 인스턴스와 상호 작용할 수 없는 경우 알림이 표시됩니다. 지원 경계에서 인스턴스 상태를 평가하도록 하려면 1. 호스트에 로그인하고 데이터베이스 상태를 파악합니다. ``` ps -eo pid,state,command | grep smon ``` 2. 필요한 경우 DB 인스턴스를 다시 시작합니다. 다시 시작에 실패하는 경우 다음 단계를 진행합니다. 3. 필요한 경우 EC2 호스트를 다시 시작합니다. DB 인스턴스가 다시 시작된 후 RDS Custom 에이전트는 DB 인스턴스가 더 이상 응답하지 않는 상태가 아님을 감지합니다. 지원 경계에 DB 인스턴스 상태를 재평가하도록 알립니다.

이벤트 ID	구성	RDS 이벤트 메시지	작업

SP-O3004	데이터베이스 로그 모드	EC2 인스턴스 [*ec2_id*]의 데이터베이스 로그 모드가 [*value_b*]로 변경되었습니다. 이 문제를 해결하려면 로그 모드를 [*value_a*]로 설정하세요.	DB 인스턴스 로그 모드를 **ARCHIVELOG** 로 변경하려면
			1. EC2 호스트에 로그인합니다.
			2. 데이터베이스에 연결하고 다음 문을 실행합니다.
			`SELECT LOG_MODE FROM V$DATABASE;`
			또는 SQL*Plus에서 다음 명령을 실행할 수 있습니다.
			`ARCHIVE LOG LIST`
			3. 다음 SQL*Plus 명령을 실행하여 일관성 있게 종료가 시작되도록 설정합니다.
			`SHUTDOWN IMMEDIATE`
			RDS Custom 에이전트가 DB 인스턴스를 자동으로 다시 시작하고 로그 모드를 ARCHIVELOG 로 설정합니다. DB 인스턴스를 30분 안에 사용할 수 있게 됩니다.

SP-O3005	Oracle home path	EC2 인스턴스 [*ec2_id*]의 Oracle home이 *new_path*로 변경되었습니다. 이 문제를 해결하려면 설정을 *old_path*로 되돌리세요.	

이벤트 ID	구성	RDS 이벤트 메시지	작업

SP-O3006	데이터베이스 고유 이름	EC2 인스턴스 [*ec2_id*]의 데이터베이스 고유 이름이 *new_value* 로 변경되었습니다. 이 문제를 해결하려면 이름을 *old_value* 로 되돌리세요.	DB 인스턴스의 데이터베이스 고유 이름을 변경하려면
			1. EC2 호스트에 로그인합니다.
			2. 데이터베이스에 연결하고 다음 문을 실행합니다.
			`SELECT DB_UNIQUE_NAME FROM V$DATABASE;`
			3. `ALTER SYSTEM SET DB_UNIQUE_NAME` 명령을 사용하여 원래 데이터베이스 고유 이름을 지정합니다.
			4. 다음 SQL 문을 실행하여 일관성 있게 종료가 시작되도록 설정합니다.
			`SHUTDOWN IMMEDIATE;`
			RDS Custom 에이전트가 DB 인스턴스를 자동으로 다시 시작하고 로그 모드를 `ARCHIVELOG` 로 설정합니다. DB 인스턴스를 30분 안에 사용할 수 있게 됩니다.

RDS Custom for Oracle 업그레이드 문제 해결

RDS Custom for Oracle 인스턴스 업그레이드에 실패할 수 있습니다. 아래에는 RDS Custom DB for Oracle DB 인스턴스를 업그레이드하는 중에 사용할 수 있는 몇 가지 중요한 기술, 파일 및 명령이 나와 있습니다.

- DB 인스턴스의 `/tmp` 디렉터리에 있는 업그레이드 출력 로그 파일을 검토합니다. 로그 이름은 DB 엔진 버전에 따라 다릅니다. 예를 들어 `catupgrd` 또는 `catup` 문자열이 포함된 로그가 표시될 수 있습니다.
- `/rdsdbdata/log/trace` 디렉터리에 있는 `alert.log` 파일을 검토합니다.
- `root` 디렉터리에서 다음 `grep` 명령을 실행하여 업그레이드 OS 프로세스를 추적합니다. 이 명령은 로그 파일이 기록되는 위치를 표시하고 업그레이드 프로세스의 상태를 확인합니다.

```
ps -aux | grep upg
```

다음은 샘플 출력을 보여줍니다.

```
root     18884  0.0  0.0 235428  8172 ?        S<   17:03   0:00 /usr/bin/
sudo -u rdsdb /rdsdbbin/scripts/oracle-control ORCL op_apply_upgrade_sh RDS-
UPGRADE/2.upgrade.sh
rdsdb    18886  0.0  0.0 153968 12164 ?        S<   17:03   0:00 /usr/bin/perl -T -
w /rdsdbbin/scripts/oracle-control ORCL op_apply_upgrade_sh RDS-UPGRADE/2.upgrade.sh
rdsdb    18887  0.0  0.0 113196  3032 ?        S<   17:03   0:00 /bin/sh /rdsdbbin/
oracle/rdbms/admin/RDS-UPGRADE/2.upgrade.sh
rdsdb    18900  0.0  0.0 113196  1812 ?        S<   17:03   0:00 /bin/sh /rdsdbbin/
oracle/rdbms/admin/RDS-UPGRADE/2.upgrade.sh
rdsdb    18901  0.1  0.0 167652 20620 ?        S<   17:03   0:07 /rdsdbbin/oracle/
perl/bin/perl catctl.pl -n 4 -d /rdsdbbin/oracle/rdbms/admin -l /tmp catupgrd.sql
root     29944  0.0  0.0 112724  2316 pts/0    S+   18:43   0:00 grep --color=auto
 upg
```

- 다음 SQL 쿼리를 실행하여 구성 요소의 현재 상태를 확인하고 데이터베이스 버전 및 DB 인스턴스에 설치된 옵션을 찾습니다.

```
SET LINESIZE 180
COLUMN COMP_ID FORMAT A15
COLUMN COMP_NAME FORMAT A40 TRUNC
COLUMN STATUS FORMAT A15 TRUNC
SELECT COMP_ID, COMP_NAME, VERSION, STATUS FROM DBA_REGISTRY ORDER BY 1;
```

다음과 유사하게 출력됩니다.

```
COMP_NAME                                STATUS               PROCEDURE
---------------------------------------- --------------------
 ------------------------------------------------
Oracle Database Catalog Views            VALID
 DBMS_REGISTRY_SYS.VALIDATE_CATALOG
Oracle Database Packages and Types       VALID
 DBMS_REGISTRY_SYS.VALIDATE_CATPROC
Oracle Text                              VALID                VALIDATE_CONTEXT
Oracle XML Database                      VALID                DBMS_REGXDB.VALIDATEXDB

4 rows selected.
```

- 다음 SQL 쿼리를 실행하여 업그레이드 프로세스를 방해할 수 있는 유효하지 않은 객체가 있는지 확인합니다.

```
SET PAGES 1000 LINES 2000
COL OBJECT FOR A40
SELECT SUBSTR(OWNER,1,12) OWNER,
       SUBSTR(OBJECT_NAME,1,30) OBJECT,
       SUBSTR(OBJECT_TYPE,1,30) TYPE, STATUS,
       CREATED
FROM   DBA_OBJECTS
WHERE  STATUS <>'VALID'
AND    OWNER IN ('SYS','SYSTEM','RDSADMIN','XDB');
```

RDS Custom for Oracle 복제본 승격 문제 해결

콘솔, `promote-read-replica` AWS CLI 명령 또는 `PromoteReadReplica` API를 사용하여 RDS Custom for Oracle의 관리형 Oracle 복제본을 승격할 수 있습니다. 기본 DB 인스턴스를 삭제하고 모든 복제본이 정상이면 RDS Custom for Oracle은 관리형 복제본을 독립 실행형 인스턴스로 자동 승격합니다. 복제본이 자동화를 일시 중지했거나 지원 경계 밖에 있는 경우 RDS Custom이 자동으로 승격할 수 있으려면 먼저 복제본을 수정해야 합니다. 자세한 내용은 RDS Custom for Oracle 복제본을 독립 실행형 DB 인스턴스로 승격 섹션을 참조하세요.

다음 상황에서는 복제본 프로모션 워크플로가 중단될 수 있습니다.

- 기본 DB 인스턴스가 `STORAGE_FULL` 상태입니다.
- 기본 데이터베이스는 모든 온라인 다시 실행 로그를 아카이브할 수 없습니다.
- Oracle 복제본에서 아카이브된 다시 실행 로그 파일과 기본 데이터베이스 간에 차이가 있습니다.

중단된 워크플로에 응답하려면 다음과 같이 하세요.

1. Oracle 복제본 DB 인스턴스의 다시 실행 로그 간격을 동기화합니다.
2. 읽기 전용 복제본을 적용된 최신 다시 실행 로그로 강제 승격시킵니다. SQL*Plus에서 다음 명령을 실행합니다.

```
ALTER DATABASE ACTIVATE STANDBY DATABASE;
SHUTDOWN IMMEDIATE
STARTUP
```

3. 지원에 문의하여 DB 인스턴스를 `available` 상태로 전환하도록 요청하세요.

Amazon RDS Custom for Oracle의 알려진 문제

RDS Custom for Oracle 작업을 수행할 경우 DB 인스턴스와 관련하여 다음과 같은 문제에 유의하세요.

- 루트 또는 dbbin 볼륨의 크기 조정은 지원되지 않습니다.

  > **Warning**
  >
  > 루트 또는 dbbin 볼륨의 크기를 수동으로 조정하지 않는 것이 좋습니다. 패치 적용 후에도 지속되는 데이터 볼륨에 모든 구성을 저장하고 RDS 스토리지 크기 조정 API만 사용하여 볼륨 크기를 조정하는 것이 좋습니다.

- 일부 RDS API는 Oracle Linux 7을 사용하는 AMI와 같이 데이터베이스 인스턴스가 이전 AMI에 있는 경우 차단될 수 있습니다. 이 문제를 해결하려면 OS 패치를 사용하여 DB 인스턴스를 최신 AMI로 패치하세요. 자세한 내용은 CEV 업그레이드 옵션 단원을 참조하십시오.
- RDS 작업을 수행하기 전에 AWS 계정에 컴퓨팅 및 스토리지에 대한 할당량이 충분한지 확인합니다.
- 데이터베이스가 생성 상태이고 데이터베이스 또는 Amazon EC2 호스트에 적극적으로 로그인하여 명령을 실행하는 경우 데이터베이스 만들기가 완료되지 않을 수 있습니다.
- 제어 파일 멀티플렉싱은 읽기 전용 복제본 문제 때문에 현재 지원되지 않습니다. 읽기 전용 복제본을 만들기 전에 소스 데이터베이스의 `CONTROL_FILES` 초기화 파라미터에 파일 이름을 하나만 지정해야 합니다.
- 데이터베이스 모드를 `PHYSICAL STANDBY`(탑재됨 또는 읽기 전용)에서 `SNAPSHOT STANDBY`(읽기/쓰기로 변환)로 변경할 수 없습니다.
- AWS 계정이 서비스 제어 정책(SCP)이 있는 AWS 조직의 일부이고 SCP에 조건 키가 포함된 경우 RDS Custom for Oracle DB 인스턴스가 다음 오류와 함께 만들어지지 않을 수 있습니다.

```
You can't create the DB instance because of incompatible resources.
The IAM instance profile role [AWSRDSCustomInstanceRole1-us-east-1] is missing the
 following permissions:
EFFECT [Allow] on ACTION(S) [ssm:DescribeAssociation, ssm:DescribeDocument,
 ssm:GetConnectionStatus,
  ssm:GetDeployablePatchSnapshotForInstance, ssmmessages:OpenControlChannel,
 ssm:GetParameters,
  ssm:ListInstanceAssociations, ssm:PutConfigurePackageResult,
 ssmmessages:CreateControlChannel,
  ssm:GetParameter, ssm:UpdateAssociationStatus, ssm:GetManifest,
 ssmmessages:CreateDataChannel,
  ssm:PutInventory, ssm:UpdateInstanceInformation, ssm:DescribeInstanceInformation,
```

```
 ssmmessages:OpenDataChannel, ssm:GetDocument, ssm:ListAssociations,
ssm:PutComplianceItems,
 ssm:UpdateInstanceAssociationStatus] for RESOURCE(S) [], EFFECT [Allow] on
ACTION(S) [ec2messages:DeleteMessage,
 ec2messages:FailMessage, ec2messages:GetEndpoint, ec2messages:AcknowledgeMessage,
ec2messages:GetMessages,
 ec2messages:SendReply] for RESOURCE(S) [], EFFECT [Allow] on ACTION(S)
[logs:CreateLogStream,
 logs:DescribeLogStreams, logs:PutRetentionPolicy, logs:PutLogEvents]
```

이 문제를 해결하려면 지원에 티켓을 만듭니다.

데이터베이스 사용자 계정의 알려진 문제

다음 사항에 주의하세요.

- RDSADMIN 및 RDS_DATAGUARD와 같이 문자열 RDS로 시작하는 데이터베이스 사용자 계정은 제거하지 마세요. RDS Custom for Oracle은 자동화를 위해 RDS 계정을 사용합니다. 이 사용자 계정을 제거하면 RDS Custom이 인스턴스를 지원되지 않는 구성 상태로 이동합니다.
- RDS Custom for Oracle DB 인스턴스의 마스터 사용자 이름은 `ModifyDBInstance` API를 사용하여 변경할 수 없습니다.
- RDS Custom for Oracle은 모든 DB 인스턴스에서 사용자 계정 자격 증명을 교체합니다. 자세한 내용은 규정 준수 프로그램을 위한 RDS Custom for Oracle 자격 증명 단원을 참조하십시오. 온프레미스 기본/대기 구성을 사용하는 경우 자격 증명 교체가 다음 리소스에 영향을 미칠 수 있습니다.
  - 수동으로 만든 대기 RDS Custom for Oracle 인스턴스

    이 문제를 해결하려면 수동 대기 데이터베이스를 삭제한 다음 API 직접 호출을 사용하여 Oracle 읽기 전용 복제본을 만듭니다. 수동 대기 데이터베이스의 보안 암호를 수동으로 관리하여 소스 DB 인스턴스와 일치하도록 합니다.
  - 수동으로 만든 교차 리전 읽기 전용 복제본

    이 문제를 해결하려면 기본 DB 인스턴스와 일치하도록 보안 암호를 수동으로 유지합니다.

파라미터 및 구성 파일의 알려진 문제

- 조정 컴퓨팅, OS 업그레이드 및 RDS Custom이 루트 볼륨을 대체하는 기타 작업을 수행한 후에 `crontab` 파일을 구성해야 합니다. `crontab`을 백업해 두는 것이 좋습니다.
- `listener.ora` 파일을 구성할 때 다음 지침을 참고하세요.

- 파일의 모든 항목이 한 줄에 있는지 확인합니다. 이 접근 방식은 인스턴스 만들기 중 들여쓰기 문제를 방지합니다.
- GLOBAL_DBNAME이 SID_NAME의 값과 동일한지 확인합니다.
- LISTENER의 값이 L_*dbname*_001 명명 규칙을 따르는지 확인합니다.
- listener.ora 파일이 데이터베이스 이름에 대한 연결을 유지하는지 확인합니다. RDS Custom은 이 연결을 사용하여 데이터베이스 시작을 확인합니다. 이 파일을 잘못 수정하면 조정 컴퓨팅 또는 패치 적용과 같은 작업이 실패할 수 있습니다.

  다음 예는 올바르게 구성된 listener.ora를 보여줍니다.

```
ADR_BASE_L_ORCL_001=/rdsdbdata/log/
USE_SID_AS_SERVICE_L_ORCL_001=ON
SID_LIST_L_ORCL_001=(SID_LIST = (SID_DESC = (SID_NAME = ORCL)(GLOBAL_DBNAME = ORCL)
 (ORACLE_HOME = /rdsdbbin/oracle.19.custom.r1.EE.1)))
SUBSCRIBE_FOR_NODE_DOWN_EVENT_L_ORCL_001=OFF
L_ORCL_001=(DESCRIPTION_LIST = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(PORT =
 XXXX)(HOST = x.x.x.x))) (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(PORT = XXXX)
(HOST = 127.0.0.1))))
```

- 서버 파라미터 파일이나 초기화 파라미터 파일에서는 설명이 지원되지 않습니다.
- 서버 파라미터 파일(/rdsdbdata/config/oracle_pfile)에서 다음 초기화 파라미터를 선언해야 합니다.
  - MEMORY_MAX_TARGET
  - MEMORY_TARGET
  - PGA_AGGREGATE_TARGET
  - PROCESSES
  - SGA_TARGET
  - USE_LARGE_PAGES

  이전 파라미터가 /rdsdbdata/config/oracle_pfile에서 선언되지 않은 경우 읽기 전용 복제본 생성 및 컴퓨팅 규모 조정이 실패할 수 있습니다.
- 서버 파라미터 파일, 감사 파일,listener.ora, tnsnames.ora 또는 sqlnet.ora 같은 구성 파일의 기호 링크는 삭제할 수 없습니다. 또한 이러한 파일의 디렉터리 구조도 수정할 수 없습니다. RDS Custom 자동화에서는 이러한 파일이 특정 디렉터리 구조에 존재할 것으로 예상합니다.

  초기화 파라미터 파일에서 서버 파라미터 파일을 생성하려면 다음 구문을 사용합니다.

```
CREATE SPFILE='/rdsdbdata/admin/$ORACLE_SID/pfile/spfile$ORACLE_SID.ora'
    FROM PFILE='/rdsdbdata/config/oracle_pfile';
```

RDS Custom for SQL Server 작업

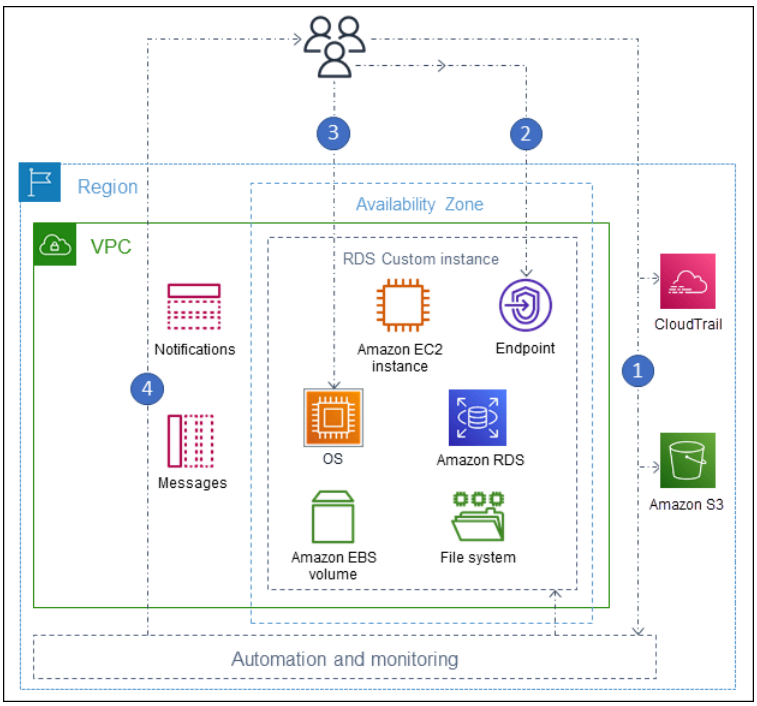
아래에서는 RDS Custom for SQL Server DB 인스턴스의 관리 및 유지 관리에 대한 지침을 확인할 수 있습니다.

주제

- RDS for SQL Server 워크플로
- Amazon RDS Custom for SQL Server 요구 사항 및 제한
- Amazon RDS Custom for SQL Server를 위한 환경 설정
- RDS Custom for SQL Server에서 기존 보유 미디어 사용(BYOM)
- RDS Custom for SQL Server 사용자 지정 엔진 버전 작업
- Amazon RDS Custom for SQL Server의 DB 인스턴스 생성 및 연결
- Amazon RDS Custom for SQL Server DB 인스턴스 관리
- RDS Custom for SQL Server를 사용하여 Microsoft Active Directory 작업
- RDS Custom for SQL Server에 대한 다중 AZ 배포 구성 및 관리
- Amazon RDS Custom for SQL Server DB 인스턴스 백업 및 복원
- Amazon RDS Custom for SQL Server DB 스냅샷 복사
- 온프레미스 데이터베이스를 SQL Server용 Amazon RDS Custom으로 마이그레이션
- Amazon RDS Custom for SQL Server DB 인스턴스 업그레이드
- Amazon RDS Custom for SQL Server의 DB 문제 해결

RDS for SQL Server 워크플로

다음 다이어그램은 RDS Custom for SQL Server의 일반적인 워크플로를 보여 줍니다.



단계는 다음과 같습니다.

1. RDS Custom에서 제공하는 엔진 버전에서 RDS Custom for SQL Server DB 인스턴스를 생성합니다.

   자세한 내용은 RDS Custom for SQL Server DB 인스턴스 생성을 참조하세요.
2. 애플리케이션을 RDS Custom DB 인스턴스 엔드포인트에 연결합니다.

   자세한 내용은 AWS Systems Manager를 사용하여 RDS Custom DB 인스턴스에 연결 및 RDP를 사용하여 RDS Custom DB 인스턴스에 연결 단원을 참조하세요.
3. (선택 사항) 호스트에 액세스하여 소프트웨어를 커스터마이징합니다.
4. RDS Custom 자동화로 생성된 알림 및 메시지를 모니터링합니다.

DB instance for RDS Custom 생성

`create-db-instance` 명령을 사용하여 RDS Custom DB 인스턴스를 생성할 수 있습니다. 생성 절차는 Amazon RDS 인스턴스를 생성과 유사합니다. 하지만 일부 파라미터는 다릅니다. 자세한 내용은 Amazon RDS Custom for SQL Server의 DB 인스턴스 생성 및 연결을 참조하세요.

데이터베이스 연결

Amazon RDS DB 인스턴스와 마찬가지로 RDS Custom for SQL Server DB 인스턴스는 VPC에 있습니다. 애플리케이션은 RDS Custom for SQL Server와 마찬가지로 SQL 서버 관리 제품군(SSMS)과 같은 클라이언트를 사용하여 RDS Custom 인스턴스에 연결합니다.

RDS Custom 커스터마이징

RDS Custom 호스트에 액세스하여 소프트웨어를 설치하거나 커스터마이징할 수 있습니다. 변경 사항과 RDS Custom 자동화 간의 충돌을 방지하려면 지정된 기간 동안 자동화를 일시 중지하면 됩니다. 이 기간 동안 RDS Custom은 모니터링 또는 인스턴스 복구를 수행하지 않습니다. 기간이 만료되면 RDS Custom은 전체 자동화를 재개합니다. 자세한 내용은 RDS Custom 자동화 일시 중지 및 다시 시작을 참조하세요.

Amazon RDS Custom for SQL Server 요구 사항 및 제한

다음에서 빠른 참조를 위해 Amazon RDS Custom for SQL Server 요구 사항 및 제한 사항에 대한 요약을 확인할 수 있습니다. 요구 사항 및 제한 사항은 관련 섹션에도 표시됩니다.

주제

- 리전 및 버전 사용 가능 여부
- RDS Custom for SQL Server 일반 요구 사항
- RDS Custom for SQL Server DB 인스턴스 클래스 지원
- RDS Custom for SQL Server 제한 사항
- RDS Custom for SQL Server DB 인스턴스에 대한 문자 집합 및 데이터 정렬 설정
- RDS Custom for SQL Server DB 인스턴스의 현지 시간대
- RDS Custom for SQL Server에서 Service Master Key 사용
- RDS Custom for SQL Server에서 변경 데이터 캡처(CDC) 지원

리전 및 버전 사용 가능 여부

기능 가용성 및 해당 지원은 각 데이터베이스 엔진의 특정 버전 및 AWS 리전마다 다릅니다. Amazon RDS Custom for SQL Server에서 사용할 수 있는 Amazon RDS의 버전 및 리전에 대한 자세한 내용은 RDS Custom for SQL Server를 지원하는 리전 및 DB 엔진 단원을 참조하세요.

RDS Custom for SQL Server 일반 요구 사항

Amazon RDS Custom for Server에 대한 다음 요구 사항을 준수해야 합니다.

- RDS Custom for SQL Server DB 인스턴스 클래스 지원에 표시된 인스턴스 클래스를 사용합니다. 지원되는 스토리지 유형은 gp2, gp3, io1 및 io2 Block Express 유형의 SSD(Solid State Drive)뿐입니다. io1, gp2, gp3의 최대 스토리지 한도는 16TiB인 반면 io2는 64TiB를 지원합니다.
- RDS Custom DB instance 생성을 위해 대칭 암호화 AWS KMS 키를 가지고 있는지 확인하세요. 자세한 내용은 대칭 암호화 AWS KMS 키 보유 여부 확인을 참조하세요.
- AWS Identity and Access Management (IAM) 역할과 인스턴스 프로파일을 생성해야 합니다. 자세한 내용은 수동으로 IAM 역할과 인스턴스 프로파일 생성 및 AWS Management Console을 사용한 자동 인스턴스 프로파일 생성(을)를 참조하세요.
- RDS Custom이 다른 AWS 서비스에 액세스하는 데 사용할 수 있는 네트워킹 구성을 제공해야 합니다. 특정 요구 사항은 2단계: 네트워킹, 인스턴스 프로파일 및 암호화 구성 단원을 참조하세요.

- RDS Custom과 Amazon RDS DB 인스턴스의 합산 수가 할당량 한도를 초과할 수 없습니다. 예를 들어, 할당량이 40개의 DB 인스턴스인 경우 RDS Custom for Server DB 인스턴스 20개와 Amazon RDS DB 인스턴스 20개를 보유할 수 있습니다.
- RDS Custom은 이름이 `do-not-delete-rds-custom-`으로 시작하는 AWS CloudTrail 추적을 자동으로 생성합니다. RDS Custom 지원 경계는 CloudTrail의 이벤트에 따라 작업이 RDS Custom 자동화에 영향을 미치는지 여부를 결정합니다. RDS Custom은 첫 번째 DB 인스턴스를 생성할 때 추적을 생성합니다. 기존 CloudTrail을 사용하려면 AWS Support에 문의하세요. 자세한 내용은 AWS CloudTrail 섹션을 참조하세요.

RDS Custom for SQL Server DB 인스턴스 클래스 지원

describe-orderable-db-instance-options 명령을 사용하여 리전에서 DB 인스턴스 클래스가 지원되는지 확인합니다.

RDS Custom for SQL Server는 다음 테이블에 나와 있는 DB 인스턴스 클래스를 지원합니다.

SQL Server 에디션	RDS 고객 지원
Enterprise Edition	db.r5.large-db.r5.24xlarge db.r5b.xlarge–db.r5b.24xlarge db.m5.xlarge-db.m5.24xlarge db.r6i.xlarge–db.r6i.32xlarge db.m6i.xlarge–db.m6i.32xlarge db.x2iedn.xlarge–db.x2iedn.32xlarge
Standard Edition	db.r5.large-db.r5.24xlarge db.r5b.large–db.r5b.8xlarge db.m5.large-db.m5.24xlarge db.r6i.large–db.r6i.8xlarge

SQL Server 에디션	RDS 고객 지원
	db.m6i.large–db.m6i.8xlarge db.x2iedn.xlarge–db.x2iedn.8xlarge
Developer Edition	db.r5.large-db.r5.24xlarge db.r5b.xlarge–db.r5b.24xlarge db.m5.xlarge-db.m5.24xlarge db.r6i.xlarge–db.r6i.32xlarge db.m6i.xlarge–db.m6i.32xlarge db.x2iedn.xlarge–db.x2iedn.32xlarge
Web Edition	db.r5.large–db.r5.4xlarge db.m5.large-db.m5.4xlarge db.r6i.large–db.r6i.4xlarge db.m6i.large–db.m6i.4xlarge db.r5b.large–db.r5b.4xlarge

다음 권장 사항은 db.x2iedn 클래스 유형에 적용됩니다.

- 생성 시 로컬 스토리지는 원시 및 할당되지 않은 디바이스입니다. 이 인스턴스 클래스와 함께 DB 인스턴스를 사용하기 전에 먼저 로컬 스토리지를 마운트하고 포맷해야 합니다. 그런 다음 최적의 성능을 보장하기 위해 `tempdb`를 구성합니다. 자세한 내용은 Optimize tempdb performance in Amazon RDS Custom for SQL Server using local instance storage를 참조하세요.
- 컴퓨팅 규모 조정, 인스턴스 교체, 스냅샷 복원 또는 시점 복구(PITR)와 같은 DB 인스턴스 작업을 실행하면 로컬 스토리지는 원시 및 할당되지 않은 상태로 되돌아갑니다. 이러한 상황에서는 드라이브를 다시 마운트하고, 다시 포맷하고, 재구성하고, `tempdb`를 사용하여 기능을 복원해야 합니다.

- 다중 AZ 인스턴스의 경우 대기 DB 인스턴스에서 구성을 수행하는 것이 좋습니다. 이렇게 하면 구성이 대기 인스턴스에 이미 설정되어 있기 때문에 장애 조치가 발생하더라도 시스템이 문제 없이 계속 작동합니다.

RDS Custom for SQL Server 제한 사항

RDS Custom for SQL Server에는 다음과 같은 제한 사항이 적용됩니다.

- Amazon RDS for RDS Custom for SQL Server DB 인스턴스에서는 읽기 전용 복제본을 생성할 수 없습니다. 그러나 다중 AZ 배포를 사용하여 고가용성을 자동으로 구성할 수 있습니다. 자세한 내용은 RDS Custom for SQL Server에 대한 다중 AZ 배포 구성 및 관리 섹션을 참조하세요.
- 기존 RDS Custom for SQL Server DB 인스턴스의 DB 인스턴스 식별자는 수정할 수 없습니다.
- 사용자 지정 엔진 버전(CEV)을 사용하여 생성되지 않은 RDS Custom for SQL Server DB 인스턴스의 경우 Microsoft Windows 운영 체제에 대한 변경의 지속성을 보장할 수 없습니다. 예를 들어 스냅샷 또는 특정 시점 복원 작업을 시작하면 이러한 변경 내용이 손실됩니다. RDS Custom for SQL Server DB 인스턴스가 CEV를 사용하여 생성된 경우에는 이러한 변경 내용이 지속됩니다.
- 일부 옵션은 지원되지 않습니다. 예를 들어 RDS Custom for SQL Server DB 인스턴스를 생성할 때 다음과 같은 작업을 수행할 수 없습니다.
  - DB 인스턴스 클래스의 코어당 CPU 코어 및 스레드 수를 변경합니다.
  - 스토리지 자동 조정을 활성화합니다.
  - 고유한 DB 파라미터 그룹, 옵션 그룹 또는 문자 집합을 지정합니다.
  - 성능 개선 도우미를 활성화합니다.
  - 마이너 버전 자동 업그레이드를 활성화합니다.
- 최대 DB 인스턴스 스토리지는 64TiB입니다.
- RDS Custom for SQL Server에는 RDS 프록시를 사용할 수 없습니다.
- RDS Custom for SQL Server DB 인스턴스에는 `describe-reserved-db-instances` API를 사용할 수 없습니다.

RDS Custom for SQL Server DB 인스턴스에 대한 문자 집합 및 데이터 정렬 설정

개요

RDS Custom for SQL Server DB 인스턴스를 사용하면 데이터를 저장하고 정렬하는 방법을 결정하는 문자 집합 및 데이터 정렬 설정을 구성할 수 있습니다. 문자 집합은 허용되는 문자를 정의하는 반면, 데이터 정렬은 데이터 정렬 및 비교를 위한 규칙을 지정합니다. 다국어 데이터로 작업하거나 특정 정렬

요구 사항이 있는 애플리케이션에 적절한 문자 집합과 데이터 정렬을 설정하는 것이 중요합니다. 예를 들어 액센트 표시된 문자를 처리하고 언어별 정렬 규칙을 정의하거나 서로 다른 로캘 전반에서 데이터 무결성을 유지해야 할 수 있습니다. 다음 섹션에서는 RDS Custom for SQL Server DB 인스턴스의 문자 집합 및 데이터 정렬 지원에 대한 정보를 제공합니다.

RDS Custom for SQL Server는 SQL_라틴, 일본어, 독일어 및 아랍어 로케일의 기존 인코딩과 UTF-8 인코딩 모두에서 다양한 서버 데이터 정렬을 지원합니다. 기본 서버 데이터 정렬은 `SQL_Latin1_General_CP1_CI_AS`이지만 지원되는 다른 데이터 정렬을 선택하여 사용할 수 있습니다. RDS for SQL Server에서 사용하는 것과 동일한 절차를 사용하여 데이터 정렬을 선택할 수 있습니다. 자세한 내용은 Amazon RDS for Microsoft SQL Server의 데이터 정렬 및 문자 세트 관리 단원을 참조하십시오.

고려 사항

RDS Custom for SQL Server에서 서버 데이터 정렬을 사용할 때는 다음과 같은 요구 사항 및 제한 사항이 적용됩니다.

- RDS Custom for SQL Server DB 인스턴스를 생성할 때 서버 데이터 정렬을 설정할 수 있습니다. DB 인스턴스를 생성한 후에는 서버 수준 데이터 정렬을 수정할 수 없습니다.
- DB 스냅샷에서 복원하거나 시점 복구(PITR)를 수행할 때는 서버 수준 데이터 정렬을 수정할 수 없습니다.
- RDS Custom for SQL Server CEV에서 DB 인스턴스를 생성할 때 DB 인스턴스는 CEV의 서버 데이터 정렬을 상속하지 않습니다. 대신 기본 서버 데이터 정렬인 `SQL_Latin1_General_CP1_CI_AS`가 사용됩니다. RDS Custom for SQL Server CEV에서 기본이 아닌 서버 데이터 정렬을 구성한 후 새 DB 인스턴스에서도 동일한 서버 데이터 정렬을 사용하려면 CEV에서 DB 인스턴스를 생성할 때 동일한 데이터 정렬을 선택해야 합니다.

> **Note**
>
> DB 인스턴스를 생성할 때 선택한 데이터 정렬이 CEV의 데이터 정렬과 다른 경우, 새 RDS Custom for SQL Server DB 인스턴스의 Microsoft SQL Server 시스템 데이터베이스가 업데이트된 데이터 정렬을 사용하도록 재구축됩니다. 재구축 프로세스는 새 RDS Custom for SQL Server DB 인스턴스에서만 수행되며 CEV 자체에는 영향을 주지 않습니다. CEV의 시스템 데이터베이스에 대한 이전 수정 사항은 시스템 데이터베이스가 재구축된 후 새 RDS Custom for SQL Server DB 인스턴스에 유지되지 않습니다. 일부 수정 사항의 예로는 `master` 데이터베이스의 사용자 정의 객체, `msdb` 데이터베이스의 예약된 작업 또는 CEV

model 데이터베이스의 기본 데이터베이스 설정 변경 등이 있습니다. 새 RDS Custom for SQL Server DB 인스턴스가 생성된 후 수정 내용을 수동으로 다시 생성할 수 있습니다.

- RDS Custom for SQL Server 사용자 지정 엔진 버전(CEV)에서 DB 인스턴스를 생성하고 CEV와 다른 데이터 정렬을 선택하는 경우, 새 DB 인스턴스에서 Microsoft SQL Server 시스템 데이터베이스를 재구축할 수 있도록 CEV 생성에 사용되는 골든 이미지(AMI)가 다음 요구 사항을 충족하는지 확인합니다.
  - SQL Server 2022의 경우 setup.exe 파일이 C:\Program Files\Microsoft SQL Server\160\Setup Bootstrap\SQL2022\setup.exe 경로에 있는지 확인합니다.
  - SQL Server 2019의 경우 setup.exe 파일이 다음 경로에 있는지 확인합니다. C:\Program Files\Microsoft SQL Server\150\Setup Bootstrap\SQL2019\setup.exe
  - master, model 및 msdb 데이터베이스의 데이터 및 로그 템플릿 사본은 기본 위치에 있어야 합니다. 자세한 정보는 Microsoft 공개 문서의 시스템 데이터베이스 재구축을 참조하세요.
  - SQL Server 데이터베이스 엔진이 NT Service\MSSQLSERVER 또는 NT AUTHORITY\NETWORK SERVICE를 서비스 계정으로 사용하는지 확인합니다. DB 인스턴스에 기본이 아닌 서버 데이터 정렬을 구성할 때 다른 계정은 C:\ 드라이브에 대한 필수 권한을 갖지 않게 됩니다.
- 새 DB 인스턴스에 대해 선택한 서버 데이터 정렬이 CEV에 구성된 데이터 정렬과 동일한 경우, 새 RDS Custom for SQL Server DB 인스턴스의 Microsoft SQL Server 시스템 데이터베이스는 재구축 프로세스를 거치지 않습니다. CEV의 시스템 데이터베이스에 대한 이전 수정 사항은 자동으로 새 RDS Custom for SQL Server DB 인스턴스에 적용됩니다.

지원되는 데이터 정렬

데이터 정렬을 다음 표에 나열된 값 중 하나로 설정할 수 있습니다.

콜레이션	설명
Arabic_100_BIN	Arabic-100, 바이너리 정렬
Arabic_100_BIN2	Arabic-100, 이진 코드 포인트 비교 정렬
Arabic_100_CI_AI	Arabic-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분
Arabic_100_CI_AI_KS	Arabic-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분

콜레이션	설명
Arabic_100_CI_AI_KS_SC	Arabic-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자
Arabic_100_CI_AI_KS_SC_UTF8	Arabic-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, UTF8
Arabic_100_CI_AI_KS_WS	Arabic-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분
Arabic_100_CI_AI_KS_WS_SC	Arabic-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자
Arabic_100_CI_AI_KS_WS_SC_UTF8	Arabic-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, UTF8
Arabic_100_CI_AI_SC	Arabic-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자
Arabic_100_CI_AI_SC_UTF8	Arabic-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, UTF8
Arabic_100_CI_AI_WS	Arabic-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분
Arabic_100_CI_AI_WS_SC	Arabic-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자
Arabic_100_CI_AI_WS_SC_UTF8	Arabic-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, UTF8
Arabic_100_CI_AS	Arabic-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분

콜레이션	설명
Arabic_100_CI_AS_KS	Arabic-100, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분
Arabic_100_CI_AS_KS_SC	Arabic-100, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자
Arabic_100_CI_AS_KS_SC_UTF8	Arabic-100, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, UTF8
Arabic_100_CI_AS_KS_WS	Arabic-100, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분
Arabic_100_CI_AS_KS_WS_SC	Arabic-100, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자
Arabic_100_CI_AS_KS_WS_SC_UTF8	Arabic-100, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, UTF8
Arabic_100_CI_AS_SC	Arabic-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자
Arabic_100_CI_AS_SC_UTF8	Arabic-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, UTF8
Arabic_100_CI_AS_WS	Arabic-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분
Arabic_100_CI_AS_WS_SC	Arabic-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자
Arabic_100_CI_AS_WS_SC_UTF8	Arabic-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, UTF8
Arabic_100_CS_AI	Arabic-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분

콜레이션	설명
Arabic_100_CS_AI_KS	Arabic-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분
Arabic_100_CS_AI_KS_SC	Arabic-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자
Arabic_100_CS_AI_KS_SC_UTF8	Arabic-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, UTF8
Arabic_100_CS_AI_KS_WS	Arabic-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분
Arabic_100_CS_AI_KS_WS_SC	Arabic-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자
Arabic_100_CS_AI_KS_WS_SC_UTF8	Arabic-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, UTF8
Arabic_100_CS_AI_SC	Arabic-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자
Arabic_100_CS_AI_SC_UTF8	Arabic-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, UTF8
Arabic_100_CS_AI_WS	Arabic-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분
Arabic_100_CS_AI_WS_SC	Arabic-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자
Arabic_100_CS_AI_WS_SC_UTF8	Arabic-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, UTF8
Arabic_100_CS_AS	Arabic-100, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분

콜레이션	설명
Arabic_100_CS_AS_KS	Arabic-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분
Arabic_100_CS_AS_KS_SC	Arabic-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자
Arabic_100_CS_AS_KS_SC_UTF8	Arabic-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, UTF8
Arabic_100_CS_AS_KS_WS	Arabic-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분
Arabic_100_CS_AS_KS_WS_SC	Arabic-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자
Arabic_100_CS_AS_KS_WS_SC_UTF8	Arabic-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, UTF8
Arabic_100_CS_AS_SC	Arabic-100, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자
Arabic_100_CS_AS_SC_UTF8	Arabic-100, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, UTF8
Arabic_100_CS_AS_WS	Arabic-100, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분
Arabic_100_CS_AS_WS_SC	Arabic-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자
Arabic_100_CS_AS_WS_SC_UTF8	Arabic-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, UTF8
Arabic_BIN	아랍어, 이진 정렬
Arabic_BIN2	아랍어, 이진 코드 포인트 비교 정렬

콜레이션	설명
Arabic_CI_AI	아랍어, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분
Arabic_CI_AI_KS	아랍어, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분
Arabic_CI_AI_KS_WS	아랍어, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분
Arabic_CI_AI_WS	아랍어, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분
Arabic_CI_AS	아랍어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Arabic_CI_AS_KS	아랍어, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분
Arabic_CI_AS_KS_WS	아랍어, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분
Arabic_CI_AS_WS	아랍어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분
Arabic_CS_AI	아랍어, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분
Arabic_CS_AI_KS	아랍어, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분
Arabic_CS_AI_KS_WS	아랍어, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분
Arabic_CS_AI_WS	아랍어, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분
Arabic_CS_AS	아랍어, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분

콜레이션	설명
Arabic_CS_AS_KS	아랍어, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분
Arabic_CS_AS_KS_WS	아랍어, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분
Arabic_CS_AS_WS	아랍어, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분
Chinese_PRC_BIN2	중국어-중화인민공화국어, 이진 코드 포인트 비교 정렬
Chinese_PRC_CI_AS	중국어-중화인민공화국어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Chinese_Taiwan_Stroke_CI_AS	중국어-대만어-스트로크, 소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Danish_Norwegian_CI_AS	덴마크어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Finnish_Swedish_CI_AS	핀란드어-스웨덴어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
French_CI_AS	프랑스어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
German_PhoneBook_100_BIN	German-PhoneBook-100, 이진 정렬
German_PhoneBook_100_BIN2	German-PhoneBook-100, 이진 코드 포인트 비교 정렬
German_PhoneBook_100_CI_AI	German-PhoneBook-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분

콜레이션	설명
German_PhoneBook_100_CI_AI_KS	German-PhoneBook-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분
German_PhoneBook_100_CI_AI_KS_SC	German-PhoneBook-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자
German_PhoneBook_100_CI_AI_KS_SC_UTF8	German-PhoneBook-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, UTF8
German_PhoneBook_100_CI_AI_KS_WS	German-PhoneBook-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분
German_PhoneBook_100_CI_AI_KS_WS_SC	German-PhoneBook-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자
German_PhoneBook_100_CI_AI_KS_WS_SC_UTF8	German-PhoneBook-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, UTF8
German_PhoneBook_100_CI_AI_SC	German-PhoneBook-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자
German_PhoneBook_100_CI_AI_SC_UTF8	German-PhoneBook-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, UTF8
German_PhoneBook_100_CI_AI_WS	German-PhoneBook-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분

콜레이션	설명
German_PhoneBook_100_CI_AI_WS_SC	German-PhoneBook-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자
German_PhoneBook_100_CI_AI_WS_SC_UTF8	German-PhoneBook-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, UTF8
German_PhoneBook_100_CI_AS	German-PhoneBook-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
German_PhoneBook_100_CI_AS_KS	German-PhoneBook-100, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분
German_PhoneBook_100_CI_AS_KS_SC	German-PhoneBook-100, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자
German_PhoneBook_100_CI_AS_KS_SC_UTF8	German-PhoneBook-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, UTF8
German_PhoneBook_100_CI_AS_KS_WS	German-PhoneBook-100, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분
German_PhoneBook_100_CI_AS_KS_WS_SC	German-PhoneBook-100, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자
German_PhoneBook_100_CI_AS_KS_WS_SC_UTF8	German-PhoneBook-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, UTF8
German_PhoneBook_100_CI_AS_SC	German-PhoneBook-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자

콜레이션	설명
German_PhoneBook_100_CI_AS_SC_UTF8	German-PhoneBook-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, UTF8
German_PhoneBook_100_CI_AS_WS	German-PhoneBook-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분
German_PhoneBook_100_CI_AS_WS_SC	German-PhoneBook-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자
German_PhoneBook_100_CI_AS_WS_SC_UTF8	German-PhoneBook-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, UTF8
German_PhoneBook_100_CS_AI	German-PhoneBook-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분
German_PhoneBook_100_CS_AI_KS	German-PhoneBook-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분
German_PhoneBook_100_CS_AI_KS_SC	German-PhoneBook-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자
German_PhoneBook_100_CS_AI_KS_SC_UTF8	German-PhoneBook-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, UTF8
German_PhoneBook_100_CS_AI_KS_WS	German-PhoneBook-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분
German_PhoneBook_100_CS_AI_KS_WS_SC	German-PhoneBook-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자

콜레이션	설명
German_PhoneBook_100_CS_AI_KS_WS_SC_UTF8	German-PhoneBook-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, UTF8
German_PhoneBook_100_CS_AI_SC	German-PhoneBook-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자
German_PhoneBook_100_CS_AI_SC_UTF8	German-PhoneBook-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, UTF8
German_PhoneBook_100_CS_AI_WS	German-PhoneBook-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분
German_PhoneBook_100_CS_AI_WS_SC	German-PhoneBook-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자
German_PhoneBook_100_CS_AI_WS_SC_UTF8	German-PhoneBook-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, UTF8
German_PhoneBook_100_CS_AS	German-PhoneBook-100, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
German_PhoneBook_100_CS_AS_KS	German-PhoneBook-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분
German_PhoneBook_100_CS_AS_KS_SC	German-PhoneBook-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자
German_PhoneBook_100_CS_AS_KS_SC_UTF8	German-PhoneBook-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, UTF8

콜레이션	설명
German_PhoneBook_100_CS_AS_KS_WS	German-PhoneBook-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분
German_PhoneBook_100_CS_AS_KS_WS_SC	German-PhoneBook-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자
German_PhoneBook_100_CS_AS_KS_WS_SC_UTF8	German-PhoneBook-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, UTF8
German_PhoneBook_BIN	German-PhoneBook, 이진 정렬
German_PhoneBook_BIN2	German-PhoneBook, 이진 코드 포인트 비교 정렬
German_PhoneBook_CI_AI	German-PhoneBook, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분
German_PhoneBook_CI_AI_KS	German-PhoneBook, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분
German_PhoneBook_CI_AI_KS_WS	German-PhoneBook, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분
German_PhoneBook_CI_AI_WS	German-PhoneBook, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분
German_PhoneBook_CI_AS	German-PhoneBook, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
German_PhoneBook_CI_AS_KS	German-PhoneBook, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분
German_PhoneBook_CI_AS_KS_WS	German-PhoneBook, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분

콜레이션	설명
German_PhoneBook_CI_AS_WS	German-PhoneBook, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분
German_PhoneBook_CS_AI	German-PhoneBook, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분
German_PhoneBook_CS_AI_KS	German-PhoneBook, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분
German_PhoneBook_CS_AI_KS_WS	German-PhoneBook, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분
German_PhoneBook_CS_AI_WS	German-PhoneBook, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분
German_PhoneBook_CS_AS	German-PhoneBook, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
German_PhoneBook_CS_AS_KS	German-PhoneBook, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분
German_PhoneBook_CS_AS_KS_WS	German-PhoneBook, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분
German_PhoneBook_CS_AS_WS	German-PhoneBook, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분
Hebrew_BIN	히브리어, 이진 정렬
Hebrew_CI_AS	히브리어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Japanese_90_BIN	Japanese-90, 이진 정렬
Japanese_90_BIN2	Japanese-90, 이진 코드 포인트 비교 정렬
Japanese_90_CI_AI	Japanese-90, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분

콜레이션	설명
Japanese_90_CI_AI_KS	Japanese-90, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분
Japanese_90_CI_AI_KS_SC	Japanese-90, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자
Japanese_90_CI_AI_KS_SC_UTF8	Japanese-90, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_90_CI_AI_KS_WS	Japanese-90, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분
Japanese_90_CI_AI_KS_WS_SC	Japanese-90, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자
Japanese_90_CI_AI_KS_WS_SC_UTF8	Japanese-90, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, UTF8
Japanese_90_CI_AI_SC	Japanese-90, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자
Japanese_90_CI_AI_SC_UTF8	Japanese-90, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_90_CI_AI_WS	Japanese-90, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분
Japanese_90_CI_AI_WS_SC	Japanese-90, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자
Japanese_90_CI_AI_WS_SC_UTF8	Japanese-90, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, UTF8

콜레이션	설명
Japanese_90_CI_AS	Japanese-90, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Japanese_90_CI_AS_KS	Japanese-90, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분
Japanese_90_CI_AS_KS_SC	Japanese-90, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자
Japanese_90_CI_AS_KS_SC_UTF8	Japanese-90, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_90_CI_AS_KS_WS	Japanese-90, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분
Japanese_90_CI_AS_KS_WS_SC	Japanese-90, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자
Japanese_90_CI_AS_KS_WS_SC_UTF8	Japanese-90, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, UTF8
Japanese_90_CI_AS_SC	Japanese-90, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자
Japanese_90_CI_AS_SC_UTF8	Japanese-90, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_90_CI_AS_WS	Japanese-90, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분
Japanese_90_CI_AS_WS_SC	Japanese-90, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자

콜레이션	설명
Japanese_90_CI_AS_WS_SC_UTF8	Japanese-90, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, UTF8
Japanese_90_CS_AI	Japanese-90, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분
Japanese_90_CS_AI_KS	Japanese-90, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분
Japanese_90_CS_AI_KS_SC	Japanese-90, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자
Japanese_90_CS_AI_KS_SC_UTF8	Japanese-90, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_90_CS_AI_KS_WS	Japanese-90, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분
Japanese_90_CS_AI_KS_WS_SC	Japanese-90, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자
Japanese_90_CS_AI_KS_WS_SC_UTF8	Japanese-90, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, UTF8
Japanese_90_CS_AI_SC	Japanese-90, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자
Japanese_90_CS_AI_SC_UTF8	Japanese-90, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_90_CS_AI_WS	Japanese-90, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분

콜레이션	설명
Japanese_90_CS_AI_WS_SC	Japanese-90, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자
Japanese_90_CS_AI_WS_SC_UTF8	Japanese-90, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, UTF8
Japanese_90_CS_AS	Japanese-90, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Japanese_90_CS_AS_KS	Japanese-90, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분
Japanese_90_CS_AS_KS_SC	Japanese-90, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자
Japanese_90_CS_AS_KS_SC_UTF8	Japanese-90, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_90_CS_AS_KS_WS	Japanese-90, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분
Japanese_90_CS_AS_KS_WS_SC	Japanese-90, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자
Japanese_90_CS_AS_KS_WS_SC_UTF8	Japanese-90, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, UTF8
Japanese_90_CS_AS_SC	Japanese-90, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자
Japanese_90_CS_AS_SC_UTF8	Japanese-90, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_90_CS_AS_WS	Japanese-90, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분

콜레이션	설명
Japanese_90_CS_AS_WS_SC	Japanese-90, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자
Japanese_90_CS_AS_WS_SC_UTF8	Japanese-90, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, UTF8
Japanese_BIN	일본어, 이진 정렬
Japanese_BIN2	일본어, 이진 코드 포인트 비교 정렬
Japanese_Bushu_Kakusu_100_BIN	Japanese-Bushu-Kakusu-100, 이진 정렬
Japanese_Bushu_Kakusu_100_BIN2	Japanese-Bushu-Kakusu-100, 이진 코드 포인트 비교 정렬
Japanese_Bushu_Kakusu_100_CI_AI	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분
Japanese_Bushu_Kakusu_100_CI_AI_KS	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분
Japanese_Bushu_Kakusu_100_CI_AI_KS_SC	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자
Japanese_Bushu_Kakusu_100_CI_AI_KS_SC_UTF8	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_Bushu_Kakusu_100_CI_AI_KS_WS	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분

콜레이션	설명
Japanese_Bushu_Kakusu_100_CI_AI_KS_WS_SC	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자
Japanese_Bushu_Kakusu_100_CI_AI_KS_WS_SC_UTF8	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, UTF8
Japanese_Bushu_Kakusu_100_CI_AI_SC	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자
Japanese_Bushu_Kakusu_100_CI_AI_SC_UTF8	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_Bushu_Kakusu_100_CI_AI_WS	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분
Japanese_Bushu_Kakusu_100_CI_AI_WS_SC	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자
Japanese_Bushu_Kakusu_100_CI_AI_WS_SC_UTF8	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, UTF8
Japanese_Bushu_Kakusu_100_CI_AS	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Japanese_Bushu_Kakusu_100_CI_AS_KS	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분

콜레이션	설명
Japanese_Bushu_Kakusu_100_CI_AS_KS_SC	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자
Japanese_Bushu_Kakusu_100_CI_AS_KS_SC_UTF8	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_Bushu_Kakusu_100_CI_AS_KS_WS	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분
Japanese_Bushu_Kakusu_100_CI_AS_KS_WS_SC	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자
Japanese_Bushu_Kakusu_100_CI_AS_KS_WS_SC_UTF8	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, UTF8
Japanese_Bushu_Kakusu_100_CI_AS_SC	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자
Japanese_Bushu_Kakusu_100_CI_AS_SC_UTF8	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_Bushu_Kakusu_100_CI_AS_WS	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분
Japanese_Bushu_Kakusu_100_CI_AS_WS_SC	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자

콜레이션	설명
Japanese_Bushu_Kakusu_100_CI_AS_WS_SC_UTF8	Japanese-Bushu-Kakusu-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, UTF8
Japanese_Bushu_Kakusu_100_CS_AI	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분
Japanese_Bushu_Kakusu_100_CS_AI_KS	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분
Japanese_Bushu_Kakusu_100_CS_AI_KS_SC	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자
Japanese_Bushu_Kakusu_100_CS_AI_KS_SC_UTF8	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_Bushu_Kakusu_100_CS_AI_KS_WS	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분
Japanese_Bushu_Kakusu_100_CS_AI_KS_WS_SC	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자
Japanese_Bushu_Kakusu_100_CS_AI_KS_WS_SC_UTF8	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, UTF8
Japanese_Bushu_Kakusu_100_CS_AI_SC	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자

콜레이션	설명
Japanese_Bushu_Kakusu_100_CS_AI_SC_UTF8	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_Bushu_Kakusu_100_CS_AI_WS	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분
Japanese_Bushu_Kakusu_100_CS_AI_WS_SC	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자
Japanese_Bushu_Kakusu_100_CS_AI_WS_SC_UTF8	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, UTF8
Japanese_Bushu_Kakusu_100_CS_AS	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Japanese_Bushu_Kakusu_100_CS_AS_KS	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분
Japanese_Bushu_Kakusu_100_CS_AS_KS_SC	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자
Japanese_Bushu_Kakusu_100_CS_AS_KS_SC_UTF8	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_Bushu_Kakusu_100_CS_AS_KS_WS	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분

콜레이션	설명
Japanese_Bushu_Kakusu_100_CS_AS_KS_WS_SC	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자
Japanese_Bushu_Kakusu_100_CS_AS_KS_WS_SC_UTF8	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, UTF8
Japanese_Bushu_Kakusu_100_CS_AS_SC	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자
Japanese_Bushu_Kakusu_100_CS_AS_SC_UTF8	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_Bushu_Kakusu_100_CS_AS_WS	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분
Japanese_Bushu_Kakusu_100_CS_AS_WS_SC	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자
Japanese_Bushu_Kakusu_100_CS_AS_WS_SC_UTF8	Japanese-Bushu-Kakusu-100, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, UTF8
Japanese_Bushu_Kakusu_140_BIN	Japanese-Bushu-Kakusu-140, 이진 정렬
Japanese_Bushu_Kakusu_140_BIN2	Japanese-Bushu-Kakusu-140, 이진 코드 포인트 비교 정렬
Japanese_Bushu_Kakusu_140_CI_AI	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분

콜레이션	설명
Japanese_Bushu_Kakusu_140_CI_AI_KS	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분
Japanese_Bushu_Kakusu_140_CI_AI_KS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_Bushu_Kakusu_140_CI_AI_KS_VSS	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분
Japanese_Bushu_Kakusu_140_CI_AI_KS_VSS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_Bushu_Kakusu_140_CI_AI_KS_WS	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분
Japanese_Bushu_Kakusu_140_CI_AI_KS_WS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_Bushu_Kakusu_140_CI_AI_KS_WS_VSS	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분
Japanese_Bushu_Kakusu_140_CI_AI_KS_WS_VSS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_Bushu_Kakusu_140_CI_AI_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분, UTF8

콜레이션	설명
Japanese_Bushu_Kakusu_140_CI_AI_VSS	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분
Japanese_Bushu_Kakusu_140_CI_AI_VSS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_Bushu_Kakusu_140_CI_AI_WS	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분
Japanese_Bushu_Kakusu_140_CI_AI_WS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_Bushu_Kakusu_140_CI_AI_WS_VSS	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분
Japanese_Bushu_Kakusu_140_CI_AI_WS_VSS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_Bushu_Kakusu_140_CI_AS	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분
Japanese_Bushu_Kakusu_140_CI_AS_KS	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분
Japanese_Bushu_Kakusu_140_CI_AS_KS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분, UTF8

콜레이션	설명
Japanese_Bushu_Kakusu_140_CI_AS_KS_VSS	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분
Japanese_Bushu_Kakusu_140_CI_AS_KS_VSS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_Bushu_Kakusu_140_CI_AS_KS_WS	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분
Japanese_Bushu_Kakusu_140_CI_AS_KS_WS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_Bushu_Kakusu_140_CI_AS_KS_WS_VSS	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분
Japanese_Bushu_Kakusu_140_CI_AS_KS_WS_VSS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_Bushu_Kakusu_140_CI_AS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_Bushu_Kakusu_140_CI_AS_VSS	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분
Japanese_Bushu_Kakusu_140_CI_AS_VSS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분, UTF8

콜레이션	설명
Japanese_Bushu_Kakusu_140_CI_AS_WS	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분
Japanese_Bushu_Kakusu_140_CI_AS_WS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_Bushu_Kakusu_140_CI_AS_WS_VSS	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분
Japanese_Bushu_Kakusu_140_CI_AS_WS_VSS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_Bushu_Kakusu_140_CS_AI	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분
Japanese_Bushu_Kakusu_140_CS_AI_KS	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분
Japanese_Bushu_Kakusu_140_CS_AI_KS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_Bushu_Kakusu_140_CS_AI_KS_VSS	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분
Japanese_Bushu_Kakusu_140_CS_AI_KS_VSS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분, UTF8

콜레이션	설명
Japanese_Bushu_Kakusu_140_CS_AI_KS_WS	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분
Japanese_Bushu_Kakusu_140_CS_AI_KS_WS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_Bushu_Kakusu_140_CS_AI_KS_WS_VSS	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분
Japanese_Bushu_Kakusu_140_CS_AI_KS_WS_VSS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_Bushu_Kakusu_140_CS_AI_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_Bushu_Kakusu_140_CS_AI_VSS	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분
Japanese_Bushu_Kakusu_140_CS_AI_VSS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_Bushu_Kakusu_140_CS_AI_WS	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분
Japanese_Bushu_Kakusu_140_CS_AI_WS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분, UTF8

콜레이션	설명
Japanese_Bushu_Kakusu_140_CS_AI_WS_VSS	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분
Japanese_Bushu_Kakusu_140_CS_AI_WS_VSS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_Bushu_Kakusu_140_CS_AS	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분
Japanese_Bushu_Kakusu_140_CS_AS_KS	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분
Japanese_Bushu_Kakusu_140_CS_AS_KS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_Bushu_Kakusu_140_CS_AS_KS_VSS	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분
Japanese_Bushu_Kakusu_140_CS_AS_KS_VSS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_Bushu_Kakusu_140_CS_AS_KS_WS	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분
Japanese_Bushu_Kakusu_140_CS_AS_KS_WS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분, UTF8

콜레이션	설명
Japanese_Bushu_Kakusu_140_CS_AS_KS_WS_VSS	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분
Japanese_Bushu_Kakusu_140_CS_AS_KS_WS_VSS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_Bushu_Kakusu_140_CS_AS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_Bushu_Kakusu_140_CS_AS_VSS	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분
Japanese_Bushu_Kakusu_140_CS_AS_VSS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_Bushu_Kakusu_140_CS_AS_WS	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분
Japanese_Bushu_Kakusu_140_CS_AS_WS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_Bushu_Kakusu_140_CS_AS_WS_VSS	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분
Japanese_Bushu_Kakusu_140_CS_AS_WS_VSS_UTF8	Japanese-Bushu-Kakusu-140, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분, UTF8

콜레이션	설명
Japanese_CI_AI	일본어, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분
Japanese_CI_AI_KS	일본어, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분
Japanese_CI_AI_KS_WS	일본어, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분
Japanese_CI_AI_WS	일본어, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분
Japanese_CI_AS	일본어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Japanese_CI_AS_KS	일본어, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분
Japanese_CI_AS_KS_WS	일본어, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분
Japanese_CI_AS_WS	일본어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분
Japanese_CS_AI	일본어, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분
Japanese_CS_AI_KS	일본어, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분
Japanese_CS_AI_KS_WS	일본어, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분
Japanese_CS_AI_WS	일본어, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분
Japanese_CS_AS	일본어, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분

콜레이션	설명
Japanese_CS_AS_KS	일본어, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분
Japanese_CS_AS_KS_WS	일본어, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분
Japanese_CS_AS_WS	일본어, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분
Japanese_Unicode_BIN	Japanese-Unicode, 이진 정렬
Japanese_Unicode_BIN2	Japanese-Unicode, 이진 코드 포인트 비교 정렬
Japanese_Unicode_CI_AI	Japanese-Unicode, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분
Japanese_Unicode_CI_AI_KS	Japanese-Unicode, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분
Japanese_Unicode_CI_AI_KS_WS	Japanese-Unicode, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분
Japanese_Unicode_CI_AI_WS	Japanese-Unicode, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분
Japanese_Unicode_CI_AS	Japanese-Unicode, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Japanese_Unicode_CI_AS_KS	Japanese-Unicode, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분
Japanese_Unicode_CI_AS_KS_WS	Japanese-Unicode, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분
Japanese_Unicode_CI_AS_WS	Japanese-Unicode, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분

콜레이션	설명
Japanese_Unicode_CS_AI	Japanese-Unicode, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분
Japanese_Unicode_CS_AI_KS	Japanese-Unicode, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분
Japanese_Unicode_CS_AI_KS_WS	Japanese-Unicode, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분
Japanese_Unicode_CS_AI_WS	Japanese-Unicode, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분
Japanese_Unicode_CS_AS	Japanese-Unicode, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Japanese_Unicode_CS_AS_KS	Japanese-Unicode, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분
Japanese_Unicode_CS_AS_KS_WS	Japanese-Unicode, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분
Japanese_Unicode_CS_AS_WS	Japanese-Unicode, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분
Japanese_XJIS_100_BIN	Japanese-XJIS-100, 이진 정렬
Japanese_XJIS_100_BIN2	Japanese-XJIS-100, 이진 코드 포인트 비교 정렬
Japanese_XJIS_100_CI_AI	Japanese-XJIS-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분
Japanese_XJIS_100_CI_AI_KS	Japanese-XJIS-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분
Japanese_XJIS_100_CI_AI_KS_SC	Japanese-XJIS-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자

콜레이션	설명
Japanese_XJIS_100_CI_AI_KS_SC_UTF8	Japanese-XJIS-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_XJIS_100_CI_AI_KS_WS	Japanese-XJIS-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분
Japanese_XJIS_100_CI_AI_KS_WS_SC	Japanese-XJIS-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자
Japanese_XJIS_100_CI_AI_KS_WS_SC_UTF8	Japanese-XJIS-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, UTF8
Japanese_XJIS_100_CI_AI_SC	Japanese-XJIS-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자
Japanese_XJIS_100_CI_AI_SC_UTF8	Japanese-XJIS-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_XJIS_100_CI_AI_WS	Japanese-XJIS-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분
Japanese_XJIS_100_CI_AI_WS_SC	Japanese-XJIS-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자
Japanese_XJIS_100_CI_AI_WS_SC_UTF8	Japanese-XJIS-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, UTF8
Japanese_XJIS_100_CI_AS	Japanese-XJIS-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분

콜레이션	설명
Japanese_XJIS_100_CI_AS_KS	Japanese-XJIS-100, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분
Japanese_XJIS_100_CI_AS_KS_SC	Japanese-XJIS-100, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자
Japanese_XJIS_100_CI_AS_KS_SC_UTF8	Japanese-XJIS-100, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_XJIS_100_CI_AS_KS_WS	Japanese-XJIS-100, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분
Japanese_XJIS_100_CI_AS_KS_WS_SC	Japanese-XJIS-100, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자
Japanese_XJIS_100_CI_AS_KS_WS_SC_UTF8	Japanese-XJIS-100, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, UTF8
Japanese_XJIS_100_CI_AS_SC	Japanese-XJIS-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자
Japanese_XJIS_100_CI_AS_SC_UTF8	Japanese-XJIS-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_XJIS_100_CI_AS_WS	Japanese-XJIS-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분
Japanese_XJIS_100_CI_AS_WS_SC	Japanese-XJIS-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자

콜레이션	설명
Japanese_XJIS_100_CI_AS_WS_SC_UTF8	Japanese-XJIS-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, UTF8
Japanese_XJIS_100_CS_AI	Japanese-XJIS-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분
Japanese_XJIS_100_CS_AI_KS	Japanese-XJIS-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분
Japanese_XJIS_100_CS_AI_KS_SC	Japanese-XJIS-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자
Japanese_XJIS_100_CS_AI_KS_SC_UTF8	Japanese-XJIS-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_XJIS_100_CS_AI_KS_WS	Japanese-XJIS-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분
Japanese_XJIS_100_CS_AI_KS_WS_SC	Japanese-XJIS-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자
Japanese_XJIS_100_CS_AI_KS_WS_SC_UTF8	Japanese-XJIS-100, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, UTF8
Japanese_XJIS_100_CS_AI_SC	Japanese-XJIS-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자
Japanese_XJIS_100_CS_AI_SC_UTF8	Japanese-XJIS-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, UTF8

콜레이션	설명
Japanese_XJIS_100_CS_AI_WS	Japanese-XJIS-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분
Japanese_XJIS_100_CS_AI_WS_SC	Japanese-XJIS-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자
Japanese_XJIS_100_CS_AI_WS_SC_UTF8	Japanese-XJIS-100, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, UTF8
Japanese_XJIS_100_CS_AS	Japanese-XJIS-100, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Japanese_XJIS_100_CS_AS_KS	Japanese-XJIS-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분
Japanese_XJIS_100_CS_AS_KS_SC	Japanese-XJIS-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자
Japanese_XJIS_100_CS_AS_KS_SC_UTF8	Japanese-XJIS-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_XJIS_100_CS_AS_KS_WS	Japanese-XJIS-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분
Japanese_XJIS_100_CS_AS_KS_WS_SC	Japanese-XJIS-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자
Japanese_XJIS_100_CS_AS_KS_WS_SC_UTF8	Japanese-XJIS-100, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, UTF8

콜레이션	설명
Japanese_XJIS_100_CS_AS_SC	Japanese-XJIS-100, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자
Japanese_XJIS_100_CS_AS_SC_UTF8	Japanese-XJIS-100, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, UTF8
Japanese_XJIS_100_CS_AS_WS	Japanese-XJIS-100, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분
Japanese_XJIS_140_BIN	Japanese-XJIS-140, 이진 정렬
Japanese_XJIS_140_BIN2	Japanese-XJIS-140, 이진 코드 포인트 비교 정렬
Japanese_XJIS_140_CI_AI	Japanese-XJIS-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분
Japanese_XJIS_140_CI_AI_KS	Japanese-XJIS-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분
Japanese_XJIS_140_CI_AI_KS_UTF8	Japanese-XJIS-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_XJIS_140_CI_AI_KS_VSS	Japanese-XJIS-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분
Japanese_XJIS_140_CI_AI_KS_VSS_UTF8	Japanese-XJIS-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분, UTF8

콜레이션	설명
Japanese_XJIS_140_CI_AI_KS_WS	Japanese-XJIS-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분
Japanese_XJIS_140_CI_AI_KS_WS_UTF8	Japanese-XJIS-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_XJIS_140_CI_AI_KS_WS_VSS	Japanese-XJIS-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분
Japanese_XJIS_140_CI_AI_KS_WS_VSS_UTF8	Japanese-XJIS-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_XJIS_140_CI_AI_UTF8	Japanese-XJIS-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_XJIS_140_CI_AI_VSS	Japanese-XJIS-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분
Japanese_XJIS_140_CI_AI_VSS_UTF8	Japanese-XJIS-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_XJIS_140_CI_AI_WS	Japanese-XJIS-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분
Japanese_XJIS_140_CI_AI_WS_UTF8	Japanese-XJIS-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분, UTF8

콜레이션	설명
Japanese_XJIS_140_CI_AI_WS_VSS	Japanese-XJIS-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분
Japanese_XJIS_140_CI_AI_WS_VSS_UTF8	Japanese-XJIS-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_XJIS_140_CI_AS	일본어-XJIS-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분
Japanese_XJIS_140_CI_AS_KS	일본어-XJIS-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분
Japanese_XJIS_140_CI_AS_KS_UTF8	일본어-XJIS-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_XJIS_140_CI_AS_KS_VSS	일본어-XJIS-140, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분
Japanese_XJIS_140_CI_AS_KS_VSS_UTF8	일본어-XJIS-140, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_XJIS_140_CI_AS_KS_WS	일본어-XJIS-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분
Japanese_XJIS_140_CI_AS_KS_WS_UTF8	Japanese-XJIS-140, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분, UTF8

콜레이션	설명
Japanese_XJIS_140_CI_AS_KS_WS_VSS	일본어-XJIS-140, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분
Japanese_XJIS_140_CI_AS_KS_WS_VSS_UTF8	Japanese-XJIS-140, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_XJIS_140_CI_AS_UTF8	일본어-XJIS-140, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_XJIS_140_CI_AS_VSS	일본어-XJIS-140, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분
Japanese_XJIS_140_CI_AS_VSS_UTF8	Japanese-XJIS-140, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_XJIS_140_CI_AS_WS	Japanese-XJIS-140, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분
Japanese_XJIS_140_CI_AS_WS_UTF8	Japanese-XJIS-140, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_XJIS_140_CI_AS_WS_VSS	Japanese-XJIS-140, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분
Japanese_XJIS_140_CI_AS_WS_VSS_UTF8	Japanese-XJIS-140, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분, UTF8

콜레이션	설명
Japanese_XJIS_140_CS_AI	Japanese-XJIS-140, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분
Japanese_XJIS_140_CS_AI_KS	Japanese-XJIS-140, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분
Japanese_XJIS_140_CS_AI_KS_UTF8	Japanese-XJIS-140, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_XJIS_140_CS_AI_KS_VSS	Japanese-XJIS-140, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분
Japanese_XJIS_140_CS_AI_KS_VSS_UTF8	Japanese-XJIS-140, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_XJIS_140_CS_AI_KS_WS	Japanese-XJIS-140, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분
Japanese_XJIS_140_CS_AI_KS_WS_UTF8	Japanese-XJIS-140, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_XJIS_140_CS_AI_KS_WS_VSS	Japanese-XJIS-140, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분
Japanese_XJIS_140_CS_AI_KS_WS_VSS_UTF8	Japanese-XJIS-140, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분, UTF8

콜레이션	설명
Japanese_XJIS_140_CS_AI_UTF8	Japanese-XJIS-140, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_XJIS_140_CS_AI_VSS	Japanese-XJIS-140, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분
Japanese_XJIS_140_CS_AI_VSS_UTF8	Japanese-XJIS-140, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_XJIS_140_CS_AI_WS	Japanese-XJIS-140, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분
Japanese_XJIS_140_CS_AI_WS_UTF8	Japanese-XJIS-140, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_XJIS_140_CS_AI_WS_VSS	Japanese-XJIS-140, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분
Japanese_XJIS_140_CS_AI_WS_VSS_UTF8	Japanese-XJIS-140, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_XJIS_140_CS_AS	Japanese-XJIS-140, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분
Japanese_XJIS_140_CS_AS_KS	Japanese-XJIS-140, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분

콜레이션	설명
Japanese_XJIS_140_CS_AS_KS_UTF8	Japanese-XJIS-140, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_XJIS_140_CS_AS_KS_VSS	Japanese-XJIS-140, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분
Japanese_XJIS_140_CS_AS_KS_VSS_UTF8	Japanese-XJIS-140, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분, UTF8
일본어_XJIS_140_CS_AS_KS_WS	Japanese-XJIS-140, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분
Japanese_XJIS_140_CS_AS_KS_WS_UTF8	Japanese-XJIS-140, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_XJIS_140_CS_AS_KS_WS_VSS	Japanese-XJIS-140, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분
Japanese_XJIS_140_CS_AS_KS_WS_VSS_UTF8	Japanese-XJIS-140, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_XJIS_140_CS_AS_UTF8	Japanese-XJIS-140, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_XJIS_140_CS_AS_VSS	Japanese-XJIS-140, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분

콜레이션	설명
Japanese_XJIS_140_CS_AS_VSS_UTF8	Japanese-XJIS-140, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분, UTF8
Japanese_XJIS_140_CS_AS_WS	Japanese-XJIS-140, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분
Japanese_XJIS_140_CS_AS_WS_UTF8	Japanese-XJIS-140, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분, UTF8
Japanese_XJIS_140_CS_AS_WS_VSS	Japanese-XJIS-140, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분
Japanese_XJIS_140_CS_AS_WS_VSS_UTF8	Japanese-XJIS-140, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분, 보조 문자, 변형 선택기 구분, UTF8
Korean_Wansung_CI_AS	한국어-완성형, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Latin1_General_100_BIN	라틴어1-일반-100, 이진 정렬
Latin1_General_100_BIN2	라틴어1-일반-100, 이진 코드 포인트 비교 정렬
Latin1_General_100_BIN2_UTF8	Latin1-General-100, 이진 코드 포인트 비교 정렬, UTF8
Latin1_General_100_CI_AS	라틴어1-일반-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Latin1_General_100_CI_AS_SC_UTF8	Latin1-General-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, UTF8
Latin1_General_BIN	라틴어1-일반, 이진 정렬

콜레이션	설명
Latin1_General_BIN2	라틴어1-일반, 이진 코드 포인트 비교 정렬
Latin1_General_CI_AI	라틴어1-일반, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분
Latin1_General_CI_AS	라틴어1-일반, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Latin1_General_CI_AS_KS	라틴어1-일반, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분
Latin1_General_CS_AS	라틴어1-일반, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Modern_Spanish_CI_AS	현대-스페인어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
SQL_1xCompat_CP850_CI_AS	라틴어1-일반, 유니코드 데이터의 경우 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비 유니코드 데이터의 경우 코드 페이지 850의 SQL Server 정렬 순서 49
SQL_Latin1_General_CP1_CI_AI	라틴어1-일반, 유니코드 데이터의 경우 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 비 유니코드 데이터의 경우 코드 페이지 1252의 SQL Server 정렬 순서 54
SQL_Latin1_General_CP1_CI_AS	라틴어1-일반, 유니코드 데이터의 경우 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비 유니코드 데이터의 경우 코드 페이지 1252의 SQL Server 정렬 순서 52
SQL_Latin1_General_CP1_CS_AS	라틴어1-일반, 유니코드 데이터의 경우 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비 유니코드 데이터의 경우 코드 페이지 1252의 SQL Server 정렬 순서 51

콜레이션	설명
SQL_Latin1_General_CP1250_CI_AS	Latin1-General, 유니코드 데이터의 경우 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 1250의 SQL Server 정렬 순서 82
SQL_Latin1_General_CP1250_CS_AS	Latin1-General, 유니코드 데이터의 경우 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 1250의 SQL Server 정렬 순서 81
SQL_Latin1_General_CP1251_CI_AS	Latin1-General, 유니코드 데이터의 경우 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 1251의 SQL Server 정렬 순서 106
SQL_Latin1_General_CP1251_CS_AS	Latin1-General, 유니코드 데이터의 경우 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 1251의 SQL Server 정렬 순서 105
SQL_Latin1_General_CP1253_CI_AI	Latin1-General, 유니코드 데이터의 경우 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 1253의 SQL Server 정렬 순서 124
SQL_Latin1_General_CP1253_CI_AS	Latin1-General, 유니코드 데이터의 경우 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 1253의 SQL Server 정렬 순서 114
SQL_Latin1_General_CP1253_CS_AS	Latin1-General, 유니코드 데이터의 경우 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 1253의 SQL Server 정렬 순서 113

콜레이션	설명
SQL_Latin1_General_CP1254_CI_AS	터키어, 유니코드 데이터의 경우 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 1254의 SQL Server 정렬 순서 130
SQL_Latin1_General_CP1254_CS_AS	터키어, 유니코드 데이터의 경우 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 1254의 SQL Server 정렬 순서 129
SQL_Latin1_General_CP1255_CI_AS	Latin1-General, 유니코드 데이터의 경우 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 1255의 SQL Server 정렬 순서 138
SQL_Latin1_General_CP1255_CS_AS	Latin1-General, 유니코드 데이터의 경우 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 1255의 SQL Server 정렬 순서 137
SQL_Latin1_General_CP1256_CI_AS	라틴어1-일반, 유니코드 데이터의 경우 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비 유니코드 데이터의 경우 코드 페이지 1256의 SQL Server 정렬 순서 146
SQL_Latin1_General_CP1256_CS_AS	Latin1-General, 유니코드 데이터의 경우 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 1256의 SQL Server 정렬 순서 145
SQL_Latin1_General_CP1257_CI_AS	Latin1-General, 유니코드 데이터의 경우 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 1257의 SQL Server 정렬 순서 154

콜레이션	설명
SQL_Latin1_General_CP1257_CS_AS	Latin1-General, 유니코드 데이터의 경우 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 1257의 SQL Server 정렬 순서 153
SQL_Latin1_General_CP437_BIN	Latin1-General, 유니코드 데이터의 경우 이진 정렬, 비유니코드 데이터의 경우 코드 페이지 437의 SQL Server 정렬 순서 30
SQL_Latin1_General_CP437_BIN2	Latin1-General, 유니코드 데이터의 경우 이진 코드 포인트 비교 정렬, 비유니코드 데이터의 경우 코드 페이지 437의 SQL Server 정렬 순서 30
SQL_Latin1_General_CP437_CI_AI	라틴어1-일반, 유니코드 데이터의 경우 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 비 유니코드 데이터의 경우 코드 페이지 437의 SQL Server 정렬 순서 34
SQL_Latin1_General_CP437_CI_AS	Latin1-General, 유니코드 데이터의 경우 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 437의 SQL Server 정렬 순서 32
SQL_Latin1_General_CP437_CS_AS	Latin1-General, 유니코드 데이터의 경우 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 437의 SQL Server 정렬 순서 31
SQL_Latin1_General_CP850_BIN	Latin1-General, 유니코드 데이터의 경우 이진 정렬, 비유니코드 데이터의 경우 코드 페이지 850의 SQL Server 정렬 순서 40
SQL_Latin1_General_CP850_BIN2	라틴어1-일반, 유니코드 데이터의 경우 이진 코드 포인트 비교, 비 유니코드 데이터의 경우 코드 페이지 850의 SQL Server 정렬 순서 40

콜레이션	설명
SQL_Latin1_General_CP850_CI_AI	라틴어1-일반, 유니코드 데이터의 경우 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 850의 SQL Server 정렬 순서 44
SQL_Latin1_General_CP850_CI_AS	라틴어1-일반, 유니코드 데이터의 경우 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비 유니코드 데이터의 경우 코드 페이지 850의 SQL Server 정렬 순서 42
SQL_Latin1_General_CP850_CS_AS	Latin1-General, 유니코드 데이터의 경우 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 850의 SQL Server 정렬 순서 41
SQL_Latin1_General_Pref_CP1_CI_AS	Latin1-General, 유니코드 데이터의 경우 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 1252의 SQL Server 정렬 순서 53
SQL_Latin1_General_Pref_CP437_CI_AS	Latin1-General, 유니코드 데이터의 경우 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 437의 SQL Server 정렬 순서 33
SQL_Latin1_General_Pref_CP850_CI_AS	Latin1-General, 유니코드 데이터의 경우 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 850의 SQL Server 정렬 순서 43
Thai_CI_AS	프랑스어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분

RDS Custom for SQL Server DB 인스턴스의 현지 시간대

RDS Custom for SQL Server DB 인스턴스의 시간대가 기본적으로 설정되어 있습니다. 현재 기본값은 협정 세계시(UTC)입니다. DB 인스턴스의 시간대를 애플리케이션의 시간대와 일치하도록 현지 시간대로 설정할 수 있습니다.

DB 인스턴스를 처음 만들 때 시간대를 설정합니다. AWS Management Console, Amazon RDS API CreateDBInstance 작업 또는 AWS CLI create-db-instance 명령을 사용하여 DB 인스턴스를 생성할 수 있습니다.

DB 인스턴스가 다중 AZ 배포의 일부인 경우 장애 조치 중에 시간대가 설정된 현지 시간대로 유지됩니다.

시점 복원을 요청할 경우 복원 시간을 지정합니다. 시간은 현지 시간대로 표시됩니다. 자세한 내용은 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 섹션을 참조하세요.

다음은 DB 인스턴스에 대해 현지 시간대를 설정할 때 적용되는 제한 사항입니다.

- 인스턴스 생성 중에 DB 인스턴스의 시간대를 구성할 수 있지만 기존 RDS Custom for SQL Server DB 인스턴스의 시간대는 수정할 수 없습니다.
- 기존 RDS Custom for SQL Server DB 인스턴스의 시간대가 수정된 경우 RDS Custom은 DB 인스턴스 상태를 `unsupported-configuration`으로 변경하고 이벤트 알림을 보냅니다.
- DB 인스턴스의 스냅샷을 다른 시간대의 DB 인스턴스로 복원할 수 없습니다.
- 한 표준 시간대의 백업 파일을 다른 표준 시간대로 복원하지 않는 것이 좋습니다. 한 표준 시간대의 백업 파일을 다른 표준 시간대로 복원하는 경우 쿼리와 애플리케이션을 감사하여 표준 시간대 변경의 영향을 확인해야 합니다. 자세한 내용은 기본 백업 및 복원 기능을 사용하여 SQL Server 데이터베이스 가져오기 및 내보내기 섹션을 참조하세요.

지원되는 시간대

현지 시간대를 다음 표에 나열된 값 중 하나로 설정할 수 있습니다.

시간대	표준 시간 오프셋	설명	참고
아프가니스탄 표준시	(UTC+04:30)	카불	이 시간대는 일광 절약 시간을 준수하지 않습니다.
알래스카 표준시	(UTC–09:00)	알래스카	

시간대	표준 시간 오프셋	설명	참고
알류샨 표준시	(UTC–10:00)	알류샨 열도	
알타이 표준시	(UTC+07:00)	바르나울, 고르노알타이스크	
아랍 표준시	(UTC+03:00)	쿠웨이트, 리야드	이 시간대는 일광 절약 시간을 준수하지 않습니다.
아라비아 표준시	(UTC+04:00)	아부다비, 무스카트	
아랍 표준시	(UTC+03:00)	바그다드	이 시간대는 일광 절약 시간을 준수하지 않습니다.
아르헨티나 표준시	(UTC–03:00)	부에노스아이레스	이 시간대는 일광 절약 시간을 준수하지 않습니다.
아스트라한 표준시	(UTC+04:00)	아스트라한, 울랴노브스크	
대서양 표준시	(UTC–04:00)	대서양 표준시(캐나다)	
AUS 중부 표준시	(UTC+09:30)	다윈	이 시간대는 일광 절약 시간을 준수하지 않습니다.
오스트레일리아 중부 표준시	(UTC+08:45)	유클라	
AUS 동부 표준시	(UTC+10:00)	캔버라, 멜버른, 시드니	
아제르바이잔 표준시	(UTC+04:00)	바쿠	
아조레스 표준시	(UTC–01:00)	아조레스	

시간대	표준 시간 오프셋	설명	참고
바이아 표준시	(UTC–03:00)	살바도르	
방글라데시 표준시	(UTC+06:00)	다카	이 시간대는 일광 절약 시간을 준수하지 않습니다.
벨라루스 표준시	(UTC+03:00)	민스크	이 시간대는 일광 절약 시간을 준수하지 않습니다.
부건빌 표준시	(UTC+11:00)	부겐빌 섬	
캐나다 중부 표준시	(UTC–06:00)	서스캐처원	이 시간대는 일광 절약 시간을 준수하지 않습니다.
카포베르데 표준시	(UTC–01:00)	카포베르데 섬	이 시간대는 일광 절약 시간을 준수하지 않습니다.
코카서스 표준시	(UTC+04:00)	예레반	
중부 오스트레일리아 표준시	(UTC+09:30)	애들레이드	
중앙 아메리카 표준시	(UTC–06:00)	중앙 아메리카	이 시간대는 일광 절약 시간을 준수하지 않습니다.
중앙 아시아 표준시	(UTC+06:00)	아스타나	이 시간대는 일광 절약 시간을 준수하지 않습니다.
브라질 중부 표준시	(UTC–04:00)	쿠이아바	
중앙 유럽 표준시	(UTC+01:00)	베오그라드, 브라티슬라바, 부다페스트, 류블랴나, 프라하	

시간대	표준 시간 오프셋	설명	참고
중앙 유럽 표준시	(UTC+01:00)	사라예보, 스코페, 바르샤바, 자그레브	
중앙 태평양 표준시	(UTC+11:00)	솔로몬 제도, 뉴칼레도니아	이 시간대는 일광 절약 시간을 준수하지 않습니다.
중부 표준시	(UTC–06:00)	중부 표준시(미국과 캐나다)	
중부 표준시(멕시코)	(UTC–06:00)	과달라하라, 멕시코시티, 몬테레이	
채텀 섬 표준시	(UTC+12:45)	채텀 섬	
중국 표준시	(UTC+08:00)	베이징, 충칭, 홍콩 특별 행정구, 우루무치	이 시간대는 일광 절약 시간을 준수하지 않습니다.
쿠바 표준시	(UTC–05:00)	하바나	
날짜 변경선 표준시	(UTC–12:00)	날짜 변경선 서쪽	이 시간대는 일광 절약 시간을 준수하지 않습니다.
E. 아프리카 표준시	(UTC+03:00)	나이로비	이 시간대는 일광 절약 시간을 준수하지 않습니다.
E. 오스트레일리아 표준시	(UTC+10:00)	브리즈번	이 시간대는 일광 절약 시간을 준수하지 않습니다.
E. 유럽 표준시	(UTC+02:00)	키시나우	
E. 남아메리카 표준시	(UTC–03:00)	브라질리아	
이스터 섬 표준시	(UTC–06:00)	이스터 섬	

시간대	표준 시간 오프셋	설명	참고
동부 표준시	(UTC–05:00)	동부 표준시(미국과 캐나다)	
동부 표준시(멕시코)	(UTC–05:00)	체투말	
이집트 표준시	(UTC+02:00)	카이로	
예카테린부르크 표준시	(UTC+05:00)	예카테린부르크	
피지 표준시	(UTC+12:00)	피지	
FLE 표준시	(UTC+02:00)	헬싱키, 키예프, 리가, 소피아, 탈린, 빌뉴스	
그루지야 표준시	(UTC+04:00)	트빌리시	이 시간대는 일광 절약 시간을 준수하지 않습니다.
GMT 표준시	(UTC)	더블린, 에든버러, 리스본, 런던	이 시간대는 그리니치 표준시(GMT)와 다릅니다. 이 시간대는 일광 절약 시간을 준수합니다.
그린란드 표준시	(UTC–03:00)	그린란드	
그리니치 표준시	(UTC)	몬로비아, 레이캬비크	이 시간대는 일광 절약 시간을 준수하지 않습니다.
GTB 표준시	(UTC+02:00)	아테네, 부쿠레슈티	
아이티 표준시	(UTC–05:00)	아이티	
하와이 표준시	(UTC–10:00)	하와이	

시간대	표준 시간 오프셋	설명	참고
인도 표준시	(UTC+05:30)	첸나이, 콜카타, 뭄바이, 뉴델리	이 시간대는 일광 절약 시간을 준수하지 않습니다.
이란 표준시	(UTC+03:30)	테헤란	
이스라엘 표준시	(UTC+02:00)	예루살렘	
요르단 표준시	(UTC+02:00)	암만	
칼리닌그라드 표준시	(UTC+02:00)	칼리닌그라드	
캄차카 표준시	(UTC+12:00)	페트로파블로프스크-캄차스키 – 이전	
대한민국 표준시	(UTC+09:00)	서울	이 시간대는 일광 절약 시간을 준수하지 않습니다.
리비아 표준시	(UTC+02:00)	트리폴리	
라인 제도 표준시	(UTC+14:00)	키리티마티 섬	
로드하우 표준시	(UTC+10:30)	로드하우 섬	
마가단 표준시	(UTC+11:00)	마가단	이 시간대는 일광 절약 시간을 준수하지 않습니다.
마가야네스 표준시	(UTC–03:00)	푼타 아레나스	
마키저스 표준시	(UTC–09:30)	마르케사스 제도	
모리셔스 표준시	(UTC+04:00)	포트루이스	이 시간대는 일광 절약 시간을 준수하지 않습니다.
중동 표준시	(UTC+02:00)	베이루트	

시간대	표준 시간 오프셋	설명	참고
몬테비데오 표준시	(UTC–03:00)	몬테비데오	
모로코 표준시	(UTC+01:00)	카사블랑카	
산지 표준시	(UTC–07:00)	산지 표준시(미국과 캐나다)	
산지 표준시(멕시코)	(UTC–07:00)	치와와, 라파스, 마사틀란	
미얀마 표준시	(UTC+06:30)	양곤(랑군)	이 시간대는 일광 절약 시간을 준수하지 않습니다.
N. 중앙 아시아 표준시	(UTC+07:00)	노보시비르스크	
나미비아 표준시	(UTC+02:00)	빈트후크	
네팔 표준시	(UTC+05:45)	카트만두	이 시간대는 일광 절약 시간을 준수하지 않습니다.
뉴질랜드 표준시	(UTC+12:00)	오클랜드, 웰링턴	
뉴펀들랜드 표준시	(UTC–03:30)	뉴펀들랜드	
노퍽 표준시	(UTC+11:00)	노퍽 섬	
북아시아 동부 표준시	(UTC+08:00)	이르쿠츠크	
북아시아 표준시	(UTC+07:00)	크라스노야르스크	
북한 표준시	(UTC+09:00)	평양	
옴스크 표준시	(UTC+06:00)	옴스크	
태평양 SA 표준시	(UTC–03:00)	산티아고	

시간대	표준 시간 오프셋	설명	참고
태평양 표준시	(UTC–08:00)	태평양 표준시(미국과 캐나다)	
태평양 표준시(멕시코)	(UTC–08:00)	바하 캘리포니아	
파키스탄 표준시	(UTC+05:00)	이슬라마바드, 카라치	이 시간대는 일광 절약 시간을 준수하지 않습니다.
파라과이 표준시	(UTC–04:00)	아순시온	
로맨스 표준시	(UTC+01:00)	브뤼셀, 코펜하겐, 마드리드, 파리	
러시아 표준 시간대 10	(UTC+11:00)	초쿠르다흐	
러시아 표준 시간대 11	(UTC+12:00)	아나디리, 페트로파블로프스크-캄차스키	
러시아 표준 시간대 3	(UTC+04:00)	이젭스크, 사마라	
러시아 표준시	(UTC+03:00)	모스크바, 상트페테르부르크, 볼고그라드	이 시간대는 일광 절약 시간을 준수하지 않습니다.
SA 동부 표준시	(UTC–03:00)	카옌, 포르탈레자	이 시간대는 일광 절약 시간을 준수하지 않습니다.
태평양 SA 표준시	(UTC–05:00)	보고타, 리마, 키토, 리오 브랑코	이 시간대는 일광 절약 시간을 준수하지 않습니다.
SA 서부 표준시	(UTC–04:00)	조지타운, 라파스, 마노스, 산후안	이 시간대는 일광 절약 시간을 준수하지 않습니다.

시간대	표준 시간 오프셋	설명	참고
생피에르 표준시	(UTC–03:00)	세인트 피에르 미켈론	
사할린 표준시	(UTC+11:00)	사할린	
사모아 표준시	(UTC+13:00)	사모아	
상투메 표준시	(UTC+01:00)	상투메	
사라토프 표준시	(UTC+04:00)	사라토프	
동남아시아 표준시	(UTC+07:00)	방콕, 하노이, 자카르타	이 시간대는 일광 절약 시간을 준수하지 않습니다.
싱가포르 표준시	(UTC+08:00)	쿠알라룸푸르, 싱가포르	이 시간대는 일광 절약 시간을 준수하지 않습니다.
남아프리카 표준시	(UTC+02:00)	하라레, 프리토리아	이 시간대는 일광 절약 시간을 준수하지 않습니다.
스리랑카 표준시	(UTC+05:30)	스리자야와르데네푸라	이 시간대는 일광 절약 시간을 준수하지 않습니다.
수단 표준시	(UTC+02:00)	하르툼	
시리아 표준시	(UTC+02:00)	다마스쿠스	
타이베이 표준시	(UTC+08:00)	타이베이	이 시간대는 일광 절약 시간을 준수하지 않습니다.
태즈메이니아 표준시	(UTC+10:00)	호바트	
토칸틴스 표준시	(UTC–03:00)	아라구아이나	

시간대	표준 시간 오프셋	설명	참고
도쿄 표준시	(UTC+09:00)	오사카, 삿포로, 도쿄	이 시간대는 일광 절약 시간을 준수하지 않습니다.
톰스크 표준시	(UTC+07:00)	톰스크	
통가 표준시	(UTC+13:00)	누쿠알로파	이 시간대는 일광 절약 시간을 준수하지 않습니다.
트란스바이칼 표준시	(UTC+09:00)	치타	
터키 표준시	(UTC+03:00)	이스탄불	
터크스 케이커스 표준시	(UTC–05:00)	터크스 케이커스	
울란바토르 표준시	(UTC+08:00)	울란바토르	이 시간대는 일광 절약 시간을 준수하지 않습니다.
미국 동부 표준시	(UTC–05:00)	인디애나(동부)	
미국 산지 표준시	(UTC–07:00)	애리조나	이 시간대는 일광 절약 시간을 준수하지 않습니다.
UTC	UTC	협정 세계시	이 시간대는 일광 절약 시간을 준수하지 않습니다.
UTC–02	(UTC–02:00)	협정 세계시–02	이 시간대는 일광 절약 시간을 준수하지 않습니다.
UTC–08	(UTC–08:00)	협정 세계시–08	
UTC–09	(UTC–09:00)	협정 세계시–09	

시간대	표준 시간 오프셋	설명	참고
UTC–11	(UTC–11:00)	협정 세계시–11	이 시간대는 일광 절약 시간을 준수하지 않습니다.
UTC+12	(UTC+12:00)	협정 세계시+12	이 시간대는 일광 절약 시간을 준수하지 않습니다.
UTC+13	(UTC+13:00)	협정 세계시+13	
베네수엘라 표준시	(UTC–04:00)	카라카스	이 시간대는 일광 절약 시간을 준수하지 않습니다.
블라디보스토크 표준시	(UTC+10:00)	블라디보스토크	
볼고그라드 표준시	(UTC+04:00)	볼고그라드	
W. 오스트레일리아 표준시	(UTC+08:00)	퍼스	이 시간대는 일광 절약 시간을 준수하지 않습니다.
W. 중앙 아프리카 표준시	(UTC+01:00)	서중앙 아프리카	이 시간대는 일광 절약 시간을 준수하지 않습니다.
W. 유럽 표준시	(UTC+01:00)	암스테르담, 베를린, 베른, 로마, 스톡홀름, 비엔나	
W. 몽골 표준시	(UTC+07:00)	호브드	
서아시아 표준시	(UTC+05:00)	아슈하바트, 타슈켄트	이 시간대는 일광 절약 시간을 준수하지 않습니다.
웨스트 뱅크 표준시	(UTC+02:00)	가자, 헤브론	

시간대	표준 시간 오프셋	설명	참고
서태평양 표준시	(UTC+10:00)	괌, 포트모르즈비	이 시간대는 일광 절약 시간을 준수하지 않습니다.
야쿠츠크 표준시	(UTC+09:00)	야쿠츠크	

RDS Custom for SQL Server에서 Service Master Key 사용

RDS Custom for SQL Server는 Service Master Key(SMK) 사용을 지원합니다. RDS Custom은 RDS Custom for SQL Server DB 인스턴스의 수명 주기 내내 동일한 SMK를 유지합니다. 동일한 SMK를 유지함으로써 DB 인스턴스는 연결된 서버 암호 및 보안 인증 정보와 같이 SMK로 암호화된 객체를 사용할 수 있습니다. 다중 AZ 배포를 사용하는 경우 RDS Custom은 기본 및 보조 DB 인스턴스 간에 SMK를 동기화하고 유지 관리합니다.

주제

- 리전 및 버전 사용 가능 여부
- 지원되는 기능
- TDE 사용
- 기능 구성
- 요구 사항 및 제한 사항

리전 및 버전 사용 가능 여부

RDS Custom for SQL Server를 사용하는 모든 리전에서 SMK 사용이 지원되며, RDS Custom에서 제공되는 모든 SQL Server 버전에서 사용할 수 있습니다. RDS Custom for SQL Server에서 사용할 수 있는 Amazon RDS의 버전 및 리전에 대한 자세한 내용은 RDS Custom for SQL Server를 지원하는 리전 및 DB 엔진 섹션을 참조하세요.

지원되는 기능

RDS Custom for SQL Server에서 SMK를 사용하는 경우 다음과 같은 기능이 지원됩니다.

- TDE(Transparent Data Encryption)
- 열 수준 암호화

- 데이터베이스 메일
- 연결된 서버
- SSIS(SQL Server Integration Services)

TDE 사용

SMK를 사용하면 스토리지에 데이터를 쓰기 전에 데이터를 암호화한 뒤에 데이터를 스토리지에서 읽을 때 다시 자동으로 해독하는 투명한 데이터 암호화(TDE)를 구성할 수 있습니다. RDS for SQL Server와 달리 RDS Custom for SQL Server DB 인스턴스에서 TDE를 구성할 때 옵션 그룹을 사용할 필요가 없습니다. 대신 인증서와 데이터베이스 암호화 키를 생성한 후 다음 명령을 실행하여 데이터베이스 수준에서 TDE를 활성화할 수 있습니다.

```
ALTER DATABASE [myDatabase] SET ENCRYPTION ON;
```

RDS for SQL Server와 함께 TDE를 사용하는 방법에 대한 자세한 내용은 SQL Server에서 TDE(투명한 데이터 암호화) 지원 섹션을 확인하세요.

Microsoft SQL Server의 TDE에 대한 자세한 내용은 Microsoft 설명서의 투명한 데이터 암호화를 참조하세요.

기능 구성

RDS Custom for SQL Server에서 SMK를 사용하는 기능을 구성하는 자세한 단계는 Amazon RDS 데이터베이스 블로그의 다음 게시물을 참조하세요.

- 연결된 서버: RDS Custom for SQL Server에서 연결된 서버 구성.
- SSIS: SSIS 패키지를 RDS Custom for SQL Server로 마이그레이션.
- TDE: RDS Custom for SQL Server에서 TDE를 사용하여 데이터 보호.

요구 사항 및 제한 사항

RDS Custom for SQL Server DB 인스턴스와 함께 SMK를 사용하는 경우 다음과 같은 요구 사항 및 제한 사항을 염두에 두세요.

- DB 인스턴스에서 SMK를 재생성하는 경우 즉시 수동 DB 스냅샷을 생성해야 합니다. 가능하면 SMK를 다시 생성하지 않는 것이 좋습니다.
- 서버 인증서 및 데이터베이스 마스터 키 암호의 백업을 유지해야 합니다. 이러한 백업을 유지 관리하지 않으면 데이터가 손실될 수 있습니다.

- SSIS를 구성하는 경우 컴퓨팅 규모 조정 또는 호스트 교체 시 SSM 문서를 사용하여 RDS Custom for SQL Server DB 인스턴스를 도메인에 조인해야 합니다.
- TDE 또는 열 암호화가 활성화되면 데이터베이스 백업이 자동으로 암호화됩니다. 스냅샷 복원 또는 특정 시점 복구를 수행하면 소스 DB 인스턴스의 SMK가 복원되어 복원에 필요한 데이터를 해독하고 복원된 인스턴스의 데이터를 다시 암호화하기 위해 새 SMK가 생성됩니다.

Microsoft SQL Server의 Service Master Key에 대한 자세한 내용은 Microsoft 설명서의 SQL Server 및 데이터베이스 암호화 키를 참조하세요.

RDS Custom for SQL Server에서 변경 데이터 캡처(CDC) 지원

개요

RDS Custom for SQL Server는 변경 데이터 캡처(CDC)에 대한 기본 지원을 제공하므로 SQL Server 테이블에서 데이터 수정 사항을 추적하고 캡처할 수 있습니다. CDC는 후속 검색 및 분석을 위해 이러한 변경 사항에 대한 자세한 메타데이터를 저장합니다. CDC 기능에 대한 자세한 내용은 Microsoft 설명서의 Change data capture를 참조하세요.

SQL Server에서 CDC 작업을 수행하려면 `sys.servers`의 로컬 서버(`server_id = 0`인 로컬 서버)와 `SERVERPROPERTY('ServerName')` 식별자 간에 일치하는 값이 필요합니다. RDS Custom for SQL Server는 인스턴스의 수명 주기 동안 이 동기화를 자동으로 유지하여 유지 관리 또는 복구 작업 중에 호스트가 교체되더라도 지속적인 CDC 작동을 보장합니다.

Important

다중 AZ 인스턴스 장애 조치 후 `SERVERPROPERTY('Servername')` 함수는 네트워크/컴퓨터 이름의 변경 사항을 자동으로 반영합니다. 그러나 `@@SERVERNAME` 함수는 `MSSQLSERVER` 서비스가 다시 시작될 때까지 이전 서버 이름을 유지합니다. `@@SERVERNAME` 쿼리는 장애 조치 후 이전 서버 이름을 반환합니다. 장애 조치 후 정확한 서버 이름을 얻으려면 다음 SQL 쿼리를 사용하세요.

```
SELECT name FROM sys.servers WHERE server_id=0
```

이 쿼리는 서비스를 다시 시작할 필요 없이 최신 서버 이름 정보를 제공합니다.

리전 및 버전 사용 가능 여부

CDC 기능은 RDS Custom for SQL Server를 사용할 수 있는 모든 AWS 리전과 RDS Custom에서 지원하는 모든 SQL Server 버전에서 지원됩니다. RDS Custom for SQL Server가 지원되는 버전 및 리전에 대한 자세한 내용은 RDS Custom for SQL Server를 지원하는 리전 및 DB 엔진 섹션을 참조하세요.

요구 사항 및 제한 사항

RDS Custom for SQL Server에서 CDC를 구현할 때는 다음 주요 고려 사항에 유의하세요.

- MS Replication과 같은 기능을 사용하도록 `sys.servers`에서 `@@SERVERNAME` 및/또는 로컬 서버를 수동으로 설정한 경우 `sys.servers`의 로컬 서버(`server_id = 0`이 있는 로컬 서버) 값이 `*.rds.amazonaws.com` 또는 `*.awsrds.*.com`과 일치하는 형식으로 설정했다면 RDS Custom for SQL Server는 `SERVERPROPERTY('ServerName')`와 일치하도록 수정을 시도하지 않습니다.
- 원격 로그인 또는 연결된 서버가 이전 호스트 이름을 적극적으로 사용하는 동안에는 RDS가 `sys.servers`의 로컬 서버(`server_id = 0`가 있는 로컬 서버)를 새 호스트 이름으로 수정할 수 없습니다. 이 제한은 두 가지 시나리오에 적용됩니다.
  - 연결된 서버가 이전 호스트 이름과 연결된 원격 로그인을 사용하여 로컬 서버에 대한 연결을 설정하는 경우
  - RDS Custom for SQL Server 인스턴스가 게시자 또는 배포자 역할을 하고 이전 호스트 이름과 연결된 로그인을 구독자 인스턴스에 연결한 경우

문제 해결

이전 서버 이름과 연결된 원격 로그인 또는 연결된 로그인을 식별하려면 다음 쿼리를 사용하세요. 적절한 CDC 기능을 보장하려면 결과를 검증하고 이러한 로그인을 제거하세요.

```
SELECT * FROM sys.remote_logins WHERE server_id=0
```

or

```
select sss.srvname,ssp.name,srl.remote_name  from sys.server_principals ssp
inner join sys.remote_logins srl on srl.local_principal_id=ssp.principal_id
inner join sys.sysservers sss  on srl.server_id = sss.srvid
where sss.srvname = @@SERVERNAME
```

Amazon RDS Custom for SQL Server를 위한 환경 설정

Amazon RDS Custom for SQL Server DB 인스턴스용 DB 인스턴스를 생성하고 관리하기 전에 다음 작업을 수행해야 합니다.

목차

- RDS Custom for SQL Server 설정에 필요한 사전 조건
  - AWS Management Console을 사용한 자동 인스턴스 프로파일 생성
- 1단계: IAM 보안 주체에 필요한 권한 부여
- 2단계: 네트워킹, 인스턴스 프로파일 및 암호화 구성
  - AWS CloudFormation을 사용한 구성
    - CloudFormation에 필요한 파라미터
    - AWS CloudFormation 템플릿 파일 다운로드
    - CloudFormation을 사용하여 리소스 구성
  - 수동으로 구성
    - 대칭 암호화 AWS KMS 키 보유 여부 확인
    - 수동으로 IAM 역할과 인스턴스 프로파일 생성
      - AWSRDSCustomSQLServerInstanceRole IAM 역할 생성
      - AWSRDSCustomSQLServerInstanceRole에 액세스 정책 추가
      - RDS Custom for SQL Server 인스턴스 프로파일 생성
      - RDS Custom for SQL Server 인스턴스 프로파일에 AWSRDSCustomSQLServerInstanceRole 추가
    - VPC 수동 구성
      - VPC 보안 그룹 구성
      - 종속 AWS 서비스의 엔드포인트 구성
      - 인스턴스 메타데이터 서비스 구성
- 크로스 인스턴스 제한 사항

> **Note**
>
> 사전 조건을 설정하고 Amazon RDS Custom for SQL Server를 시작하는 방법에 대한 단계별 튜토리얼을 보려면 Get started with Amazon RDS Custom for SQL Server using an

CloudFormation template (Network setup) 및 Explore the prerequisites required to create an Amazon RDS Custom for SQL Server instance를 참조하세요.

RDS Custom for SQL Server 설정에 필요한 사전 조건

RDS Custom for SQL Server DB 인스턴스를 생성하기 전에 환경이 이 주제에 설명된 요구 사항을 충족하는지 확인합니다. CloudFormation 템플릿을 사용하여 AWS 계정 내에서 사전 조건을 설정할 수도 있습니다. 자세한 내용은 AWS CloudFormation을 사용한 구성 섹션을 참조하세요.

RDS Custom for SQL Server를 사용하려면 다음과 같은 사전 조건을 구성해야 합니다.

- 인스턴스 생성에 필요한 AWS Identity and Access Management(IAM) 권한을 구성합니다. RDS에 `create-db-instance` 요청을 수행하는 데 필요한 AWS Identity and Access Management(IAM) 사용자 또는 역할입니다.
- RDS Custom for SQL Server DB 인스턴스에서 요구하는 사전 조건 리소스를 구성합니다.
  - RDS Custom 인스턴스의 암호화에 필요한 AWS KMS 키를 구성합니다. RDS Custom을 사용하려면 인스턴스 생성 시 암호화를 위한 고객 관리형 키가 필요합니다. RDS Custom DB 인스턴스 생성 요청 시 KMS 키 ARN, ID, 별칭 ARN 또는 별칭 이름이 `kms-key-id` 파라미터로 전달됩니다.
  - RDS Custom for SQL Server DB 인스턴스 내에서 필요한 권한을 구성합니다. RDS Custom은 DB 인스턴스 생성 시 인스턴스 프로파일을 DB 인스턴스에 연결하고 이를 DB 인스턴스 내 자동화에 사용합니다. 인스턴스 프로파일 이름은 RDS Custom 생성 요청에서 `custom-iam-instance-profile`로 설정됩니다. AWS Management Console에서 인스턴스 프로파일을 생성하거나 인스턴스 프로파일을 수동으로 생성할 수 있습니다. 자세한 내용은 AWS Management Console을 사용한 자동 인스턴스 프로파일 생성 및 수동으로 IAM 역할과 인스턴스 프로파일 생성(을)를 참조하세요.
  - RDS Custom for SQL Server의 요구 사항에 따라 네트워킹 설정을 구성합니다. RDS Custom 인스턴스는 인스턴스 생성 시 제공하는 서브넷(DB 서브넷 그룹으로 구성)에 있습니다. 이러한 서브넷은 RDS Custom 인스턴스가 RDS 자동화에 필요한 서비스와 통신할 수 있도록 허용해야 합니다.

> **Note**
>
> 위에서 언급한 요구 사항에 대해 계정 수준 권한을 제한하는 서비스 제어 정책(SCP)이 없어야 합니다.

사용 중인 계정이 AWS Organization의 일부인 경우 계정 수준 권한을 제한하는 서비스 제어 정책(SCP)이 있을 수 있습니다. SCP가 다음 절차를 사용하여 생성한 사용자 및 역할에 대한 권한을 제한하지 않도록 합니다.
SCP에 대한 자세한 내용은 AWS Organizations 사용 설명서에서 서비스 제어 정책(SCP)을 참조하세요. describe-organization AWS CLI 명령을 사용하여 계정이 AWS Organization의 일부인지 확인합니다.
AWS Organizations에 대한 자세한 내용을 알아보려면 AWS Organizations Organizations 사용 설명서의 AWS Organizations란 무엇입니까?를 참조하세요.

RDS Custom for SQL Server에 적용되는 일반 요구 사항은 RDS Custom for SQL Server 일반 요구 사항 섹션을 참조하세요.

AWS Management Console을 사용한 자동 인스턴스 프로파일 생성

RDS Custom을 사용하여 RDS Custom for SQL Server DB 인스턴스를 시작하려면 인스턴스 프로파일을 생성하고 구성해야 합니다. AWS Management Console을 사용하면 한 번에 새 인스턴스 프로파일을 만들고 연결할 수 있습니다. 이 옵션은 데이터베이스 생성, 스냅샷 복원 및 특정 시점으로 복원 콘솔 페이지의 RDS Custom 보안 섹션에서 사용할 수 있습니다. 새 인스턴스 프로파일 생성을 선택하여 인스턴스 프로파일 이름 접미사를 제공합니다. AWS Management Console은 RDS Custom 자동화 작업에 필요한 권한이 있는 새 인스턴스 프로파일을 생성합니다. 새 인스턴스 프로파일을 자동으로 만들려면 로그인한 AWS Management Console 사용자에게 `iam:CreateInstanceProfile`, `iam:AddRoleToInstanceProfile`, `iam:CreateRole` 및 `iam:AttachRolePolicy` 권한이 있어야 합니다.

> **Note**
>
> 이 옵션은 AWS Management Console에서만 사용할 수 있습니다. CLI 또는 SDK를 사용하는 경우 RDS Custom 제공 CloudFormation 템플릿을 사용하거나 인스턴스 프로파일을 수동으로 생성하세요. 자세한 내용은 수동으로 IAM 역할과 인스턴스 프로파일 생성 섹션을 참조하세요.

1단계: IAM 보안 주체에 필요한 권한 부여

RDS Custom 인스턴스를 생성하기에 충분한 액세스 권한을 가지고 있는지 확인하세요. 콘솔 또는 CLI를 사용하여 RDS Custom for SQL Server DB 인스턴스를 생성하는 데 필요한 IAM 역할 또는 IAM 사용자(IAM 보안 주체라고 함)는 DB 인스턴스를 성공적으로 생성하기 위해 다음 정책 중 하나를 가지고 있어야 합니다.

- `AdministratorAccess` 정책
- 다음과 같은 추가 권한이 있는 `AmazonRDSFullAccess` 정책:

```
iam:SimulatePrincipalPolicy
cloudtrail:CreateTrail
cloudtrail:StartLogging
s3:CreateBucket
s3:PutBucketPolicy
s3:PutBucketObjectLockConfiguration
s3:PutBucketVersioning
kms:CreateGrant
kms:DescribeKey
kms:Decrypt
kms:ReEncryptFrom
kms:ReEncryptTo
kms:GenerateDataKeyWithoutPlaintext
kms:GenerateDataKey
ec2:DescribeImages
ec2:RunInstances
ec2:CreateTags
```

RDS Custom은 인스턴스 생성 시 이러한 권한을 사용합니다. 이러한 권한은 RDS Custom 작업에 필요한 리소스를 계정에 구성합니다.

`kms:CreateGrant` 권한에 대한 자세한 내용은 AWS KMS key 관리 섹션을 참조하세요.

다음 샘플 JSON 정책은 필요한 권한을 부여합니다.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ValidateIamRole",
            "Effect": "Allow",
            "Action": "iam:SimulatePrincipalPolicy",
            "Resource": "*"
        },
        {
            "Sid": "CreateCloudTrail",
```

```
            "Effect": "Allow",
            "Action": [
                "cloudtrail:CreateTrail",
                "cloudtrail:StartLogging"
            ],
            "Resource": "arn:aws:cloudtrail:*:*:trail/do-not-delete-rds-custom-*"
        },
        {
            "Sid": "CreateS3Bucket",
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutBucketPolicy",
                "s3:PutBucketObjectLockConfiguration",
                "s3:PutBucketVersioning"
            ],
            "Resource": "arn:aws:s3:::do-not-delete-rds-custom-*"
        },
        {
            "Sid": "CreateKmsGrant",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        }
    ]
}
```

IAM 보안 주체가 사용자 지정 엔진 버전(CEV)을 사용하려면 다음과 같은 추가 권한이 필요합니다.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ConfigureKmsKeyEncryptionPermission",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
```

```
                "kms:DescribeKey",
                "kms:Decrypt",
                "kms:ReEncryptFrom",
                "kms:ReEncryptTo",
                "kms:GenerateDataKeyWithoutPlaintext",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:region:account_id:key/key_id"
        },
        {
            "Sid": "CreateEc2Instance",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeImages",
                "ec2:RunInstances",
                "ec2:CreateTags"
            ],
            "Resource": "*"
        }
    ]
}
```

인스턴스를 생성할 때 사용하는 계정의 ID로 *account_id*를 바꿉니다. 인스턴스를 생성할 때 사용하는 AWS 리전으로 *region*을 바꿉니다. 고객 관리형 키 ID로 *key_id*를 바꿉니다. 필요에 따라 키를 여러 개 추가할 수 있습니다.

EC2 인스턴스를 시작하는 데 필요한 리소스 수준 권한에 대한 자세한 내용은 인스턴스 실행(RunInstances)을 참조하세요.

또한 IAM 보안 주체는 IAM 역할에 `iam:PassRole` 권한이 필요합니다. RDS Custom DB 인스턴스를 생성하려면 요청의 `custom-iam-instance-profile` 파라미터에 전달된 인스턴스 프로파일에 첨부해야 합니다. 인스턴스 프로파일과 연결된 역할은 나중에 2단계: 네트워킹, 인스턴스 프로파일 및 암호화 구성에서 생성됩니다.

> **Note**
>
> 서비스 제어 정책(SCP), 권한 경계 또는 IAM 보안 주체와 연결된 세션 정책에서 이전에 나열된 권한을 제한하지 않는지 확인합니다.

2단계: 네트워킹, 인스턴스 프로파일 및 암호화 구성

다음 프로세스 중 하나를 사용하여 IAM 인스턴스 프로파일 역할, Virtual Private Cloud(VPC) 및 AWS KMS 대칭 암호화 키를 구성할 수 있습니다.

- AWS CloudFormation을 사용한 구성(권장)
- 수동으로 구성

> **Note**
>
> 계정이 AWS Organizations에 속한 경우 서비스 제어 정책(SCP)에서 인스턴스 프로파일 역할에 필요한 권한을 제한하지 않는지 확인해야 합니다.
> 이 주제의 네트워킹 구성은 공개적으로 액세스할 수 없는 DB 인스턴스에서 가장 잘 작동하도록 설계되었습니다. VPC 외부에서 이러한 DB 인스턴스에 직접 연결할 수 없습니다.

AWS CloudFormation을 사용한 구성

설정을 간소화하기 위해 AWS CloudFormation 템플릿 파일을 사용하여 CloudFormation 스택을 만들 수 있습니다. CloudFormation 템플릿은 RDS Custom의 요구 사항에 따라 모든 네트워킹, 인스턴스 프로파일, 암호화 리소스를 생성합니다.

스택을 생성하는 방법은 AWS CloudFormation 사용 설명서의 AWS CloudFormation 콘솔에서 스택 생성을 참조하세요.

AWS CloudFormation 템플릿을 사용하여 Amazon RDS Custom for SQL Server를 시작하는 방법에 대한 자습서는 AWS 데이터베이스 블로그에서 AWS CloudFormation 템플릿을 사용하여 Amazon RDS Custom for SQL Server 시작하기를 참조하세요.

주제

- CloudFormation에 필요한 파라미터
- AWS CloudFormation 템플릿 파일 다운로드
- CloudFormation을 사용하여 리소스 구성

CloudFormation에 필요한 파라미터

CloudFormation으로 RDS Custom 사전 조건 리소스를 구성하려면 다음 파라미터가 필요합니다.

Parameter Group	파라미터 이름	기본 값	설명
가용성 구성	사전 조건 설정을 위한 가용성 구성을 선택합니다.	Multi-AZ	RDS Custom 인스턴스에 대해 사전 조건을 단일 AZ 구성으로 설정할지 또는 다중 AZ 구성으로 설정할지 지정합니다. 이 구성에 다중 AZ DB 인스턴스가 하나 이상 필요한 경우 다중 AZ 구성을 사용해야 합니다.
네트워크 구성	VPC에 대한 IPv4 CIDR 블록	10.0.0.0/16	VPC에 대해 IPv4 CIDR 블록(또는 IP 주소 범위)을 지정합니다. 이 VPC는 RDS Custom DB 인스턴스를 생성하고 사용할 수 있도록 구성되어 있습니다.
	두 프라이빗 서브넷 중 첫 번째에 대한 IPv4 CIDR 블록	10.0.128.0/20	첫 번째 프라이빗 서브넷에 대해 IPv4 CIDR 블록(또는 IP 주소 범위)을 지정합니다. RDS Custom DB 인스턴스를 생성할 수 있는 두 서브넷 중 하나입니다. 인터넷에 액세스할 수 없는 프라이빗 서브넷입니다.
	두 프라이빗 서브넷 중 두 번째에 대한 IPv4 CIDR 블록	10.0.144.0/20	두 번째 프라이빗 서브넷에 대해 IPv4 CIDR 블록(또는 IP 주소 범위)을 지정합니다.

Parameter Group	파라미터 이름	기본 값	설명
			RDS Custom DB 인스턴스를 생성할 수 있는 두 서브넷 중 하나입니다. 인터넷에 액세스할 수 없는 프라이빗 서브넷입니다.
	퍼블릭 서브넷에 대한 IPv4 CIDR 블록	10.0.0.0/20	퍼블릭 서브넷에 대해 IPv4 CIDR 블록(또는 IP 주소 범위)을 지정합니다. RDS Custom DB 인스턴스와 연결할 수 있는 EC2 인스턴스를 생성할 수 있는 서브넷 중 하나입니다. 인터넷에 액세스할 수 있는 퍼블릭 서브넷입니다.
RDP 액세스 구성	소스의 IPv4 CIDR 블록	-	소스의 IPv4 CIDR 블록(또는 IP 주소 범위)을 지정합니다. 퍼블릭 서브넷에서 EC2 인스턴스에 RDP를 연결하는 IP 범위입니다. 설정하지 않으면 EC2 인스턴스에 대한 RDP 연결이 구성되지 않습니다.

Parameter Group	파라미터 이름	기본 값	설명
	RDS Custom for SQL Server 인스턴스에 대한 RDP 액세스 설정	아니요	EC2 인스턴스에서 RDS Custom for SQL Server 인스턴스로의 RDP 연결을 활성화할지 지정합니다. EC2 인스턴스에서 DB 인스턴스로의 RDP 연결은 기본적으로 구성되지 않습니다.

CloudFormation으로 생성된 리소스

기본 설정을 사용하여 CloudFormation 스택을 성공적으로 생성하면 AWS 계정에 다음 리소스가 생성됩니다.

- RDS Custom에서 관리하는 데이터의 암호화를 위한 대칭 암호화 KMS 키입니다.
- 인스턴스 프로파일은 RDS Custom에서 요구하는 권한을 제공하기 위해 `AmazonRDSCustomInstanceProfileRolePolicy`를 통해 IAM 역할에 연결됩니다. 자세한 내용은 AWS 관리형 정책 참조 안내서의 AmazonRDSCustomServiceRolePolicy를 참조하세요.
- CIDR 범위가 CloudFormation 파라미터로 지정된 VPC입니다. 기본값은 `10.0.0.0/16`입니다.
- 파라미터에 CIDR 범위가 지정된 2개의 프라이빗 서브넷과 AWS 리전에 있는 2개의 서로 다른 가용 영역입니다. 서브넷 CIDR의 기본값은 `10.0.128.0/20` 및 `10.0.144.0/20`입니다.
- 파라미터에 CIDR 범위가 지정된 퍼블릭 서브넷 1개. 서브넷 CIDR의 기본값은 10.0.0.0/20입니다. EC2 인스턴스는 이 서브넷에 있으며 RDS Custom 인스턴스에 연결하는 데 사용할 수 있습니다.
- Amazon 도메인 이름 시스템(DNS) 서버에 대한 도메인 이름 확인 기능이 있는 VPC에 설정된 DHCP 옵션입니다.
- 2개의 프라이빗 서브넷과 연결되며 인터넷에 액세스할 수 없는 라우팅 테이블입니다.
- 퍼블릭 서브넷과 연결되며 인터넷에 액세스할 수 있는 라우팅 테이블
- 퍼블릭 서브넷에 대한 인터넷 액세스를 허용하기 위해 VPC와 연결되는 인터넷 게이트웨이
- 두 프라이빗 서브넷과 연결되는 네트워크 액세스 제어 목록(ACL)과 HTTPS와 VPC 내의 DB 포트로 제한된 액세스

- RDS Custom 인스턴스와 연결할 VPC 보안 그룹입니다. 아웃바운드 HTTPS의 경우 RDS Custom과 EC2 인스턴스 보안 그룹의 인바운드 DB 포트에 필요한 AWS 서비스 엔드포인트로 액세스가 제한됩니다.
- 퍼블릭 서브넷의 EC2 인스턴스에 연결되는 VPC 보안 그룹. 아웃바운드 DB 포트의 경우 액세스가 RDS Custom 인스턴스 보안 그룹으로 제한됩니다.
- RDS Custom에 필요한 AWS 서비스 엔드포인트에 대해 생성된 VPC 엔드포인트와 연결할 VPC 보안 그룹입니다.
- RDS Custom 인스턴스가 생성되는 DB 서브넷 그룹입니다. 이 템플릿으로 생성된 2개의 프라이빗 서브넷이 DB 서브넷 그룹에 추가됩니다.
- RDS Custom에 필요한 각 AWS 서비스 엔드포인트에 대한 VPC 엔드포인트입니다.

가용성 구성을 multi-az로 설정하면 위 목록 외에도 다음과 같은 리소스가 생성됩니다.

- 프라이빗 서브넷 간 통신을 허용하는 네트워크 ACL 규칙
- RDS Custom 인스턴스와 연결된 VPC 보안 그룹 내 다중 AZ 포트에 대한 인바운드 및 아웃바운드 액세스
- 다중 AZ 통신에 필요한 AWS 서비스 엔드포인트로의 VPC 엔드포인트 연결

또한 RDP 액세스 구성을 설정하면 다음과 같은 리소스가 생성됩니다.

- 소스 IP 주소에서 퍼블릭 서브넷에 대한 RDP 액세스 구성:
  - 소스 IP에서 퍼블릭 서브넷으로의 RDP 연결을 허용하는 네트워크 ACL 규칙
  - 소스 IP에서 EC2 인스턴스와 연결된 VPC 보안 그룹으로의 RDP 포트 인그레스 액세스
- 퍼블릭 서브넷의 EC2 인스턴스에서 프라이빗 서브넷의 RDS Custom 인스턴스로의 RDP 액세스 구성:
  - 퍼블릭 서브넷에서 프라이빗 서브넷으로의 RDP 연결을 허용하는 네트워크 ACL 규칙
  - EC2 인스턴스와 연결된 VPC 보안 그룹에서 RDS Custom 인스턴스와 연결된 VPC 보안 그룹으로의 RDP 포트에 대한 인바운드 액세스

다음 절차에 따라 RDS Custom for SQL Server용 CloudFormation 스택을 생성합니다.

AWS CloudFormation 템플릿 파일 다운로드

템플릿 파일을 다운로드하려면

1. custom-sqlserver-onboard.zip 링크의 컨텍스트 메뉴(마우스 오른쪽 버튼 클릭)를 열고 Save Link As(다른 이름으로 링크 저장)를 선택합니다.
2. 파일을 컴퓨터에 저장하고 압축을 풉니다.

CloudFormation을 사용하여 리소스 구성

CloudFormation을 사용하여 리소스를 구성하려면

1. https://console.aws.amazon.com/cloudformation에서 CloudFormation 콘솔을 엽니다.
2. 스택 생성 마법사를 시작하려면 스택 생성(Create Stack)을 선택합니다.

   스택 생성(Create Stack) 페이지가 표시됩니다.
3. 사전 조건 - 템플릿 준비(Prerequisite - Prepare template)에서 템플릿 준비 완료를 선택합니다.
4. 템플릿 지정(Specify template)에서 다음 작업을 수행합니다.
   a. 템플릿 소스로 템플릿 파일 업로드를 선택합니다.
   b. 파일 선택에서 올바른 파일을 찾아 선택합니다.
5. 다음을 선택합니다.

   스택 세부 정보 지정(Specify stack details) 페이지가 나타납니다.
6. 스택 이름에 **`rds-custom-sqlserver`**을 입력합니다.
7. 파라미터(Parameters)에서 다음을 수행합니다.
   a. 기본 옵션을 유지하려면 다음(Next)을 선택합니다.
   b. 옵션을 변경하려면 적절한 가용성 구성, 네트워킹 구성 및 RDP 액세스 구성을 선택한 후 다음을 선택합니다.

      파라미터를 변경하기 전에 각 파라미터에 대한 설명을 주의 깊게 살펴봅니다.

> **Note**
>
> 이 CloudFormation 스택에서 하나 이상의 다중 AZ 인스턴스를 생성하는 경우, CloudFormation 스택 파라미터 사전 조건 설정을 위한 가용성 구성 선택이 `Multi-AZ`로

설정되어 있는지 확인하세요. CloudFormation 스택을 단일 AZ로 생성하는 경우, 첫 번째 다중 AZ 인스턴스를 생성하기 전에 CloudFormation 스택을 다중 AZ 구성으로 업데이트하세요.

8. 스택 옵션 구성 페이지에서 다음을 선택합니다.
9. rds-custom-sqlserver 검토(Review rds-custom-sqlserver) 페이지에서 다음을 수행합니다.
   a. 기능에서 이 사용자 지정 이름을 사용하여 AWS CloudFormation이 IAM 리소스를 생성할 수 있음에 동의합니다 확인란을 선택합니다.
   b. 스택 생성을 선택합니다.

> **Note**
>
> 이 AWS CloudFormation 스택에서 생성된 리소스를 리소스 페이지에서 직접 업데이트하지 마세요. 이렇게 하면 AWS CloudFormation 템플릿을 사용하여 이러한 리소스에 향후 업데이트를 적용할 수 없습니다.

CloudFormation은 RDS Custom for SQL Server에 필요한 리소스를 생성합니다. 스택 생성에 실패하면 이벤트(Events) 탭에서 실패한 리소스 생성 및 상태 이유를 확인할 수 있습니다.

콘솔의 이 CloudFormation 스택에 대한 출력(Outputs) 탭에는 RDS Custom for SQL Server DB 인스턴스를 생성하기 위해 파라미터로 전달할 모든 리소스 정보가 있어야 합니다. RDS Custom DB 인스턴스용 CloudFormation에서 생성한 VPC 보안 그룹과 DB 서브넷 그룹을 사용해야 합니다. 기본적으로 RDS는 기본 VPC 보안 그룹을 연결하려고 하는데, 여기에 필요한 액세스 권한이 없을 수 있습니다.

CloudFormation을 사용하여 리소스를 만든 경우 수동으로 구성을 건너뛸 수 있습니다.

CloudFormation 스택 업데이트

CloudFormation 스택을 만든 후 일부 구성을 업데이트할 수도 있습니다. 업데이트할 수 있는 구성은 다음과 같습니다.

- RDS Custom for SQL Server에 대한 가용성 구성
  - 사전 조건 설정을 위한 가용성 구성 선택: 단일 AZ 구성과 다중 AZ 구성 간에 전환하려면 이 파라미터를 업데이트하세요. 하나 이상의 다중 AZ 인스턴스에 대해 이 CloudFormation 스택을 사용하는 경우 다중 AZ 구성을 선택하도록 스택을 업데이트해야 합니다.
- RDS Custom for SQL Server에 대한 RDP 액세스 구성

- 소스의 IPv4 CIDR 블록: 이 파라미터를 업데이트하여 소스의 IPv4 CIDR 블록(또는 IP 주소 범위)을 업데이트할 수 있습니다. 이 파라미터를 공백으로 설정하면 소스 CIDR 블록에서 퍼블릭 서브넷으로의 RDP 액세스 구성이 제거됩니다.
- RDS Custom for SQL Server에 대한 RDP 액세스 설정: EC2 인스턴스에서 RDS Custom for SQL Server 인스턴스로의 RDP 연결을 활성화하거나 비활성화합니다.

CloudFormation 스택 삭제

스택에서 리소스를 사용하는 모든 RDS Custom 인스턴스를 삭제한 후 CloudFormation 스택을 삭제할 수 있습니다. RDS Custom은 CloudFormation 스택을 추적하지 않으므로 스택 리소스를 사용하는 DB 인스턴스가 있는 경우 스택 삭제를 차단하지 않습니다. 스택을 삭제할 때 스택 리소스를 사용하는 RDS Custom DB 인스턴스가 없는지 확인하세요.

> **Note**
>
> CloudFormation 스택을 삭제하면 KMS 키를 제외하고 스택에서 생성된 모든 리소스가 삭제됩니다. KMS 키는 삭제 대기 중 상태가 되고 30일 후에 삭제됩니다. KMS 키를 유지하려면 30일 유예 기간 동안 CancelKeyDeletion 작업을 수행합니다.

수동으로 구성

리소스를 수동으로 구성하도록 선택한 경우 다음 태스크를 수행합니다.

> **Note**
>
> 설정을 간소화하려는 경우, 수동 구성 대신 AWS CloudFormation 템플릿 파일을 사용하여 CloudFormation 스택을 만들 수 있습니다. 자세한 내용은 AWS CloudFormation을 사용한 구성 섹션을 참조하세요.
> AWS CLI를 사용하여 이 섹션을 완료할 수도 있습니다. 사용하려면 최신 CLI를 다운로드하고 설치하세요.

주제

- 대칭 암호화 AWS KMS 키 보유 여부 확인
- 수동으로 IAM 역할과 인스턴스 프로파일 생성
- VPC 수동 구성

대칭 암호화 AWS KMS 키 보유 여부 확인

대칭 암호화 AWS KMS key는 RDS Custom에 필수입니다. RDS Custom for SQL Server DB 인스턴스를 생성할 때는 KMS 키 식별자를 `kms-key-id` 파라미터로 제공해야 합니다. 자세한 내용은 Amazon RDS Custom for SQL Server의 DB 인스턴스 생성 및 연결 섹션을 참조하세요.

다음과 같은 옵션이 있습니다:

- 기존에 AWS 계정에 고객 관리형 KMS 키가 있는 경우 RDS Custom과 함께 사용할 수 있습니다. 별도로 조치를 취할 필요가 없습니다.
- 다른 RDS Custom 엔진에 대해 고객 관리형 대칭 암호화 KMS 키를 이미 생성한 경우 동일한 KMS 키를 재사용할 수 있습니다. 별도로 조치를 취할 필요가 없습니다.
- 계정에 사용 중이던 고객 관리형 대칭 암호화 KMS 키가 없는 경우 AWS Key Management Service 개발자 가이드의 키 생성 지침에 따라 KMS 키를 생성합니다.
- CEV 또는 RDS Custom DB 인스턴스를 생성하고 있으며 KMS 키가 다른 AWS 계정에 있는 경우 반드시 AWS CLI를 사용해야 합니다. 교차 계정 KMS 키에는 AWS 콘솔을 사용할 수 없습니다.

> **Important**
>
> RDS Custom은 AWS 관리형 KMS 키를 지원하지 않습니다.

사용하는 대칭 암호화 키는 `kms:Decrypt` 및 `kms:GenerateDataKey` 작업에 액세스할 수 있는 권한을 IAM 인스턴스 프로파일의 AWS Identity and Access Management(IAM) 역할에 제공해야 합니다. 계정에 대칭 암호화 키를 새로 사용하는 경우에는 변경할 필요가 없습니다. 아니면 대칭 암호화 키의 정책이 이러한 작업에 액세스할 권한을 부여할 수 있는지 확인하세요.

자세한 내용은 4단계: RDS Custom for Oracle의 IAM 구성 섹션을 참조하세요.

수동으로 IAM 역할과 인스턴스 프로파일 생성

인스턴스 프로파일을 수동으로 생성하고 이를 사용하여 RDS Custom 인스턴스를 시작할 수 있습니다. AWS Management Console에서 인스턴스를 생성하려는 경우 이 섹션을 건너뛰세요. AWS Management Console을 사용하면 인스턴스 프로파일을 생성하여 RDS Custom DB 인스턴스에 연결할 수 있습니다. 자세한 내용은 AWS Management Console을 사용한 자동 인스턴스 프로파일 생성 섹션을 참조하세요.

인스턴스 프로파일을 수동으로 생성할 때는 인스턴스 프로파일 이름을 `custom-iam-instance-profile` 파라미터로 `create-db-instance` CLI 명령에 전달하세요. RDS Custom은 이 인스턴스 프로파일과 연결된 역할을 사용하여 자동화를 실행함으로써 인스턴스를 관리합니다.

RDS Custom for SQL Server에 대한 IAM 인스턴스 프로파일과 IAM 역할을 생성하려면

1. Amazon EC2에서 역할을 위임하는 데 사용할 수 있는 신뢰 정책을 통해 이름이 `AWSRDSCustomSQLServerInstanceRole`인 IAM 역할을 생성합니다.
2. AWS 관리형 정책 `AmazonRDSCustomInstanceProfileRolePolicy`를 `AWSRDSCustomSQLServerInstanceRole`에 추가합니다.
3. 이름이 `AWSRDSCustomSQLServerInstanceProfile`인 RDS Custom for SQL Server의 IAM 인스턴스 프로파일을 생성합니다.
4. 인스턴스 프로파일에 `AWSRDSCustomSQLServerInstanceRole` 역할을 추가합니다.

AWSRDSCustomSQLServerInstanceRole IAM 역할 생성

다음 예제에서는 `AWSRDSCustomSQLServerInstanceRole` 역할을 생성합니다. 신뢰 정책을 사용하면 Amazon EC2가 역할을 위임하도록 할 수 있습니다.

```
aws iam create-role \
    --role-name AWSRDSCustomSQLServerInstanceRole \
    --assume-role-policy-document '{
        "Version": "2012-10-17",
          "Statement": [
            {
              "Action": "sts:AssumeRole",
              "Effect": "Allow",
              "Principal": {
                  "Service": "ec2.amazonaws.com"
              }
            }
          ]
        }'
```

AWSRDSCustomSQLServerInstanceRole에 액세스 정책 추가

필요한 권한을 제공하려면 AWS 관리형 정책 `AmazonRDSCustomInstanceProfileRolePolicy`를 `AWSRDSCustomSQLServerInstanceRole`에 연결하세요.

AmazonRDSCustomInstanceProfileRolePolicy는 RDS Custom 인스턴스가 메시지를 송수신하고 다양한 자동화 작업을 수행할 수 있도록 합니다.

> **Note**
>
> SCP 또는 인스턴스 프로파일 역할과 관련된 권한 경계가 액세스 정책의 권한을 제한하지 않는지 확인합니다.

다음 예에서는 AWS 관리형 정책 AmazonRDSCustomInstanceProfileRolePolicy를 AWSRDSCustomSQLServerInstanceRole 역할에 연결합니다.

```
aws iam attach-role-policy \
    --role-name AWSRDSCustomSQLServerInstanceRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonRDSCustomInstanceProfileRolePolicy
```

RDS Custom for SQL Server 인스턴스 프로파일 생성

인스턴스 프로파일은 단일 IAM 역할을 포함하는 컨테이너입니다. RDS Custom은 인스턴스 프로파일을 사용하여 인스턴스에 역할을 전달합니다.

AWS Management Console을 사용하여 Amazon EC2 역할을 생성하는 경우, 콘솔이 자동으로 인스턴스 프로파일을 생성하여 역할 생성 시 역할과 동일한 이름을 부여합니다. 다음과 같이 인스턴스 프로파일을 생성하고 이름을 AWSRDSCustomSQLServerInstanceProfile로 지정합니다.

```
aws iam create-instance-profile \
    --instance-profile-name AWSRDSCustomSQLServerInstanceProfile
```

RDS Custom for SQL Server 인스턴스 프로파일에 AWSRDSCustomSQLServerInstanceRole 추가

이전에 만든 AWSRDSCustomSQLServerInstanceProfile 프로파일에 AWSRDSCustomInstanceRoleForRdsCustomInstance 역할을 추가합니다.

```
aws iam add-role-to-instance-profile \
    --instance-profile-name AWSRDSCustomSQLServerInstanceProfile \
    --role-name AWSRDSCustomSQLServerInstanceRole
```

VPC 수동 구성

RDS Custom DB 인스턴스는 Amazon EC2 인스턴스 또는 Amazon RDS 인스턴스와 마찬가지로 Amazon VPC 서비스를 기반으로 하는 Virtual Private Cloud(VPC)에 있습니다. 자체 VPC를 제공하고 구성해야 인스턴스 네트워킹 설정을 완벽하게 제어할 수 있습니다.

RDS Custom은 DB 인스턴스에서 다른 AWS 서비스로 통신을 전송합니다. RDS Custom DB 인스턴스를 생성하는 서브넷에서 다음 서비스에 액세스할 수 있는지 확인합니다.

- Amazon CloudWatch(`com.amazonaws.`*`region`*`.monitoring`)
- Amazon CloudWatch Logs(`com.amazonaws.`*`region`*`.logs`)
- Amazon CloudWatch Events(`com.amazonaws.`*`region`*`.events`)
- Amazon EC2(`com.amazonaws.`*`region`*`.ec2` 및 `com.amazonaws.`*`region`*`.ec2messages`)
- Amazon EventBridge(`com.amazonaws.`*`region`*`.events`)
- Amazon S3(`com.amazonaws.`*`region`*`.s3`)
- AWS Secrets Manager (`com.amazonaws.`*`region`*`.secretsmanager`)
- AWS Systems Manager (`com.amazonaws.`*`region`*`.ssm` 및 `com.amazonaws.`*`region`*`.ssmmessages`)

다중 AZ 배포를 생성하는 경우

- Amazon Simple Queue Service(`com.amazonaws.`*`region`*`.sqs`)

필요한 서비스와 RDS Custom이 통신할 수 없는 경우 다음 이벤트가 게시됩니다.

```
Database instance in incompatible-network. SSM Agent connection not available. Amazon
 RDS can't connect to the dependent AWS services.
```

```
Database instance in incompatible-network. Amazon RDS can't connect to dependent AWS
 services. Make sure port 443 (HTTPS) allows outbound connections, and try again.
 "Failed to connect to the following services: s3 events"
```

`incompatible-network` 오류를 방지하려면 RDS Custom DB 인스턴스와 AWS 서비스 간의 통신과 관련된 VPC 구성 요소가 다음 요구 사항을 충족하는지 확인합니다.

- DB 인스턴스는 포트 443에서 다른 AWS 서비스로 아웃바운드 연결을 생성할 수 있습니다.

- VPC는 RDS Custom DB 인스턴스에서 시작된 요청에 대한 수신 응답을 허용합니다.
- RDS Custom는 각 AWS 서비스에 대한 엔드포인트의 도메인 이름을 정확하게 확인할 수 있습니다.

다른 RDS Custom DB 엔진의 VPC를 이미 구성한 경우 해당 VPC를 재사용하고 이 프로세스를 건너뛸 수 있습니다.

주제

- VPC 보안 그룹 구성
- 종속 AWS 서비스의 엔드포인트 구성
- 인스턴스 메타데이터 서비스 구성

VPC 보안 그룹 구성

보안 그룹은 VPC 인스턴스에 대한 가상 방화벽 역할을 하며 인바운드 및 아웃바운드 트래픽을 모두 제어합니다. RDS Custom DB 인스턴스에는 네트워크 인터페이스에 연결되어 인스턴스를 보호하는 보안 그룹이 있습니다. 보안 그룹이 HTTPS를 통해 RDS Custom과 다른 AWS 서비스 간의 트래픽을 허용하는지 확인해야 합니다. 인스턴스 생성 요청 시 이 보안 그룹을 `vpc-security-group-ids` 파라미터로 전달합니다.

RDS Custom의 보안 그룹을 구성하는 방법

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/vpc에서 Amazon VPC 콘솔을 엽니다.
2. RDS Custom에서 기본 보안 그룹을 사용하거나 고유한 보안 그룹을 생성하도록 허용합니다.

   자세한 지침은 보안 그룹을 생성하여 VPC 내부의 DB 인스턴스에 대한 액세스를 제공 섹션을 참조하세요.
3. 보안 그룹이 포트 443에서 아웃바운드 연결을 허용하는지 확인합니다. RDS Custom에서 종속 AWS 서비스와 통신하려면 이 포트가 필요합니다.
4. 프라이빗 VPC가 있고 VPC 엔드포인트를 사용하는 경우 DB 인스턴스와 연결된 보안 그룹이 포트 443에서 VPC 엔드포인트로의 아웃바운드 연결을 허용하는지 확인합니다. 또한 VPC 엔드포인트와 연결된 보안 그룹이 DB 인스턴스에서 포트 443의 인바운드 연결을 허용하는지 확인합니다.

   수신 연결을 허용하지 않으면 RDS Custom 인스턴스가 AWS Systems Manager 및 Amazon EC2 엔드포인트에 연결할 수 없습니다. 자세한 내용은 AWS Systems Manager 사용 설명서의 Virtual Private Cloud 엔드포인트 생성을 참조하세요.

5. RDS Custom for SQL Server 다중 AZ 인스턴스의 경우 DB 인스턴스와 연결된 보안 그룹이 포트 1120에서 이 보안 그룹 자체에 인바운드 및 아웃바운드 연결을 허용하는지 확인합니다. 이는 다중 AZ RDS Custom for SQL Server DB 인스턴스의 피어 호스트 연결에 필요합니다.

보안 그룹에 대한 자세한 내용은 Amazon VPC 개발자 안내서의 VPC의 보안 그룹을 참조하세요.

종속 AWS 서비스의 엔드포인트 구성

다음 지침에 따라 모든 서비스에 대한 엔드포인트를 VPC에 추가하는 것이 좋습니다. 그러나 VPC가 AWS 서비스 엔드포인트와 통신할 수 있도록 하는 모든 솔루션을 사용할 수 있습니다. 예를 들어, 네트워크 주소 변환(NAT) 또는 AWS Direct Connect를 사용할 수 있습니다.

RDS Custom이 작동하는 AWS 서비스의 엔드포인트를 구성하려면

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. 탐색 모음에서 리전 선택기를 사용하여 AWS 리전을 선택합니다.
3. 탐색 창에서 엔드포인트를 선택합니다. 기본 창에서 Create Endpoint(엔드포인트 생성)를 선택합니다.
4. 서비스 범주(Service category)에서 AWS 서비스를 선택합니다.
5. 서비스 이름으로 테이블에 나와 있는 엔드포인트를 선택합니다.
6. VPC에서 VPC를 선택합니다.
7. 서브넷에서 포함하고자 하는 각 가용 영역의 서브넷을 선택합니다.

   VPC 엔드포인트는 여러 가용 영역을 아우를 수 있습니다. AWS는 선택한 각 서브넷에 VPC 엔드포인트에 대한 탄력적 네트워크 인터페이스를 생성합니다. 각 네트워크 인터페이스에는 도메인 이름 시스템(DNS) 호스트 이름과 프라이빗 IP 주소가 있습니다.
8. 보안 그룹(Security group)에서 보안 그룹을 선택하거나 새로 만듭니다.

   보안 그룹을 사용하면 엔드포인트에 대한 액세스를 제어하므로 방화벽을 사용하는 것과 마찬가지입니다. 보안 그룹이 포트 443에서 DB 인스턴스로부터 인바운드 연결을 허용하는지 확인합니다. VPC 보안 그룹에 대한 자세한 내용은 Amazon VPC 사용 설명서의 VPC의 보안 그룹을 참조하세요.
9. 필요에 따라 정책을 VPC 엔드포인트에 연결할 수 있습니다. 엔드포인트정책은 연결 중인 AWS 서비스에 대한 액세스를 제어할 수 있습니다. 기본 정책은 모든 요청이 엔드포인트를 통과하도록 허용합니다. 사용자 지정 정책을 사용하는 경우 정책에서 DB 인스턴스의 요청이 허용되는지 확인합니다.
10. 엔드포인트 생성을 선택합니다.

다음 테이블에서는 VPC가 아웃바운드 통신에 사용해야 하는 엔드포인트 목록을 찾는 방법을 안내합니다.

Service	엔드포인트 형식	참고 및 링크
AWS Systems Manager	다음 엔드포인트 형식을 사용합니다. • ssm.*region*.amazonaws.com • ssmmessages. *region*.amazonaws.com	각 리전의 엔드포인트 목록은 Amazon Web Services 일반 참조에서 AWS Systems Manager 엔드포인트 및 할당량을 참조하세요.
AWS Secrets Manager	secretsmanager. *region*.amazonaws.com 엔드포인트 형식을 사용합니다.	각 리전의 엔드포인트 목록은 Amazon Web Services 일반 참조에서 AWS Secrets Manager 엔드포인트 및 할당량을 참조하세요.
Amazon CloudWatch	다음 엔드포인트 형식을 사용합니다. • CloudWatch 지표의 경우 monitoring. *region*.amazonaws.com 을 사용합니다. • CloudWatch Events의 경우 events.*region*.amazonaws.com 을 사용합니다. • CloudWatch Logs의 경우 logs.*region*.amazonaws.com 을 사용합니다.	모든 리전의 엔드포인트 목록은 다음을 참조하세요. • Amazon Web Services 일반 참조의 Amazon CloudWatch 엔드포인트 및 할당량 • Amazon Web Services 일반 참조의 Amazon CloudWatch Logs 엔드포인트 및 할당량 • Amazon Web Services 일반 참조의 Amazon CloudWatch Events 엔드포인트 및 할당량
Amazon EC2	다음 엔드포인트 형식을 사용합니다. • ec2.*region*.amazonaws.com • ec2messages. *region*.amazonaws.com	각 리전의 엔드포인트 목록은 Amazon Web Services 일반 참조의 Amazon Elastic Compute Cloud 엔드포인트 및 할당량을 참조하세요.
Amazon S3	s3.*region*.amazonaws.com 엔드포인트 형식을 사용합니다.	각 리전의 엔드포인트 목록은 Amazon Web Services 일반 참조에서 Amazon Simple Storage

Service	엔드포인트 형식	참고 및 링크
		Service 엔드포인트 및 할당량을 참조하세요. Amazon S3용 게이트웨이 엔드포인트에 대해 자세히 알아보려면 Amazon VPC 개발자 가이드에서 Amazon S3용 엔드포인트를 참조하세요. 액세스 포인트를 생성하는 방법은 Amazon VPC 개발자 안내서의 액세스 포인트 생성을 참조하세요. Amazon S3용 게이트웨이 엔드포인트를 생성하는 방법은 게이트웨이 VPC 엔드포인트를 참조하세요.
Amazon Simple Queue Service	`sqs.region.amazonaws.com` 엔드포인트 형식 사용	각 리전의 엔드포인트 목록은 Amazon Simple Queue Service endpoints and quotas를 참조하세요.

인스턴스 메타데이터 서비스 구성

인스턴스가 다음을 수행할 수 있는지 확인하세요.

- 인스턴스 메타데이터 서비스 버전 2(IMDSv2)를 사용하여 인스턴스 메타데이터 서비스에 액세스합니다.
- 포트 80(HTTP)을 통한 IMDS 링크 IP 주소로의 아웃바운드 통신을 허용합니다.
- IMDSv2 링크인 `http://169.254.169.254`에서 인스턴스 메타데이터를 요청합니다.

자세한 내용은 https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/configuring-instance-metadata-service.htmlAmazon EC2 사용 설명서의 IMDSv2 사용을 참조하세요.

크로스 인스턴스 제한 사항

위 단계에 따라 인스턴스 프로파일을 만들면 AWS 관리형 정책 `AmazonRDSCustomInstanceProfileRolePolicy`를 사용하여 RDS Custom에 필요한 권한을 제공하므로 인스턴스 관리 및 모니터링을 자동화할 수 있습니다. 이 관리형 정책은 RDS Custom이 자동화를 실행하는 데 필요한 리소스에만 권한을 부여하도록 합니다. 관리형 정책을 사용하여 새 기능을 지원하고 수동 개입 없이 기존 인스턴스 프로파일에 자동으로 적용되도록 하여 보안 요구 사항을 충족하는 것이 좋습니다. 자세한 내용은 AWS AWS 관리형 정책: AmazonRDSCustomInstanceProfileRolePolicy를 참조하세요.

`AmazonRDSCustomInstanceProfileRolePolicy` 관리형 정책은 인스턴스 프로파일을 크로스 계정 액세스로 제한하지만, 동일한 계정 내 RDS Custom 인스턴스에서 일부 RDS Custom 관리 리소스에 대한 액세스는 허용할 수 있습니다. 요구 사항에 따라 권한 경계를 사용하여 크로스 인스턴스 액세스를 추가로 제한할 수 있습니다. 권한 경계는 ID 기반 정책을 통해 엔터티에 부여할 수 있는 최대 권한을 정의하지만, 그 자체로 권한을 부여하지는 않습니다. 자세한 내용은 경계를 사용한 유효 권한 평가를 참조하세요.

예를 들어 다음 경계 정책은 인스턴스 프로파일 역할이 특정 AWS KMS 키에만 액세스하도록 제한하고 다른 AWS KMS 키를 사용하는 인스턴스 전체에서 RDS Custom 관리형 리소스에 대한 액세스를 제한합니다.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyOtherKmsKeyAccess",
            "Effect": "Deny",
            "Action": "kms:*",
            "NotResource": "arn:aws:kms:region:acct_id:key/KMS_key_ID"
        },
        {
            "Sid": "NoBoundarySetByDefault",
            "Effect": "Allow",
            "Action": "*",
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> `AmazonRDSCustomInstanceProfileRolePolicy`기 RDS Custom에 부여하는 권한을 권한 경계가 차단하지 않도록 하세요.

RDS Custom for SQL Server에서 기존 보유 미디어 사용(BYOM)

RDS Custom for SQL Server는 라이선스 포함(LI) 및 기존 보유 미디어 사용(BYOM)이라는 두 가지 라이선스 모델을 지원합니다.

BYOM을 사용하여 다음 작업을 수행할 수 있습니다.

1. 지원되는 누적 업데이트 (CU)와 함께 자체 Microsoft SQL Server 바이너리를 AWS EC2 Windows AMI에 제공하고 설치합니다.
2. AMI를 골든 이미지로 저장합니다. 골든 이미지는 사용자 지정 엔진 버전(CEV)을 생성하는 데 사용할 수 있는 템플릿입니다.
3. 골든 이미지로 CEV를 만듭니다.
4. CEV를 사용하여 새 RDS Custom for SQL Server DB 인스턴스를 생성합니다.

그러면 Amazon RDS가 사용자를 대신해 DB 인스턴스를 관리합니다.

> **Note**
>
> 또한 라이선스 포함(LI) RDS Custom for SQL Server DB 인스턴스가 있는 경우 이 DB 인스턴스의 SQL Server 소프트웨어를 BYOM과 함께 사용할 수 없습니다. 자체 SQL Server 바이너리를 BYOM에 가져와야 합니다.

RDS Custom for SQL Server에서 BYOM 사용 시 요구 사항

RDS Custom for SQL Server를 사용하는 사용자 지정 엔진 버전에 대한 일반 요구 사항이 BYOM에도 적용됩니다. 자세한 내용은 RDS Custom for SQL Server CEV 요구 사항 단원을 참조하십시오.

BYOM을 사용할 때는 다음과 같은 추가 요구 사항을 충족해야 합니다.

- 지원되는 다음 에디션(SQL Server 2022 또는 2019 Enterprise, Standard, Developer Edition) 중 하나를 사용하세요.
- SQL Server 시스템 관리자(SA) 서버 역할 권한을 `NT AUTHORITY\SYSTEM`에 부여합니다.
- EC2 Windows Server OS를 UTC 시간에 맞게 구성하여 유지합니다.

  기본 Amazon EC2 Windows 인스턴스는 기본적으로 UTC 표준 시간대로 설정됩니다. Windows 인스턴스의 시간을 보고 변경하는 것에 대한 자세한 내용은 Amazon EC2 사용 설명서의 Windows 인스턴스의 시간 설정을 참조하세요.

- SSM 연결을 허용하려면 TCP 포트 1433과 UDP 포트 1434를 엽니다.

RDS Custom for SQL Server에서 BYOM 사용 시 제한 사항

RDS Custom for SQL Server에 대한 일반 제한 사항이 BYOM에도 적용됩니다. 자세한 내용은 Amazon RDS Custom for SQL Server 요구 사항 및 제한 단원을 참조하십시오.

BYOM에는 다음과 같은 추가 제한 사항이 적용됩니다.

- 기본 SQL Server 인스턴스(MSSQLSERVER)만 지원됩니다. 이름이 지정된 SQL Server 인스턴스는 지원되지 않습니다. RDS Custom for SQL Server는 기본 SQL Server 인스턴스만 검색하고 모니터링합니다.
- 각 AMI에서는 SQL Server를 한 번만 설치할 수 있습니다. 서로 다른 SQL Server 버전을 여러 개 설치하는 것은 지원되지 않습니다.
- SQL Server Web Edition은 BYOM에서 지원되지 않습니다.
- SQL Server 에디션의 평가 버전은 BYOM에서 지원되지 않습니다. SQL Server를 설치할 때 평가 버전 사용 확인란을 선택하지 마시기 바랍니다.
- 기능 가용성 및 해당 지원은 각 데이터베이스 엔진의 특정 버전 및 AWS 리전 리전에 따라 다릅니다. 자세한 내용은 RDS Custom for SQL Server CEV에 대한 리전 가용성 및 RDS Custom for SQL Server CEV에 대한 버전 지원 단원을 참조하세요.

BYOM을 사용하여 RDS Custom for SQL Server DB 인스턴스 생성

BYOM을 사용하여 RDS Custom for SQL Server DB 인스턴스를 준비하고 생성하려면 기존 보유 미디어를 사용(BYOM)하여 CEV 준비 섹션을 참조하세요.

RDS Custom for SQL Server 사용자 지정 엔진 버전 작업

RDS Custom for SQL Server용 사용자 지정 엔진 버전(CEV)은 Microsoft SQL Server가 포함된 Amazon Machine Image(AMI)입니다.

CEV 워크플로의 기본 단계는 다음과 같습니다.

1. CEV의 기본 이미지로 사용할 AWS EC2 Windows AMI를 선택합니다. 사전 설치된 Microsoft SQL Server를 사용하거나 자체 미디어를 가져와 SQL Server를 직접 설치할 수 있습니다.
2. 운영 체제(OS)에 다른 소프트웨어를 설치하고 기업 요구 사항에 맞게 OS 및 SQL Server의 구성을 사용자 지정합니다.
3. AMI를 골든 이미지로 저장
4. 골든 이미지로 커스텀 엔진 버전(CEV)을 만듭니다.
5. CEV를 사용하여 새 RDS Custom for SQL Server DB 인스턴스를 생성합니다.

이렇게 하면 Amazon RDS가 사용자를 대신해 이러한 DB 인스턴스를 관리합니다.

CEV를 사용하면 OS와 데이터베이스의 기본 구성을 원하는 대로 유지할 수 있습니다. CEV를 사용하면 타사 에이전트 설치 또는 기타 OS 사용자 지정 같은 호스트 구성이 RDS Custom for SQL Server DB 인스턴스에서 유지됩니다. CEV를 사용하면 동일한 구성의 RDS Custom for SQL Server DB 플릿을 빠르게 배포할 수 있습니다.

주제

- RDS Custom for SQL Server용 CEV 생성 준비
- RDS Custom for SQL Server CEV 생성
- RDS Custom for SQL Server용 CEV 수정
- Amazon RDS Custom for SQL Server의 CEV 세부 정보 보기
- RDS Custom for SQL Server용 CEV 삭제

RDS Custom for SQL Server용 CEV 생성 준비

사전 설치된 라이선스 포함(LI) Microsoft SQL Server가 속한 Amazon Machine Image(AMI)를 사용하거나 자체 SQL Server 설치 미디어(BYOM)가 설치될 AMI를 사용하여 CEV를 생성할 수 있습니다.

CEV 준비

다음 절차에 따라 기존 보유 미디어 사용(BYOM) 또는 사전 설치된 Microsoft SQL Server(LI)를 사용하여 CEV를 생성합니다.

기존 보유 미디어를 사용(BYOM)하여 CEV 준비

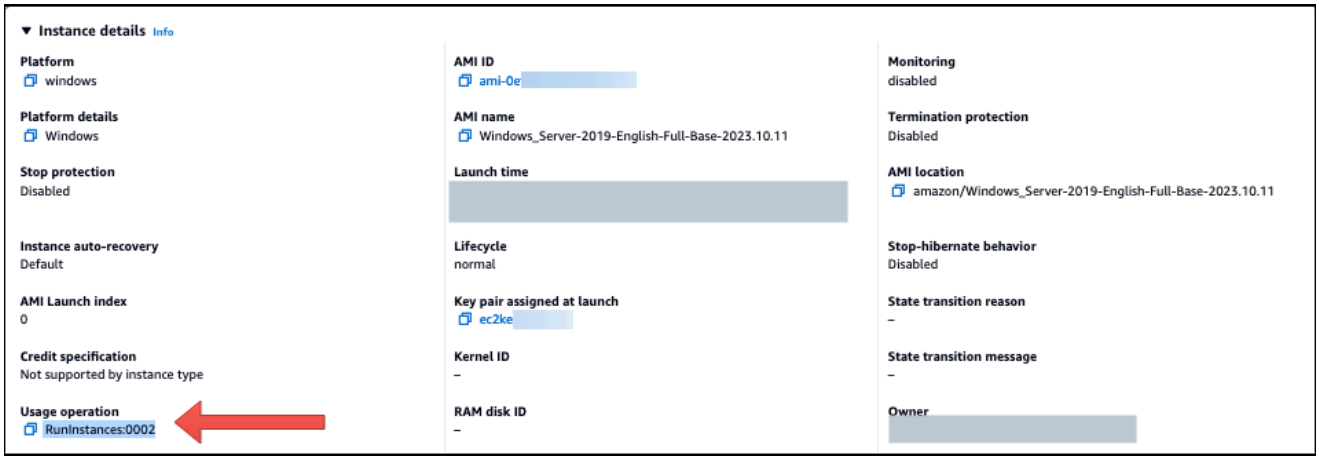
다음 단계에서는 Windows Server 2019 Base가 포함된 AMI를 예로 사용합니다.

BYOM을 사용하여 CEV를 만드는 방법

1. Amazon EC2 콘솔 대시보드에서 인스턴스 시작을 선택합니다.
2. 이름에 인스턴스 이름을 입력합니다.
3. 빠른 시작에서 Windows를 선택합니다.
4. Microsoft Windows Server 2019 Base를 선택합니다.
5. 적절한 인스턴스 유형, 키 페어, 네트워크 및 스토리지 설정을 선택하고 인스턴스를 시작합니다.
6. EC2 인스턴스를 시작하거나 생성한 후 4단계에서 올바른 Windows AMI를 선택했는지 확인합니다.

   a. Amazon EC2 콘솔에서 EC2 인스턴스를 선택합니다.

   b. 세부 정보 섹션에서 사용 작업을 확인하고 해당 작업이 RunInstances:0002로 설정되어 있는지 확인합니다.



7. EC2 인스턴스에 로그인하고 SQL Server 설치 미디어를 해당 인스턴스에 복사합니다.

> **Note**
>
> SQL Server Developer 에디션을 사용하여 CEV를 구축하는 경우 Microsoft Visual Studio 구독을 사용하여 설치 미디어를 구해야 할 수 있습니다.

8. SQL Server를 설치합니다. 다음을 수행하세요.
   a. RDS Custom for SQL Server에서 BYOM 사용 시 요구 사항 및 RDS Custom for SQL Server CEV에 대한 버전 지원를 검토합니다.
   b. 인스턴스 루트 디렉터리를 기본값인 `C:\Program Files\Microsoft SQL Server\`로 설정합니다. 이 디렉터리를 변경하지 마세요.
   c. SQL Server 데이터베이스 엔진 계정 이름을 `NT Service\MSSQLSERVER` 또는 `NT AUTHORITY\NETWORK SERVICE`로 설정합니다.
   d. SQL Server 시작 모드를 수동으로 설정합니다.
   e. SQL Server 인증 모드를 혼합으로 선택합니다.
   f. 기본 데이터 디렉터리 및 TempDB 위치에 대한 기존 설정을 그대로 유지합니다.
9. SQL Server 시스템 관리자(SA) 서버 역할 권한을 `NT AUTHORITY\SYSTEM`에 부여합니다.

```
USE [master]
GO
EXEC master..sp_addsrvrolemember @loginame = N'NT AUTHORITY\SYSTEM' , @rolename =
 N'sysadmin'
GO
```

10. 추가 소프트웨어를 설치하거나 요구 사항에 맞게 OS 및 데이터베이스 구성을 사용자 지정합니다.
11. EC2 인스턴스에서 Sysprep을 실행합니다. 자세한 내용은 Create an Amazon EC2 AMI using Windows Sysprep을 참조하세요.
12. 설치된 SQL Server 버전, 기타 소프트웨어 및 사용자 지정이 포함된 AMI를 저장합니다. 이것이 골든 이미지가 됩니다.
13. 생성한 이미지의 AMI ID를 제공하여 새 CEV를 생성합니다. 자세한 단계는 RDS Custom for SQL Server CEV 생성 섹션을 참조하세요.
14. CEV를 사용하여 새 RDS Custom for SQL Server DB 인스턴스를 생성합니다. 자세한 단계는 CEV에서 RDS Custom for SQL Server DB 인스턴스 생성 섹션을 참조하세요.

사전 설치된 SQL Server(LI)를 사용하여 CEV 준비

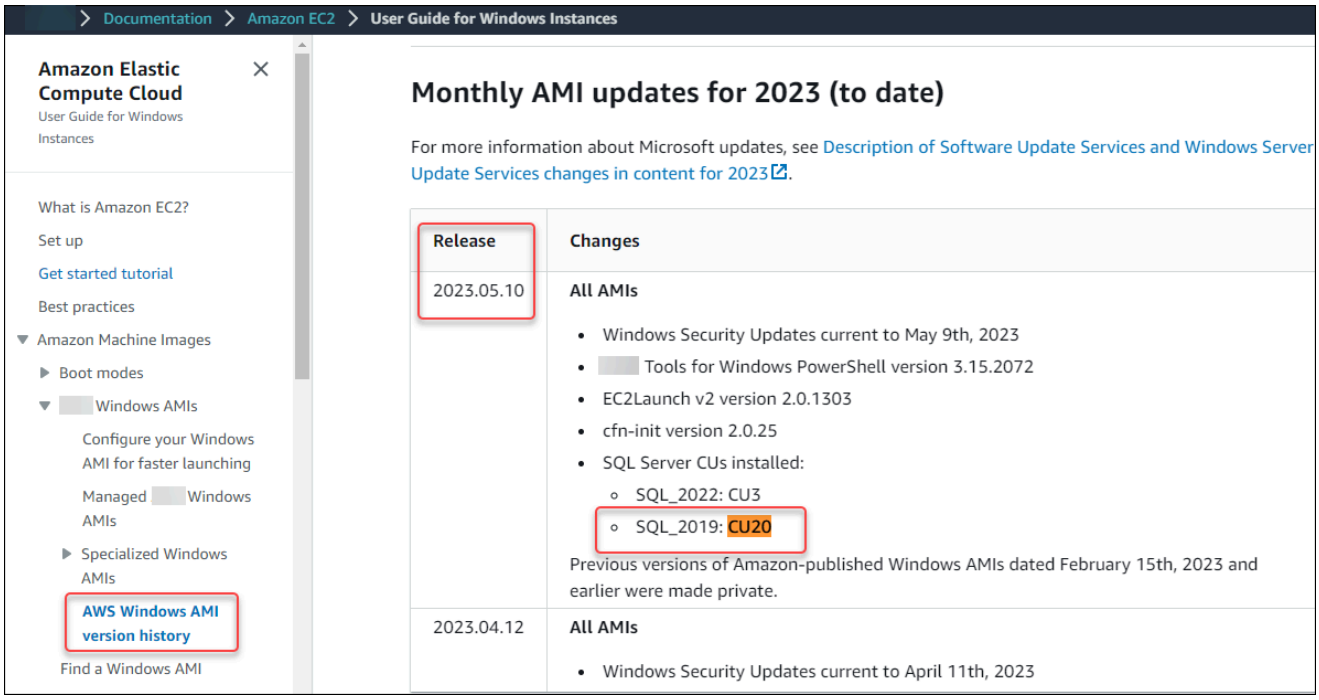
사전 설치된 Microsoft SQL Server(LI)를 사용하여 CEV를 생성하는 다음 단계에서는 SQL Server CU20 릴리스 번호가 `2023.05.10`인 AMI를 예로 사용합니다. CEV를 생성할 때는 릴리스 번호가 가장 최신인 AMI를 선택하세요. 이렇게 하면 지원되는 버전의 Windows Server 및 SQL Server를 최신 누적 업데이트(CU)와 함께 사용할 수 있습니다.

사전 설치된 Microsoft SQL Server(LI)를 사용하여 CEV를 만드는 방법

1. 라이선스 포함(LI) Microsoft Windows Server 및 SQL Server가 있는 최신 AWS EC2 Windows Amazon Machine Image(AMI)를 선택합니다.

   a. Windows AMI 버전 기록에서 CU20을 검색합니다.

   b. 릴리스 번호를 기록해 두세요. SQL Server 2019 CU20의 릴리스 번호는 `2023.05.10`입니다.



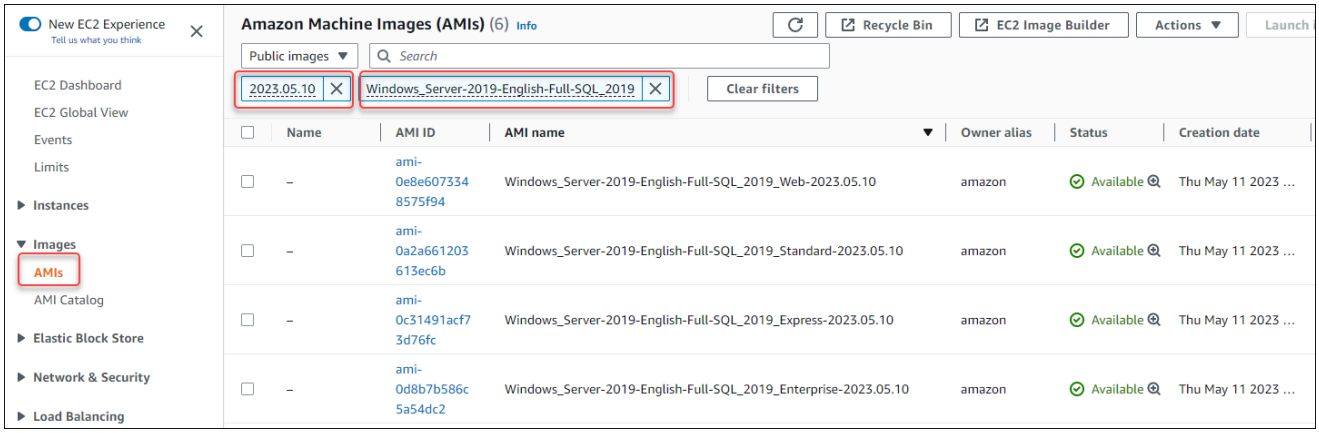
   c. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.

   d. Amazon EC2 콘솔의 왼쪽 탐색 창에서 Images(이미지)를 선택하고 AMIs를 선택합니다.

   e. 퍼블릭 이미지를 선택합니다.

   f. 검색 상자에 `2023.05.10`를 입력합니다. AMI 목록이 나타납니다.

   g. 검색 상자에 `Windows_Server-2019-English-Full-SQL_2019`를 입력하여 결과를 필터링합니다. 다음과 같은 결과가 표시됩니다.



h. 사용할 SQL Server 에디션이 있는 AMI를 선택합니다.

2. 선택한 AMI에서 EC2 인스턴스를 생성하거나 시작합니다.
3. EC2 인스턴스에 로그인하여 추가 소프트웨어를 설치하거나 요구 사항에 맞게 OS 및 데이터베이스 구성을 사용자 지정합니다.
4. EC2 인스턴스에서 Sysprep을 실행합니다. Sysprep을 사용한 AMI 준비에 대한 자세한 내용은 Sysprep을 사용하여 표준화된 Amazon Machine Image(AMI) 생성을 참조하세요.
5. 설치된 SQL Server 버전, 기타 소프트웨어 및 사용자 지정이 포함된 AMI를 저장합니다. 이것이 골든 이미지가 됩니다.
6. 생성한 이미지의 AMI ID를 제공하여 새 CEV를 생성합니다. CEV 생성에 대한 자세한 단계는 RDS Custom for SQL Server CEV 생성 섹션을 참조하세요.
7. CEV를 사용하여 새 RDS Custom for SQL Server DB 인스턴스를 생성합니다. 자세한 단계는 CEV에서 RDS Custom for SQL Server DB 인스턴스 생성 섹션을 참조하세요.

RDS Custom for SQL Server CEV에 대한 리전 가용성

RDS Custom for SQL Server에 대한 사용자 지정 엔진 버전(CEV) 지원은 다음 AWS 리전에서 제공됩니다.

- 미국 동부(오하이오)
- 미국 동부(버지니아 북부)
- 미국 서부(오리건)
- 미국 서부(캘리포니아 북부)
- 아시아 태평양(뭄바이)
- 아시아 태평양(오사카)
- 아시아 태평양(서울)

- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(도쿄)
- 캐나다(중부)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(파리)
- 유럽(스톡홀름)
- 남아메리카(상파울루)

RDS Custom for SQL Server CEV에 대한 버전 지원

RDS Custom for SQL Server용 CEV 생성은 다음 AWS EC2 Windows AMI에서 지원됩니다.

- 사전 설치된 미디어를 사용하는 CEV의 경우 라이선스 포함(LI) Microsoft Windows Server 2019(OS) 및 SQL Server 2022 또는 2019가 포함된 AWS EC2 Windows AMI
- 기존 보유 미디어를 사용(BYOM)하는 CEV의 경우 Microsoft Windows Server 2019(OS)가 포함된 AWS EC2 Windows AMI

RDS Custom for SQL Server용 CEV 생성은 다음 운영 체제(OS) 및 데이터베이스 에디션에 지원됩니다.

- 사전 설치된 미디어를 사용하는 CEV의 경우:
  - SQL Server 2022 Enterprise, Standard 또는 Web과 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19.
  - SQL Server 2019 Enterprise, Standard 또는 Web과 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32
- 기존 보유 미디어를 사용(BYOM)하는 CEV의 경우:
  - SQL Server 2022 Enterprise, Standard 또는 Developer와 CU9, CU13, CU14-GDR, CU15-GDR, CU16, CU17, CU18 및 CU19.
  - SQL Server 2019 Enterprise, Standard 또는 Developer와 CU8, CU17, CU18, CU20, CU24, CU26, CU28-GDR, CU29-GDR, CU30 및 CU32

- 사전 설치된 미디어 또는 기존 보유 미디어를 사용(BYOM)하는 CEV의 경우 OS로 Windows Server 2019가 유일하게 지원됨

자세한 내용은 AWS Windows AMI 버전 기록을 참조하세요.

RDS Custom for SQL Server CEV 요구 사항

다음 요구 사항은 RDS Custom for SQL Server용 CEV 생성에 적용됩니다.

- CEV를 생성하는 데 사용하는 AMI는 RDS Custom for SQL Server에서 지원하는 OS 및 데이터베이스 구성을 기반으로 해야 합니다. 지원되는 구성에 대한 자세한 내용은 Amazon RDS Custom for SQL Server 요구 사항 및 제한 섹션을 참조하세요.
- CEV에는 고유한 이름이 있어야 합니다. 기존 CEV와 이름이 같은 CEV는 생성할 수 없습니다.
- SQL Server 메이저 버전 + 마이너 버전 + 사용자 지정 문자열이라는 이름 지정 패턴을 사용하여 CEV 이름을 지정해야 합니다. 메이저 버전+마이너 버전은 AMI와 함께 제공된 SQL Server 버전과 일치해야 합니다. 예를 들어 SQL Server 2019 CU17을 사용하는 AMI의 이름은 15.00.4249.2.my_cevtest로 지정할 수 있습니다.
- Sysprep을 사용한 AMI를 준비해야 합니다. Sysprep을 사용한 AMI 준비에 대한 자세한 내용은 Sysprep을 사용하여 표준화된 Amazon Machine Image(AMI) 생성을 참조하세요.
- AMI의 수명 주기를 관리할 책임은 사용자에게 있습니다. CEV에서 생성된 RDS Custom for SQL Server DB 인스턴스에는 AMI 사본이 저장되지 않습니다. CEV를 생성하는 데 사용한 AMI에 대한 포인터를 유지합니다. RDS Custom for SQL Server DB 인스턴스가 계속 작동하려면 AMI가 있어야 합니다.

RDS Custom for SQL Server CEV 제한 사항

RDS Custom for SQL Server를 사용하는 사용자 지정 엔진 버전에는 다음과 같은 제한 사항이 적용됩니다.

- 연결된 DB 인스턴스 또는 DB 스냅샷 같은 리소스가 있는 경우 CEV를 삭제할 수 없습니다.
- RDS Custom for SQL Server DB 인스턴스를 생성하려면 CEV는 상태가 `pending-validation`, `available`, `failed` 또는 `validating`이어야 합니다. CEV 상태가 `incompatible-image-configuration`인 경우에는 CEV를 사용하여 RDS Custom for SQL Server DB 인스턴스를 생성할 수 없습니다.
- 새 CEV를 사용하도록 RDS Custom for SQL Server DB 인스턴스를 수정하려면 CEV 상태가 `available`이어야 합니다.

- 기존 RDS Custom for SQL Server DB 인스턴스에서 AMI 또는 CEV를 생성할 수 없습니다.
- 다른 AMI를 사용하도록 기존 CEV를 수정할 수는 없습니다. 그러나 RDS Custom for SQL Server DB 인스턴스를 수정하여 다른 CEV를 사용할 수는 있습니다. 자세한 내용은 RDS Custom for SQL Server DB 인스턴스 수정 섹션을 참조하세요.
- DB 인스턴스 생성 중에 제공된 KMS 키와 다른 고객 관리형 KMS 키로 AMI 또는 CEV를 암호화하는 것은 지원되지 않습니다.
- CEV의 교차 리전 복사는 지원되지 않습니다.
- CEV의 교차 계정 복사는 지원되지 않습니다.
- 삭제한 CEV는 복원 또는 복구할 수 없습니다. 하지만 동일한 AMI에서 새 CEV를 생성할 수 있습니다.
- RDS Custom for SQL Server DB 인스턴스는 SQL Server 데이터베이스 파일을 D:\ 드라이브에 저장합니다. CEV와 연결된 AMI는 Microsoft SQL Server 시스템 데이터베이스 파일을 C:\ 드라이브에 저장해야 합니다.
- RDS Custom for SQL Server DB 인스턴스는 SQL Server 구성 변경 사항을 유지합니다. CEV에서 생성한 RDS Custom for SQL Server DB 인스턴스에서 실행 중인 운영 체제의 구성 변경 사항은 유지되지 않습니다. OS의 구성을 영구적으로 변경하고 이를 새 기본 구성으로 유지해야 하는 경우, 새 CEV를 생성한 다음 새 CEV를 사용하도록 DB 인스턴스를 수정하십시오.

  > **Important**
  >
  > 새 CEV를 사용하도록 RDS Custom for SQL Server DB 인스턴스를 수정하는 작업은 오프라인 작업입니다. 즉시 수정을 수행하거나 주간 유지 관리 기간 중에 수정하도록 예약할 수 있습니다.

- CEV를 수정할 때 Amazon RDS는 해당 수정 사항을 관련 RDS Custom for SQL Server DB 인스턴스에 푸시하지 않습니다. 신규 또는 업데이트된 CEV를 사용하려면 각 RDS Custom for SQL Server DB 인스턴스를 수정해야 합니다. 자세한 내용은 RDS Custom for SQL Server DB 인스턴스 수정 섹션을 참조하세요.
-

  > **Important**
  >
  > CEV에서 사용하는 AMI가 삭제되면 호스트 교체가 필요할 수 있는 모든 수정(예: 컴퓨팅 크기 조정)이 실패하게 됩니다. 이렇게 되면 RDS Custom for SQL Server DB 인스턴스가 RDS 지원 경계 외부에 배치됩니다. CEV와 연결된 AMI는 삭제하지 않는 것이 좋습니다.

RDS Custom for SQL Server CEV 생성

AWS Management Console 또는 AWS CLI를 사용하여 사용자 지정 엔진 버전(CEV)을 생성할 수 있습니다. 그런 다음 CEV를 사용하여 RDS Custom for SQL Server DB 인스턴스를 생성할 수 있습니다.

Amazon Machine Image(AMI)가 CEV와 동일한 AWS 계정 및 리전에 있는지 확인합니다. 그렇지 않으면 CEV 생성 프로세스가 실패합니다.

자세한 내용은 Amazon RDS Custom for SQL Server의 DB 인스턴스 생성 및 연결 단원을 참조하십시오.

> **Important**
>
> CEV를 만드는 단계는 사전 설치된 SQL Server로 만든 AMI 및 기존 보유 미디어를 사용(BYOM)하여 만든 AMI와 동일합니다.

콘솔

CEV를 생성하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 사용자 지정 엔진 버전(Custom engine versions)을 선택합니다.

   사용자 지정 엔진 버전(Custom engine versions) 페이지에는 현재 존재하는 모든 CEV가 표시됩니다. CEV를 생성하지 않았다면 테이블은 비어 있습니다.
3. Create custom engine version(사용자 지정 엔진 버전 생성)을 선택합니다.
4. Engine type(엔진 유형)에서 Microsoft SQL Server를 선택합니다.
5. 에디션에서 사용할 DB 엔진 에디션을 선택합니다.
6. Major version(메이저 버전)에서 AMI에 설치된 메이저 엔진 버전을 선택합니다.
7. 버전 세부 정보(Version details)의 사용자 지정 엔진 버전 이름(Custom engine version name)에 유효한 이름을 입력합니다.

   이름 형식은 *major-engine-version*.*minor-engine-version*.*customized_string*입니다. 1~50개의 영숫자, 밑줄, 대시 및 마침표를 사용할 수 있습니다. 예를 들어, **15.00.4249.2.my_cevtest**을 이름으로 입력할 수 있습니다.

필요에 따라 CEV에 대한 설명을 입력합니다.

8. Installation Media(설치 미디어)에서 CEV를 생성할 때 사용할 AMI ID를 찾거나 입력합니다.
9. Tags(태그) 섹션에서 CEV를 식별하는 태그를 추가합니다.
10. Create custom engine version(사용자 지정 엔진 버전 생성)을 선택합니다.

사용자 지정 엔진 버전(Custom engine versions) 페이지가 표시됩니다. CEV가 pending-validation(검증 보류 중) 상태로 표시됩니다.

AWS CLI

AWS CLI를 사용하여 CEV를 생성하려면 create-custom-db-engine-version 명령을 실행합니다.

다음 옵션이 필요합니다.

- `--engine`
- `--engine-version`
- `--image-id`

다음 옵션도 지정할 수 있습니다.

- `--description`
- `--region`
- `--tags`

다음 예제에서는 `15.00.4249.2.my_cevtest`이라는 CEV를 생성합니다. CEV의 이름이 주요 엔진 버전 번호로 시작하는지 확인하세요.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-custom-db-engine-version \
    --engine custom-sqlserver-ee \
    --engine-version 15.00.4249.2.my_cevtest \
    --image-id ami-0r93cx31t5r596482 \
    --description "Custom SQL Server EE 15.00.4249.2 cev test"
```

다음 부분 출력은 엔진, 파라미터 그룹 및 기타 정보를 보여줍니다.

```
"DBEngineVersions": [
    {
    "Engine": "custom-sqlserver-ee",
    "MajorEngineVersion": "15.00",
    "EngineVersion": "15.00.4249.2.my_cevtest",
    "DBEngineDescription": "Microsoft SQL Server Enterprise Edition for RDS Custom for
 SQL Server",
    "DBEngineVersionArn": "arn:aws:rds:us-east-1:<my-account-id>:cev:custom-sqlserver-
ee/15.00.4249.2.my_cevtest/a1234a1-123c-12rd-bre1-1234567890",
    "DBEngineVersionDescription": "Custom SQL Server EE 15.00.4249.2 cev test",

    "Image": [
        "ImageId": "ami-0r93cx31t5r596482",
        "Status": "pending-validation"
     ],
    "CreateTime": "2022-11-20T19:30:01.831000+00:00",
    "SupportsLogExportsToCloudwatchLogs": false,
    "SupportsReadReplica": false,
    "Status": "pending-validation",
    "SupportsParallelQuery": false,
    "SupportsGlobalDatabases": false,
    "TagList": []
    }
]
```

CEV 생성 프로세스가 실패하면 RDS Custom for SQL Server가 `Creation failed for custom engine version` *`major-engine-version.cev_name`* 메시지와 함께 RDS-EVENT-0198을 발급합니다. 메시지에는 실패(예: 이벤트가 누락된 파일 인쇄) 관련 세부 정보가 포함되어 있습니다. CEV 생성 문제 해결 아이디어를 찾으려면 RDS Custom for SQL Server의 CEV 오류 문제 해결 단원을 참조하세요.

CEV에서 RDS Custom for SQL Server DB 인스턴스 생성

CEV를 성공적으로 생성하면 CEV status(CEV 상태)에 `pending-validation`이 표시됩니다. 이제 CEV를 사용하여 새 RDS Custom for SQL Server DB 인스턴스를 생성할 수 있습니다. CEV에서 새 RDS Custom for SQL Server DB 인스턴스를 생성하려면 RDS Custom for SQL Server DB 인스턴스 생성 단원을 참조하세요.

CEV의 수명 주기

CEV 수명 주기에는 다음 상태가 포함됩니다.

CEV 상태	설명	문제 해결 제안
`pending-validation`	CEV가 생성되었고 연결된 AMI의 검증이 보류 중입니다. CEV는 RDS Custom for SQL Server DB 인스턴스가 생성될 때까지 `pending-validation` 상태를 유지됩니다.	기존 작업이 없는 경우 CEV에서 새 RDS Custom for SQL Server DB 인스턴스를 생성하세요. RDS Custom for SQL Server DB 인스턴스를 생성할 때 시스템은 CEV에 연결된 AMI 검증을 시도합니다.
`validating`	새 CEV를 기반으로 하는 RDS Custom for SQL Server DB 인스턴스의 생성 작업이 진행 중입니다. RDS Custom for SQL Server DB 인스턴스를 생성할 때 시스템은 CEV의 연결된 AMI를 검증하려고 합니다.	RDS Custom for SQL Server DB 인스턴스 생성 작업이 완료될 때까지 기다리세요. RDS EVENTS 콘솔을 사용하여 문제 해결을 위한 세부 이벤트 메시지를 검토할 수 있습니다.
`available`	CEV가 성공적으로 검증되었습니다. CEV에서 RDS Custom for SQL Server	CEV는 추가 검증을 하지 않아도 됩니다. CEV는 RDS Custom for SQL Server DB 인스턴스를 추가로 생성하거나 기존 인스턴스를 수정하는 데 사용할 수 있습니다.

CEV 상태	설명	문제 해결 제안	
	DB 인스턴스가 성공적으로 생성되면 CEV가 `available` 상태가 됩니다.		
`inactive`	CEV가 비활성 상태로 수정되었습니다.	이 CEV에서는 RDS Custom DB 인스턴스를 생성하거나 업그레이드할 수 없습니다. 또한 이 CEV에서는 DB 스냅샷을 복원하여 새로운 RDS Custom DB 인스턴스를 생성할 수 없습니다. 상태를 `ACTIVE`로 변경하는 방법에 대한 자세한 내용은 RDS Custom for SQL Server용 CEV 수정 단원을 참조하세요.	
`failed`	AMI를 검증하기 전에 이 CEV의 DB 인스턴스 생성 단계가 실패했습니다. 또는 CEV에서 사용하는 기본 AMI가 사용 가능한 상태가 아닙니다.	시스템에서 DB 인스턴스를 생성할 수 없는 근본 원인을 해결하세요. 자세한 오류 메시지를 확인하고 새 DB 인스턴스를 다시 생성해 보세요. CEV에서 사용하는 기본 AMI가 사용 가능한 상태인지 확인하세요.	

CEV 상태	설명	문제 해결 제안	
`incompatible-image-configuration`	AMI를 검증하는 중 오류가 발생했습니다.	오류의 기술적 세부 정보를 확인하세요. 이 CEV를 사용하여 AMI를 다시 검증할 수는 없습니다. 다음을 검토하세요. 권장 사항: • SQL Server 메이저 버전 + 마이너 버전 + 사용자 지정 문자열이라는 필수 이름 지정 패턴을 사용하여 CEV의 이름을 지정했는지 확인합니다. • CEV 이름의 SQL Server 버전이 AMI와 함께 제공된 버전과 일치하는지 확인합니다. • OS 빌드 버전이 최소 필수 빌드 버전을 충족하는지 확인합니다. • OS 메이저 버전이 최소 필수 메이저 버전을 충족하는지 확인합니다. 올바른 정보를 사용하여 새 CEV를 생성합니다. 필요한 경우 지원되는 AMI를 사용하여 새 EC2 인스턴스를 생성하고 이 인스턴스에서 Sysprep 프로세스를 실행합니다.	

RDS Custom for SQL Server용 CEV 수정

AWS Management Console 또는 AWS CLI를 사용하여 CEV를 수정하고, CEV 설명 또는 가용 상태를 수정할 수 있습니다. CEV는 다음 상태 값 중 하나를 가집니다.

- `available` - 이 CEV를 사용하여 새로운 RDS Custom DB 인스턴스를 생성하거나 DB 인스턴스를 업그레이드할 수 있습니다. 이 상태는 새로 생성된 CEV의 기본 상태입니다.
- `inactive` - 이 CEV에서는 RDS Custom DB 인스턴스를 생성하거나 업그레이드할 수 없습니다. 이 CEV에서는 DB 스냅샷을 복원하여 새로운 RDS Custom DB 인스턴스를 생성할 수 없습니다.

CEV 상태를 `available`에서 `inactive`로, 또는 `inactive`에서 `available`로 변경할 수 있습니다. 상태를 `INACTIVE`로 변경하면 CEV를 실수로 사용하는 일을 방지하고 중단된 CEV를 다시 사용할 수 있습니다.

콘솔

CEV를 수정하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 사용자 지정 엔진 버전(Custom engine versions)을 선택합니다.
3. 수정하려는 설명 또는 상태가 지정된 CEV를 선택합니다.
4. 작업에서 수정을 선택합니다.
5. 다음을 원하는 대로 변경합니다.
   - CEV 상태 설정(CEV status settings)에서 새로운 가용 상태를 선택합니다.
   - 버전 설명(Version description)에 새로운 설명을 입력합니다.
6. CEV 수정(Modify CEV)을 선택합니다.

   CEV가 사용 중인 경우 콘솔에 CEV 상태를 수정할 수 없습니다(You can't modify the CEV status)가 표시됩니다. 문제를 해결한 후 다시 시도하세요.

사용자 지정 엔진 버전(Custom engine versions) 페이지가 표시됩니다.

AWS CLI

AWS CLI를 사용하여 CEV를 수정하려면 modify-custom-db-engine-version 명령을 실행합니다. describe-db-engine-versions 명령을 실행하여 수정할 CEV를 찾을 수 있습니다.

다음 옵션이 필요합니다.

- `--engine`
- `--engine-version` *`cev`*(*`cev`*는 수정하려는 사용자 지정 엔진 버전의 이름)
- `--status` *`status`*(*`status`*는 CEV에 할당하려는 가용 상태)

다음 예제에서는 이름이 `15.00.4249.2.my_cevtest`로 지정된 CEV를 현재 상태에서 `inactive`로 변경합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-custom-db-engine-version \
    --engine custom-sqlserver-ee \
    --engine-version 15.00.4249.2.my_cevtest \
    --status inactive
```

Windows의 경우:

```
aws rds modify-custom-db-engine-version ^
    --engine custom-sqlserver-ee ^
    --engine-version 15.00.4249.2.my_cevtest ^
    --status inactive
```

새 CEV를 사용하도록 RDS Custom for SQL Server DB 인스턴스를 수정하려면 새 CEV를 사용하도록 RDS Custom for SQL Server DB 인스턴스 수정 섹션을 참조하세요.

새 CEV를 사용하도록 RDS Custom for SQL Server DB 인스턴스 수정

기존 RDS Custom for SQL Server DB 인스턴스를 수정하여 다른 CEV를 사용할 수 있습니다. 다음과 같은 변경을 적용할 수 있습니다.

- CEV 변경
- DB 인스턴스 클래스 변경
- 백업 보존 기간 및 백업 기간 변경
- 유지 관리 기간 변경

콘솔

RDS Custom for SQL Server DB 인스턴스를 수정하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 수정하려는 DB 인스턴스를 선택합니다.
4. Modify(수정)를 선택합니다.

5. 필요에 따라 다음과 같이 변경합니다.

   a. DB 엔진 버전(DB engine version)에서 다른 CEV를 선택합니다.

   b. DB 인스턴스 클래스(DB instance class) 값을 변경합니다. 지원되는 클래스는 RDS Custom for SQL Server DB 인스턴스 클래스 지원 섹션을 참조하세요.

   c. 백업 보존 기간(Backup retention period) 값을 변경합니다.

   d. 백업 기간(Backup window)의 경우 시작 시간(Start time)과 기간(Duration) 값을 설정합니다.

   e. DB 인스턴스 유지 관리 기간(DB instance maintenance window)의 경우 시작일(Start day), 시작 시간(Start time), 기간(Duration) 값을 설정합니다.

6. 계속(Continue)을 선택합니다.

7. 즉시 적용(Apply immediately) 또는 예약된 다음 유지 관리 기간에 적용(Apply during the next scheduled maintenance window)을 선택합니다.

8. Modify DB instance(DB 인스턴스 수정)를 선택합니다.

> **Note**
>
> 특정 CEV의 DB 인스턴스를 다른 CEV로 수정하는 경우(예를 들어 마이너 버전 업그레이드를 하는 경우), SQL Server 시스템 데이터베이스와 관련 데이터 및 구성은 현재 RDS Custom for SQL Server DB 인스턴스의 항목이 그대로 유지됩니다.

AWS CLI

AWS CLI로 DB 인스턴스를 수정해 다른 CEV를 사용할 수 있게 하려면 modify-db-instance 명령을 실행합니다.

다음 옵션이 필요합니다.

- `--db-instance-identifier`
- `--engine-version` *cev*(*cev*는 DB 인스턴스의 변경 대상인 사용자 지정 엔진 버전의 이름입니다).

다음 예제에서는 `my-cev-db-instance`라는 이름의 CEV를 사용하도록 `15.00.4249.2.my_cevtest_new`라는 이름의 DB 인스턴스를 수정하고 변경 사항을 즉시 적용합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier my-cev-db-instance \
    --engine-version 15.00.4249.2.my_cevtest_new \
    --apply-immediately
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-cev-db-instance ^
    --engine-version 15.00.4249.2.my_cevtest_new ^
    --apply-immediately
```

Amazon RDS Custom for SQL Server의 CEV 세부 정보 보기

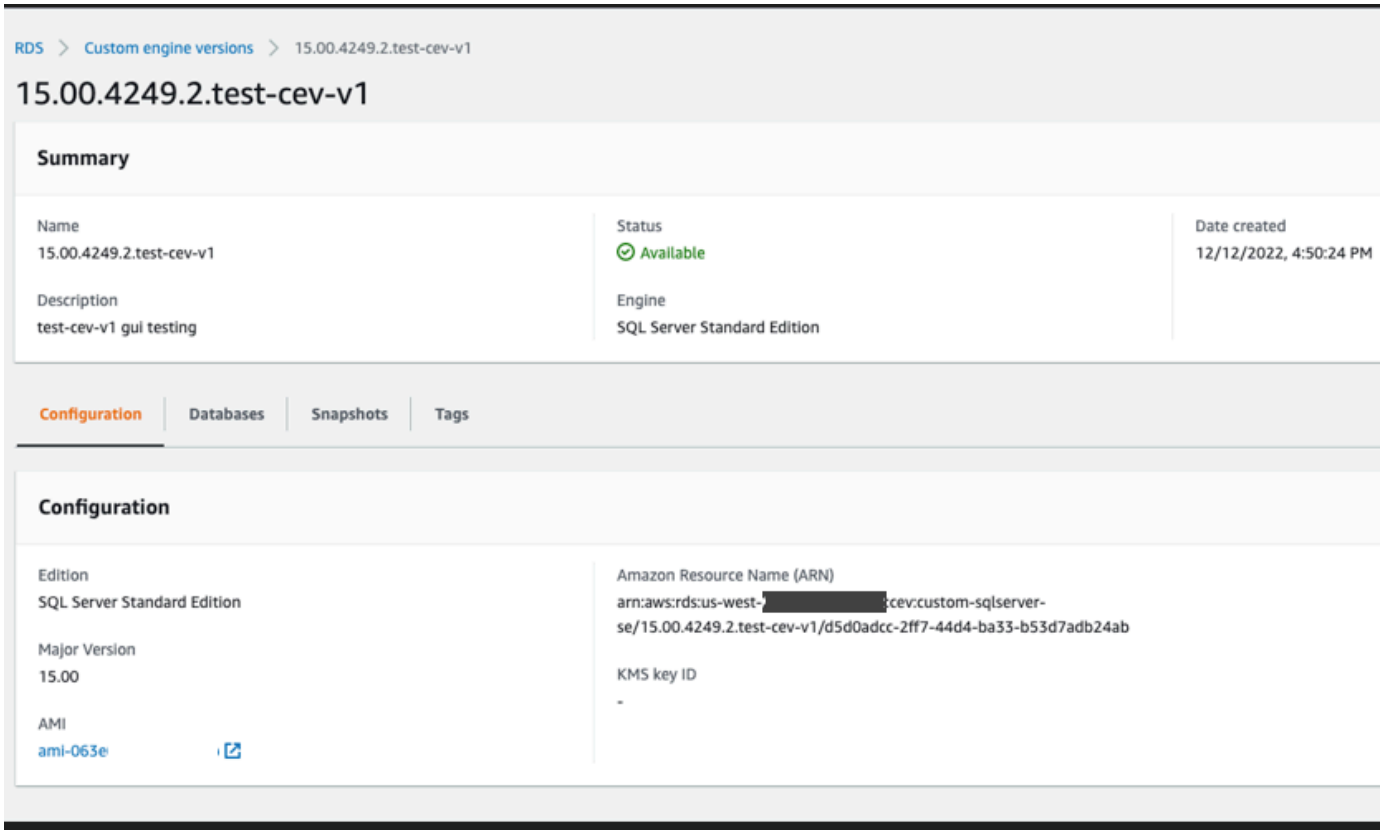
AWS Management Console 또는 AWS CLI를 사용하여 CEV에 대한 세부 정보를 볼 수 있습니다.

콘솔

CEV 세부 정보를 보려면

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 사용자 지정 엔진 버전(Custom engine versions)을 선택합니다.

   사용자 지정 엔진 버전(Custom engine versions) 페이지에는 현재 존재하는 모든 CEV가 표시됩니다. CEV를 생성한 적이 없으면 페이지가 비어 있습니다.
3. 보려는 CEV의 이름을 선택합니다.
4. 세부 정보를 보려면 Configuration(구성)을 선택합니다.



AWS CLI

AWS CLI를 사용하여 CEV에 대한 세부 정보를 보려면 describe-db-engine-versions 명령을 실행합니다.

다음 옵션도 지정할 수 있습니다.

- `--include-all`, 수명 주기 상태가 있는 모든 CEV를 볼 수 있습니다. `--include-all` 옵션이 없으면 `available` 수명 주기 상태의 CEV만 반환됩니다.

```
aws rds describe-db-engine-versions --engine custom-sqlserver-ee --engine-version
 15.00.4249.2.my_cevtest --include-all
{
    "DBEngineVersions": [
        {
            "Engine": "custom-sqlserver-ee",
            "MajorEngineVersion": "15.00",
            "EngineVersion": "15.00.4249.2.my_cevtest",
            "DBParameterGroupFamily": "custom-sqlserver-ee-15.0",
```

```
            "DBEngineDescription": "Microsoft SQL Server Enterprise Edition for custom
 RDS",
            "DBEngineVersionArn": "arn:aws:rds:us-east-1:{my-account-id}:cev:custom-
sqlserver-ee/15.00.4249.2.my_cevtest/a1234a1-123c-12rd-bre1-1234567890",
            "DBEngineVersionDescription": "Custom SQL Server EE 15.00.4249.2 cev test",
            "Image": {
                "ImageId": "ami-0r93cx31t5r596482",
                "Status": "pending-validation"
            },
            "DBEngineMediaType": "AWS Provided",
            "CreateTime": "2022-11-20T19:30:01.831000+00:00",
            "ValidUpgradeTarget": [],
            "SupportsLogExportsToCloudwatchLogs": false,
            "SupportsReadReplica": false,
            "SupportedFeatureNames": [],
            "Status": "pending-validation",
            "SupportsParallelQuery": false,
            "SupportsGlobalDatabases": false,
            "TagList": [],
            "SupportsBabelfish": false
        }
    ]
}
```

필터를 사용하여 특정 수명 주기 상태의 CEV를 볼 수 있습니다. 예를 들어 수명 주기 상태가 `pending-validation`, `available` 또는 `failed`인 CEV를 볼 수 있습니다.

```
aws rds describe-db-engine-versions engine custom-sqlserver-ee
                region us-west-2 include-all query 'DBEngineVersions[?Status ==
 pending-validation ||
                Status == available || Status == failed]'
```

RDS Custom for SQL Server용 CEV 삭제

AWS Management Console 또는 AWS CLI를 사용하여 CEV를 삭제할 수 있습니다. 이 작업은 일반적으로 몇 분 정도 걸립니다.

CEV를 삭제하기 전에 CEV가 다음 중 하나에 속하지 않는지 확인하세요.

- RDS Custom DB 인스턴스
- RDS Custom DB 인스턴스의 스냅샷
- RDS Custom DB 인스턴스의 자동 백업

콘솔

CEV를 삭제하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 사용자 지정 엔진 버전(Custom engine versions)을 선택합니다.
3. 삭제하려는 설명 또는 상태가 지정된 CEV를 선택합니다.
4. 작업에 대해 삭제를 선택합니다.

   cev_name을 삭제하시겠습니까?(Delete cev_name?) 대화 상자가 표시됩니다.
5. **delete me**를 입력한 다음 삭제를 선택합니다.

   사용자 지정 엔진 버전(Custom engine versions) 페이지의 배너에서 CEV가 삭제 중임을 표시합니다.

AWS CLI

AWS CLI를 사용하여 CEV를 삭제하려면 delete-custom-db-engine-version 명령을 실행합니다.

다음 옵션이 필요합니다.

- `--engine custom-sqlserver-ee`
- `--engine-version` *cev*(*cev*는 삭제할 사용자 지정 엔진 버전의 이름)

다음 예제에서는 이름이 `15.00.4249.2.my_cevtest`인 CEV를 삭제합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds delete-custom-db-engine-version \
    --engine custom-sqlserver-ee \
    --engine-version 15.00.4249.2.my_cevtest
```

Windows의 경우:

```
aws rds delete-custom-db-engine-version ^
    --engine custom-sqlserver-ee ^
```

```
    --engine-version 15.00.4249.2.my_cevtest
```

Amazon RDS Custom for SQL Server의 DB 인스턴스 생성 및 연결

RDS Custom DB 인스턴스를 생성한 다음 AWS Systems Manager 또는 원격 데스크톱 프로토콜(RDP)을 사용하여 연결할 수 있습니다.

> **Important**
>
> RDS Custom for SQL Server DB 인스턴스를 생성하거나 연결하기 전에 Amazon RDS Custom for SQL Server를 위한 환경 설정에 나와 있는 작업을 완료해야 합니다.
> RDS Custom DB 인스턴스를 생성할 때 태그를 지정할 수 있지만, RDS Custom 자동화에 필요한 `AWSRDSCustom` 태그를 생성하거나 수정해서는 안 됩니다. 자세한 내용은 RDS Custom for SQL Server 리소스 태깅 단원을 참조하십시오.
> RDS Custom for SQL Server DB 인스턴스를 처음 생성할 때 다음 오류가 표시될 수 있습니다. 서비스 연결 역할이 생성되고 있습니다. 나중에 다시 시도해 주세요. 이 경우에는 몇 분 정도 기다렸다가 다시 DB 인스턴스를 생성하면 됩니다.

주제

- RDS Custom for SQL Server DB 인스턴스 생성
- RDS Custom 서비스 연결 역할
- AWS Systems Manager를 사용하여 RDS Custom DB 인스턴스에 연결
- RDP를 사용하여 RDS Custom DB 인스턴스에 연결

RDS Custom for SQL Server DB 인스턴스 생성

AWS Management Console 또는 AWS CLI를 사용하여 SQL Server DB 인스턴스용 Amazon RDS Custom을 생성할 수 있습니다. 생성 절차는 Amazon RDS DB 인스턴스를 생성하는 절차와 유사합니다.

자세한 내용은 Amazon RDS DB 인스턴스 생성 단원을 참조하십시오.

콘솔

RDS Custom for SQL Server DB 인스턴스를 생성하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택합니다.

3. 데이터베이스 생성을 선택합니다.
4. 데이터베이스 생성 방법으로 표준 생성(Standard Create)을 선택합니다.
5. 엔진 옵션(Engine options)에서 엔진 유형으로 Microsoft SQL Server를 선택합니다.
6. 데이터베이스 관리 유형(Database management type)에서 Amazon RDS Custom을 선택합니다.
7. 에디션(Edition) 섹션에서 사용하려는 DB 엔진 에디션을 선택합니다.
8. (선택 사항) CEV에서 DB 인스턴스를 만들어야 하는 경우 Use custom engine version (CEV)(사용자 지정 엔진 버전(CEV) 사용) 확인란을 선택합니다. 드롭다운 목록에서 CEV를 선택합니다.
9. 데이터베이스 버전은 기본값 버전을 유지합니다.
10. Templates(템플릿)의 경우, Production(프로덕션)을 선택합니다.
11. 설정(Settings) 섹션에서 고유한 DB 인스턴스 식별자(DB instance identifier) 이름을 입력합니다.
12. 마스터 암호를 입력하려면 다음과 같이 하세요.

    a. Settings(설정) 섹션에서 Credential Settings(보안 인증 정보 설정)를 엽니다.

    b. Auto generate a password(암호 자동 생성) 확인란의 선택을 취소합니다.

    c. 마스터 사용자 이름(Master username) 값을 변경하고 마스터 암호(Master password) 및 암호 확인(Confirm password)에 동일한 암호를 입력합니다.

    기본적으로 새로운 RDS Custom DB 인스턴스는 마스터 사용자를 위해 자동 생성된 암호를 사용합니다.

13. DB 인스턴스 크기(DB instance size) 섹션에서 DB 인스턴스 클래스(DB instance class) 값을 선택합니다.

    지원되는 클래스는 RDS Custom for SQL Server DB 인스턴스 클래스 지원 섹션을 참조하세요.

14. 스토리지(Storage) 설정을 선택합니다.
15. RDS Custom 보안을 위해서는 다음을 수행합니다.

    a. IAM 인스턴스 프로파일에 RDS Custom for SQL Server DB 인스턴스의 인스턴스 프로파일을 선택하는 옵션이 두 개 있습니다.

        1. 새 인스턴스 프로파일 생성을 선택하고 인스턴스 프로파일 이름 접미사를 제공합니다. 자세한 내용은 AWS Management Console을 사용한 자동 인스턴스 프로파일 생성 단원을 참조하십시오.
        2. 기존 인스턴스 프로파일을 선택합니다. 드롭다운 목록에서 `AWSRDSCustom`으로 시작하는 인스턴스 프로파일을 선택합니다.

b. 암호화(Encryption)의 경우 키 ARN 입력(Enter a key ARN)을 선택하여 사용 가능한 AWS KMS 키를 나열합니다. 그런 다음 목록에서 키를 선택합니다.

   AWS KMS 키는 RDS Custom에 필수입니다. 자세한 내용은 대칭 암호화 AWS KMS 키 보유 여부 확인 단원을 참조하십시오.

16. 나머지 섹션에서 원하는 대로 RDS Custom DB 인스턴스 설정을 지정합니다. 각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하세요. 다음 설정은 콘솔에 표시되지 않으며 지원되지 않습니다.

    - 프로세서 기능
    - Storage autoscaling(스토리지 Autoscaling)
    - 가용성 및 지속성(Availability & durability)
    - 데이터베이스 인증(Database authentication)의 암호 및 Kerberos 인증(Password and Kerberos authentication) 옵션(암호 인증(Password authentication)만 지원)
    - 추가 구성(Additional configuration)의 데이터베이스 옵션(Database options) 그룹
    - 성능 개선 도우미
    - 로그 내보내기
    - 마이너 버전 자동 업그레이드 활성화
    - 삭제 방지

    백업 보존 기간은 지원되지만, 0일을 선택할 수 없습니다.

17. Create database(데이터베이스 생성)를 선택합니다.

    자격 증명 세부 정보 보기(View credential details) 버튼이 데이터베이스 페이지에 표시됩니다.

    RDS Custom DB 인스턴스의 마스터 사용자 이름 및 암호를 보려면 자격 증명 세부 정보 보기(View credential details)를 선택합니다.

    DB 인스턴스를 마스터 사용자로 연결하려면 화면에 나타난 사용자 이름과 암호를 사용합니다.

    > **Important**
    >
    > 마스터 사용자 암호를 다시 볼 수는 없습니다. 따라서 기록을 해두지 않으면 이를 변경해야 합니다. RDS Custom DB 인스턴스가 사용 가능한 상태가 되고 난 후에 마스터 사용자

암호를 변경하려면 DB 인스턴스를 수정하면 됩니다. DB 인스턴스 수정에 대한 자세한 내용은 Amazon RDS Custom for SQL Server DB 인스턴스 관리 단원을 참조하세요.

18. 데이터베이스(Databases)를 선택하여 RDS Custom DB 인스턴스 목록을 확인합니다.
19. 방금 생성한 RDS Custom DB 인스턴스를 선택합니다.

    RDS 콘솔에 새로운 RDS Custom DB 인스턴스의 세부 정보가 표시됩니다.

    - RDS Custom DB 인스턴스를 만들고 사용할 준비가 될 때까지 DB 인스턴스의 상태는 생성 중(creating)입니다. 상태가 available로 변경되면 DB 인스턴스에 연결할 수 있습니다. 할당된 인스턴스 클래스 및 스토리지에 따라 새 DB 인스턴스를 사용할 수 있게 되기까지 몇 분 정도 걸릴 수 있습니다.
    - 역할(Role)에는 인스턴스(RDS Custom)(Instance (RDS Custom)) 값이 있습니다.
    - RDS Custom 자동화 모드(RDS Custom automation mode)에는 완전 자동화(Full automation) 값이 있습니다. 이 설정은 DB 인스턴스가 자동 모니터링 및 인스턴스 복구를 제공함을 의미합니다.

AWS CLI

create-db-instance AWS CLI 명령을 사용하여 RDS Custom DB 인스턴스를 생성합니다.

다음 옵션이 필요합니다.

- `--db-instance-identifier`
- `--db-instance-class`(지원되는 인스턴스 클래스 목록은 RDS Custom for SQL Server DB 인스턴스 클래스 지원 섹션 참조)
- `--engine`(`custom-sqlserver-ee`, `custom-sqlserver-se` 또는 `custom-sqlserver-web`)
- `--kms-key-id`
- `--custom-iam-instance-profile`

다음 예제에서는 `my-custom-instance`라는 RDS Custom for SQL Server DB 인스턴스를 생성합니다. 백업 보존 기간은 3일로 설정합니다.

> **Note**
>
> 사용자 지정 엔진 버전(CEV)에서 DB 인스턴스를 만들려면 `--engine-version` 파라미터에 기존 CEV 이름을 제공합니다. 예제: `--engine-version 15.00.4249.2.my_cevtest`

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance \
    --engine custom-sqlserver-ee \
    --engine-version 15.00.4073.23.v1 \
    --db-instance-identifier my-custom-instance \
    --db-instance-class db.m5.xlarge \
    --allocated-storage 20 \
    --db-subnet-group mydbsubnetgroup \
    --master-username myuser \
    --master-user-password mypassword \
    --backup-retention-period 3 \
    --no-multi-az \
    --port 8200 \
    --kms-key-id mykmskey \
    --custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance
```

Windows의 경우:

```
aws rds create-db-instance ^
    --engine custom-sqlserver-ee ^
    --engine-version 15.00.4073.23.v1 ^
    --db-instance-identifier my-custom-instance ^
    --db-instance-class db.m5.xlarge ^
    --allocated-storage 20 ^
    --db-subnet-group mydbsubnetgroup ^
    --master-username myuser ^
    --master-user-password mypassword ^
    --backup-retention-period 3 ^
    --no-multi-az ^
    --port 8200 ^
    --kms-key-id mykmskey ^
    --custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

`describe-db-instances` 명령을 사용하여 인스턴스에 대한 세부 정보를 가져옵니다.

```
aws rds describe-db-instances --db-instance-identifier my-custom-instance
```

다음 부분 출력은 엔진, 파라미터 그룹 및 기타 정보를 보여줍니다.

```
{
    "DBInstances": [
        {
            "PendingModifiedValues": {},
            "Engine": "custom-sqlserver-ee",
            "MultiAZ": false,
            "DBSecurityGroups": [],
            "DBParameterGroups": [
                {
                    "DBParameterGroupName": "default.custom-sqlserver-ee-15",
                    "ParameterApplyStatus": "in-sync"
                }
            ],
            "AutomationMode": "full",
            "DBInstanceIdentifier": "my-custom-instance",
            "TagList": []
        }
    ]
}
```

RDS Custom 서비스 연결 역할

서비스 연결 역할을 사용하여 AWS 계정의 리소스에 대한 Amazon RDS Custom 액세스 권한을 부여합니다. 필요한 권한을 수동으로 추가할 필요가 없어 RDS Custom을 보다 쉽게 사용할 수 있습니다. RDS Custom은 서비스 연결 역할의 권한을 정의하므로, 달리 정의하지 않으면 RDS Custom만 해당 역할을 맡을 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며, 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

RDS Custom DB 인스턴스를 생성하면 Amazon RDS 및 RDS Custom 서비스 연결 역할이 모두 생성되어(존재하지 않았던 경우) 사용됩니다. 자세한 내용은 Amazon RDS에 서비스 연결 역할 사용 단원을 참조하십시오.

RDS Custom for SQL Server DB 인스턴스를 처음 생성할 때 다음 오류가 표시될 수 있습니다. 서비스 연결 역할이 생성되고 있습니다. 나중에 다시 시도해 주세요. 이 경우에는 몇 분 정도 기다렸다가 다시 DB 인스턴스를 생성하면 됩니다.

AWS Systems Manager를 사용하여 RDS Custom DB 인스턴스에 연결

RDS Custom DB 인스턴스를 생성한 후 AWS Systems Manager 세션 관리자를 사용하여 연결할 수 있습니다. 세션 관리자는 브라우저 기반 셸이나 AWS CLI를 통해 Amazon EC2 인스턴스를 관리하는 데 사용할 수 있는 Systems Manager 기능입니다. 자세한 내용은 AWS Systems Manager 세션 관리자를 참조하세요.

콘솔

세션 관리자를 사용하여 DB 인스턴스에 연결하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스(Databases)를 선택한 후 연결하려는 RDS Custom DB 인스턴스를 선택합니다.
3. Configuration(구성)을 선택합니다.
4. DB 인스턴스의 리소스 ID(Resource ID) 값을 기록해 둡니다. 예를 들어, 리소스 ID는 `db-ABCDEFGHIJKLMNOPQRS0123456`일 수 있습니다.
5. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
6. 탐색 창에서 Instances(인스턴스)를 선택합니다.
7. EC2 인스턴스의 이름을 찾은 다음 연결된 인스턴스 ID를 선택합니다. 예를 들어, 인스턴스 ID는 `i-abcdefghijklm01234`일 수 있습니다.
8. 연결을 선택합니다.
9. 세션 관리자(Session Manager)를 선택합니다.
10. Connect(연결)를 선택합니다.

    세션에 대한 창이 열립니다.

AWS CLI

AWS CLI를 사용하여 RDS Custom DB 인스턴스에 연결할 수 있습니다. 이 기술을 사용하려면 AWS CLI용 세션 관리자 플러그인이 필요합니다. 플러그인 설치 방법은 AWS CLI용 세션 관리자 플러그인 설치를 참조하세요.

RDS Custom DB 인스턴스의 DB 리소스 ID를 찾으려면 describe-db-instances를 사용합니다.

```
aws rds describe-db-instances \
    --query 'DBInstances[*].[DBInstanceIdentifier,DbiResourceId]' \
    --output text
```

다음 샘플 출력은 RDS Custom 인스턴스의 리소스 ID를 보여줍니다. 접두사는 db-입니다.

```
db-ABCDEFGHIJKLMNOPQRS0123456
```

DB 인스턴스의 EC2 인스턴스 ID를 찾으려면 `aws ec2 describe-instances`를 사용합니다. 다음 예제에는 리소스 ID로 `db-ABCDEFGHIJKLMNOPQRS0123456`이 사용됩니다.

```
aws ec2 describe-instances \
    --filters "Name=tag:Name,Values=db-ABCDEFGHIJKLMNOPQRS0123456" \
    --output text \
    --query 'Reservations[*].Instances[*].InstanceId'
```

다음 샘플 출력에는 EC2 인스턴스 ID가 나와 있습니다.

```
i-abcdefghijklm01234
```

`--target` 파라미터의 EC2 인스턴스 ID를 제공하는 `aws ssm start-session` 명령을 사용합니다.

```
aws ssm start-session --target "i-abcdefghijklm01234"
```

성공적으로 연결되면 다음과 같이 표시됩니다.

```
Starting session with SessionId: yourid-abcdefghijklm1234
[ssm-user@ip-123-45-67-89 bin]$
```

RDP를 사용하여 RDS Custom DB 인스턴스에 연결

RDS Custom DB 인스턴스를 생성한 후 RDP 클라이언트를 사용하여 이 인스턴스에 연결할 수 있습니다. 이 절차는 Amazon EC2 인스턴스에 연결하는 절차와 동일합니다. 자세한 내용은 Windows 인스턴스에 연결을 참조하세요.

DB 인스턴스에 연결하려면 인스턴스와 연결된 키 페어가 필요합니다. 키 페어는 RDS Custom에서 생성합니다. 페어 이름은 접두사 `do-not-delete-rds-custom-`*`DBInstanceIdentifier`*를 사용하며, AWS Secrets Manager는 프라이빗 키를 비밀 키로 저장합니다.

다음 단계의 작업을 완료합니다.

1. RDP 연결을 허용하도록 DB 인스턴스 구성.
2. 비밀 키 검색.
3. RDP 유틸리티를 사용하여 EC2 인스턴스에 연결.

RDP 연결을 허용하도록 DB 인스턴스 구성

RDP 연결을 허용하려면 VPC 보안 그룹을 구성하고 호스트에 방화벽 규칙을 설정합니다.

VPC 보안 그룹 구성

DB 인스턴스와 연결된 VPC 보안 그룹이 전송 제어 프로토콜(TCP)용 포트 3389에서 인바운드 연결을 허용하는지 확인합니다. VPC 보안 그룹을 구성하는 방법은 VPC 보안 그룹 구성 섹션을 참조하세요.

호스트에서 방화벽 규칙 설정

TCP용 포트 3389에서 인바운드 연결을 허용하려면 호스트에 방화벽 규칙을 설정합니다. 다음 예제에서 이 작업을 수행하는 방법을 보여줍니다.

특정 `-Profile` 값(`Public`, `Private` 또는 `Domain`)을 사용하는 것이 좋습니다. `Any`을 사용하면 세 값을 모두 지정하는 것입니다. 쉼표로 구분된 값 조합을 지정할 수도 있습니다. 방화벽 규칙 설정에 대한 자세한 내용은 Microsoft 설명서의 Set-NetFirewallRule을 참조하세요.

Systems Manager 세션 관리자를 사용하여 방화벽 규칙을 설정하는 방법

1. AWS Systems Manager를 사용하여 RDS Custom DB 인스턴스에 연결에 표시된 대로 세션 관리자에 연결합니다.
2. 다음 명령을 실행합니다.

```
Set-NetFirewallRule -DisplayName "Remote Desktop - User Mode (TCP-In)" -Direction
 Inbound -LocalAddress Any -Profile Any
```

Systems Manager CLI 명령을 사용하여 방화벽 규칙 설정

1. 다음 명령을 사용하여 호스트에서 RDP를 엽니다.

```
OPEN_RDP_COMMAND_ID=$(aws ssm send-command --region $AWS_REGION \
    --instance-ids $RDS_CUSTOM_INSTANCE_EC2_ID \
    --document-name "AWS-RunPowerShellScript" \
    --parameters '{"commands":["Set-NetFirewallRule -DisplayName \"Remote Desktop -
 User Mode (TCP-In)\" -Direction Inbound -LocalAddress Any -Profile Any"]}' \
    --comment "Open RDP port" | jq -r ".Command.CommandId")
```

2. 출력에서 반환된 명령 ID를 사용하여 이전 명령의 상태를 가져옵니다. 다음 쿼리를 사용하여 명령 ID를 반환하려면 jq 플러그인이 설치되어 있는지 확인합니다.

```
aws ssm list-commands \
    --region $AWS_REGION \
    --command-id $OPEN_RDP_COMMAND_ID
```

비밀 키 검색

AWS Management Console 또는 AWS CLI 중 하나를 사용하여 비밀 키를 검색할 수 있습니다.

콘솔

비밀 키를 검색하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스(Databases)를 선택한 후 연결하려는 RDS Custom DB 인스턴스를 선택합니다.
3. Configuration(구성) 탭을 선택합니다.
4. DB 인스턴스의 DB 인스턴스 ID를 기록해 둡니다(예: *my-custom-instance*).
5. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
6. 탐색 창에서 Instances(인스턴스)를 선택합니다.

7. EC2 인스턴스의 이름을 찾은 다음 연결된 인스턴스 ID를 선택합니다.

   이 예제에서 인스턴스 ID는 `i-abcdefghijklm01234`입니다.

8. 세부 정보(Details)에서 키 페어 이름(Key pair name)을 찾습니다. 페어 이름에는 DB 식별자가 포함됩니다. 이 예제에서 페어 이름은 `do-not-delete-rds-custom-`*`my-custom-instance-0d726c`*입니다.

9. 인스턴스 요약에서 퍼블릭 IPv4 DNS(Public IPv4 DNS)를 찾습니다. 예를 들어, 퍼블릭 DNS는 `ec2-12-345-678-901.us-east-2.compute.amazonaws.com`일 수 있습니다.

10. https://console.aws.amazon.com/secretsmanager/에서 AWS Secrets Manager 콘솔을 엽니다.

11. 키 페어와 이름이 같은 비밀 키를 선택합니다.

12. Retrieve secret value(보안 암호 값 검색)를 선택합니다.

AWS CLI

프라이빗 키를 검색하는 방법

1. `aws rds describe-db-instances` 명령을 호출하여 RDS Custom DB 인스턴스 목록을 가져옵니다

```
aws rds describe-db-instances \
    --query 'DBInstances[*].[DBInstanceIdentifier,DbiResourceId]' \
    --output text
```

2. 샘플 출력에서 DB 인스턴스 식별자를 선택합니다(예: `do-not-delete-rds-custom-`*`my-custom-instance`*).

3. `aws ec2 describe-instances` 명령을 호출하여 DB 인스턴스의 EC2 인스턴스 ID를 찾습니다. 다음 예제에서는 EC2 인스턴스 이름을 사용하여 DB 인스턴스를 설명합니다.

```
aws ec2 describe-instances \
    --filters "Name=tag:Name,Values=do-not-delete-rds-custom-my-custom-instance" \
    --output text \
    --query 'Reservations[*].Instances[*].InstanceId'
```

   다음 샘플 출력에는 EC2 인스턴스 ID가 나와 있습니다.

```
i-abcdefghijklm01234
```

4. EC2 인스턴스 ID를 지정하여 다음 예제에서와 같이 키 이름을 찾습니다.

```
aws ec2 describe-instances \
    --instance-ids i-abcdefghijklm01234 \
    --output text \
    --query 'Reservations[*].Instances[*].KeyName'
```

다음 샘플 출력은 접두사 do-not-delete-rds-custom-*DBInstanceIdentifier*를 사용하는 키 이름을 보여줍니다.

```
do-not-delete-rds-custom-my-custom-instance-0d726c
```

RDP 유틸리티를 사용하여 EC2 인스턴스에 연결

Amazon EC2 사용 설명서의 RDP를 사용하여 Windows 인스턴스에 연결에 나와 있는 절차를 따릅니다. 이 절차에서는 프라이빗 키가 포함된 .pem 파일을 생성한 것으로 가정합니다.

Amazon RDS Custom for SQL Server DB 인스턴스 관리

Amazon RDS Custom for SQL Server은 Amazon RDS DB 인스턴스에 대한 일반적인 관리 작업의 하위 집합을 지원합니다. 그런 다음 AWS Management Console 및 AWS CLI를 사용하여 지원되는 RDS Custom for SQL Server 관리 작업에 대한 지침을 확인할 수 있습니다.

주제

- RDS Custom 자동화 일시 중지 및 다시 시작
- RDS Custom for SQL Server DB 인스턴스 수정
- RDS Custom for SQL Server DB 인스턴스 스토리지 수정
- RDS Custom for SQL Server 리소스 태깅
- RDS Custom for SQL Server DB 인스턴스 삭제
- RDS Custom for SQL Server DB 인스턴스 시작 및 중지

RDS Custom 자동화 일시 중지 및 다시 시작

RDS Custom은 RDS Custom for SQL Server DB 인스턴스에 대한 모니터링 및 인스턴스 복구를 자동으로 제공합니다. 인스턴스를 사용자 지정해야 하는 경우 다음을 수행하세요.

1. 지정된 기간 동안 RDS Custom 자동화를 일시 중지합니다. 일시 중지하면 사용자 지정한 인스턴스가 RDS Custom 자동화를 방해하지 않도록 할 수 있습니다.
2. 필요에 따라 RDS Custom for SQL Server DB 인스턴스를 사용자 지정합니다.
3. 다음 중 하나를 수행하세요.
   - 수동으로 자동화를 다시 시작합니다.
   - 일시 중지 기간이 끝날 때까지 기다립니다. 이 경우 RDS Custom은 모니터링 및 인스턴스 복구를 자동으로 재개합니다.

> **Important**
>
> RDS Custom for SQL Server DB 인스턴스를 수정할 때 지원되는 자동화 작업은 자동화 일시 중지 및 재개입니다.

콘솔

RDS Custom 자동화를 일시 중지하거나 다시 시작하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스(Databases)를 선택한 다음 변경하려는 RDS Custom DB 인스턴스를 선택합니다.
3. Modify(수정)를 선택합니다. Modify DB instance(DB 인스턴스 수정) 페이지가 나타납니다.
4. RDS Custom 자동화 모드(RDS Custom automation mode)로 다음 옵션 중 하나를 선택합니다.
   - 일시 중지됨(Paused)을 선택하면 RDS Custom DB 인스턴스에 대한 모니터링 및 인스턴스 복구가 일시 중지됩니다. 자동화 모드 지속 시간(Automation mode duration)으로 원하는 일시 중지 기간(분)을 입력합니다. 최소값은 60분(기본값)입니다. 최대값은 1,440분입니다.
   - 완전 자동화(Full automation)를 선택하면 자동화가 재개됩니다.
5. 계속(Continue)을 선택하여 수정 사항을 요약한 내용을 확인합니다.

   RDS Custom에서 변경 사항을 즉시 적용한다는 메시지가 표시됩니다.
6. 변경 내용이 정확할 경우 DB 인스턴스 수정(Modify DB instance)을 선택합니다. 또는 뒤로를 선택하여 변경 내용을 편집하거나 취소를 선택하여 변경 내용을 취소합니다.

   RDS 콘솔에 수정 사항에 대한 세부 정보가 표시됩니다. 자동화를 일시 중지한 경우 RDS Custom DB 인스턴스의 상태(Status)가 자동화 일시 중지됨(Automation paused)으로 표시됩니다.
7. (선택 사항) 탐색 창에서 데이터베이스(Databases)를 선택한 후 RDS Custom DB 인스턴스를 선택합니다.

   요약(Summary) 창에서 RDS Custom 자동화 모드(RDS Custom automation mode)는 자동화 상태를 나타냅니다. 자동화가 일시 중지되면 값이 일시 중지됨(Paused)이 됩니다. ***num***분 후에 자동화가 재개됩니다.

AWS CLI

RDS Custom 자동화를 일시 중지하거나 재개하려면 `modify-db-instance` AWS CLI 명령을 사용합니다. 필수 파라미터 `--db-instance-identifier`를 사용하여 DB 인스턴스를 식별합니다. 다음 파라미터를 사용하여 자동화 모드를 제어합니다.

- `--automation-mode`는 DB 인스턴스의 일시 정지 상태를 지정합니다. 유효한 값은 자동화를 일시 중지하는 `all-paused`와 다시 재개하는 `full`입니다.
- `--resume-full-automation-mode-minutes`는 일시 중지 기간을 지정합니다. 기본값은 60분입니다.

> **Note**
>
> `--no-apply-immediately` 또는 `--apply-immediately`에 관계없이 RDS Custom은 가능한 한 빨리 비동기식으로 수정 사항을 적용합니다.

명령 응답에서 `ResumeFullAutomationModeTime`은 재개 시간을 UTC 타임스탬프로 나타냅니다. 자동화 모드가 `all-paused`인 경우 `modify-db-instance`를 사용하여 자동화 모드를 재개하거나 일시 중지 기간을 연장할 수 있습니다. 다른 `modify-db-instance` 옵션은 지원되지 않습니다.

다음 예제에서는 `my-custom-instance`에 대한 자동화를 90분 동안 일시 중지합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier my-custom-instance \
    --automation-mode all-paused \
    --resume-full-automation-mode-minutes 90
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --automation-mode all-paused ^
    --resume-full-automation-mode-minutes 90
```

다음 예제에서는 일시 중지 기간을 30분 더 연장합니다. 30분이 `ResumeFullAutomationModeTime`에 표시된 원래 시간에 추가됩니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier my-custom-instance \
    --automation-mode all-paused \
    --resume-full-automation-mode-minutes 30
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --automation-mode all-paused ^
    --resume-full-automation-mode-minutes 30
```

다음 예제에서는 `my-custom-instance`에 대한 전체 자동화를 재개합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier my-custom-instance \
    --automation-mode full \
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --automation-mode full
```

다음 부분 샘플 출력에서 보류 중인 `AutomationMode` 값은 `full`입니다.

```
{
    "DBInstance": {
        "PubliclyAccessible": true,
        "MasterUsername": "admin",
        "MonitoringInterval": 0,
        "LicenseModel": "bring-your-own-license",
        "VpcSecurityGroups": [
            {
                "Status": "active",
                "VpcSecurityGroupId": "0123456789abcdefg"
            }
        ],
```

```
        "InstanceCreateTime": "2020-11-07T19:50:06.193Z",
        "CopyTagsToSnapshot": false,
        "OptionGroupMemberships": [
            {
                "Status": "in-sync",
                "OptionGroupName": "default:custom-oracle-ee-19"
            }
        ],
        "PendingModifiedValues": {
            "AutomationMode": "full"
        },
        "Engine": "custom-oracle-ee",
        "MultiAZ": false,
        "DBSecurityGroups": [],
        "DBParameterGroups": [
            {
                "DBParameterGroupName": "default.custom-oracle-ee-19",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        ...
        "ReadReplicaDBInstanceIdentifiers": [],
        "AllocatedStorage": 250,
        "DBInstanceArn": "arn:aws:rds:us-west-2:012345678912:db:my-custom-instance",
        "BackupRetentionPeriod": 3,
        "DBName": "ORCL",
        "PreferredMaintenanceWindow": "fri:10:56-fri:11:26",
        "Endpoint": {
            "HostedZoneId": "ABCDEFGHIJKLMNO",
            "Port": 8200,
            "Address": "my-custom-instance.abcdefghijk.us-west-2.rds.amazonaws.com"
        },
        "DBInstanceStatus": "automation-paused",
        "IAMDatabaseAuthenticationEnabled": false,
        "AutomationMode": "all-paused",
        "EngineVersion": "19.my_cev1",
        "DeletionProtection": false,
        "AvailabilityZone": "us-west-2a",
        "DomainMemberships": [],
        "StorageType": "gp2",
        "DbiResourceId": "db-ABCDEFGHIJKLMNOPQRSTUVW",
        "ResumeFullAutomationModeTime": "2020-11-07T20:56:50.565Z",
        "KmsKeyId": "arn:aws:kms:us-west-2:012345678912:key/
aa111a11-111a-11a1-1a11-1111a11a1a1a",
```

```
        "StorageEncrypted": false,
        "AssociatedRoles": [],
        "DBInstanceClass": "db.m5.xlarge",
        "DbInstancePort": 0,
        "DBInstanceIdentifier": "my-custom-instance",
        "TagList": []
    }
```

RDS Custom for SQL Server DB 인스턴스 수정

RDS Custom for SQL Server DB 인스턴스를 수정하는 절차는 Amazon RDS에서와 유사하지만, 변경할 수 있는 내용은 다음으로 제한됩니다.

- DB 인스턴스 클래스 변경
- 백업 보존 기간 및 백업 기간 변경
- 유지 관리 기간 변경
- 새로운 버전을 사용할 수 있을 때 DB 엔진 버전 업그레이드
- 할당된 스토리지, 프로비저닝된 IOPS, 스토리지 유형 변경
- 다중 AZ 배포 허용 및 제거

RDS Custom for SQL Server DB 인스턴스를 수정할 때는 다음과 같은 제한이 적용됩니다.

- Custom DB 옵션 및 파라미터 그룹은 지원되지 않습니다.
- RDS Custom DB 인스턴스에 수동으로 연결하는 모든 스토리지 볼륨은 지원 경계를 벗어납니다.

  자세한 내용은 RDS Custom 지원 범위 단원을 참조하십시오.

콘솔

RDS Custom for SQL Server DB 인스턴스를 수정하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 수정하려는 DB 인스턴스를 선택합니다.
4. Modify(수정)를 선택합니다.
5. 필요에 따라 다음과 같이 변경합니다.

a. DB engine version(DB 엔진 버전)에서 새 버전을 선택합니다.

b. DB 인스턴스 클래스(DB instance class) 값을 변경합니다. 지원되는 클래스는 RDS Custom for SQL Server DB 인스턴스 클래스 지원 섹션을 참조하세요.

c. 백업 보존 기간(Backup retention period) 값을 변경합니다.

d. 백업 기간(Backup window)의 경우 시작 시간(Start time)과 기간(Duration) 값을 설정합니다.

e. DB 인스턴스 유지 관리 기간(DB instance maintenance window)의 경우 시작일(Start day), 시작 시간(Start time), 기간(Duration) 값을 설정합니다.

6. 계속(Continue)을 선택합니다.
7. 즉시 적용(Apply immediately) 또는 예약된 다음 유지 관리 기간에 적용(Apply during the next scheduled maintenance window)을 선택합니다.
8. Modify DB instance(DB 인스턴스 수정)를 선택합니다.

AWS CLI

RDS Custom for SQL Server DB 인스턴스를 수정하려면 modify-db-instance AWS CLI 명령을 사용하면 됩니다. 필요한 경우 다음 파라미터를 설정합니다.

- `--db-instance-class` - 지원되는 클래스는 RDS Custom for SQL Server DB 인스턴스 클래스 지원 섹션을 참조하세요.
- `--engine-version` - 업그레이드할 데이터베이스 엔진의 버전 번호입니다.
- `--backup-retention-period` - 자동화된 백업을 유지하는 기간으로, 0~35일까지 설정할 수 있습니다.
- `--preferred-backup-window` - 자동화된 백업이 생성되는 일일 시간 범위입니다.
- `--preferred-maintenance-window` - 시스템 유지 관리를 실행할 수 있는 주 단위 기간(UTC)입니다.
- `--apply-immediately` – `--apply-immediately`를 사용하여 스토리지 변경 사항을 바로 적용합니다.

  그 밖에 다음 유지 관리 기간에 스토리지 변경 사항을 적용하려면 `--no-apply-immediately`(기본값)를 사용합니다.

RDS Custom for SQL Server DB 인스턴스 스토리지 수정

RDS Custom for SQL Server DB 인스턴스의 스토리지 수정은 Amazon RDS DB 인스턴스의 스토리지 수정과 유사하지만 다음 작업만 수행할 수 있습니다.

- 할당된 스토리지 크기를 늘립니다.
- 스토리지 유형을 변경합니다. 범용 또는 프로비저닝된 IOPS 같은 사용 가능한 스토리지 유형을 사용할 수 있습니다. 프로비저닝된 IOPS는 gp3, io1 및 io2 Block Express 스토리지 유형에서 지원됩니다.
- 프로비저닝된 IOPS를 지원하는 볼륨 유형을 사용하는 경우 프로비저닝된 IOPS를 변경합니다.

RDS Custom for SQL Server DB 인스턴스의 스토리지 수정에는 다음과 같은 제한이 적용됩니다.

- RDS Custom for SQL Server에 할당된 최소 스토리지 크기는 20GiB입니다. io1, gp2, gp3의 최대 스토리지 한도는 16TiB인 반면 io2는 64TiB를 지원합니다.
- Amazon RDS와 마찬가지로 할당된 스토리지를 줄일 수는 없는데, 이는 Amazon Elastic Block Store(Amazon EBS) 볼륨의 제한입니다. 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 작업 섹션을 참조하세요.
- RDS Custom for SQL Server DB 인스턴스에는 스토리지 autoscaling이 지원되지 않습니다.
- RDS Custom DB 인스턴스에 수동으로 연결하는 스토리지 볼륨은 스토리지 크기 조정 대상으로 간주되지 않습니다. RDS에서 제공하는 기본 데이터 볼륨, 즉 D 드라이브만 스토리지 크기 조정 대상으로 간주됩니다.

  자세한 내용은 RDS Custom 지원 범위 섹션을 참조하세요.
- 스토리지 크기 조정은 일반적으로 DB 인스턴스의 중단이나 성능 저하를 일으키지 않습니다. DB 인스턴스에 대한 스토리지 크기를 수정하면 DB 인스턴스의 상태가 스토리지 최적화로 됩니다.
- 스토리지 최적화는 몇 시간이 걸릴 수 있습니다. 6시간 또는 인스턴스에서 스토리지 최적화가 완료된 시간 둘 중 더 긴 시간 동안은 스토리지를 추가로 수정할 수 없습니다. 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 작업 섹션을 참조하세요.

스토리지에 대한 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 단원을 참조하세요.

스토리지 수정에 대한 일반적인 정보는 Amazon RDS DB 인스턴스 스토리지 작업 섹션을 참조하세요.

Important

Amazon EC2 콘솔 또는 Amazon EBS 콘솔 또는 API를 사용하여 RDS Custom for SQL Server DB 인스턴스의 스토리지를 수정하지 마세요. Amazon RDS 콘솔 또는 API 외부에서 직접 스토리지를 수정하면 데이터베이스가 `unsupported-configuration` 상태가 됩니다.
Amazon EC2 또는 Amazon EBS를 사용하여 직접 스토리지를 변경하는 경우 Amazon RDS는 데이터베이스 인스턴스 상태를 추적하거나 관리할 수 없습니다. 이로 인해 다음가 같은 문제가 발생할 수 있습니다.

- 고가용성 장애 조치 메커니즘이 제대로 작동하지 않음
- 데이터베이스 복제 설정이 중단됨
- 중복 기능이 실패함

데이터베이스를 지원되는 상태로 유지하려면 Amazon RDS 콘솔 또는 API를 통해서만 스토리지를 수정하세요. 복구 단계는 RDS Custom for SQL Server에서 지원되지 않는 구성 문제 해결 섹션을 참조하세요.

콘솔

RDS Custom for SQL Server DB 인스턴스 스토리지를 수정하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 수정하려는 DB 인스턴스를 선택합니다.
4. Modify(수정)를 선택합니다.
5. 필요에 따라 다음과 같이 변경합니다.
   a. Allocated storage(할당된 스토리지)에 새로운 값을 입력합니다. 현재 값보다 커야 하며 20GiB~16TiB여야 합니다.
   b. 스토리지 유형(Storage type) 값을 변경합니다. 범용 또는 프로비저닝된 IOPS 같은 사용 가능한 스토리지 유형을 사용할 수 있습니다. 프로비저닝된 IOPS는 gp3, io1 및 io2 Block Express 스토리지 유형에서 지원됩니다.
   c. 프로비저닝된 IOPS를 지원하는 볼륨 유형을 지정하는 경우 프로비저닝된 IOPS 값을 정의할 수 있습니다.

6. Continue(계속)를 선택합니다.
7. 즉시 적용(Apply immediately) 또는 예약된 다음 유지 관리 기간에 적용(Apply during the next scheduled maintenance window)을 선택합니다.
8. Modify DB instance(DB 인스턴스 수정)를 선택합니다.

AWS CLI

RDS Custom for SQL Server DB 인스턴스의 스토리지를 수정하려면 modify-db-instance AWS CLI 명령을 사용하면 됩니다. 필요한 경우 다음 파라미터를 설정합니다.

- `--allocated-storage` - DB 인스턴스에 할당할 스토리지 크기(GiB)입니다. 현재 값보다 커야 하며 20~16,384GiB여야 합니다.
- `--storage-type` - 스토리지 유형(예: gp2, gp3, io1 또는 io2)입니다.
- `--iops` - DB 인스턴스의 프로비저닝된 IOPS입니다. 프로비저닝된 IOPS를 지원하는 스토리지 유형(예: gp3, io1, io2)에만 지정할 수 있습니다.
- `--apply-immediately` – `--apply-immediately`를 사용하여 스토리지 변경 사항을 바로 적용합니다.

  그 밖에 다음 유지 관리 기간에 스토리지 변경 사항을 적용하려면 `--no-apply-immediately`(기본값)를 사용합니다.

다음 예에서는 my-custom-instance의 스토리지 크기를 200GiB로, 스토리지 유형을 io1로, 프로비저닝된 IOPS를 3000으로 변경합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier my-custom-instance \
    --storage-type io1 \
    --iops 3000 \
    --allocated-storage 200 \
    --apply-immediately
```

Windows의 경우:

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

```
    --db-instance-identifier my-custom-instance ^
    --storage-type io1 ^
    --iops 3000 ^
    --allocated-storage 200 ^
    --apply-immediately
```

RDS Custom for SQL Server 리소스 태깅

Amazon RDS 리소스와 마찬가지로 RDS Custom 리소스에 태그를 지정할 수 있지만, 몇 가지 중요한 차이점이 있습니다.

- RDS Custom 자동화에 필요한 `AWSRDSCustom` 태그를 생성하거나 수정해서는 안 됩니다. 이렇게 하면 자동화가 중단될 수 있습니다.
- `Name` 태그는 접두사가 `do-not-delete-rds-custom`인 RDS Custom 리소스에 추가됩니다. 고객이 전달한 키 값을 덮어씁니다.
- 생성 중에 RDS Custom DB 인스턴스에 추가된 태그는 다른 모든 관련 RDS Custom 리소스로 전파됩니다.
- DB 인스턴스 생성 후 RDS Custom 리소스에 태그를 추가하면 태그가 전파되지 않습니다.

리소스 태그 지정에 대한 일반적인 정보는 Amazon RDS 리소스에 태그 지정 섹션을 참조하세요.

RDS Custom for SQL Server DB 인스턴스 삭제

RDS Custom for SQL Server DB 인스턴스를 삭제하려면 다음을 수행하세요.

- DB 인스턴스 이름을 입력합니다.
- DB 인스턴스의 최종 DB 스냅샷을 생성하는 옵션을 선택하거나 선택 취소합니다.
- 자동화된 백업을 유지하는 옵션을 선택하거나 선택 취소합니다.

콘솔이나 CLI를 사용하여 RDS Custom for SQL Server DB 인스턴스를 삭제할 수 있습니다. 해당 DB 인스턴스를 삭제하는 데 필요한 시간은 백업 보존 기간(즉, 삭제할 백업 수), 삭제되는 데이터의 양과 최종 스냅샷 생성 여부에 따라 달라질 수 있습니다.

> **Warning**
>
> RDS Custom for SQL Server DB 인스턴스를 삭제하면 EC2 인스턴스 및 관련 Amazon EBS 볼륨이 영구적으로 삭제됩니다. 이러한 리소스를 종료하거나 삭제해서는 안 됩니다. 종료하거나 삭제하면 삭제 및 최종 스냅샷 생성이 실패할 수 있습니다.

> **Note**
>
> DB 인스턴스가 `creating`, `failed`, `incompatible-create`, `incompatible-restore` 또는 `incompatible-network` 상태일 때는 최종 DB 스냅샷을 생성할 수 없습니다. 자세한 내용은 Amazon RDS DB 인스턴스 상태 보기 단원을 참조하십시오.

> **Important**
>
> 최종 스냅샷을 선택하는 경우 DB 인스턴스 삭제가 진행되는 동안 DB 인스턴스에 데이터를 쓰지 않는 것이 좋습니다. 일단 DB 인스턴스 삭제가 시작되면 데이터 변경 사항이 최종 스냅샷에 캡처되지 않을 수 있습니다.

콘솔

RDS Custom DB 인스턴스를 삭제하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택한 후 삭제하려는 RDS Custom for SQL Server DB 인스턴스를 선택합니다. RDS Custom for SQL Server DB 인스턴스는 Instance (RDS Custom for SQL Server)(인스턴스(RDS Custom for SQL Server))를 표시합니다.
3. 작업에 대해 삭제를 선택합니다.
4. 최종 스냅샷을 생성하려면 Create final snapshot(최종 스냅샷 생성)을 선택하고 Final snapshot name(최종 스냅샷 이름)에 이름을 입력합니다.
5. 자동 백업을 보관하려면 Retain automated backups(자동 백업 보관)를 선택합니다.
6. 상자에 **`delete me`**를 입력합니다.
7. 삭제를 선택합니다.

AWS CLI

delete-db-instance AWS CLI 명령을 사용하여 RDS Custom for SQL Server DB 인스턴스를 삭제할 수 있습니다. 필수 파라미터 `--db-instance-identifier`를 사용하여 DB 인스턴스를 식별합니다. 나머지 파라미터는 Amazon RDS DB 인스턴스와 동일합니다.

다음 예제에서는 이름이 `my-custom-instance`인 RDS Custom for SQL Server DB 인스턴스를 삭제하고 최종 스냅샷을 생성하며 자동 백업을 유지합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds delete-db-instance \
    --db-instance-identifier my-custom-instance \
    --no-skip-final-snapshot \
    --final-db-snapshot-identifier my-custom-instance-final-snapshot \
    --no-delete-automated-backups
```

Windows의 경우:

```
aws rds delete-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --no-skip-final-snapshot ^
    --final-db-snapshot-identifier my-custom-instance-final-snapshot ^
    --no-delete-automated-backups
```

최종 스냅샷을 찍으려면 `--final-db-snapshot-identifier` 옵션이 필요하며 이 옵션을 지정해야 합니다.

최종 스냅샷을 건너뛰려면 명령에서 `--no-skip-final-snapshot` 옵션 및 `--final-db-snapshot-identifier` 옵션 대신 `--skip-final-snapshot` 옵션을 지정합니다.

자동 백업을 삭제하려면 명령에서 `--no-delete-automated-backups` 옵션 대신 `--delete-automated-backups` 옵션을 지정합니다.

RDS Custom for SQL Server DB 인스턴스 시작 및 중지

RDS Custom for SQL Server DB 인스턴스를 시작 및 중지할 수 있습니다. RDS Custom for SQL Server DB 인스턴스를 중지 및 시작할 때도 RDS for SQL Server DB 인스턴스의 일반적인 요구 사항

및 제한 사항이 동일하게 적용됩니다. 자세한 내용은 Amazon RDS DB 인스턴스의 일시적 중지 단원을 참조하십시오.

RDS Custom for SQL Server DB 인스턴스를 시작 및 중지할 때도 다음과 같은 고려 사항이 적용됩니다.

- DB 인스턴스가 `STOPPED` 상태인 동안에는 RDS Custom for SQL Server DB 인스턴스의 EC2 인스턴스 속성을 수정하는 작업이 지원되지 않습니다.
- 단일 가용 영역에 구성되어 있는 경우 RDS Custom for SQL Server DB 인스턴스를 중지 및 시작할 수 있습니다. 다중 AZ 구성에서는 RDS Custom for SQL Server DB 인스턴스를 중지할 수 없습니다.
- RDS Custom for SQL Server DB 인스턴스를 중지할 때 `SYSTEM` 스냅샷이 생성됩니다. RDS Custom for SQL Server DB 인스턴스를 다시 시작하면 스냅샷이 자동 삭제됩니다.
- RDS Custom for SQL Server DB 인스턴스를 중지할 때 EC2 인스턴스를 삭제하면 RDS Custom for SQL Server DB 인스턴스가 다시 시작될 때 `C:` 드라이브가 교체됩니다.
- RDS Custom for SQL Server DB 인스턴스를 중지해도 인스턴스 유형을 수정하지 않는 한 `C:\` 드라이브, 호스트 이름 및 사용자 지정 구성은 그대로 유지됩니다.
- 다음 작업을 수행하면 RDS Custom에서 DB 인스턴스를 지원 경계가 아닌 영역에 배치하게 되며, 이 경우에도 DB 인스턴스 시간에 대한 요금이 부과됩니다.
  - Amazon RDS가 중지되어 있는 동안 기본 EC2 인스턴스를 시작합니다. 문제를 해결하려면 `start-db-instance` Amazon RDS API를 호출하거나, EC2를 중지하여 RDS Custom 인스턴스가 `STOPPED` 상태가 되도록 합니다.
  - RDS Custom for SQL Server DB 인스턴스가 `ACTIVE` 상태일 때 기본 EC2 인스턴스를 중지합니다.

DB 인스턴스의 중지 및 시작에 관한 자세한 내용은 Amazon RDS DB 인스턴스의 일시적 중지 및 이전에 중지된 Amazon RDS DB 인스턴스 시작 단원을 참조하세요.

RDS Custom for SQL Server를 사용하여 Microsoft Active Directory 작업

RDS Custom for SQL Server를 사용하면 인스턴스를 자체 관리형 Active Directory(AD) 또는 AWS Managed Microsoft AD에 조인할 수 있습니다. 이는 온프레미스 데이터 센터, Amazon EC2 또는 다른 클라우드 서비스 공급자와 같이 AD가 호스팅되는 위치에 관계없이 적용됩니다.

사용자 및 서비스 인증의 경우, 중간 도메인 및 포리스트 신뢰를 사용하지 않고도 RDS Custom for SQL Server DB 인스턴스에서 NTLM 또는 Kerberos 인증을 사용할 수 있습니다. 사용자가 자체 조인된 Active Directory로 RDS Custom for SQL Server DB 인스턴스에서 인증을 시도하면 인증 요청이 자체 관리형 AD 또는 사용자가 지정한 AWS Managed Microsoft AD로 전달됩니다.

다음 섹션에서는 RDS Custom for SQL Server에 자체 관리형 Active Directory 및 AWS 관리형 Active Directory를 사용하여 작업하는 정보를 찾을 수 있습니다.

주제

- 리전 및 버전 사용 가능 여부
- 자체 관리형 또는 온프레미스 AD 구성
- AWS Directory Service를 사용하여 Microsoft Active Directory 구성
- 네트워크 구성 포트 규칙
- 네트워크 검증
- RDS Custom for SQL Server 인스턴스에 대한 Windows 인증 설정
- 도메인에서 DB 인스턴스 관리
- 도메인 멤버십 이해
- Active Directory 문제 해결

리전 및 버전 사용 가능 여부

RDS Custom for SQL Server는 RDS Custom for SQL Server가 지원되는 모든 리전에서 NTLM 또는 Kerberos를 사용하여 자체 관리형 AD 및 AWS Managed Microsoft AD를 모두 지원합니다. 자세한 내용은 RDS Custom을 지원하는 리전 및 DB 엔진 단원을 참조하십시오.

자체 관리형 또는 온프레미스 AD 구성

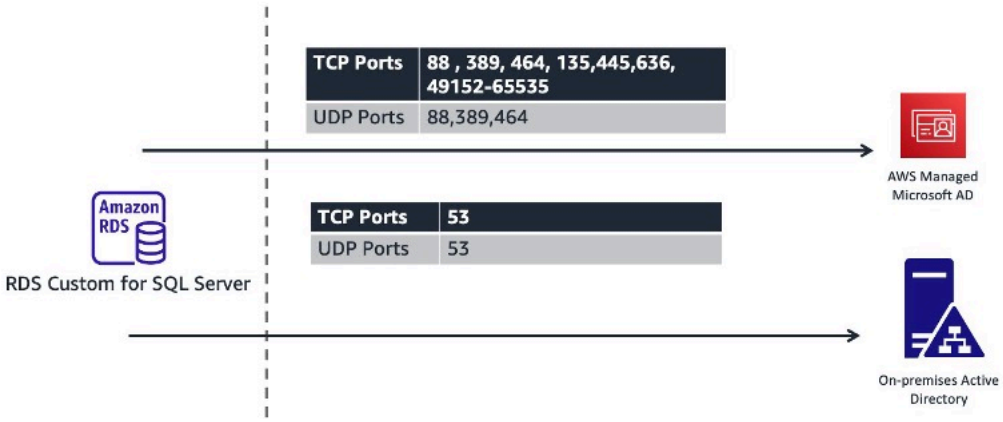
온프레미스 또는 자체 관리형 Microsoft AD를 RDS Custom for SQL Server DB 인스턴스에 조인하려면 Active Domain을 다음과 같이 구성해야 합니다.

- 자체 관리형 또는 온프레미스 AD에서 RDS Custom for SQL Server DB 인스턴스와 연결된 VPC의 서브넷을 정의합니다. VPC의 서브넷과 AD 사이트의 서브넷 간에 충돌이 없는지 확인합니다.
- AD 도메인 컨트롤러의 도메인 기능 수준은 Windows Server 2008 R2 이상입니다.
- AD 도메인 이름은 단일 레이블 도메인(SLD) 형식일 수 없습니다. RDS Custom for SQL Server는 SLD 도메인을 지원하지 않습니다.
- AD의 정규화된 도메인 이름(FQDN)이 47자를 초과하면 안 됩니다.

네트워크 연결 구성

다음과 같은 방식으로 자체 관리형 또는 온프레미스 AD 네트워크 연결을 구성합니다.

- RDS Custom for SQL Server 인스턴스가 실행 중인 Amazon VPC와 AD 간의 연결을 설정합니다. AWS Direct Connect, AWS VPN, AWS Transit Gateway 및 VPC 피어링을 사용합니다.
- 자체 관리형 또는 온프레미스 AD에 대해 RDS Custom for SQL Server 보안 그룹 및 네트워크 ACL 포트의 트래픽을 허용합니다. 자세한 내용은 네트워크 구성 포트 규칙 단원을 참조하십시오.



DNS 확인 구성

자체 관리형 또는 온프레미스 AD를 사용하여 DNS 확인을 구성하려면 다음 요구 사항을 설정합니다.

- 자체 호스팅된 Active Directory의 정규화된 도메인 이름(FQDN)을 확인하도록 VPC 내에서 DNS 확인을 구성합니다. FQDN의 예는 `corp.example.local`입니다. DNS 확인을 구성하려면 Amazon Route 53 아웃바운드 엔드포인트 및 해석기 규칙이 있는 특정 도메인에 대한 쿼리를 전달하도록 VPC DNS 해석기를 구성합니다. 자세한 내용은 DNS 레코드를 확인하도록 Route 53 Resolver 아웃바운드 엔드포인트 구성을 참조하세요.

- VPC와 온프레미스 리소스를 모두 활용하는 워크로드의 경우 온프레미스에서 호스팅되는 DNS 레코드를 확인해야 합니다. 온프레미스 리소스는 AWS에서 호스팅되는 이름을 확인해야 할 수 있습니다.

  하이브리드 클라우드 설정을 생성하려면 해석기 엔드포인트와 조건부 전달을 사용하여 온프레미스 리소스와 사용자 지정 VPC 간의 DNS 쿼리를 확인합니다. 자세한 내용은 Amazon Route 53 개발자 안내서의 VPC와 네트워크 간 DNS 쿼리 해석을 참조하세요.

> **Important**
>
> RDS Custom for SQL Server에서 네트워크 인터페이스의 DNS 해석기 설정을 수정하면 DNS 지원 VPC 엔드포인트가 더 이상 제대로 작동하지 않습니다. 인터넷 액세스가 없는 프라이빗 서브넷 내의 인스턴스에는 DNS 지원 VPC 엔드포인트가 필요합니다.

AWS Directory Service를 사용하여 Microsoft Active Directory 구성

AWS Managed Microsoft AD는 AWS에서 완전 관리형 Microsoft Active Directory를 생성하며, Windows Server 2019에 의해 구동되고 2012 R2 포리스트 및 도메인 기능 수준에서 작동합니다. AWS Directory Service는 Amazon VPC의 서로 다른 서브넷에 도메인 컨트롤러를 생성하여 장애가 발생하더라도 디렉터리를 쉽게 사용할 수 있도록 합니다.

AWS Managed Microsoft AD를 사용하여 디렉터리를 생성하려면 AWS Directory Service 관리 안내서의 AWS Managed Microsoft AD 시작하기를 참조하세요.

네트워크 연결 구성

디렉터리와 DB 인스턴스 사이에 VPC 간 트래픽 활성화

디렉터리와 DB 인스턴스를 동일한 VPC에 배치하려면 이 단계를 건너뛰고 네트워크 구성 포트 규칙의 다음 단계로 이동하세요.

디렉터리와 DB 인스턴스를 서로 다른 VPC에 배치하려면 VPC 피어링 또는 AWS Transit Gateway를 사용하여 VPC 간 트래픽을 구성하세요. VPC 피어링 사용에 대한 자세한 내용은 Amazon VPC 피어링 가이드의 VPC 피어링이란 무엇인가요? 및 Amazon VPC Transit Gateways의 AWS Transit Gateway란 무엇인가요?를 참조하세요.

VPC 피어링을 사용하여 VPC 간 트래픽 활성화

1. 네트워크 트래픽이 양방향으로 흐를 수 있도록 적절한 VPC 라우팅 규칙을 설정합니다.

2. DB 인스턴스의 보안 그룹이 디렉터리의 보안 그룹에서 인바운드 트래픽을 수신할 수 있도록 허용합니다. 자세한 내용은 네트워크 구성 포트 규칙 단원을 참조하십시오.
3. 네트워크 액세스 제어 목록(ACL)은 트래픽을 차단해서는 안 됩니다.

다른 AWS 계정가 디렉터리를 소유하는 경우 디렉터리를 공유해야 합니다. AWS Directory Service 관리 안내서의 자습서: 원활한 EC2 도메인 조인을 위해 AWS Managed Microsoft AD 공유에 따라 RDS Custom for SQL Server 인스턴스가 있는 AWS 계정와 디렉터리를 공유합니다.

AWS 계정 간에 디렉터리 공유

1. DB 인스턴스용 계정을 사용하여 AWS Directory Service 콘솔에 로그인하고 계속하기 전에 도메인이 SHARED 상태가 되었는지 확인합니다.
2. DB 인스턴스용 계정을 사용하여 AWS Directory Service 콘솔에 로그인한 후 디렉터리 ID 값을 기록해 둡니다. 이 ID를 사용하여 DB 인스턴스를 도메인에 조인합니다.

DNS 확인 구성

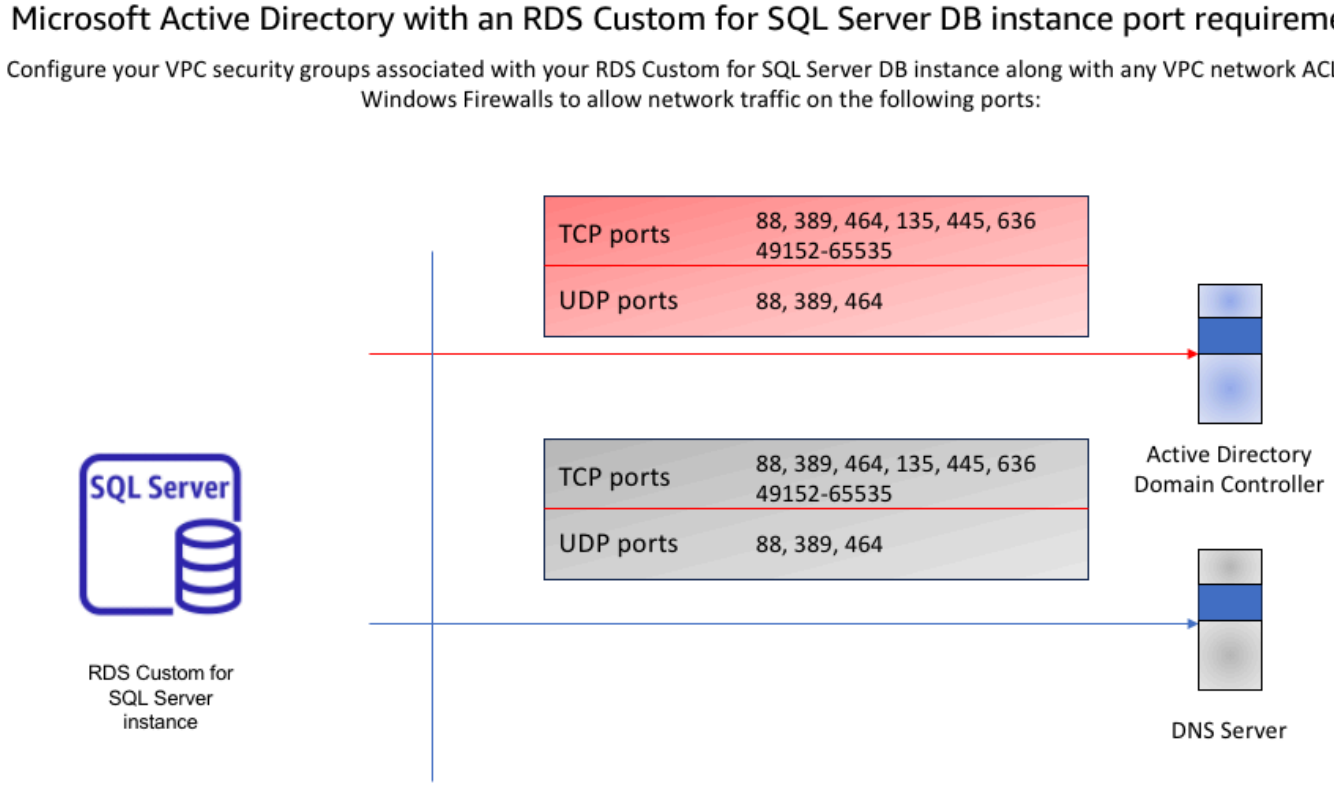
AWS Managed Microsoft AD를 사용하여 디렉터리를 생성하면 AWS Directory Service가 자동으로 두 개의 도메인 컨트롤러를 생성하고 사용자를 대신하여 DNS 서비스를 추가합니다.

기존 AWS Managed Microsoft AD가 있거나 RDS Custom for SQL Server DB 인스턴스가 아닌 다른 VPC에서 시작할 계획이 있는 경우 Route 53 아웃바운드 및 해석기 규칙이 있는 특정 도메인에 대한 쿼리를 전달하도록 VPC DNS 해석기를 구성합니다. DNS 레코드를 확인하도록 Route 53 Resolver 아웃바운드 엔드포인트 구성을 참조하세요.

네트워크 구성 포트 규칙

다음 네트워크 구성을 충족하는지 확인합니다.

- RDS Custom for SQL Server DB 인스턴스를 생성하려는 Amazon VPC와 자체 관리형 Active Directory 또는 AWS Managed Microsoft AD 간에 연결이 구성되었습니다. 자체 관리형 Active Directory의 경우, AWS Direct Connect, AWS VPN, VPC 피어링 또는 AWS Transit Gateway를 사용하여 연결을 설정합니다. AWS Managed Microsoft AD의 경우, VPC 피어링을 사용하여 연결을 설정합니다.
- RDS Custom for SQL Server DB 인스턴스를 만드는 서브넷의 보안 그룹과 VPC 네트워크 ACL이 다음 다이어그램에 표시된 방향으로 포트를 통한 트래픽을 허용하는지 확인합니다.



다음 테이블에는 각 포트의 역할이 나와 있습니다.

프로토콜	포트	역할
TCP/UDP	53	도메인 이름 시스템(DNS)
TCP/UDP	88	Kerberos 인증
TCP/UDP	464	암호 변경/설정
TCP/UDP	389	Lightweight Directory Access Protocol(LDAP)
TCP	135	분산 컴퓨팅 환경/엔드포인트 매퍼(DCE/EPMAP)
TCP	445	디렉터리 서비스 SMB 파일 공유

프로토콜	포트	역할
TCP	636	Lightweight Directory Access Protocol over TLS/SSL(LDAPS)
TCP	49,152~65,535	RPC용 임시 포트

- 일반적으로 도메인 DNS 서버는 AD 도메인 컨트롤러에 있습니다. 이 기능을 사용하기 위해 VPC DHCP 옵션 세트를 구성할 필요는 없습니다. 자세한 정보는 Amazon VPC 사용 설명서의 DHCP 옵션 세트를 참조하세요.

> **Important**
>
> VPC 네트워크 ACL을 사용하는 경우 RDS Custom for SQL Server DB 인스턴스의 동적 포트(49152~65535)를 통한 아웃바운드 트래픽도 허용해야 합니다. 이러한 트래픽 규칙이 각 AD 도메인 컨트롤러, DNS 서버 및 RDS Custom for SQL Server DB 인스턴스에 적용되는 방화벽에도 반영되는지 확인하세요.
>
> VPC 보안 그룹에서는 네트워크 트래픽이 시작되는 방향으로만 포트를 열어야 하지만, 대부분의 Windows 방화벽과 VPC 네트워크 ACL에서는 포트가 양방향으로 열려 있어야 합니다.

네트워크 검증

RDS Custom 인스턴스를 자체 관리형 또는 AWS Managed Microsoft AD에 조인하기 전에 RDS Custom for SQL Server 인스턴스를 시작하려는 위치와 동일한 VPC의 EC2 인스턴스에서 다음을 확인합니다.

- 정규화된 도메인 이름(FQDN)을 도메인 컨트롤러 IP로 확인할 수 있는지 확인합니다.

```
nslookup corp.example.com
```

명령은 유사한 출력을 반환해야 합니다.

```
Server:  ip-10-0-0-2.us-west-2.compute.internal
Address:  25.0.0.2

Non-authoritative answer:
```

```
Name:    corp.example.com
Addresses:  40.0.9.25 (DC1 IP)
            40.0.50.123 (DC2 IP)
```

- RDS Custom 인스턴스를 시작하는 VPC의 EC2 인스턴스에서 AWS 서비스를 확인합니다.

```
$region='input-your-aws-region'
$domainFQDN='input-your-domainFQDN'

function Test-DomainPorts {
    param (
        [string]$Domain,
        [array]$Ports
    )

    foreach ($portInfo in $Ports) {
        try {
            $conn = New-Object System.Net.Sockets.TcpClient
            $connectionResult = $conn.BeginConnect($Domain, $portInfo.Port, $null,
 $null)
            $success = $connectionResult.AsyncWaitHandle.WaitOne(1000) # 1 second
 timeout
            if ($success) {
                $conn.EndConnect($connectionResult)
                $result = $true
            } else {
                $result = $false
            }
        }
        catch {
            $result = $false
        }
        finally {
            if ($null -ne $conn) {
                $conn.Close()
            }
        }
        Write-Host "$($portInfo.Description) port open: $result"
    }
}

# Check if ports can be reached
$ports = @(
    @{Port = 53;   Description = "DNS"},
```

```
    @{Port = 88;   Description = "Kerberos"},
    @{Port = 389;  Description = "LDAP"},
    @{Port = 445;  Description = "SMB"},
    @{Port = 5985; Description = "WinRM"},
    @{Port = 636;  Description = "LDAPS"},
    @{Port = 3268; Description = "Global Catalog"},
    @{Port = 3269; Description = "Global Catalog over SSL"},
    @{Port = 9389; Description = "AD DS"}
)

function Test-DomainReachability {
    param (
        [string]$DomainName
    )

    try {
        $dnsResults = Resolve-DnsName -Name $DomainName -ErrorAction Stop
        Write-Host "Domain $DomainName is successfully resolving to following IP
 addresses: $($dnsResults.IpAddress)"
        Write-Host ""
        return $true
    }
    catch {
        Write-Host ""
        Write-Host "Error Message: $($_.Exception.Message)"
        Write-Host "Domain $DomainName reachability check failed, please Configure
 DNS resolution"
        return $false
    }
}

$domain = (Get-WmiObject Win32_ComputerSystem).Domain
if ($domain -eq 'WORKGROUP') {
    Write-Host ""
    Write-Host "Host $env:computername is still part of WORKGROUP and not part of any
 domain"
    }
else {
    Write-Host ""
    Write-Host "Host $env:computername is joined to $domain domain"
    Write-Host ""
    }
```

```
$isReachable = Test-DomainReachability -DomainName $domainFQDN
if ($isReachable) {
    write-Host "Checking if domain $domainFQDN is reachable on required ports  "
    Test-DomainPorts -Domain $domainFQDN -Ports $ports
}
else {
    Write-Host "Port check skipped. Domain not reachable"
}



# Get network adapter configuration
$networkConfig = Get-WmiObject Win32_NetworkAdapterConfiguration |
                 Where-Object { $_.IPEnabled -eq $true } |
                 Select-Object -First 1

# Check DNS server settings
$dnsServers = $networkConfig.DNSServerSearchOrder

if ($dnsServers) {
    Write-Host "`nDNS Server settings:"
    foreach ($server in $dnsServers) {
        Write-Host "  - $server"
    }
} else {
    Write-Host "`nNo DNS servers configured or unable to retrieve DNS server
 information."
}

write-host ""

# Checks reachability to dependent services
$services = "s3", "ec2", "secretsmanager", "logs", "events", "monitoring", "ssm",
 "ec2messages", "ssmmessages"

function Get-TcpConnectionAsync {
    param (
        $ServicePrefix,
        $region
    )
    $endpoint = "${ServicePrefix}.${region}.amazonaws.com"
    $tcp = New-Object Net.Sockets.TcpClient
    $result = $false
```

```
    try {
        $connectTask = $tcp.ConnectAsync($endpoint, 443)
        $timedOut = $connectTask.Wait(3000)
        $result = $tcp.Connected
    }
    catch {
        $result = $false
    }
    return $result
}

foreach ($service in $services) {
    $validationResult = Get-TcpConnectionAsync -ServicePrefix $service -Region
 $region
    Write-Host "Reachability to $service is $validationResult"
}
```

`TcpTestSucceeded` 값은 `s3`, `ec2`, `secretsmanager`, `logs`, `events`, `monitoring`, `ssm`, `ec2messages`, `ssmmessages`에 대해 `True`를 반환해야 합니다.

RDS Custom for SQL Server 인스턴스에 대한 Windows 인증 설정

AD 도메인에 조인된 RDS Custom for SQL Server DB 인스턴스를 소유한 AWS 계정에 대해 전용 OU 및 해당 OU 범위의 서비스 보안 인증을 만드는 것이 좋습니다. 전용 OU 및 서비스 보안 인증을 지정하면 권한 충돌을 피하고 최소 권한 원칙을 따릅니다.

Active Directory 수준 그룹 정책은 AWS 자동화 및 권한과 충돌할 수 있습니다. RDS Custom for SQL Server에 대해 생성한 OU에만 적용되는 GPO를 선택하는 것이 좋습니다.

- 자체 관리형 또는 온프레미스 AD에서 OU 및 AD 도메인 사용자를 생성하려면 도메인 컨트롤러를 도메인 관리자로 연결할 수 있습니다.
- AWS Directory Service 디렉터리에서 사용자 및 그룹을 생성하려면 관리 인스턴스에 연결하고 사용자 및 그룹 생성 권한이 있는 사용자로 로그인해야 합니다. 자세한 내용은 AWS Directory Service 관리 안내서의 AWS Managed Microsoft AD에서의 사용자 및 그룹 관리를 참조하세요.
- Amazon EC2 Windows Server 인스턴스에서 Active Directory를 관리하려면 EC2 인스턴스에서 Active Directory 도메인 서비스 및 Active Directory Lightweight Directory 서비스 도구를 설치해야 합니다. 자세한 내용은 AWS Directory Service 관리 안내서의 AWS Managed Microsoft AD용 Active Directory 관리 도구 설치를 참조하세요.

- 이러한 도구는 관리가 용이하도록 RDS Custom for SQL Server DB 인스턴스가 아닌 별도의 EC2 인스턴스에 설치하는 것이 좋습니다.

AD 도메인 서비스 계정의 요구 사항은 다음과 같습니다.

- AD 도메인에는 컴퓨터를 도메인에 연결할 수 있는 권한이 위임된 서비스 계정이 있어야 합니다. 도메인 서비스 계정은 특정 작업을 수행할 권한이 위임된 AD의 사용자 계정입니다.
- RDS Custom for SQL Server에 조인하려는 조직 단위의 도메인 서비스 계정에 다음 권한을 위임합니다.
  - 검증된 DNS 호스트 이름 쓰기 기능
  - 서비스 보안 주체 이름에 쓸 수 있는 검증된 기능
  - 컴퓨터 객체 생성 및 삭제
- 자체 관리형 및 온프레미스 AD의 경우 도메인 서비스 계정은 'AWS위임 도메인 이름 시스템 관리자' 그룹의 멤버여야 합니다.
- AWS Managed Microsoft AD의 경우 도메인 서비스 계정은 'DnsAdmins' 그룹의 멤버여야 합니다.

이러한 권한은 컴퓨터 객체를 자체 관리형 AD 및 AWS Managed Microsoft AD에 조인하는 데 필요한 최소 권한 집합입니다. 자세한 내용은 Microsoft Windows Server 설명서의 오류: 제어를 위임받은 관리자가 아닌 사용자가 컴퓨터를 도메인 컨트롤러에 조인하려고 하면 액세스가 거부됨 항목을 참조하세요.

> **Important**
>
> DB 인스턴스를 만든 후에는 RDS Custom for SQL Server가 조직 단위(OU)에서 생성한 컴퓨터 객체를 옮기지 마세요. 연결된 객체를 이동하면 RDS Custom for SQL Server DB 인스턴스가 잘못 구성될 수 있습니다. Amazon RDS에서 생성한 컴퓨터 객체를 이동해야 하는 경우 ModifyDBInstance 작업을 사용하여 컴퓨터 객체의 원하는 위치로 도메인 파라미터를 수정하세요.

주제

- 1단계: AD에서 조직 단위(OU) 생성
- 2단계: AD 도메인 사용자 생성
- 3단계: 자체 관리형 또는 AWS Managed Microsoft AD에서 AD 사용자에게 제어 권한 위임
- 4단계: 보안 암호 생성

- 5단계: RDS Custom for SQL Server DB 인스턴스 생성 또는 수정
- 6단계: Windows 인증 SQL Server 로그인 생성
- 7단계: Kerberos 또는 NTLM 인증 사용

1단계: AD에서 조직 단위(OU) 생성

다음 단계에 따라 AD에서 조직 단위를 생성합니다.

AD에서 OU 생성

1. 도메인 관리자로 도메인 AD에 연결합니다.
2. Active Directory 사용자 및 컴퓨터를 열고 OU를 생성할 도메인을 선택합니다.
3. 도메인을 마우스 오른쪽 버튼으로 클릭하고 새로 만들기를 선택한 다음 조직 단위를 선택합니다.
4. OU 이름을 입력합니다.

   컨테이너가 실수로 삭제되지 않도록 보호를 활성화합니다.
5. 확인을 선택합니다. 새 OU는 도메인 아래에 표시됩니다.

AWS Managed Microsoft AD의 경우 이 OU의 이름은 디렉터리 생성 시 입력한 NetBIOS 이름에 근거를 둡니다. 이 OU는 AWS가 소유하며, 전체 제어 권한을 부여한 모든 AWS 관련 디렉터리 객체가 포함됩니다. 기본적으로 이 OU에는 컴퓨터와 사용자라는 두 개의 하위 OU가 있습니다. RDS Custom이 생성하는 새 OU는 NetBIOS를 기반으로 하는 OU의 하위 OU입니다.

2단계: AD 도메인 사용자 생성

도메인 사용자 보안 인증 정보는 Secrets Manager에서 보안 암호로 사용됩니다.

AD에서 AD 도메인 사용자 생성

1. Active Directory 사용자 및 컴퓨터를 열고 사용자를 생성할 도메인과 OU를 선택합니다.
2. 사용자 객체를 마우스 오른쪽 버튼으로 클릭하고 새로 만들기, 사용자 순으로 선택합니다.
3. 사용자의 이름, 성 및 로그인 이름을 입력합니다. 다음을 클릭합니다.
4. 사용자의 암호를 입력합니다. 다음 로그인 시 사용자가 암호를 변경해야 함 또는 계정이 비활성화됨을 선택하지 마세요. 다음을 클릭합니다.
5. 확인을 클릭합니다. 새 사용자는 도메인 아래에 표시됩니다.

3단계: 자체 관리형 또는 AWS Managed Microsoft AD에서 AD 사용자에게 제어 권한 위임

도메인의 AD 도메인 사용자에게 제어 권한을 위임하려면

1. Active Directory 사용자 및 컴퓨터 MMC 스냅인을 열고 도메인을 선택합니다.
2. 이전에 만든 OU를 마우스 오른쪽 버튼으로 클릭하고 제어 위임을 선택합니다.
3. 제어 위임 마법사에서 다음을 클릭합니다.
4. 사용자 또는 그룹 섹션에서 추가를 클릭합니다.
5. 사용자, 컴퓨터 또는 그룹 선택에서 생성한 AD 사용자를 입력하고 이름 확인을 클릭합니다. AD 사용자 확인에 성공하면 확인을 클릭합니다.
6. 사용자 또는 그룹 섹션에서 AD 사용자가 추가되었는지 확인하고 다음을 클릭합니다.
7. 위임할 작업 섹션에서 위임할 사용자 지정 작업 생성을 선택하고 다음을 클릭합니다.
8. Active Directory 객체 유형 섹션에서:

   폴더의 다음 객체만을 선택합니다.

   컴퓨터 객체를 선택합니다.

   이 폴더에서 선택한 객체 생성을 선택합니다.

   이 폴더에서 선택한 객체 삭제를 선택하고 다음을 클릭합니다.
9. 권한 섹션에서:

   일반을 선택한 상태로 유지합니다.

   DNS 호스트 이름에 대한 검증된 쓰기를 선택합니다.

   서비스 보안 주체 이름에 대한 검증된 쓰기를 선택하고 다음을 클릭합니다.
10. 제어 위임 마법사를 완료하려면에서 설정을 확인한 다음 마침을 클릭합니다.

4단계: 보안 암호 생성

Active Directory에 포함하려는 RDS Custom for SQL Server DB 인스턴스가 포함된 동일한 AWS 계정 및 리전에 보안 암호를 만들어야 합니다. 2단계: AD 도메인 사용자 생성에서 생성된 AD 도메인 사용자의 자격 증명을 저장합니다.

Console

- AWS Secrets Manager에서 새 암호 저장을 선택합니다.
- 보안 암호 유형에서 다른 유형의 보안 암호를 선택합니다.
- 키/값 쌍의 경우 2개의 키를 추가합니다.
  - 첫 번째 키 값에는 SELF_MANAGED_ACTIVE_DIRECTORY_USERNAME을 입력하고 AD 사용자의 사용자 이름(도메인 접두사 제외)을 입력합니다.
  - 두 번째 키에 SELF_MANAGED_ACTIVE_DIRECTORY_PASSWORD를 입력하고 도메인에 AD 사용자의 암호를 입력합니다.
- 암호화 키에 RDS Custom for SQL Server 인스턴스를 생성하는 데 사용한 것과 동일한 AWS KMS 키를 입력합니다.
- 보안 암호 이름에서 do-not-delete-rds-custom-으로 시작하는 보안 암호 이름을 선택하여 인스턴스 프로파일이 이 보안 암호에 액세스할 수 있도록 합니다. 보안 암호에 다른 이름을 선택하려면 보안 암호 이름에 액세스하도록 RDSCustomInstanceProfile을 업데이트합니다.
- (선택 사항) 설명에 보안 암호 이름에 대한 설명을 입력합니다.
- 태그 Key="AWSRDSCustom",Value="custom-sqlserver" 추가
- 저장을 클릭한 후 다음을 클릭합니다.
- 교체 설정 구성에서 기본값을 유지하고 다음을 선택합니다.
- 보안 암호 설정을 검토하고 저장을 클릭합니다.
- 새 보안 암호를 선택하고 보안 암호 ARN의 값을 복사합니다. 다음 단계에서 이를 사용하여 Active Directory를 설정합니다.

CLI

CLI에서 다음 명령을 실행하여 보안 암호를 생성합니다.

```
# Linux based
aws secretsmanager create-secret \
--name do-not-delete-rds-custom-DomainUserCredentails \
--description "Active directory user credentials for managing RDS Custom" \
--secret-string "{\"SELF_MANAGED_ACTIVE_DIRECTORY_USERNAME\":\"tester\",
\"SELF_MANAGED_ACTIVE_DIRECTORY_PASSWORD\":\"xxxxxxxx\"}" \
--kms-key-id <RDSCustomKMSKey> \
--tags Key="AWSRDSCustom",Value="custom-sqlserver"

# Windows based
```

```
aws secretsmanager create-secret ^
--name do-not-delete-rds-custom-DomainUserCredentails ^
--description "Active directory user credentials for managing RDS Custom" ^
--secret-string "{\"SELF_MANAGED_ACTIVE_DIRECTORY_USERNAME\":\"tester\",
\"SELF_MANAGED_ACTIVE_DIRECTORY_PASSWORD\":\"xxxxxxxx\"}" ^
--kms-key-id <RDSCustomKMSKey> ^
--tags Key="AWSRDSCustom",Value="custom-sqlserver"
```

5단계: RDS Custom for SQL Server DB 인스턴스 생성 또는 수정

디렉터리에서 사용할 RDS Custom for SQL Server DB 인스턴스를 생성하거나 수정합니다. 콘솔, CLI 또는 RDS API를 사용하여 DB 인스턴스를 디렉터리에 연결할 수 있습니다. 이 작업을 다음 중 한 가지 방법으로 수행할 수 있습니다.

- 콘솔, create-db-instance CLI 명령 또는 CreateDBInstance RDS API 작업을 사용하여 새 SQL Server DB 인스턴스를 생성합니다.

  지침은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
- 콘솔, modify-db-instance CLI 명령 또는 ModifyDBInstance RDS API 작업을 사용하여 기존 SQL Server DB 인스턴스를 수정합니다.

  지침은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.
- 콘솔, restore-db-instance-from-db-snapshot CLI 명령 또는 RestoreDBInstanceFromDBSnapshot RDS API 작업을 사용하여 DB 스냅샷에서 SQL Server DB 인스턴스를 복원합니다.

  지침은 DB 인스턴스 복원 섹션을 참조하세요.
- 콘솔, restore-db-instance-to-point-in-time CLI 명령 또는 RestoreDBInstanceToPointInTime RDS API 작업을 사용하여 SQL Server DB 인스턴스를 특정 시점으로 복구합니다.

  지침은 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 섹션을 참조하세요.

> **Note**
>
> RDS Custom for SQL Server 인스턴스가 이미 AD에 수동으로 조인된 경우 네트워크 구성 포트 규칙, 네트워크 검증의 설정을 확인하고 1~4단계를 완료합니다. 도메인 조인 보안 인증 정보 및 구성이 RDS Custom에 등록되어 CNAME를 모니터링, 등록 및 복구 작업을 수행할 수 있

도록 `--domain-fqdn`, `--domain-ou` 및 `--domain-auth-secret-arn`을 AD로 업데이트 합니다.

AWS CLI를 사용하는 경우 생성한 도메인 디렉터리를 DB 인스턴스에서 사용하려면 다음과 같은 파라미터가 필요합니다.

- `--domain-fqdn` 파라미터에는 자체 관리형 AD의 정규화된 도메인 이름을 사용합니다.
- `--domain-ou` 파라미터에는 자체 관리형 AD에서 만든 OU를 사용합니다.
- `--domain-auth-secret-arn` 파라미터에는 생성한 보안 암호 ARN의 값을 사용합니다.

Important

자체 관리형 AD 도메인에 조인하거나 자체 관리형 AD 도메인 또는 AWS Managed Microsoft AD에서 제거하도록 DB 인스턴스를 수정한 경우 변경 사항이 적용되려면 DB 인스턴스를 재부팅해야 합니다. 변경 사항을 즉시 적용하거나 다음 유지 관리 기간까지 기다릴 수 있습니다. 즉시 적용 옵션을 선택하면 단일 AZ DB 인스턴스에 가동 중단이 발생합니다. 다중 AZ DB 클러스터는 재부팅을 완료하기 전에 장애 조치를 수행합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

다음 CLI 명령은 새 RDS Custom for SQL Server DB 인스턴스를 만들어 자체 관리형 또는 AWS Managed Microsoft AD 도메인에 조인합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance  \
--engine custom-sqlserver-se \
--engine-version 15.00.4312.2.v1 \
--db-instance-identifier my-custom-instance \
--db-instance-class db.m5.large \
--allocated-storage 100 --storage-type io1 --iops 1000 \
--master-username my-master-username \
--master-user-password my-master-password \
--kms-key-id  my-RDSCustom-key-id \
--custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance  \
--domain-fqdn "corp.example.com" \
--domain-ou "OU=RDSCustomOU,DC=corp,DC=example,DC=com" \
```

```
--domain-auth-secret-arn "arn:aws:secretsmanager:region:account-number:secret:do-not-delete-rds-custom-my-AD-test-secret-123456" \
--db-subnet-group-name my-DB-subnet-grp \
--vpc-security-group-ids  my-securitygroup-id \
--no-publicly-accessible \
--backup-retention-period 3 \
--port 8200 \
--region us-west-2 \
--no-multi-az
```

Windows의 경우:

```
aws rds create-db-instance  ^
--engine custom-sqlserver-se ^
--engine-version 15.00.4312.2.v1 ^
--db-instance-identifier my-custom-instance ^
--db-instance-class db.m5.large ^
--allocated-storage 100 --storage-type io1 --iops 1000 ^
--master-usernamemy-master-username ^
--master-user-password my-master-password ^
--kms-key-id  my-RDSCustom-key-id ^
--custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance  ^
--domain-fqdn "corp.example.com" ^
--domain-ou "OU=RDSCustomOU,DC=corp,DC=example,DC=com" ^
--domain-auth-secret-arn "arn:aws:secretsmanager:region:account-number:secret:do-not-delete-rds-custom-my-AD-test-secret-123456" ^
--db-subnet-group-name my-DB-subnet-grp ^
--vpc-security-group-ids  my-securitygroup-id ^
--no-publicly-accessible ^
--backup-retention-period 3 ^
--port 8200 ^
--region us-west-2 ^
--no-multi-az
```

Important

AWS Managed Microsoft AD에 대한 NetBIOS가 corpexample인 경우 OU 자체로 표시됩니다. 이전에 생성된 새 OU는 중첩 OU로 표시됩니다. AWS Managed Microsoft AD의 경우 `--domain-ou`을

"OU=RDSCustomOU,OU=corpexample,DC=corp,DC=example,DC=com"으로 설정합니다.

다음 명령은 Active Directory 도메인을 사용하도록 기존 RDS Custom for SQL Server DB 인스턴스를 수정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier my-custom-instance \
    --domain-fqdn "corp.example.com" \
    --domain-ou "OU=RDSCustomOU,DC=corp,DC=example,DC=com" \
    --domain-auth-secret-arn "arn:aws:secretsmanager:region:account-number:secret:do-not-delete-rds-custom-my-AD-test-secret-123456" \
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --domain-fqdn "corp.example.com" ^
    --domain-ou "OU=RDSCustomOU,DC=corp,DC=example,DC=com" ^
    --domain-auth-secret-arn "arn:aws:secretsmanager:region:account-number:secret:do-not-delete-rds-custom-my-AD-test-secret-123456" ^
```

다음 CLI 명령은 Active Directory 도메인에서 RDS Custom for SQL Server DB 인스턴스를 제거합니다.

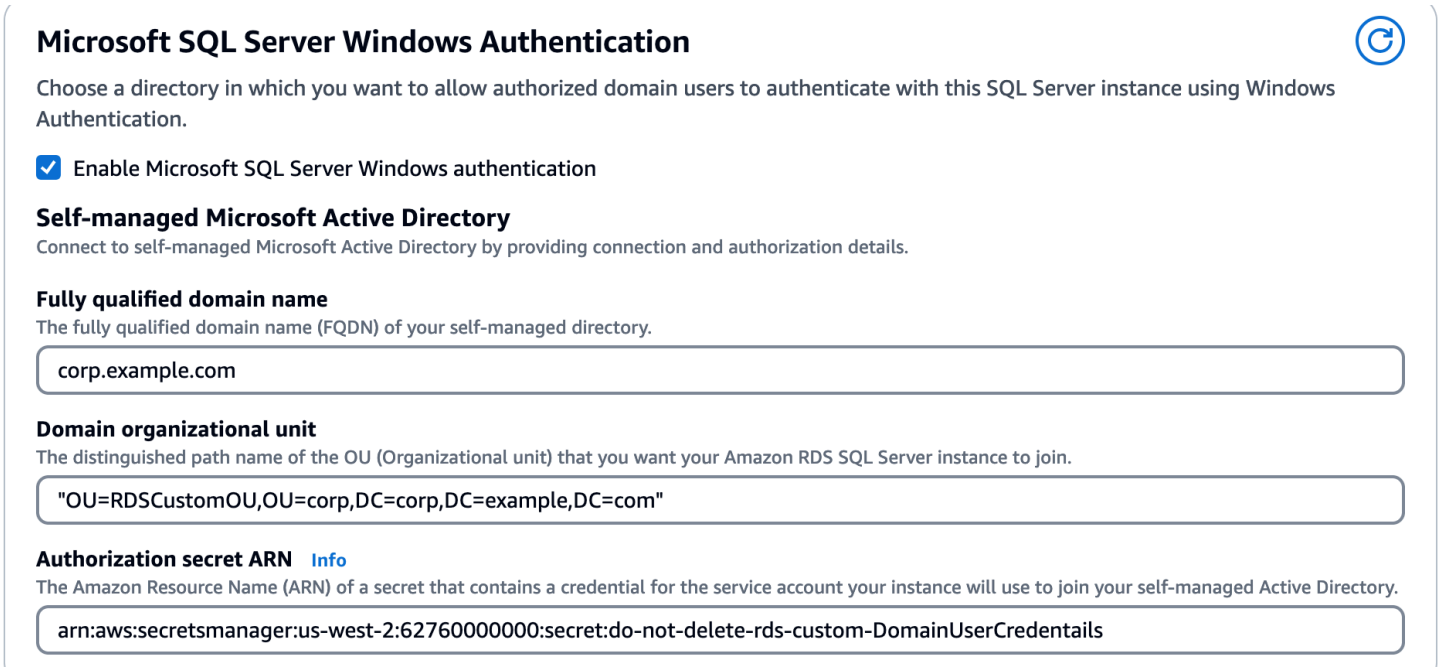
대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier my-custom-instance \
    --disable-domain
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --disable-domain
```

콘솔을 사용하여 인스턴스를 생성하거나 수정할 때 Microsoft SQL Server Windows 인증 활성화를 클릭하여 다음 옵션을 확인합니다.



도메인 FQDN이 도메인 컨트롤러 IP 주소로 확인되고 있는지 확인하는 것은 사용자의 책임입니다. 도메인 컨트롤러 IP가 확인되지 않으면 도메인 조인 작업은 실패하지만 RDS Custom for SQL Server 인스턴스 만들기에 성공합니다. 문제 해결 정보는 Active Directory 문제 해결를 참조하세요.

6단계: Windows 인증 SQL Server 로그인 생성

다른 DB 인스턴스의 경우와 같은 방법으로 Amazon RDS 마스터 사용자 보안 인증 정보를 사용하여 SQL Server DB 인스턴스에 연결합니다. DB 인스턴스는 AD 도메인에 조인되므로 SQL Server 로그인 및 사용자를 프로비저닝할 수 있습니다. 이 작업은 AD 도메인의 AD 사용자 및 그룹 유틸리티에서 수행합니다. 데이터베이스 권한은 이러한 Windows 로그인에 부여되거나 취소되는 표준 SQL Server 권한을 통해 관리됩니다.

AD 사용자가 SQL Server로 인증하려면 AD 사용자 또는 사용자가 속한 Active Directory 그룹에 대한 SQL Server Windows 로그인이 있어야 합니다. 세분화된 액세스 제어는 이러한 SQL Server 로그인에 대한 권한을 부여하거나 취소하여 처리합니다. SQL Server 로그인이 없거나 이러한 로그인이 있는 AD 그룹에 속하지 않은 AD 사용자는 SQL Server DB 인스턴스에 액세스할 수 없습니다.

AD SQL Server 로그인을 생성하려면 `ALTER ANY LOGIN` 권한이 필요합니다. 이 권한으로 로그인을 생성하지 않은 경우 SQL Server 인증을 사용하여 DB 인스턴스의 마스터 사용자로 연결하고 마스터 사용자의 맥락에서 AD SQL Server 로그인을 생성합니다.

다음과 같은 데이터 정의 언어(DDL) 명령을 실행하여 AD 사용자 또는 그룹에 대한 SQL Server 로그인을 생성할 수 있습니다.

```
USE [master]
GO
CREATE LOGIN [mydomain\myuser] FROM WINDOWS WITH DEFAULT_DATABASE = [master],
 DEFAULT_LANGUAGE = [us_english];
GO
```

도메인의 사용자(사람 및 애플리케이션)는 이제 Windows 인증을 사용하여 도메인이 조인된 클라이언트 컴퓨터의 RDS Custom for SQL Server 인스턴스에 연결할 수 있습니다.

7단계: Kerberos 또는 NTLM 인증 사용

RDS 엔드포인트를 사용한 NTLM 인증

각 Amazon RDS DB 인스턴스에는 엔드포인트가 있으며, 각 엔드포인트에는 DB 인스턴스의 DNS 이름과 포트 번호가 있습니다. SQL 클라이언트 애플리케이션을 사용해 DB 인스턴스에 연결하려면 DB 인스턴스에 연결할 수 있는 DNS 이름과 포트 번호가 필요합니다. NTLM 인증을 사용하여 인증하려면 RDS 엔드포인트에 연결해야 합니다.

계획된 데이터베이스 유지 관리 또는 예기치 않은 서비스 중단이 발생할 경우 Amazon RDS가 최신 보조 데이터베이스로 자동으로 장애 조치를 수행하므로 수동 개입 없이 작업을 빠르게 재개할 수 있습니다. 기본 인스턴스 및 보조 인스턴스는 동일한 엔드포인트를 사용합니다. 이 엔드포인트의 물리적 네트워크 주소는 장애 조치 프로세스의 일환으로 보조로 전환됩니다. 장애 조치가 발생하는 경우 애플리케이션을 다시 구성할 필요가 없습니다.

Kerberos 인증

RDS Custom for SQL Server에 대한 Kerberos 기반 인증을 수행하려면 특정 서비스 보안 주체 이름(SPN)에 연결해야 합니다. 그러나 장애 조치 이벤트 후 애플리케이션이 새 SPN을 인식하지 못할 수 있습니다. 이를 해결하기 위해 RDS Custom for SQL Server는 Kerberos 기반 엔드포인트를 제공합니다.

Kerberos 기반 엔드포인트는 특정 형식을 따릅니다. RDS 엔드포인트가 *rds-instance-name*.*account-region-hash*.*aws-region*.rds.amazonaws.com인 경우 해당 Kerberos 기반 엔드포인트는 *rds-instance-name*.*account-region-hash*.*aws-region*.awsrds.*fully qualified domain name (FQDN)*입니다.

예를 들어 RDS 엔드포인트가 ad-test.cocv6zwtircu.us-east-1.rds.amazonaws.com이고 도메인 이름이 corp-ad.company.com인 경우 Kerberos 기반 엔드포인트는 ad-test.cocv6zwtircu.us-east-1.awsrds.corp-ad.company.com입니다.

이 Kerberos 기반 엔드포인트는 기본 SQL Server 인스턴스의 새 SPN을 가리키도록 엔드포인트가 자동으로 업데이트되므로 장애 조치 이벤트 후에도 Kerberos를 사용하여 SQL Server 인스턴스로 인증하는 데 사용할 수 있습니다.

CNAME 찾기

CNAME를 찾으려면 도메인 컨트롤러에 연결하고 DNS Manager를 엽니다. 정방향 조회 영역 및 FQDN으로 이동합니다.

awsrds, aws-region, 계정 및 리전별 해시를 탐색합니다.

RDS Custom EC2 인스턴스를 연결하고 CNAME를 사용하여 로컬에서 데이터베이스에 연결하려고 하면 연결에서 Kerberos 대신 NTLM 인증을 사용합니다.

원격 클라이언트에서 CNAME를 연결한 후 NTLM 연결이 반환되면 필요한 포트가 허용 목록에 있는지 확인합니다.

연결에서 Kerberos를 사용 중인지 확인하려면 다음 쿼리를 실행합니다.

```
SELECT net_transport, auth_scheme
    FROM sys.dm_exec_connections
    WHERE session_id = @@SSPID;
```

도메인에서 DB 인스턴스 관리

콘솔, AWS CLI 또는 Amazon RDS API를 사용하여 DB 인스턴스 및 DB 인스턴스와 도메인과의 관계를 관리할 수 있습니다. 예를 들어, DB 인스턴스를 도메인 내로, 도메인 외부로 또는 도메인 간에 이동할 수 있습니다.

예를 들어 Amazon RDS API를 사용하여 다음을 수행할 수 있습니다.

- 실패한 멤버십에 대한 도메인 조인을 다시 시도하려면 ModifyDBInstance API 작업을 사용하고 현재 멤버십의 디렉터리 ID를 지정합니다.
- 멤버십에 대한 IAM 역할 이름을 업데이트하려면 `ModifyDBInstance` API 작업을 사용하고 현재 멤버십의 디렉터리 ID 및 새 IAM 역할을 지정합니다.
- 도메인에서 DB 인스턴스를 제거하려면 `ModifyDBInstance` API 작업을 사용하고 `none`을 도메인 파라미터로 지정합니다.
- 한 도메인에서 다른 도메인으로 DB 인스턴스를 이동하려면 `ModifyDBInstance` API 작업을 사용하여 새 도메인의 도메인 식별자를 도메인 파라미터로 지정합니다.

- 각 DB 인스턴스에 대한 멤버십을 나열하려면 DescribeDBInstances API 작업을 사용합니다.

RDS Custom for SQL Server DB 인스턴스 복원 및 Active Directory 도메인에 추가

DB 스냅샷을 복원하거나 SQL Server DB 인스턴스에 대한 특정 시점 복구(PITR)를 수행한 후 Active Directory 도메인에 추가할 수 있습니다. DB 인스턴스가 복원된 후 5단계: RDS Custom for SQL Server DB 인스턴스 생성 또는 수정에 설명된 프로세스를 사용하여 DB 인스턴스를 AD 도메인에 추가하도록 인스턴스를 수정합니다.

도메인 멤버십 이해

DB 인스턴스를 생성하거나 수정한 경우 해당 인스턴스는 도메인의 구성원이 됩니다. AWS 콘솔은 DB 인스턴스에 대한 도메인 멤버십의 상태를 나타냅니다. DB 인스턴스의 상태는 다음 중 한 가지가 될 수 있습니다.

- joined – 인스턴스가 도메인의 구성원입니다.
- joining – 인스턴스가 도메인 구성원이 되기 위한 과정을 진행하고 있습니다.
- pending-join – 인스턴스 멤버십이 보류 중입니다.
- pending-maintenance-join - AWS에서 다음 예약된 유지 관리 기간 동안 인스턴스를 도메인의 멤버로 만들려고 시도합니다.
- pending-removal – 도메인에서 인스턴스 제거 작업이 보류 중입니다.
- pending-maintenance-removal - AWS에서 다음 예약된 유지 관리 기간 동안 도메인에서 인스턴스를 제거하려고 시도합니다.
- failed – 구성 문제가 발생하여 인스턴스가 도메인에 조인되지 않았습니다. 인스턴스 수정 명령을 다시 실행하기 전에 구성을 확인하고 수정합니다.
- removing – 인스턴스를 도메인에서 제거하고 있습니다.

네트워크 연결 문제 또는 잘못된 IAM 역할로 인해 도메인 구성원 되기 요청이 실패할 수 있습니다. 예를 들어, DB 인스턴스를 생성하거나 기존 인스턴스를 수정하여 DB 인스턴스가 도메인의 멤버가 되려는 시도를 못하게 할 수 있습니다. 이 경우 명령을 다시 실행하여 DB 인스턴스를 생성 또는 수정하거나 새로 생성된 인스턴스를 수정하여 도메인에 조인할 수 있습니다.

Active Directory 문제 해결

다음은 AD를 설정하거나 수정할 때 발생할 수 있는 문제입니다.

오류 코드	설명	일반적인 원인	문제 해결 제안
오류 2/0x2	시스템이 지정된 파일을 찾을 수 없습니다.	`–domain-ou` 파라미터로 지정된 조직 단위(OU)의 형식 또는 위치가 잘못되었습니다. AWS Secrets Manager를 통해 지정된 도메인 서비스 계정에는 OU에 가입하는 데 필요한 권한이 없습니다.	`–domain-ou` 파라미터를 검토합니다. 도메인 서비스 계정에 OU에 대한 올바른 권한이 있는지 확인합니다.
오류 5/0x5	액세스가 거부되었습니다.	도메인 서비스 계정에 대한 권한이 잘못 구성되었거나 컴퓨터 계정이 이미 도메인에 있습니다.	도메인의 도메인 서비스 계정 권한을 검토하고 RDS 컴퓨터 계정이 도메인에 중복되지 않았는지 확인합니다. RDS Custom for SQL Server DB 인스턴스에서 `SELECT @@SERVERNAME` 을 실행하여 RDS 컴퓨터 계정의 이름을 확인할 수 있습니다. 다중 AZ를 사용하는 경우 장애 조치를 사용하여 재부팅한 다음 RDS 컴퓨터 계정을 다시 확인합니다. 자세한 내용은 DB 인스턴스 재부팅 단원을 참조하십시오.
오류 87/0x57	파라미터가 올바르지 않습니다.	AWS Secrets Manager를 통해 지정된 도메인 서비스 계정에 올바른 권한이 없습니다. 사용자 프로필도 손상되었을 수 있습니다.	도메인 서비스 계정의 요구 사항을 검토합니다.
오류 234/0xEA	지정된 조직 단위	`–domain-ou` 파라미터로 지정된 OU가 AD에 존재하지 않습니다.	`–domain-ou` 파라미터를 검토하고 지정된 OU가 AD에 있는지 확인합니다.

오류 코드	설명	일반적인 원인	문제 해결 제안
	(OU)가 없습니다.		
오류 1326/0x52E	사용자 이름 또는 암호가 잘못되었습니다.	AWS Secrets Manager에 제공된 도메인 서비스 계정 보안 인증 정보에 알 수 없는 사용자 이름이나 잘못된 암호가 있습니다. AD에서 도메인 계정을 사용하지 않도록 설정할 수도 있습니다.	AWS Secrets Manager에 제공된 보안 인증 정보가 올바르고 Active Directory에서 도메인 계정이 활성화되어 있는지 확인합니다.
오류 1355/0x54B	지정된 도메인이 존재하지 않거나 해당 주소를 찾을 수 없습니다.	도메인이 중지되었거나, 지정된 DNS IP 집합에 연결할 수 없거나, 지정된 FQDN에 연결할 수 없습니다.	`–domain-dns-ips` 및 `–domain-fqdn` 파라미터를 검토하여 올바른지 확인합니다. RDS Custom for SQL Server DB 인스턴스의 네트워킹 구성을 검토하고 AD에 연결할 수 있는지 확인합니다.
오류 1722/0x6BA	RPC 서버를 사용할 수 없습니다.	AD 도메인의 RPC 서비스에 연결하는 중 문제가 발생했습니다. 서비스 또는 네트워크 문제일 수 있습니다.	RPC 서비스가 도메인 컨트롤러에서 실행되고 있고 TCP 포트 135 및 49152-65535 에서 RDS Custom for SQL Server DB 인스턴스의 도메인에 연결할 수 있는지 확인합니다.

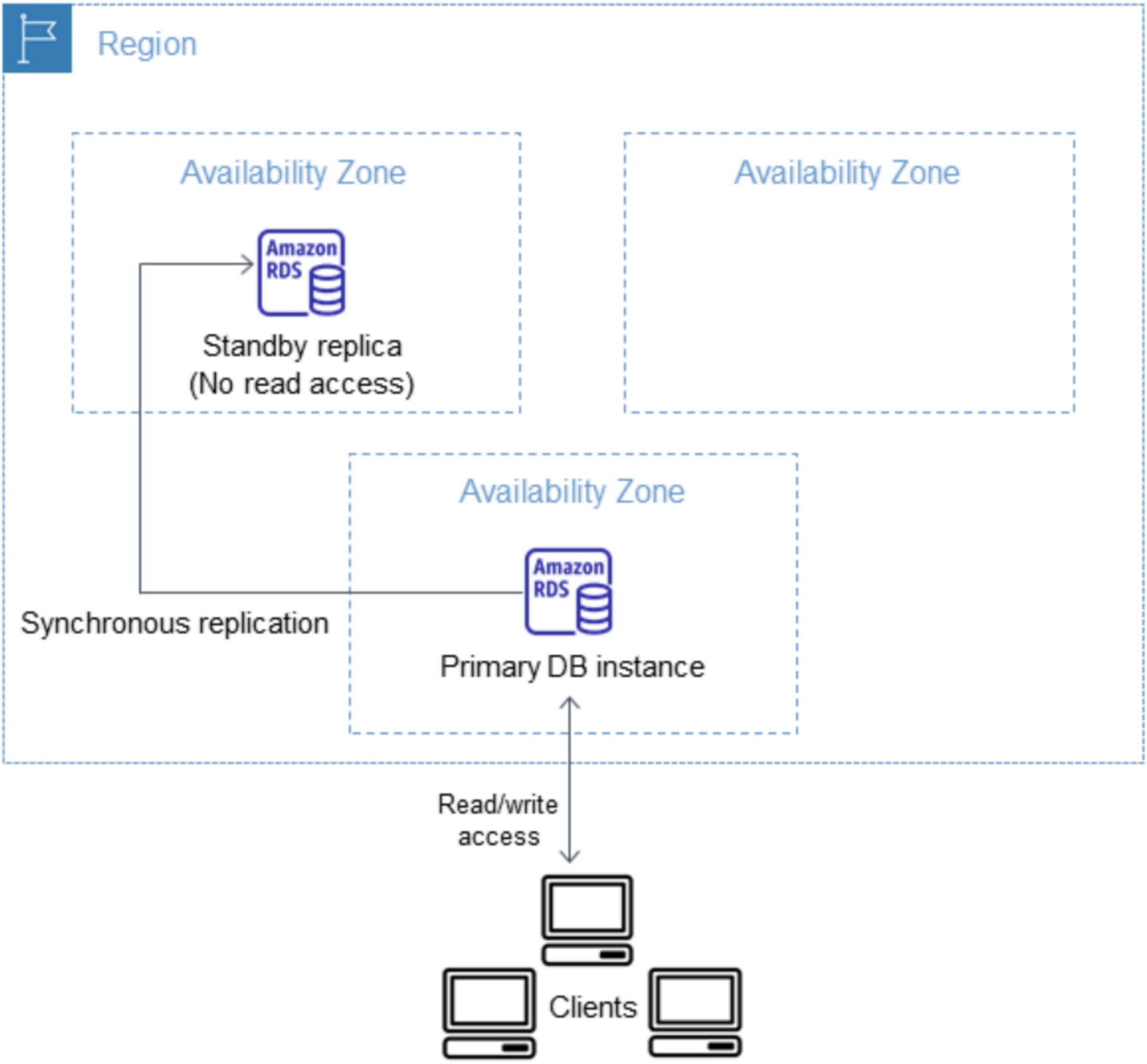
오류 코드	설명	일반적인 원인	문제 해결 제안
오류 2224/0x8B0	계정이 이미 있습니다.	AD에 추가하려는 컴퓨터 계정이 이미 있습니다.	RDS Custom for SQL Server DB 인스턴스에서 `SELECT @@SERVERNAME` 을 실행하여 컴퓨터 계정을 식별한 다음 AD에서 해당 계정을 신중히 제거합니다.
오류 2242/0x8c2	이 사용자의 암호가 만료되었습니다.	AWS Secrets Manager를 통해 지정한 도메인 서비스 계정의 암호가 만료되었습니다.	RDS Custom for SQL Server DB 인스턴스를 AD에 가입시키는 데 사용되는 도메인 서비스 계정의 암호를 업데이트합니다.

RDS Custom for SQL Server에 대한 다중 AZ 배포 구성 및 관리

RDS Custom for SQL Server에 대한 다중 AZ DB 인스턴스 배포에서 Amazon RDS는 자동으로 서로 다른 가용 영역에 동기식 대기 복제본을 프로비저닝하고 유지합니다. 기본 DB 인스턴스는 가용 영역 전체에서 대기 복제본으로 동기식으로 복제되어 데이터 중복성을 제공합니다.

> **Important**
>
> RDS Custom for SQL Server에 대한 다중 AZ 배포는 RDS for SQL Server에 대한 다중 AZ 배포와 다릅니다. RDS for SQL Server에 대한 다중 AZ와 달리, RDS Custom은 사용자의 계정 내에서 실행되는데 이는 권한이 필요하므로 다중 AZ DB 인스턴스를 만들기 전에 RDS Custom for SQL Server의 사전 조건을 설정해야 합니다.
> 사전 조건을 완료하지 않으면 다중 AZ DB 인스턴스가 실행되지 않거나, 단일 AZ DB 인스턴스로 자동으로 되돌아갈 수 있습니다. 사전 조건에 대한 자세한 내용은 RDS Custom for SQL Server를 사용한 다중 AZ 배포의 사전 조건 단원을 참조하세요.

DB 인스턴스를 고가용성으로 실행하면 계획된 시스템 유지 관리 중 가용성을 높일 수 있습니다. 계획된 데이터베이스 유지 관리 또는 예기치 않은 서비스 중단이 발생할 경우, Amazon RDS가 최신 보조 DB 인스턴스로 자동으로 장애 조치를 수행합니다. 이 기능을 통해 수동 개입 없이 데이터베이스 작업을 빠르게 재개할 수 있습니다. 기본 인스턴스 및 예비 인스턴스는 동일한 엔드포인트를 사용합니다. 이 엔드포인트의 물리적 네트워크 주소는 장애 조치 프로세스의 일환으로 보조 복제본으로 전환됩니다. 장애 조치가 발생하는 경우 애플리케이션을 다시 구성할 필요가 없습니다.



RDS Custom DB 인스턴스 생성 시 다중 AZ를 지정하여 RDS Custom for SQL Server 다중 AZ 배포를 생성할 수 있습니다. 콘솔을 사용하여 DB 인스턴스를 수정하고 다중 AZ 옵션을 지정함으로써 기존 RDS Custom for SQL Server DB 인스턴스를 다중 AZ 배포로 변환할 수 있습니다. 또한, AWS CLI 또는 Amazon RDS API를 사용하여 다중 AZ DB 인스턴스 배포를 지정할 수도 있습니다.

RDS 콘솔에 예비 복제본(보조 AZ)의 가용 영역이 표시됩니다. `describe-db-instances` CLI 명령 또는 `DescribeDBInstances` API 작업을 사용하여 보조 AZ를 찾을 수도 있습니다.

다중 AZ DB 인스턴스 배포를 사용하는 RDS Custom for SQL Server DB 인스턴스는 단일 AZ 배포에 비해 쓰기 및 커밋 대기 시간이 길어질 수 있습니다. 이러한 대기 시간 증가는 DB 인스턴스 간의 동기 데이터 복제로 인해 발생할 수 있습니다. AWS는 가용 영역 간 지연 시간이 짧은 네트워크 연결을 제공하도록 설계되었지만 배포가 예비 복제본으로 장애 조치될 경우 지연 시간이 변경될 수 있습니다.

> **Note**
>
> 프로덕션 워크로드의 경우 빠르고 일관된 성능을 제공할 수 있도록 프로비저닝된 IOPS(초당 입/출력 작업)와 함께 DB 인스턴스 클래스를 사용하는 것이 좋습니다. DB 인스턴스 클래스에 대한 자세한 내용은 Amazon RDS Custom for SQL Server 요구 사항 및 제한 섹션을 참조하세요.

주제

- 리전 및 버전 사용 가능 여부
- RDS Custom for SQL Server를 사용한 다중 AZ 배포의 제한 사항
- RDS Custom for SQL Server를 사용한 다중 AZ 배포의 사전 조건
- RDS Custom for SQL Server에 대한 다중 AZ 배포 생성
- RDS Custom for SQL Server 단일 AZ 배포를 다중 AZ 배포로 수정
- RDS Custom for SQL Server 다중 AZ 배포를 단일 AZ 배포로 수정하려면
- RDS Custom for SQL Server 다중 AZ 배포에 대한 장애 조치 프로세스

리전 및 버전 사용 가능 여부

다음과 같은 SQL Server 에디션의 경우 RDS Custom for SQL Server에 대한 다중 AZ 배포가 지원됩니다.

- SQL Server 2022 및 2019: Enterprise, Standard, Web, Developer Edition

> **Note**
>
> RDS Custom for SQL Server의 다중 AZ 배포는 SQL Server 2019 CU8(15.00.4073.23) 이하 버전에서는 지원되지 않습니다.

RDS Custom for SQL Server에 대한 다중 AZ 배포는 RDS Custom for SQL Server를 사용하는 모든 리전에서 사용할 수 있습니다. RDS Custom for SQL Server에 대한 다중 AZ 배포를 리전에서 사용할 수 있는지에 대한 자세한 내용은 RDS Custom for SQL Server를 지원하는 리전 및 DB 엔진를 참조하세요.

RDS Custom for SQL Server를 사용한 다중 AZ 배포의 제한 사항

RDS Custom for SQL Server를 사용한 다중 AZ 배포에는 다음과 같은 제한 사항이 있습니다.

- 리전 간 다중 AZ 배포는 지원되지 않습니다.
- 보조 DB 인스턴스가 데이터베이스 읽기 작업을 허용하도록 구성할 수 없습니다.
- 다중 AZ 배포와 함께 사용자 지정 엔진 버전(CEV)을 사용할 경우 보조 DB 인스턴스도 동일한 CEV를 사용합니다. 보조 DB 인스턴스는 다른 CEV를 사용할 수 없습니다.

RDS Custom for SQL Server를 사용한 다중 AZ 배포의 사전 조건

기존 RDS Custom for SQL Server 단일 AZ 배포가 있을 경우, 이 배포를 다중 AZ 배포로 수정하기 전에 다음과 같은 추가 사전 조건이 필요합니다. 사전 조건을 수동으로 완료하거나 제공된 CloudFormation 템플릿을 사용하여 완료하도록 선택할 수 있습니다. 최신 CloudFormation 템플릿에는 단일 AZ 배포와 다중 AZ 배포를 위한 사전 조건이 포함되어 있습니다.

> **Important**
>
> 설정을 간소화하려면 네트워크 설정 지침에 제공된 최신 AWS CloudFormation 템플릿 파일을 사용하여 사전 조건을 생성하는 것이 좋습니다. 자세한 내용은 AWS CloudFormation을 사용한 구성 단원을 참조하십시오.

> **Note**
>
> 기존 RDS Custom for SQL Server 단일 AZ 배포를 다중 AZ 배포로 수정하는 경우 이러한 사전 조건을 완료해야 합니다. 사전 조건을 완료하지 않으면 다중 AZ 설정이 실패합니다. 사전 조건을 완료하려면 RDS Custom for SQL Server 단일 AZ 배포를 다중 AZ 배포로 수정의 단계를 따릅니다.

- 포트 1120을 허용하도록 RDS 보안 그룹 인바운드 및 아웃바운드 규칙을 업데이트합니다.

- 프라이빗 네트워크 액세스 제어 목록(ACL)에 DB 인스턴스 VPC의 TCP 포트 `0-65535`를 허용하는 규칙을 추가합니다.
- RDS Custom for SQL Server DB 인스턴스가 SQS와 통신할 수 있도록 새 Amazon SQS VPC 엔드포인트를 생성합니다.
- 인스턴스 프로파일 역할에서 SQS 권한을 업데이트합니다.

RDS Custom for SQL Server에 대한 다중 AZ 배포 생성

RDS Custom for SQL Server 다중 AZ 배포를 생성하려면 Amazon RDS Custom for SQL Server의 DB 인스턴스 생성 및 연결의 단계를 따릅니다.

> **Important**
>
> 설정을 간소화하려면 네트워크 설정 지침에 제공된 최신 AWS CloudFormation 템플릿 파일을 사용하는 것이 좋습니다. 자세한 내용은 AWS CloudFormation을 사용한 구성 단원을 참조하십시오.

다중 AZ 배포 생성을 완료하려면 몇 분 정도 걸립니다.

RDS Custom for SQL Server 단일 AZ 배포를 다중 AZ 배포로 수정

기존 RDS Custom for SQL Server DB 인스턴스를 단일 AZ 배포에서 다중 AZ 배포로 수정할 수 있습니다. DB 인스턴스를 수정할 경우 Amazon RDS는 여러 가지 작업을 수행합니다.

- 기본 DB 인스턴스의 스냅샷을 생성합니다.
- 스냅샷에서 스탠바이 복제본용 새 볼륨을 생성합니다. 이러한 볼륨은 백그라운드에서 초기화되며 데이터가 완전히 초기화된 후에 최대 볼륨 성능이 달성됩니다.
- 기본 DB 인스턴스와 보조 DB 인스턴스 간의 동기 블록 수준 복제를 활성화합니다.

> **Important**
>
> 사용량이 가장 많은 기간에는 프로덕션 DB 인스턴스의 RDS Custom for SQL Server DB 인스턴스를 단일 AZ에서 다중 AZ 배포로 수정하지 않는 것이 좋습니다.

AWS는 스냅샷을 사용하여 대기 인스턴스를 생성하면 단일 AZ에서 다중 AZ로 변환할 때 가동 중지 시간을 피할 수 있지만, 다중 AZ로 변환하는 동안과 이후에 성능에 영향을 미칠 수 있습니다. 이는 쓰기 대기 시간에 민감한 워크로드에 상당한 영향을 미칠 수 있습니다. 이 기능을 사용하면 스냅샷에서 대용량 볼륨을 신속하게 복원할 수 있지만, 이 경우 동기식 복제로 인해 I/O 작업의 지연 시간이 증가할 수 있습니다. 이러한 지연 시간은 데이터베이스 성능에 영향을 줄 수 있습니다.

> **Note**
>
> 2024년 8월 29일 이전에 RDS Custom for SQL Server DB 인스턴스를 생성한 경우 수정하기 전에 최신 마이너 버전으로 패치합니다.
>
> - SQL Server 2019 인스턴스의 경우 DB 엔진 버전을 `15.00.4410.1.v1` 이상으로 업그레이드합니다.
> - SQL Server 2022 인스턴스의 경우 DB 엔진 버전을 `16.00.4150.1.v1` 이상으로 업그레이드합니다.

주제

- CloudFormation을 사용하여 단일 AZ를 다중 AZ 배포로 수정하기 위한 사전 조건 구성
- 단일 AZ를 다중 AZ 배포로 수동으로 수정하기 위한 사전 조건 구성
- RDS 콘솔, AWS CLI 또는 RDS API를 사용하여 수정합니다.

CloudFormation을 사용하여 단일 AZ를 다중 AZ 배포로 수정하기 위한 사전 조건 구성

다중 AZ 배포를 사용하려면 사전 조건이 포함된 최신 CloudFormation 템플릿을 적용하거나, 최신 사전 조건을 수동으로 구성해야 합니다. 최신 CloudFormation 사전 조건 템플릿을 이미 적용한 경우에는 이 단계를 건너뛰어도 됩니다.

CloudForma를 사용하여 RDS Custom for SQL Server 다중 AZ 배포를 구성하려면

1. https://console.aws.amazon.com/cloudformation에서 CloudFormation 콘솔을 엽니다.
2. 스택 생성 마법사를 시작하려면 단일 AZ 배포를 만드는 데 사용했던 기존 스택을 선택하고 Update(업데이트)를 선택합니다.

   스택 업데이트 페이지가 표시됩니다.
3. 사전 조건 - 템플릿 준비에서 현재 템플릿 교체를 선택합니다.
4. 템플릿 지정(Specify template)에서 다음 작업을 수행합니다.

a. 최신 AWS CloudFormation 템플릿 파일을 다운로드합니다. custom-sqlserver-onboard.zip 링크의 컨텍스트 메뉴(마우스 오른쪽 버튼 클릭)를 열고 Save Link As(다른 이름으로 링크 저장)를 선택합니다.

b. `custom-sqlserver-onboard.json` 파일을 컴퓨터에 저장하고 압축을 풉니다.

c. 템플릿 소스로 템플릿 파일 업로드를 선택합니다.

d. 파일 선택(Choose file)에서 `custom-sqlserver-onboard.json`을 찾아 선택합니다.

5. 다음을 선택합니다.

   스택 세부 정보 지정(Specify stack details) 페이지가 나타납니다.

6. 기본 옵션을 유지하려면 다음(Next)을 선택합니다.

   고급 옵션 페이지가 나타납니다.

7. 기본 옵션을 유지하려면 다음(Next)을 선택합니다.

8. 기본 옵션을 유지하려면 다음(Next)을 선택합니다.

9. 변경 사항 검토 페이지에서 다음을 수행합니다.

   a. 기능에서 이 사용자 지정 이름을 사용하여 AWS CloudFormation이 IAM 리소스를 생성할 수 있음에 동의합니다 확인란을 선택합니다.

   b. 제출을 선택합니다.

10. 업데이트가 성공했는지 확인합니다. 작업이 성공하면 `UPDATE_COMPLETE`가 표시됩니다.

업데이트가 실패하면 업데이트 프로세스에 지정된 새 구성이 모두 롤백됩니다. 기존 리소스는 계속 사용할 수 있습니다. 예를 들어 번호가 18과 19인 네트워크 ACL 규칙을 추가했는데 같은 번호로 된 기존 규칙이 있는 경우, 업데이트 시 `Resource handler returned message: "The network acl entry identified by 18 already exists.`라는 오류가 반환됩니다. 이러한 시나리오가 발생하면 18보다 작은 숫자를 사용하도록 기존 ACL 규칙을 수정한 후 업데이트를 다시 시도해볼 수 있습니다.

단일 AZ를 다중 AZ 배포로 수동으로 수정하기 위한 사전 조건 구성

> **Important**
>
> 설정을 간소화하려면 네트워크 설정 지침에 제공된 최신 AWS CloudFormation 템플릿 파일을 사용하는 것이 좋습니다. 자세한 내용은 CloudFormation을 사용하여 단일 AZ를 다중 AZ 배포로 수정하기 위한 사전 조건 구성 섹션을 참조하세요.

사전 조건을 수동으로 구성하도록 선택한 경우 다음 태스크를 수행합니다.

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. Endpoint(엔드포인트)를 선택합니다. 엔드포인트 생성페이지가 나타납니다.
3. 서비스 범주에서 AWS 서비스를 선택합니다.
4. 서비스에서 *SQS#* 검색합니다.
5. VPC에서 RDS Custom for SQL Server DB 인스턴스가 배포된 VPC를 선택합니다.
6. 서브넷에서 RDS Custom for SQL Server DB 인스턴스가 배포된 서브넷을 선택합니다.
7. 보안 그룹에서 *-vpc-endpoint-sg* 그룹을 선택합니다.
8. 정책에서 사용자 지정을 선택합니다.
9. 사용자 지정 정책에서 *AWS ###*, *##*, *accountId*, *IAM-Instance-role*을 고유한 값으로 바꿉니다.

JSON

```
                {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Condition": {
                    "StringLike": {
                        "aws:ResourceTag/AWSRDSCustom": "custom-sqlserver"
                    }
                },
                "Action": [
                    "SQS:SendMessage",
                    "SQS:ReceiveMessage",
                    "SQS:DeleteMessage",
                    "SQS:GetQueueUrl"
                ],
                "Resource": "arn:${AWS::Partition}:sqs:${AWS::Region}:
${AWS::AccountId}:do-not-delete-rds-custom-*",
                "Effect": "Allow",
                "Principal": {
                    "AWS": "arn:${AWS::Partition}:iam::
${AWS::AccountId}:role/{IAM-Instance-role}"
                }
            }
```

```
        ]
    }
```

10. Amazon SQS에 액세스할 수 있는 권한으로 인스턴스 프로파일을 업데이트합니다. *AWS ###*, *##*, *accountId*를 고유한 값으로 바꿉니다.

```
                        {
    "Sid": "SendMessageToSQSQueue",
    "Effect": "Allow",
    "Action": [
      "SQS:SendMessage",
      "SQS:ReceiveMessage",
      "SQS:DeleteMessage",
      "SQS:GetQueueUrl"

    ],
    "Resource": [
      {
        "Fn::Sub": "arn:${AWS::Partition}:sqs:${AWS::Region}:${AWS::AccountId}:do-
not-delete-rds-custom-*"
      }
    ],
    "Condition": {
      "StringLike": {
        "aws:ResourceTag/AWSRDSCustom": "custom-sqlserver"
      }
    }
  }
                        >
```

11. 포트 1120을 허용하도록 Amazon RDS 보안 그룹 인바운드 및 아웃바운드 규칙을 업데이트합니다.

    a. 보안 그룹에서 *-rds-custom-instance-sg* 그룹을 선택합니다.

    b. 인바운드 규칙의 경우 소스 *-rds-custom-instance-sg* 그룹의 포트 *1120*을 허용하는 사용자 지정 TCP 규칙을 만듭니다.

    c. 아웃바운드 규칙의 경우 대상 *-rds-custom-instance-sg* 그룹의 포트 *1120*을 허용하는 사용자 지정 TCP 규칙을 만듭니다.

12. 프라이빗 네트워크 액세스 제어 목록(ACL)에 DB 인스턴스 소스 서브넷의 TCP 포트 `0-65535`를 허용하는 규칙을 추가합니다.

> **Note**
>
> 인바운드 규칙 및 아웃바운드 규칙을 생성할 때는 기존의 가장 높은 규칙 번호를 기록해 둡니다. 새로 생성하는 규칙의 규칙 번호 100보다 작아야 하며 기존 규칙 번호와 일치하지 않아야 합니다.

    a. 네트워크 ACL에서 *`-private-network-acl`* 그룹을 선택합니다.
    b. 인바운드 규칙의 경우 *`privatesubnet1`* 및 *`privatesubnet2`*에서 나오는 소스와 함께 TCP 포트 `0-65535`를 허용하는 모든 TCP 규칙을 생성합니다.
    c. 아웃바운드 규칙의 경우 대상인 *`privatesubnet1`* 및 *`privatesubnet2`*로 향하는 TCP 포트 `0-65535`를 허용하는 모든 TCP 규칙을 생성합니다.

RDS 콘솔, AWS CLI 또는 RDS API를 사용하여 수정합니다.

사전 조건을 완료한 후에는 RDS 콘솔, AWS CLI 또는 RDS API를 사용하여 RDS Custom for SQL Server DB 인스턴스를 단일 AZ에서 다중 AZ 배포로 수정할 수 있습니다.

콘솔

RDS Custom for SQL Server 단일 AZ 배포를 다중 AZ 배포로 수정하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 콘솔에서 데이터베이스를 선택합니다.

   데이터베이스 창이 표시됩니다.
3. 수정하려는 RDS Custom for SQL Server DB 인스턴스를 선택합니다.
4. 작업에서 다중 AZ 배포로 변환을 선택합니다.
5. 확인 페이지에서 즉시 적용을 선택하여 변경 사항을 즉시 적용합니다. 이 옵션을 선택하면 다운타임이 발생하지 않지만 성능이 영향을 받을 수 있습니다. 다음 유지 관리 기간에 업데이트를 적용하도록 선택할 수도 있습니다. 자세한 내용은 수정 예약 설정 사용 섹션을 참조하세요.
6. 확인 페이지에서 다중 AZ로 변환을 선택합니다.

AWS CLI

AWS CLI를 사용하여 다중 AZ DB 인스턴스 배포로 변환하려면 modify-db-instance 명령을 호출하고 `--multi-az` 옵션을 설정합니다. DB 인스턴스 식별자와 수정하려는 기타 옵션 값을 지정합니다. 각 옵션에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하세요.

Example

다음 코드는 `--multi-az` 옵션을 포함하여 `mycustomdbinstance`를 수정합니다. 변경 사항은 `--no-apply-immediately`를 사용하여 다음 유지 관리 기간에 적용됩니다. 변경 사항을 바로 적용하려면 `--apply-immediately`를 사용합니다. 자세한 내용은 수정 예약 설정 사용 섹션을 참조하세요.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mycustomdbinstance \
    --multi-az \
    --no-apply-immediately
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mycustomdbinstance ^
    --multi-az  \ ^
    --no-apply-immediately
```

RDS API

RDS API를 사용하여 다중 AZ DB 인스턴스 배포로 변환하려면 ModifyDBInstance 작업을 호출하고 `MultiAZ` 파라미터를 true로 설정합니다.

RDS Custom for SQL Server 다중 AZ 배포를 단일 AZ 배포로 수정하려면

기존 RDS Custom for SQL Server DB 인스턴스를 다중 AZ 배포에서 단일 AZ 배포로 수정할 수 있습니다.

콘솔

RDS Custom for SQL Server DB 인스턴스를 다중 AZ 배포에서 단일 AZ 배포로 수정할 수 있습니다.

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.

2. Amazon RDS 콘솔에서 데이터베이스를 선택합니다.

   데이터베이스 창이 표시됩니다.

3. 수정하려는 RDS Custom for SQL Server DB 인스턴스를 선택합니다.
4. 다중 AZ 배포의 경우 No(아니요)를 선택합니다.
5. 확인 페이지에서 즉시 적용을 선택하여 변경 사항을 즉시 적용합니다. 이 옵션을 선택하면 다운타임이 발생하지 않지만 성능이 영향을 받을 수 있습니다. 다음 유지 관리 기간에 업데이트를 적용하도록 선택할 수도 있습니다. 자세한 내용은 수정 예약 설정 사용 단원을 참조하십시오.
6. 확인 페이지에서 DB 인스턴스 수정을 선택합니다.

AWS CLI

AWS CLI를 사용하여 다중 AZ 배포를 단일 AZ 배포로 수정하려면 modify-db-instance 명령을 호출하고 `--no-multi-az` 옵션을 포함합니다. DB 인스턴스 식별자와 수정하려는 기타 옵션 값을 지정합니다. 각 옵션에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하세요.

Example

다음 코드는 `--no-multi-az` 옵션을 포함하여 `mycustomdbinstance`를 수정합니다. 변경 사항은 `--no-apply-immediately`를 사용하여 다음 유지 관리 기간에 적용됩니다. 변경 사항을 바로 적용하려면 `--apply-immediately`를 사용합니다. 자세한 내용은 수정 예약 설정 사용 단원을 참조하십시오.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mycustomdbinstance \
    --no-multi-az  \
    --no-apply-immediately
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mycustomdbinstance ^
    --no-multi-az \ ^
    --no-apply-immediately
```

RDS API

RDS API를 사용하여 다중 AZ 배포를 단일 AZ 배포로 수정하려면 modify-db-instance 작업을 호출하고 `MultiAZ` 파라미터를 `false`로 설정합니다.

RDS Custom for SQL Server 다중 AZ 배포에 대한 장애 조치 프로세스

계획되거나 계획되지 않은 DB 인스턴스의 운영 중단으로 인해 인프라 장애가 발생한 경우, 다중 AZ를 설정하면 Amazon RDS는 자동으로 다른 가용 영역의 대기 복제본으로 전환됩니다. 장애 조치가 완료되는 데 소요되는 시간은 프라이머리 DB 인스턴스를 사용할 수 없게 된 시점의 데이터베이스 활동 및 기타 조건에 따라 달라집니다. 장애 조치에 소요되는 시간은 일반적으로 60~120초입니다. 그러나 트랜잭션의 규모가 크거나 복구 프로세스가 복잡한 경우 장애 조치에 소요되는 시간이 증가할 수 있습니다. 장애 조치가 완료되면 RDS 콘솔에 새 가용 영역이 표시되는 데 시간이 더 걸릴 수 있습니다.

> **Note**
>
> 장애 조치를 사용하면 DB 인스턴스를 재부팅할 때 장애 조치를 수동으로 강제 적용할 수 있습니다. DB 인스턴스 재부팅에 대한 자세한 내용은 DB 인스턴스 재부팅을 참조하세요.

Amazon RDS는 자동으로 장애 조치를 취하여 관리자의 개입 없이 데이터베이스 작업을 신속하게 재개할 수 있도록 합니다. 다음 표에 설명된 조건 중 하나가 발생하면 기본 DB 인스턴스는 자동으로 예비 복제본으로 전환됩니다. 이 장애 조치 이유는 RDS 이벤트 로그에서 확인할 수 있습니다.

장애 조치 이유	설명
`The operating system for the RDS Custom for SQL Server Multi-AZ DB instance is being patched in an offline operation`	OS 패치 또는 보안 업데이트를 위한 유지 관리 기간 동안 장애 조치가 트리거되었습니다. 자세한 내용은 DB 인스턴스 유지 관리 단원을 참조하십시오.
`The primary host of the RDS Custom for SQL Server Multi-`	다중 AZ DB 인스턴스 배포에서 손상된 프라이머리 DB 인스턴스를 감지하여 장애 조치를 수행했습니다.

장애 조치 이유	설명
`AZ DB instance is unhealthy.`	
`The primary host of the RDS Custom for SQL Server Multi-AZ DB instance is unreachable due to loss of network connectivity.`	RDS 모니터링이 기본 DB 인스턴스에 대한 네트워크 연결 실패를 감지하여 장애 조치를 트리거했습니다.
`The RDS Custom for SQL Server Multi-AZ DB instance was modified by the customer.`	DB 인스턴스 수정 때문에 장애 조치가 트리거되었습니다. 자세한 내용은 RDS Custom for SQL Server DB 인스턴스 수정 단원을 참조하십시오.
`The storage volume of the primary host of the RDS Custom for SQL Server Multi-AZ DB instance experienced a failure.`	다중 AZ DB 인스턴스 배포가 프라이머리 DB 인스턴스에서 스토리지 문제를 감지하여 장애 조치를 수행했습니다.
`The user requested a failover of the RDS Custom for SQL Server Multi-AZ DB instance.`	RDS Custom for SQL Server 다중 AZ DB 인스턴스가 장애 조치로 인해 재부팅되었습니다. 자세한 내용은 DB 인스턴스 재부팅 단원을 참조하십시오.

장애 조치 이유	설명
`The RDS Custom for SQL Server Multi-AZ primary DB instance is busy or unresponsive.`	기본 DB 인스턴스가 응답하지 않습니다. 다음 단계를 시도하는 것이 좋습니다. • 이벤트 로그 및 CloudWatch 로그에서 과도한 CPU, 메모리 또는 스왑 공간 사용량을 검사합니다. 자세한 내용은 Amazon RDS 이벤트 알림 작업 단원을 참조하십시오. • Amazon RDS 이벤트에서 트리거되는 규칙을 생성합니다. 자세한 내용은 Amazon RDS 이벤트에서 트리거되는 규칙 생성 단원을 참조하십시오. • 워크로드를 평가하여 적절한 DB 인스턴스 클래스를 사용 중인지 확인합니다. 자세한 내용은 DB 인스턴스 클래스 단원을 참조하십시오.

다음 단계에 따라 다중 AZ DB 인스턴스가 장애 조치를 수행했는지 확인할 수 있습니다.

- 장애 조치가 시작되었음을 이메일 또는 SMS로 사용자에게 알리도록 DB 이벤트 구독을 설정합니다. 이벤트에 대한 자세한 내용은 Amazon RDS 이벤트 알림 작업 단원을 참조하세요.
- RDS 콘솔 또는 API 작업을 사용하여 DB 이벤트를 확인합니다.
- RDS 콘솔, CLI 또는 API 작업을 사용하여 RDS Custom for SQL Server 다중 AZ DB 인스턴스 배포의 현재 상태를 확인합니다.

RDS Custom for SQL Server 다중 AZ 배포를 사용하는 애플리케이션의 TTL(Time To Live) 설정

장애 조치 메커니즘은 DB 인스턴스의 Domain Name System(DNS) 레코드가 예비 DB 인스턴스를 가리키도록 자동으로 변경합니다. 그 결과 DB 인스턴스의 기존 연결을 모두 재설정해야 합니다. DNS 캐시 TTL(Time-to-Live) 구성 값이 낮은지 확인하고, 애플리케이션이 오랜 시간 동안 DNS를 캐시하지 않는지 확인합니다. TTL 값이 높으면 장애 조치 후 애플리케이션이 DB 인스턴스에 빠르게 다시 연결되지 않을 수 있습니다.

Amazon RDS Custom for SQL Server DB 인스턴스 백업 및 복원

Amazon RDS와 마찬가지로 RDS Custom은 백업 보존이 실행되면 RDS Custom for SQL Server DB 인스턴스의 자동 백업을 생성하고 저장합니다. 또한, 수동으로 DB 인스턴스를 백업할 수도 있습니다. 자동 백업은 스냅샷 백업과 트랜잭션 로그 백업으로 구성됩니다. 스냅샷 백업은 지정된 백업 기간 동안 DB 인스턴스의 전체 스토리지 볼륨에 대해 수행됩니다. PITR 대상 데이터베이스의 트랜잭션 로그 백업은 정기적으로 수행됩니다. RDS Custom은 지정한 백업 보존 기간에 따라 DB 인스턴스의 자동 백업을 저장합니다. 자동 백업을 사용하여 DB 인스턴스를 백업 보존 기간 내의 특정 시점으로 복구할 수 있습니다.

스냅샷을 수동으로 백업할 수도 있습니다. 언제든지 이러한 스냅샷 백업에서 새 DB 인스턴스를 생성할 수 있습니다. DB 스냅샷 수동 생성에 대한 자세한 내용은 RDS Custom for SQL Server 스냅샷 생성 섹션을 참조하세요.

스냅샷 백업은 운영상 전체 백업으로 제공되지만, 증분 스토리지 사용에 대해서만 요금이 청구됩니다. RDS Custom DB 인스턴스의 첫 번째 스냅샷에는 전체 DB 인스턴스의 데이터가 포함되며, 동일한 데이터베이스의 후속 스냅샷은 증분식이며, 마지막 스냅샷 이후 변경된 데이터만 저장됩니다.

주제

- RDS Custom for SQL Server 스냅샷 생성
- RDS Custom for SQL Server DB 스냅샷에서 복원
- RDS Custom for SQL Server 인스턴스를 특정 시점으로 복원
- RDS Custom for SQL Server 스냅샷 삭제
- RDS Custom for SQL Server 자동 백업 삭제

RDS Custom for SQL Server 스냅샷 생성

RDS Custom for SQL Server는 DB 인스턴스의 스토리지 볼륨 스냅샷을 생성하여 개별 데이터베이스뿐만 아니라 전체 DB 인스턴스를 백업합니다. 스냅샷을 생성할 때 백업할 RDS Custom for SQL Server DB 인스턴스를 지정하고, 나중에 복원할 수 있도록 스냅샷에 이름을 지정합니다.

스냅샷을 생성할 때 RDS Custom for SQL Server는 (`D:`) 볼륨에 대한 Amazon EBS 스냅샷을 생성하며, 이는 DB 인스턴스에 연결된 데이터베이스 볼륨입니다. 스냅샷을 특정 DB 인스턴스에 쉽게 연결할 수 있도록 `DBSnapshotIdentifier`, `DbiResourceId` 및 `VolumeType`으로 태그가 지정됩니다.

DB 스냅샷을 생성하면 I/O가 잠시 중단됩니다. 이러한 일시 중단은 DB 인스턴스의 크기와 클래스에 따라 몇 초에서 몇 분까지 지속될 수 있습니다. 스냅샷 생성 시간은 데이터베이스의 총 수와 크기에 따라

달라집니다. 시점 복구(PITR) 작업에 사용할 수 있는 데이터베이스 수에 대한 자세한 내용은 인스턴스 클래스 유형별 PITR 대상 데이터베이스 수 섹션을 참조하세요.

스냅샷에는 전체 스토리지 볼륨이 포함되므로 임시 파일과 같은 파일의 크기도 스냅샷 생성 시간에 영향을 미칩니다. 스냅샷을 생성하는 방법에 대한 자세한 내용은 Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성 섹션을 참조하세요.

콘솔 또는 AWS CLI를 사용하여 RDS Custom for SQL Server 스냅샷을 생성할 수 있습니다.

콘솔

RDS Custom 스냅샷을 생성하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
3. RDS Custom DB 인스턴스 목록에서 스냅샷을 생성할 인스턴스를 선택합니다.
4. 작업에서 스냅샷 만들기를 선택합니다.

   [DB 스냅샷 생성(Take DB Snapshot)] 창이 나타납니다.
5. 스냅샷 이름(Snapshot name)에 스냅샷 이름을 입력합니다.
6. [스냅샷 생성(Take Snapshot)]을 선택합니다.

AWS CLI

create-db-snapshot AWS CLI 명령을 사용하여 RDS Custom DB 인스턴스의 스냅샷을 생성합니다.

다음과 같은 옵션을 지정할 수 있습니다.

- `--db-instance-identifier` - 백업할 RDS Custom DB 인스턴스를 식별합니다.
- `--db-snapshot-identifier` - 나중에 복원할 수 있도록 RDS Custom 스냅샷의 이름을 지정합니다.

이 예제에서는 *`my-custom-instance`*라는 RDS Custom DB 인스턴스에 대해 *`my-custom-snapshot`*이라는 DB 스냅샷을 생성합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-snapshot \
    --db-instance-identifier my-custom-instance \
    --db-snapshot-identifier my-custom-snapshot
```

Windows의 경우:

```
aws rds create-db-snapshot ^
    --db-instance-identifier my-custom-instance ^
    --db-snapshot-identifier my-custom-snapshot
```

RDS Custom for SQL Server DB 스냅샷에서 복원

RDS Custom for SQL Server DB 인스턴스를 복원하는 경우 DB 스냅샷의 이름과 새로운 인스턴스의 이름을 입력합니다. 스냅샷에서 기존 RDS Custom DB 인스턴스로 복원할 수 없습니다. 복원 시 새로운 RDS Custom for SQL Server DB 인스턴스가 생성됩니다.

스냅샷에서 복원하면 스토리지 볼륨이 스냅샷을 생성한 시점으로 복원됩니다. 여기에는 (D:) 볼륨에 있던 모든 데이터베이스와 기타 파일이 포함됩니다.

콘솔

DB 스냅샷에서 RDS Custom DB 인스턴스를 복원하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.
3. 복원 원본으로 사용할 DB 스냅샷을 선택합니다.
4. 작업에서 스냅샷 복원을 선택합니다.
5. DB 인스턴스 복원(Restore DB instance) 페이지의 DB 인스턴스 식별자(DB instance identifier)에 복원된 RDS Custom DB 인스턴스의 이름을 입력합니다.
6. DB 인스턴스 복원을 선택합니다.

AWS CLI

restore-db-instance-from-db-snapshot AWS CLI 명령을 사용하여 RDS Custom DB 스냅샷을 복원합니다.

복원하려는 스냅샷이 프라이빗 DB 인스턴스용인 경우 db-subnet-group-name 및 no-publicly-accessible을 모두 올바르게 지정해야 합니다. 그렇지 않으면 DB 인스턴스의 기본값에 공개적으로 액세스할 수 있게 됩니다. 다음 옵션이 필요합니다.

- db-snapshot-identifier - 복구할 스냅샷을 식별합니다.
- db-instance-identifier - DB 스냅샷에서 생성할 RDS Custom DB 인스턴스의 이름을 지정합니다.
- custom-iam-instance-profile - RDS Custom for Oracle DB 인스턴스의 기본 Amazon EC2 인스턴스와 연결된 인스턴스 프로필을 지정합니다.

다음 코드는 my-custom-instance에 대한 my-custom-snapshot이라는 스냅샷을 복원합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds restore-db-instance-from-db-snapshot \
  --db-snapshot-identifier my-custom-snapshot \
  --db-instance-identifier my-custom-instance \
  --custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance \
  --no-publicly-accessible
```

Windows의 경우:

```
aws rds restore-db-instance-from-db-snapshot ^
  --db-snapshot-identifier my-custom-snapshot ^
  --db-instance-identifier my-custom-instance ^
  --custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance ^
  --no-publicly-accessible
```

RDS Custom for SQL Server 인스턴스를 특정 시점으로 복원

DB 인스턴스를 특정 시점(PITR)으로 복원하여 새로운 DB 인스턴스를 생성할 수 있습니다. PITR을 지원하려면 DB 인스턴스에 백업 보존 기능이 활성화되어 있어야 합니다.

RDS Custom for SQL Server DB 인스턴스의 가장 빠른 복원 가능 시간은 여러 요소에 따라 달라지지만, 일반적으로 현재 시간에서 5분 이내입니다. DB 인스턴스의 최근 복원 가능 시간을 확인하려면 AWS CLIdescribe-db-instances 명령을 사용한 후 DB 인스턴스의 LatestRestorableTime 필드에

반환되는 값을 살펴봅니다. Amazon RDS 콘솔에서 각 DB 인스턴스의 복원 가능한 최신 시간을 보려면 [자동 백업을 선택합니다.

백업 보존 기간 중 어느 특정 시점으로든 복원할 수 있습니다. 각 DB 인스턴스의 복원 가능한 가장 빠른 시간을 보려면 Amazon RDS 콘솔에서 자동 백업을 선택합니다.

PITR에 대한 일반적인 정보는 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 섹션을 참조하세요.

주제

- RDS Custom for SQL Server에 대한 PITR 고려 사항
- 인스턴스 클래스 유형별 PITR 대상 데이터베이스 수
- 데이터베이스를 PITR에 포함되지 않도록 설정
- Amazon S3의 트랜잭션 로그
- AWS Management Console, AWS CLI 또는 RDS API를 사용하여 PITR 복원합니다.

RDS Custom for SQL Server에 대한 PITR 고려 사항

RDS Custom for SQL Server에서 PITR은 다음과 같은 중요한 면에서 Amazon RDS의 PITR과 다릅니다.

- PITR은 DB 인스턴스의 데이터베이스만 복원합니다. C: 드라이브의 운영 체제 또는 파일은 복원되지 않습니다.
- RDS Custom for SQL Server DB 인스턴스의 경우 데이터베이스가 자동으로 백업되며 다음 조건에서만 PITR을 사용할 수 있습니다.
  - 데이터베이스가 온라인 상태입니다.
  - 복구 모델이 FULL로 설정되어 있습니다.
  - 쓰기가 가능합니다.
  - D: 드라이브에 실존 파일이 있습니다.
  - `rds_pitr_blocked_databases` 테이블에 등록되어 있지 않습니다. 자세한 내용은 데이터베이스를 PITR에 포함되지 않도록 설정 단원을 참조하십시오.
- PITR 대상 데이터베이스는 데이터베이스 ID의 순서에 따라 결정됩니다. RDS Custom for SQL Server는 DB 인스턴스당 최대 5,000개의 데이터베이스를 허용합니다. 그러나 RDS Custom for SQL Server DB 인스턴스에 대해 PITR 작업을 통해 복원되는 최대 데이터베이스 수는 인스턴스 클래스 유형에 따라 달라집니다. 자세한 내용은 인스턴스 클래스 유형별 PITR 대상 데이터베이스 수 단원을 참조하십시오.

PITR에 포함되지 않는 다른 데이터베이스는 PITR에 사용되는 자동 스냅샷 백업을 비롯하여 DB 스냅샷에서 복원할 수 있습니다.

- 새로운 데이터베이스를 추가하거나, 데이터베이스 이름을 바꾸거나, PITR 대상 데이터베이스를 복원하면 DB 인스턴스의 스냅샷이 시작됩니다.
- PITR 대상인 최대 데이터베이스 수는 데이터베이스 인스턴스가 규모 조정 컴퓨팅 작업을 거칠 때 대상 인스턴스 클래스 유형에 따라 달라집니다. 인스턴스의 더 많은 데이터베이스가 PITR에 적합하도록 인스턴스를 확장하면 새 스냅샷이 생성됩니다.
- 복원된 데이터베이스의 이름은 소스 DB 인스턴스와 동일합니다. 다른 이름을 지정할 수 없습니다.
- `AWSRDSCustomSQLServerIamRolePolicy`는 다른 AWS 서비스에 대한 액세스가 필요합니다. 자세한 내용은 AWSRDSCustomSQLServerInstanceRole에 액세스 정책 추가 단원을 참조하십시오.
- RDS Custom for SQL Server에는 표준 시간대 변경이 지원되지 않습니다. 운영 체제 또는 DB 인스턴스 시간대를 변경하면 PITR(및 기타 자동화)이 작동하지 않습니다.

인스턴스 클래스 유형별 PITR 대상 데이터베이스 수

다음 표에는 인스턴스 클래스 유형에 따라 PITR에 사용할 수 있는 최대 데이터베이스 수가 나와 있습니다.

인스턴스 클래스 유형	PITR에 사용할 수 있는 최대 데이터베이스 수
db.*.large	100
db.*.xlarge to db.*.2xlarge	150
db.*.4xlarge to db.*.8xlarge	300
db.*.12xlarge to db.*.16xlarge	600

인스턴스 클래스 유형	PITR에 사용할 수 있는 최대 데이터베이스 수				
db.*.24xlarge, db.*32xlarge	1000				

* 서로 다른 인스턴스 클래스 유형을 나타냅니다.

DB 인스턴스에서 PITR에 사용할 수 있는 최대 데이터베이스 수는 인스턴스 클래스 유형에 따라 다릅니다. RDS Custom for SQL Server에서 지원하는 최대 인스턴스 클래스 유형의 경우 100개부터 1,000개까지 다양합니다. SQL 서버 시스템 데이터베이스(`master`, `model`, `msdb`, `tempdb`)는 이 제한에 포함되지 않습니다. 대상 인스턴스 클래스 유형에 따라 DB 인스턴스를 확장하거나 축소할 때 RDS Custom은 PITR에 사용할 수 있는 데이터베이스 수를 자동으로 업데이트합니다. RDS Custom for SQL Server는 DB 인스턴스에서 PITR에 사용할 수 있는 최대 데이터베이스 수가 변경될 때 `RDS-EVENT-0352`를 전송합니다. 자세한 내용은 사용자 지정 엔진 버전 이벤트 단원을 참조하십시오.

> **Note**
>
> 100개 이상의 데이터베이스에 대한 PITR 지원은 2023년 8월 26일 이후에 생성된 DB 인스턴스에서만 사용할 수 있습니다. 2023년 8월 26일 이전에 생성된 인스턴스의 경우 PITR에 사용할 수 있는 최대 데이터베이스 수는 인스턴스 클래스에 관계없이 100개입니다. 2023년 8월 26일 이전에 생성된 DB 인스턴스의 데이터베이스 100개 이상에 대해 PITR 지원을 활성화하려면 다음 작업을 수행할 수 있습니다.
>
> - DB 엔진 버전을 15.00.4322.2.v1 이상으로 업그레이드

PITR 작업 중에 RDS Custom은 복원 시 소스 DB 인스턴스에서 PITR에 속했던 모든 데이터베이스를 복원합니다. 대상 DB 인스턴스가 복원 작업을 완료하고 백업 보존을 활성화하면 대상 DB 인스턴스에서 PITR에 사용할 수 있는 최대 데이터베이스 수를 기준으로 DB 인스턴스가 백업을 시작합니다.

예를 들어, DB 인스턴스가 200개의 데이터베이스가 있는 `db.*.xlarge`에서 실행되는 경우는 다음과 같습니다.

1. RDS Custom for SQL Server는 PITR 백업을 위해 데이터베이스 ID를 기준으로 정렬된 처음 150개의 데이터베이스를 선택합니다.

2. db.*.4xlarge까지 확장하도록 인스턴스를 수정합니다.
3. 컴퓨팅 확장 작업이 완료되면 RDS Custom for SQL Server는 데이터베이스 ID를 기준으로 정렬된 처음 300개의 데이터베이스를 PITR 백업용으로 선택합니다. 이제 PITR 요구 사항 조건을 충족하는 200개 데이터베이스 각각이 PITR을 사용할 수 있습니다.
4. 이제 다시 db.*.xlarge로 인스턴스 규모를 축소합니다.
5. 컴퓨팅 축소 작업이 완료되면 RDS Custom for SQL Server는 데이터베이스 ID를 기준으로 정렬된 처음 150개의 데이터베이스를 PITR 백업용으로 다시 선택합니다.

데이터베이스를 PITR에 포함되지 않도록 설정

PITR에서 개별 데이터베이스를 제외하도록 선택할 수 있습니다. 이렇게 하려면 `database_id` 값을 `rds_pitr_blocked_databases` 테이블에 넣으면 됩니다. 다음 SQL 스크립트를 사용하여 테이블을 생성합니다.

rds_pitr_blocked_databases 테이블을 생성하는 방법

- 다음 SQL 스크립트를 실행합니다.

```
create table msdb..rds_pitr_blocked_databases
(
database_id INT NOT NULL,
database_name SYSNAME NOT NULL,
db_entry_updated_date datetime NOT NULL DEFAULT GETDATE(),
db_entry_updated_by SYSNAME NOT NULL DEFAULT CURRENT_USER,
PRIMARY KEY (database_id)
);
```

대상 데이터베이스 및 대상이 아닌 데이터베이스 목록은 Amazon S3 버킷 `do-not-delete-rds-custom-`*`$ACCOUNT_ID`*`-`*`$REGION`*`-`*`unique_identifier`*의 `RDSCustomForSQLServer/Instances/`*`DB_instance_resource_ID`*`/TransactionLogMetadata` 디렉터리에 있는 `RI.End` 파일을 참조하세요. `RI.End` 파일에 대한 자세한 내용은 Amazon S3의 트랜잭션 로그를 참조하세요.

다음 SQL 스크립트를 사용하여 PITR 대상인 데이터베이스 목록을 확인할 수도 있습니다. `@limit` 변수를 인스턴스 클래스의 PITR에 사용할 수 있는 최대 데이터베이스 수로 설정합니다. 자세한 내용은 인스턴스 클래스 유형별 PITR 대상 데이터베이스 수 단원을 참조하십시오.

DB 인스턴스 클래스의 PITR 대상인 데이터베이스 목록을 확인하려면

- 다음 SQL 스크립트를 실행합니다.

```
DECLARE @Limit INT;
SET @Limit = (insert-database-instance-limit-here);

USE msdb;
IF (EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA = 'dbo' AND
 TABLE_NAME = 'rds_pitr_blocked_databases'))
    WITH TABLE0 AS (
        SELECT hdrs.database_id as DatabaseId, sdb.name as DatabaseName,
 'ALWAYS_ON_NOT_WRITABLE_REPLICA' as Reason, NULL as DatabaseNameOnPitrTable
        FROM sys.dm_hadr_database_replica_states hdrs
        INNER JOIN sys.databases sdb ON sdb.database_id = hdrs.database_id
        WHERE (hdrs.is_local = 1 AND hdrs.is_primary_replica = 0)
        OR (sys.fn_hadr_is_primary_replica (sdb.name) = 1 AND DATABASEPROPERTYEX
 (sdb.name, 'Updateability') = 'READ_ONLY')
    ),
    TABLE1 as (
            SELECT dbs.database_id as DatabaseId, sysdbs.name as DatabaseName,
 'OPTOUT' as Reason,
            CASE WHEN dbs.database_name = sysdbs.name THEN NULL ELSE
 dbs.database_name END AS DatabaseNameOnPitrTable
            FROM msdb.dbo.rds_pitr_blocked_databases dbs
            INNER JOIN sys.databases sysdbs ON dbs.database_id = sysdbs.database_id
            WHERE sysdbs.database_id > 4
            ),
    TABLE2 as (
            SELECT
            db.name AS DatabaseName,
            db.create_date AS CreateDate,
            db.state_desc AS DatabaseState,
            db.database_id AS DatabaseId,
            rs.database_guid AS DatabaseGuid,
            rs.last_log_backup_lsn AS LastLogBackupLSN,
            rs.recovery_fork_guid RecoveryForkGuid,
            rs.first_recovery_fork_guid AS FirstRecoveryForkGuid,
            db.recovery_model_desc AS RecoveryModel,
            db.is_auto_close_on AS IsAutoClose,
            db.is_read_only as IsReadOnly,
            NEWID() as FileName,
            CASE WHEN(db.state_desc = 'ONLINE'
```

```
                    AND db.recovery_model_desc != 'SIMPLE'
                    AND((db.is_auto_close_on = 0 and db.collation_name IS NOT NULL)
 OR db.is_auto_close_on = 1))
                    AND db.is_read_only != 1
                    AND db.user_access = 0
                    AND db.source_database_id IS NULL
                    AND db.is_in_standby != 1
                    THEN 1 ELSE 0 END AS IsPartOfSnapshot,
            CASE WHEN db.source_database_id IS NULL THEN 0 ELSE 1 END AS
 IsDatabaseSnapshot
            FROM sys.databases db
            INNER JOIN sys.database_recovery_status rs
            ON db.database_id = rs.database_id
            WHERE DB_NAME(db.database_id) NOT IN('tempdb') AND
            db.database_id NOT IN (SELECT DISTINCT DatabaseId FROM TABLE1) AND
            db.database_id NOT IN (SELECT DISTINCT DatabaseId FROM TABLE0)
        ),
        TABLE3 as(
            Select @Limit+count(DatabaseName) as TotalNumberOfDatabases from TABLE2
 where TABLE2.IsPartOfSnapshot=1 and DatabaseName in ('master','model','msdb')
        )
        SELECT TOP(SELECT TotalNumberOfDatabases from TABLE3)
 DatabaseName,CreateDate,DatabaseState,DatabaseId from TABLE2 where
 TABLE2.IsPartOfSnapshot=1
        ORDER BY TABLE2.DatabaseID ASC
ELSE
    WITH TABLE0 AS (
        SELECT hdrs.database_id as DatabaseId, sdb.name as DatabaseName,
 'ALWAYS_ON_NOT_WRITABLE_REPLICA' as Reason, NULL as DatabaseNameOnPitrTable
        FROM sys.dm_hadr_database_replica_states hdrs
        INNER JOIN sys.databases sdb ON sdb.database_id = hdrs.database_id
        WHERE (hdrs.is_local = 1 AND hdrs.is_primary_replica = 0)
        OR (sys.fn_hadr_is_primary_replica (sdb.name) = 1 AND DATABASEPROPERTYEX
 (sdb.name, 'Updateability') = 'READ_ONLY')
    ),
    TABLE1 as (
            SELECT
            db.name AS DatabaseName,
            db.create_date AS CreateDate,
            db.state_desc AS DatabaseState,
            db.database_id AS DatabaseId,
            rs.database_guid AS DatabaseGuid,
            rs.last_log_backup_lsn AS LastLogBackupLSN,
            rs.recovery_fork_guid RecoveryForkGuid,
```

```
            rs.first_recovery_fork_guid AS FirstRecoveryForkGuid,
            db.recovery_model_desc AS RecoveryModel,
            db.is_auto_close_on AS IsAutoClose,
            db.is_read_only as IsReadOnly,
            NEWID() as FileName,
            CASE WHEN(db.state_desc = 'ONLINE'
                    AND db.recovery_model_desc != 'SIMPLE'
                    AND((db.is_auto_close_on = 0 and db.collation_name IS NOT NULL)
OR db.is_auto_close_on = 1))
                    AND db.is_read_only != 1
                    AND db.user_access = 0
                    AND db.source_database_id IS NULL
                    AND db.is_in_standby != 1
                    THEN 1 ELSE 0 END AS IsPartOfSnapshot,
            CASE WHEN db.source_database_id IS NULL THEN 0 ELSE 1 END AS
IsDatabaseSnapshot
            FROM sys.databases db
            INNER JOIN sys.database_recovery_status rs
            ON db.database_id = rs.database_id
            WHERE DB_NAME(db.database_id) NOT IN('tempdb') AND
            db.database_id NOT IN (SELECT DISTINCT DatabaseId FROM TABLE0)
        ),
        TABLE2 as(
            SELECT @Limit+count(DatabaseName) as TotalNumberOfDatabases from TABLE1
where TABLE1.IsPartOfSnapshot=1 and DatabaseName in ('master','model','msdb')
        )
        select top(select TotalNumberOfDatabases from TABLE2)
DatabaseName,CreateDate,DatabaseState,DatabaseId from TABLE1 where
TABLE1.IsPartOfSnapshot=1
        ORDER BY TABLE1.DatabaseID ASC
```

> **Note**
>
> 심볼 링크만 있는 데이터베이스도 PITR 작업 대상인 데이터베이스에서 제외됩니다. 위 쿼리는 이 기준에 따라 필터링되지 않습니다.

Amazon S3의 트랜잭션 로그

백업 보존 기간에 따라 RDS Custom for SQL Server DB 인스턴스에 대한 트랜잭션 로그가 자동으로 추출되어 Amazon S3에 업로드되는지 여부가 결정됩니다. 0이 아닌 값은 자동 백업이 생성되고 RDS Custom 에이전트가 트랜잭션 로그를 5분마다 S3에 업로드함을 의미합니다.

S3의 트랜잭션 로그 파일은 DB 인스턴스를 생성할 때 입력한 AWS KMS key를 사용하여 유후 상태로 암호화됩니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 서버 측 암호화를 사용하여 데이터 보호를 참조하세요.

각 데이터베이스의 트랜잭션 로그는 `do-not-delete-rds-custom-`*`$ACCOUNT_ID`*`-`*`$REGION`*`-`*`unique_identifier`*라는 이름이 지정된 S3 버킷에 업로드됩니다. S3 버킷의 `RDSCustomForSQLServer/Instances/`*`DB_instance_resource_ID`* 디렉터리에는 다음 두 개의 하위 디렉터리가 있습니다.

- `TransactionLogs` - 각 데이터베이스 및 해당 메타데이터에 대한 트랜잭션 로그를 포함합니다.

  트랜잭션 로그 파일 이름은 *`yyyyMMddHHmm`*`.`*`database_id`*`.`*`timestamp`*와 같은 패턴을 따릅니다.

  ```
  202110202230.11.1634769287
  ```

  `_metadata`라는 접미사가 있는 동일한 파일 이름에는 로그 시퀀스 번호, 데이터베이스 이름 및 `RdsChunkCount`와 같은 트랜잭션 로그에 대한 정보가 포함되어 있습니다. `RdsChunkCount`는 단일 트랜잭션 로그 파일을 나타내는 실제 파일 수를 결정합니다. 트랜잭션 로그 파일의 물리적 청크를 의미하는 접미사 `_0001`, `_0002` 등이 있는 파일이 표시될 수 있습니다. 청크된 트랜잭션 로그 파일을 사용하려면 청크를 다운로드한 후 병합해야 합니다.

  다음과 같은 파일이 있는 경우를 생각해 봅니다.

  - `202110202230.11.1634769287`
  - `202110202230.11.1634769287_0001`
  - `202110202230.11.1634769287_0002`
  - `202110202230.11.1634769287_metadata`

  `RdsChunkCount`은 3입니다. 파일 병합 순서는 `202110202230.11.1634769287`, `202110202230.11.1634769287_0001`, `202110202230.11.1634769287_0002`입니다.
- `TransactionLogMetadata` - 트랜잭션 로그 추출의 각 반복에 대한 메타데이터 정보를 포함합니다.

RI.End 파일에는 트랜잭션 로그가 추출된 모든 데이터베이스 및 존재하지만 트랜잭션 로그가 추출되지 않은 모든 데이터베이스에 대한 정보가 들어 있습니다. 예를 들어, RI.End 파일 이름은 *yyyyMMddHHmm*.RI.End.*timestamp* 패턴을 따릅니다.

```
202110202230.RI.End.1634769281
```

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 PITR 복원합니다.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 RDS Custom for SQL Server DB 인스턴스를 특정 시점으로 복원할 수 있습니다.

콘솔

RDS Custom DB 인스턴스를 지정된 시간으로 복원하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 자동 백업(Automated backups)을 선택합니다.
3. 복원하려는 RDS Custom DB 인스턴스를 선택합니다.
4. 작업에서 특정 시점으로 복구를 선택합니다.

   특정 시점으로 복구 창이 나타납니다.
5. 최근 복원 가능 시간을 선택하여 가능한 최근 시간으로 복원하거나, 사용자 지정을 선택하여 시간을 선택합니다.

   사용자 지정(Custom)을 선택한 경우 인스턴스를 복원하려는 날짜와 시간을 입력합니다.

   시간은 현지 시간대로 표시됩니다. 즉, 협정 세계시(UTC)에서 오프셋으로 표시됩니다. 예를 들어 UTC-5는 동부 표준시/하절기 중부 표준시입니다.
6. DB 인스턴스 식별자(DB instance identifier)에는 복원된 대상 RDS Custom DB 인스턴스의 이름을 입력합니다. 이름은 고유해야 합니다.
7. 필요에 따라 DB 인스턴스 클래스와 같은 기타 옵션을 선택합니다.
8. 특정 시점으로 복구를 선택합니다.

AWS CLI

새로운 RDS Custom DB 인스턴스를 생성하려면 restore-db-instance-to-point-in-time AWS CLI 명령을 사용하여 DB 인스턴스를 지정된 시간으로 복원합니다.

다음 옵션 중 하나를 사용하여 복원 원본으로 사용할 백업을 지정합니다.

- `--source-db-instance-identifier` *mysourcedbinstance*
- `--source-dbi-resource-id` *dbinstanceresourceID*
- `--source-db-instance-automated-backups-arn` *backupARN*

`custom-iam-instance-profile` 옵션은 필수입니다.

다음 예제는 지정된 시간을 기준으로 `my-custom-db-instance`를 `my-restored-custom-db-instance`라는 새로운 DB 인스턴스로 복원합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds restore-db-instance-to-point-in-time \
    --source-db-instance-identifier my-custom-db-instance\
    --target-db-instance-identifier my-restored-custom-db-instance \
    --custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance \
    --restore-time 2022-10-14T23:45:00.000Z
```

Windows의 경우:

```
aws rds restore-db-instance-to-point-in-time ^
    --source-db-instance-identifier my-custom-db-instance ^
    --target-db-instance-identifier my-restored-custom-db-instance ^
    --custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance ^
    --restore-time 2022-10-14T23:45:00.000Z
```

RDS Custom for SQL Server 스냅샷 삭제

RDS Custom for SQL Server에서 관리하는 DB 스냅샷이 더 이상 필요하지 않으면 삭제할 수 있습니다. Amazon RDS와 RDS Custom DB 인스턴스의 삭제 절차는 동일합니다.

바이너리 및 루트 볼륨에 대한 Amazon EBS 스냅샷은 계정에서 실행 중인 일부 인스턴스 또는 다른 RDS Custom for SQL Server 스냅샷에 연결될 수 있어 계정에 더 오래 남아 있습니다. 이러한 EBS 스

냅샷은 기존 RDS Custom for SQL Server 리소스(DB 인스턴스 또는 백업)와 더 이상 관련이 없는 경우 자동으로 삭제됩니다.

콘솔

RDS Custom DB 인스턴스의 스냅샷을 삭제하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.
3. 삭제하고 싶은 DB 스냅샷을 선택합니다.
4. 작업(Actions)에서 스냅샷 삭제(Delete snapshot)를 선택합니다.
5. 확인 페이지에서 삭제를 선택합니다.

AWS CLI

RDS Custom 스냅샷을 삭제하려면 AWS CLI 명령 delete-db-snapshot을 사용합니다.

다음 옵션이 필요합니다.

- `--db-snapshot-identifier` - 삭제할 스냅샷입니다.

다음 예제에서는 `my-custom-snapshot` DB 스냅샷을 삭제합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds delete-db-snapshot \
  --db-snapshot-identifier my-custom-snapshot
```

Windows의 경우:

```
aws rds delete-db-snapshot ^
  --db-snapshot-identifier my-custom-snapshot
```

RDS Custom for SQL Server 자동 백업 삭제

보관된 RDS Custom for SQL Server 자동 백업이 더 이상 필요하지 않으면 삭제할 수 있습니다. 절차는 Amazon RDS 백업을 삭제하는 절차와 동일합니다.

콘솔

보관된 자동 백업을 삭제하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 자동 백업(Automated backups)을 선택합니다.
3. Retained(보관됨)를 선택합니다.
4. 삭제하려는 보관된 자동 백업을 선택합니다.
5. [Actions]에 대해 [Delete]를 선택합니다.
6. 확인 페이지에서 **`delete me`**를 입력하고 삭제를 선택합니다.

AWS CLI

delete-db-instance-automated-backup이라는 AWS CLI 명령을 사용하여 보관된 자동 백업을 삭제할 수 있습니다.

다음 옵션을 사용하여 보관된 자동 백업을 삭제할 수 있습니다.

- `--dbi-resource-id` - 소스 RDS Custom DB 인스턴스의 리소스 식별자입니다.

  보관된 자동 백업의 소스 DB 인스턴스에 대한 리소스 식별자는 AWS CLI 명령 describe-db-instance-automated-backups를 사용하여 찾을 수 있습니다.

다음 예시에서는 소스 DB 인스턴스 리소스 식별자가 `custom-db-123ABCEXAMPLE`인 보관된 자동 백업을 삭제합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds delete-db-instance-automated-backup \
    --dbi-resource-id custom-db-123ABCEXAMPLE
```

Windows의 경우:

```
aws rds delete-db-instance-automated-backup ^
    --dbi-resource-id custom-db-123ABCEXAMPLE
```

Amazon RDS Custom for SQL Server DB 스냅샷 복사

RDS Custom for SQL Server에서는 자동 백업 및 수동으로 DB 스냅샷을 복사할 수 있습니다. 스냅샷을 복사한 후 생성한 복사본은 수동 스냅샷입니다. 자동 백업 또는 수동 스냅샷의 복사본을 여러 개 만들 수 있지만, 각 복사본에는 고유한 식별자가 있어야 합니다.

RDS Custom for SQL Server를 사용하는 다른 AWS 리전에서 동일한 AWS 계정 내의 스냅샷만 복사할 수 있습니다. 다음 연산자는 현재 지원되지 않습니다.

- 동일한 AWS 리전에서 DB 스냅샷을 복사합니다.
- AWS 계정 간에 DB 스냅샷을 복사합니다.

RDS Custom for SQL Server는 증분 스냅샷 복사를 지원합니다. 자세한 내용은 증분 스냅샷 복사 시 고려 사항 섹션을 참조하세요.

주제

- 제한 사항
- 암호화 처리
- 리전 간 복사
- 사용자 지정 엔진 버전(CEV)으로 만든 DB 인스턴스의 스냅샷
- IAM 보안 주체에 필요한 권한 부여
- DB 스냅샷 복사

제한 사항

RDS Custom for SQL Server용 DB 스냅샷을 복사하는 데는 다음과 같은 제한이 적용됩니다.

- 대상 스냅샷이 제공되기 전에 소스 스냅샷을 삭제하면 스냅샷 복사가 실패할 수 있습니다. 소스 스냅샷을 삭제하기 전에 대상 스냅샷의 상태가 `AVAILABLE`인지 확인하세요.
- DB 스냅샷 복사 요청에서 옵션 그룹 이름을 지정하거나 옵션 그룹을 복사할 수 없습니다.
- 복사 프로세스 이전 또는 복사 프로세스 중에 소스 DB 스냅샷의 종속 AWS 리소스를 삭제하면 스냅샷 복사 요청이 비동기적으로 실패할 수 있습니다.
  - 계정의 RDS Custom 관리형 S3 버킷에 저장된 소스 DB 인스턴스의 Service Master Key(SMK) 백업 파일을 삭제하면 DB 스냅샷 복사가 비동기적으로 성공합니다. 하지만 TDE 지원 데이터베이스와 같이 SMK에 의존하는 SQL Server 기능에는 문제가 있습니다. 자세한 내용은 RDS Custom for

SQL Server에서 TDE 지원 데이터베이스의 PENDING_RECOVERY 상태 문제 해결 섹션을 참조하세요.

- 동일한 AWS 리전에서 DB 스냅샷을 복사하는 것은 현재 지원되지 않습니다.
- AWS 계정 간 DB 스냅샷을 복사하는 것은 현재 지원되지 않습니다.

Amazon RDS의 DB 스냅샷 복사 제한은 RDS Custom for SQL Server에도 적용됩니다. 자세한 내용은 제한 사항 섹션을 참조하세요.

암호화 처리

모든 RDS Custom for SQL Server DB 인스턴스 및 DB 스냅샷은 KMS 키로 암호화됩니다. 암호화된 스냅샷은 암호화된 스냅샷에만 복사할 수 있으므로, DB 스냅샷 복사 요청을 위해 대상 AWS 리전에서 유효한 KMS 키를 지정해야 합니다.

소스 스냅샷은 복사 프로세스 전체에서 암호화를 유지합니다. Amazon RDS는 봉투 암호화를 사용하여 지정된 대상 AWS 리전 KMS 키를 통해 복사 작업 중에 데이터를 보호합니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 봉투 암호화를 참조하세요.

리전 간 복사

AWS 리전 간에 DB 스냅샷을 복사할 수 있습니다. 하지만 리전 간 스냅샷 복사와 관련한 특정 제약 및 고려 사항이 있습니다.

스냅샷 복사를 위해 AWS 리전 간에 통신할 수 있도록 RDS 권한 부여

리전 간 DB 스냅샷 복사 요청이 성공적으로 처리되면 RDS가 복사를 시작합니다. RDS가 소스 스냅샷에 액세스할 수 있도록 권한 부여 요청이 생성됩니다. 이 권한 부여 요청은 소스 DB 스냅샷을 대상 DB 스냅샷에 연결합니다. 이렇게 하면 RDS가 지정된 대상 스냅샷에만 복사할 수 있습니다.

RDS는 서비스 연결 IAM 역할의 `rds:CrossRegionCommunication` 권한을 사용하여 권한 부여를 확인합니다. 복사가 승인되면 RDS는 소스 리전과 통신하여 복사 작업을 완료할 수 있습니다.

RDS는 이전에 CopyDBSnapshot 요청에 의해 승인되지 않은 DB 스냅샷에 액세스할 권한이 없습니다. 복사가 완료되면 권한 부여가 취소됩니다.

RDS는 서비스 연결 역할을 사용하여 소스 리전에서 승인을 확인합니다. 복사 프로세스 중에 서비스 연결 역할을 삭제하면 복사가 실패합니다.

자세한 내용은 AWS Identity and Access Management 사용 설명서의 서비스 연결 역할 사용을 참조하세요.

AWS Security Token Service 보안 인증 사용

전역 AWS Security Token Service(AWS STS) 엔드포인트의 세션 토큰은 기본적으로 활성화된 AWS 리전(상용 리전)에서만 유효합니다. `assumeRole`에서 AWS STS API 작업을 통해 얻은 자격 증명을 사용하는 경우 소스 리전이 옵트인 리전이면 리전별 엔드포인트를 사용합니다. 그렇지 않으면 요청이 실패합니다. 자격 증명은 두 리전 모두에서 유효해야 하며, 이는 리전 AWS STS 엔드포인트를 사용할 때만 옵트인 리전에 적용됩니다.

전역 엔드포인트를 사용하려면 작업의 두 리전 모두에 대해 전역 엔드포인트를 사용하도록 설정되어 있어야 합니다. 글로벌 엔드포인트를 AWS STS 계정 설정의 모든 AWS 리전에서 `Valid`로 설정합니다.

자세한 내용은 AWS Identity and Access Management 사용 설명서의 AWS 리전에서 AWS STS 관리를 참조하세요.

사용자 지정 엔진 버전(CEV)으로 만든 DB 인스턴스의 스냅샷

사용자 지정 엔진 버전(CEV)을 사용하는 DB 인스턴스의 DB 스냅샷의 경우 RDS는 CEV를 DB 스냅샷과 연결합니다. AWS 리전에서 CEV와 연결된 소스 DB 스냅샷을 복사하려면 RDS가 소스 DB 스냅샷과 함께 CEV를 대상 리전으로 복사합니다.

동일한 CEV와 연결된 여러 DB 스냅샷을 동일한 대상 리전에 복사하는 경우 첫 번째 복사 요청은 연결된 CEV를 복사합니다. 다음 요청의 복사 프로세스는 처음에 복사한 CEV를 찾아 다음 DB 스냅샷 사본과 연결합니다. 기존 CEV 사본은 DB 스냅샷 복사본과 연결할 수 있는 `AVAILABLE` 상태여야 합니다.

CEV와 연결된 DB 스냅샷을 복사하려면 요청자의 IAM 정책에 DB 스냅샷 복사와 연결된 CEV 복사를 모두 승인할 권한이 있어야 합니다. 연결된 CEV 복사를 허용하려면 요청자의 IAM 정책에 다음과 같은 권한이 필요합니다.

- `rds:CopyCustomDBEngineVersion` - 요청자 IAM 보안 주체는 소스 CEV를 소스 DB 스냅샷과 함께 대상 리전에 복사할 수 있는 권한이 있어야 합니다. 요청자 IAM 보안 주체가 소스 CEV를 복사할 권한이 없는 경우 권한 부여 오류로 인해 스냅샷 복사 요청이 실패합니다.
- `ec2:CreateTags` - 소스 CEV의 기본 EC2 AMI가 CEV 사본의 일부로 대상 리전에 복사됩니다. RDS Custom은 AMI를 복사하기 전에 AMI에 `AWSRDSCustom` 태그를 지정하려고 합니다. 요청자 IAM 보안 주체가 소스 리전의 소스 CEV를 기반으로 하는 AMI에 대해 태그를 생성할 권한이 있는지 확인하세요.

CEV 복사 권한에 대한 자세한 내용은 IAM 보안 주체에 필요한 권한 부여 섹션을 참조하세요.

IAM 보안 주체에 필요한 권한 부여

RDS Custom for SQL Server DB 스냅샷을 복사할 수 있는 충분한 액세스 권한을 가지고 있는지 확인하세요. 콘솔 또는 CLI를 사용하여 DB 스냅샷을 복사하는 IAM 역할 또는 사용자(IAM 보안 주체라고 함)는 DB 인스턴스를 성공적으로 생성하기 위해 다음 중 하나의 정책을 가지고 있어야 합니다.

- `AdministratorAccess` 정책
- 다음과 같은 추가 권한이 있는 `AmazonRDSFullAccess` 정책:

```
s3:CreateBucket
s3:GetBucketPolicy
s3:PutBucketPolicy
kms:CreateGrant
kms:DescribeKey
ec2:CreateTags
```

RDS Custom은 AWS 리전에서 스냅샷을 복사할 때 이러한 권한을 사용합니다. 이러한 권한은 RDS Custom 작업에 필요한 리소스를 계정에 구성합니다. `kms:CreateGrant` 권한에 대한 자세한 내용은 AWS KMS key 관리 섹션을 참조하세요.

다음 샘플 JSON 정책은 `AmazonRDSFullAccess` 정책 외에 필요한 권한을 부여합니다.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateS3BucketAndReadWriteBucketPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutBucketPolicy",
                "s3:GetBucketPolicy"
            ],
            "Resource": "arn:aws:s3:::do-not-delete-rds-custom-*"
        },
        {
            "Sid": "CreateKmsGrant",
            "Effect": "Allow",
```

```
            "Action": [
                "kms:CreateGrant",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CreateEc2Tags",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 서비스 제어 정책(SCP), 권한 경계 또는 IAM 보안 주체와 연결된 세션 정책에서 나열된 권한을 제한하지 않는지 확인합니다.

요청자의 IAM 정책에 컨텍스트 키가 있는 조건을 사용하는 경우 특정 조건으로 인해 요청이 실패할 수 있습니다. IAM 정책 조건으로 인한 일반적인 위험에 대한 자세한 내용은 리전 간 DB 스냅샷 복사 요청 섹션을 참조하세요.

DB 스냅샷 복사

다음 절차에 따라 DB 스냅샷을 복사합니다. 각 AWS 계정에 대해 AWS 리전 간에 DB 스냅샷을 한 번에 20개까지 복사할 수 있습니다. DB 스냅샷을 다른 AWS 리전으로 복사하면 그 AWS 리전에 유지되는 수동 DB 스냅샷이 생성됩니다. 소스 AWS 리전 외부로 DB 스냅샷을 복사하면 Amazon RDS 데이터 전송 요금이 발생합니다. 데이터 전송 요금에 대한 자세한 정보는 Amazon RDS 요금을 참조하십시오.

새 AWS 리전에 DB 스냅샷 사본이 생성된 후 DB 스냅샷 사본은 해당 AWS 리전의 다른 모든 DB 스냅샷과 똑같이 동작합니다.

AWS Management Console, AWS CLI 또는 Amazon RDS API를 사용하여 DB 스냅샷을 복사할 수 있습니다.

Console

다음 프로시저는 AWS Management Console을 사용하여 RDS Custom for SQL Server DB 스냅샷을 복사합니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.
3. 복사하려는 RDS Custom for SQL Server DB 스냅샷을 선택합니다.
4. 작업 드롭다운에서 스냅샷 복사를 선택합니다.

RDS > Snapshots > Copy snapshot

Copy snapshot

Settings

Source DB snapshot
DB snapshot identifier for the snapshot being copied.
db1-snapshot

New DB snapshot identifier
DB snapshot identifier for the new snapshot.
copied-db1-snapshot
Must start with a letter and only contain letters, digits, or hyphens.

Destination Region Info
Choose a Region

Resource tags Info
☐ Copy tags from the source DB snapshot

Depending on the amount of data to be copied and the Region you choose, this operation can take several hours to complete. The display on the progress bar might be delayed until setup is complete.

Encryption

Encryption Info
☑ Enable Encryption
Choose to encrypt the copy of the source DB snapshot. Master key IDs and aliases appear in the list after they have been created using KMS. You cannot remove encryption from an encrypted DB snapshot.

AWS KMS key Info
Enter a key ARN

Amazon Resource Name (ARN)
[redacted]
Example: arn:aws:kms:<region>:<accountID>:key/<key-id>

Account
[redacted]

KMS key ID
[redacted]

5. DB 스냅샷을 다른 AWS 리전에 복사하려면 대상 리전을 필요한 값으로 설정합니다.

   > **Note**
   >
   > 대상 AWS 리전에는 소스 AWS 리전과 동일한 가용 데이터베이스 엔진 버전이 있어야 합니다.

6. 새 DB 스냅샷 식별자에 DB 스냅샷의 고유 이름을 입력합니다. 자동 백업 또는 수동 스냅샷의 복사본을 여러 개 만들 수 있지만, 각 복사본에는 고유한 식별자가 있어야 합니다.
7. (선택 사항) [Copy Tags]를 선택하여 스냅샷의 태그와 값을 스냅샷 사본에 복사합니다.
8. 암호화에 대해 DB 스냅샷 복사본을 암호화하는 데 사용할 KMS 키 식별자를 지정합니다.

   > **Note**
   >
   > RDS Custom for SQL Server는 모든 DB 스냅샷을 암호화합니다. 암호화되지 않은 DB 스냅샷을 생성할 수 없습니다.

9. 스냅샷 복사를 선택합니다.

RDS Custom for SQL Server는 선택한 AWS 리전에 DB 인스턴스의 DB 스냅샷 복사본을 생성합니다.

AWS CLI

AWS CLI 명령 copy-db-snapshot을 사용하여 RDS Custom for SQL Server DB 스냅샷을 복사할 수 있습니다. 스냅샷을 새 AWS 리전으로 복사하는 경우 새 AWS 리전에서 명령을 실행합니다. 다음 옵션을 사용하여 DB 스냅샷을 복사할 수 있습니다. 시나리오에 따라 필요하지 않은 옵션도 있습니다.

- `--source-db-snapshot-identifier` - 소스 DB 스냅샷의 식별자입니다.
  - 소스 스냅샷이 사본과 다른 AWS 리전에 있는 경우 유효한 DB 스냅샷 ARN을 지정합니다. 예를 들어 `arn:aws:rds:us-west-2:123456789012:snapshot:instance1-snapshot-12345678`입니다.
- `--target-db-snapshot-identifier` – DB 스냅샷의 새 사본의 식별자입니다.
- `--kms-key-id` – 암호화된 DB 스냅샷의 KMS 키 식별자입니다. KMS 키 식별자는 KMS 키의 Amazon 리소스 이름(ARN), 키 식별자 또는 키 별칭입니다.

  - 암호화된 스냅샷을 다른 AWS 리전에 복사하는 경우 대상 AWS 리전에 대해 KMS 키를 지정해야 합니다. KMS 키는 생성된 AWS 리전에만 적용되며, 다중 리전 키를 사용하지 않는 한 다른 AWS 리전의 한 AWS 리전에서 암호화 키를 사용할 수 없습니다. 다중 리전 KMS 키에 대한 자세한 내용은 AWS KMS에서 다중 리전 키 사용을 참조하세요.
- `--copy-tags` - 소스 스냅샷의 태그와 값을 스냅샷 복사본에 포함합니다.

RDS Custom for SQL Server DB 스냅샷의 복사에는 다음 옵션이 지원되지 않습니다.

- `--copy-option-group`
- `--option-group-name`
- `--pre-signed-url`
- `--target-custom-availability-zone`

다음 코드 예제에서는 암호화된 DB 스냅샷을 미국 서부(오리건) 리전에서 미국 동부(버지니아 북부) 리전으로 복사합니다. 대상(us-east-1) 리전에서 명령을 실행합니다.

Linux, macOS 또는 Unix의 경우는 다음과 같습니다.

```
aws rds copy-db-snapshot \
     --region us-east-1 \
    --source-db-snapshot-identifier arn:aws:rds:us-
west-2:123456789012:snapshot:instance1-snapshot-12345678 \
    --target-db-snapshot-identifier mydbsnapshotcopy \
    --kms-key-id a1b2c3d4-1234-5678-wxyz-a1b2c3d4d5e6
```

Windows의 경우:

```
aws rds copy-db-snapshot ^
     --region us-east-1 ^
    --source-db-snapshot-identifier arn:aws:rds:us-
west-2:123456789012:snapshot:instance1-snapshot-12345678 ^
    --target-db-snapshot-identifier mydbsnapshotcopy ^
    --kms-key-id a1b2c3d4-1234-5678-wxyz-a1b2c3d4d5e6
```

RDS API

Amazon RDS API 작업 CopyDBSnapshot을 사용하여 RDS Custom for SQL Server DB 스냅샷을 복사할 수 있습니다. 스냅샷을 새 AWS 리전으로 복사하는 경우 새 AWS 리전에서 작업을 수행합니

다. 다음 파라미터를 사용하여 DB 스냅샷을 복사할 수 있습니다. 모든 파라미터가 필요한 것은 아닙니다.

- `SourceDBSnapshotIdentifier` - 소스 DB 스냅샷의 식별자입니다.
  - 소스 스냅샷이 사본과 다른 AWS 리전에 있는 경우 유효한 DB 스냅샷 ARN을 지정합니다. 예를 들어 `arn:aws:rds:us-west-2:123456789012:snapshot:instance1-snapshot-12345678`입니다.
- `TargetDBSnapshotIdentifier` – DB 스냅샷의 새 사본의 식별자입니다.
- `KmsKeyId` – 암호화된 DB 스냅샷의 KMS 키 식별자입니다. KMS 키 식별자는 KMS 키의 Amazon 리소스 이름(ARN), 키 식별자 또는 키 별칭입니다.
  - 암호화된 스냅샷을 다른 AWS 리전에 복사하는 경우 대상 AWS 리전에 대해 KMS 키를 지정해야 합니다. KMS 키는 생성된 AWS 리전에만 적용되며, 다중 리전 키를 사용하지 않는 한 다른 AWS 리전의 한 AWS 리전에서 암호화 키를 사용할 수 없습니다. 다중 리전 KMS 키에 대한 자세한 내용은 AWS KMS에서 다중 리전 키 사용을 참조하세요.
- `CopyTags` - 소스 스냅샷의 태그와 값을 스냅샷 사본에 복사하려면 이 파라미터를 `true`로 설정합니다. 기본값은 `false`입니다.

다음 옵션에는 RDS Custom for SQL Server DB 스냅샷 복사가 지원되지 않습니다.

- `CopyOptionGroup`
- `OptionGroupName`
- `PreSignedUrl`
- `TargetCustomAvailabilityZone`

다음 코드는 US East (N. Virginia) 리전에 `mydbsnapshotcopy`라는 새 이름으로 스냅샷 복사본을 생성합니다.

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CopyDBSnapshot
    &KmsKeyId=a1b2c3d4-1234-5678-wxyz-a1b2c3d4d5e6
    &SourceDBSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-
west-2%3A123456789012%3Asnapshot%3Ainstance1-snapshot-12345678
    &TargetDBSnapshotIdentifier=mydbsnapshotcopy
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
    &X-Amz-Date=20161117T221704Z
```

```
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-
date
    &X-Amz-
Signature=da4f2da66739d2e722c85fcfd225dc27bba7e2b8dbea8d8612434378e52adccf
```

온프레미스 데이터베이스를 SQL Server용 Amazon RDS Custom으로 마이그레이션

다음 프로세스를 따라 기본 백업 및 복원을 사용하여 온프레미스 Microsoft SQL Server 데이터베이스를 SQL Server용 Amazon RDS Custom으로 마이그레이션할 수 있습니다.

1. 온프레미스 DB 인스턴스에서 데이터베이스의 전체 백업을 수행합니다.
2. Amazon S3로 백업 파일을 업로드합니다.
3. S3의 백업 파일을 SQL Server DB 인스턴스용 RDS Custom으로 다운로드합니다.
4. SQL Server DB 인스턴스용 RDS Custom에서 다운로드한 백업 파일을 사용하여 데이터베이스를 복원합니다.

이 프로세스에서는 기본 전체 백업 및 복원을 사용하여 온프레미스에서 SQL Server용 RDS Custom으로 데이터베이스를 마이그레이션하는 방법을 설명합니다. 마이그레이션 프로세스 중에 전환 시간을 줄이기 위해 디퍼렌셜 또는 로그 백업을 사용하는 것도 고려할 수 있습니다.

RDS for SQL Server의 기본 백업 및 복원에 대한 일반적인 정보는 기본 백업 및 복원 기능을 사용하여 SQL Server 데이터베이스 가져오기 및 내보내기 섹션을 참조하세요.

주제

- 사전 조건
- 온프레미스 데이터베이스 백업
- Amazon S3로 백업 파일 업로드
- Amazon S3에서 백업 파일 다운로드
- SQL Server DB 인스턴스용 RDS Custom으로 백업 파일 복원

사전 조건

데이터베이스를 마이그레이션하기 전에 다음 작업을 수행합니다.

1. SQL Server DB 인스턴스용 RDS Custom에 대해 원격 데스크톱 연결(RDP)을 구성합니다. 자세한 내용은 RDP를 사용하여 RDS Custom DB 인스턴스에 연결 섹션을 참조하세요.
2. 데이터베이스 백업 파일을 업로드 및 다운로드할 수 있도록 Amazon S3에 액세스할 권한을 구성합니다.

온프레미스 데이터베이스 백업

SQL Server 기본 백업을 사용하여 온프레미스 DB 인스턴스의 데이터베이스 전체 백업을 수행할 수 있습니다.

다음 예제에서는 백업 파일 크기를 줄이는 COMPRESSION 옵션이 지정된 mydatabase라는 이름의 데이터베이스 백업을 보여줍니다.

온프레미스 데이터베이스를 백업하는 방법

1. SQL Server Management Studio(SSMS)를 사용하여 온프레미스 SQL Server 인스턴스에 연결합니다.
2. T-SQL 명령을 실행합니다.

```
backup database mydatabase to
disk ='C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\Backup\mydb-full-compressed.bak'
with compression;
```

Amazon S3로 백업 파일 업로드

AWS Management Console을 사용하여 mydb-full-compressed.bak 백업 파일을 Amazon S3로 업로드합니다.

S3로 백업 파일을 업로드하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 버킷(Buckets)에서 백업 파일을 업로드할 버킷의 이름을 선택합니다.
3. 업로드를 선택합니다.
4. 업로드 창에서 다음 중 하나를 수행합니다.

   - mydb-full-compressed.bak를 업로드(Upload) 창으로 끌어다 놓습니다.
   - 파일 추가(Add file)와 mydb-full-compressed.bak를 차례로 선택한 다음 열기(Open)를 선택합니다.

   Amazon S3는 백업 파일을 S3 객체로 업로드합니다. 업로드가 완료되면 업로드: 상태 페이지에서 성공 메시지를 볼 수 있습니다.

Amazon S3에서 백업 파일 다운로드

콘솔을 사용하여 S3의 백업 파일을 SQL Server DB 인스턴스용 RDS Custom으로 다운로드할 수 있습니다.

S3에서 백업 파일을 다운로드하는 방법

1. RDP를 사용하여 SQL Server DB 인스턴스용 RDS Custom에 연결합니다.
2. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
3. 버킷(Buckets) 목록에서 백업 파일이 들어 있는 버킷의 이름을 선택합니다.
4. `mydb-full-compressed.bak` 백업 파일을 선택합니다.
5. 작업(Actions)에서 다음으로 다운로드(Download as)를 선택합니다.
6. 제공된 링크에 대한 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 나서 다른 이름으로 저장(Save As)을 선택합니다.
7. `mydb-full-compressed.bak`를 `D:\rdsdbdata\BACKUP` 디렉터리에 저장합니다.

SQL Server DB 인스턴스용 RDS Custom으로 백업 파일 복원

SQL Server 기본 복원을 사용하여 백업 파일을 SQL Server DB 인스턴스용 RDS Custom으로 복원할 수 있습니다.

이 예제에서는 데이터 및 로그 파일 디렉터리가 온프레미스 DB 인스턴스와 다르기 때문에 `MOVE` 옵션이 지정됩니다.

백업 파일을 복원하는 방법

1. SSMS를 사용하여 SQL Server DB 인스턴스용 RDS Custom에 연결합니다.
2. T-SQL 명령을 실행합니다.

```
restore database mydatabase from disk='D:\rdsdbdata\BACKUP\mydb-full-
compressed.bak'
with move 'mydatabase' to 'D:\rdsdbdata\DATA\mydatabase.mdf',
move 'mydatabase_log' to 'D:\rdsdbdata\DATA\mydatabase_log.ldf';
```

Amazon RDS Custom for SQL Server DB 인스턴스 업그레이드

새 DB 엔진 버전을 사용하도록 수정하여 Amazon RDS Custom for SQL Server DB 인스턴스를 업그레이드할 수 있습니다. DB 인스턴스 업그레이드에 대한 일반적인 정보는 DB 인스턴스 엔진 버전 업그레이드 섹션을 참조하세요.

주제

- RDS Custom for SQL Server 업그레이드 개요
- 메이저 및 마이너 엔진 버전 업그레이드
- 데이터베이스 호환성 수준

RDS Custom for SQL Server 업그레이드 개요

Amazon RDS Custom for SQL Server는 메이저 및 마이너 버전 업그레이드를 지원합니다. 마이너 버전 업그레이드에는 보안 패치, 버그 수정 및 엔진 개선 기능이 포함될 수 있습니다. Microsoft는 이러한 업데이트를 누적 업데이트(CU)로 릴리스합니다. 메이저 버전 업그레이드는 SQL Server 2019에서 2022로 업그레이드하는 것처럼 버전 간에 새로운 기능과 엔진 변경 사항을 도입합니다. 두 업그레이드를 즉시 적용하거나 예약된 유지 관리 기간 동안 적용할 수 있습니다. 잠재적인 이전 버전과의 호환성 문제를 방지하기 위해 업그레이드하기 전에 비프로덕션 환경에서 애플리케이션을 테스트하는 것이 좋습니다.

RDS Custom for SQL Server에서는 RDS 제공 엔진 버전(RPEV) 또는 사용자 지정 엔진 버전(CEV)을 업그레이드할 수 있습니다.

- RDS 제공 엔진 버전(RPEV)에는 최신 운영 체제(OS) 패치와 SQL Server 누적 업데이트(CU)가 포함되어 있습니다.
- 사용자 지정 엔진 버전(CEV)의 경우 2단계 프로세스를 따라야 합니다. 먼저 대상 SQL Server 버전으로 새 CEV를 생성합니다. RDS Custom for SQL Server용 CEV 생성 준비 섹션을 참조하세요. 이 대상 버전은 현재 버전과 같거나 더 최신 버전이어야 합니다. 새 CEV가 생성되면 이 새 버전을 사용하도록 데이터베이스 인스턴스를 수정합니다. 자세한 내용은 다중 AZ를 사용하여 Amazon RDS Custom for SQL Server CEV에 대한 마이너 버전 업그레이드 수행을 참조하세요.

  실행 중인 RDS Custom 인스턴스에 SQL Server 누적 인플레이스 업데이트를 적용하지 마세요. 특정 SQL Server 버전(예: SQL Server 2022 CU16)으로 CEV를 생성한 경우 인스턴스에 직접 최신 누적 업데이트를 적용하면 지원 경계에서 벗어나 `SP-S3006` 오류를 보고합니다. CEV를 사용하여 기존 SQL Server 인스턴스를 패치하려면 원하는 누적 업데이트가 포함된 새 CEV를 생성한 다음 기존 인스턴스를 수정하여 새 CEV로 전환합니다.

다중 AZ 배포에서 RDS Custom for SQL Server DB 인스턴스를 업그레이드하는 경우 RDS Custom for SQL Server가 인스턴스에 대한 롤링 업그레이드를 수행합니다. 이 접근 방식은 한 번에 하나의 인스턴스를 업그레이드하여 가동 중지 시간을 최소화합니다. RDS는 롤링 업그레이드를 위해 다음 작업을 수행합니다.

1. 대기 DB 인스턴스를 업그레이드합니다.
2. 업그레이드된 대기 DB 인스턴스로 장애 조치하여 이를 새 기본 DB 인스턴스로 만듭니다.
3. 새 대기 DB 인스턴스를 업그레이드합니다.

다중 AZ 배포의 DB 인스턴스 가동 중지 시간은 장애 조치에 걸리는 시간입니다.

RDS Custom for SQL Server DB 인스턴스를 업그레이드할 때는 다음과 같은 제한 사항이 적용됩니다.

- Custom DB 옵션 및 파라미터 그룹은 지원되지 않습니다.
- RDS Custom for SQL Server DB 인스턴스에 연결하는 추가 스토리지 볼륨은 업그레이드 후에 연결되지 않습니다.
- CEV의 경우 SQL Server 누적 업데이트의 인플레이스 적용은 지원되지 않으므로 인스턴스가 지원 경계에서 제거됩니다.

메이저 및 마이너 엔진 버전 업그레이드

메이저 및 마이너 엔진 버전 업그레이드는 모두 되돌릴 수 없으며 항상 더 최신 버전으로 업그레이드를 수행해야 합니다. 사용 가능한 대상 버전을 식별하려면 AWS Management Console을 사용하고 DB 인스턴스를 수정할 때 사용 가능한 버전 중에서 선택합니다. 또는 `describe-db-engine-versions` CLI 명령 또는 DescribeDBEngineVersions RDS API 명령을 사용합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-engine-versions \
    --engine custom-sqlserver-se \
    --engine-version 15.00.4322.2.v1 \
    --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" \
    --output table
```

Windows의 경우:

```
aws rds describe-db-engine-versions ^
```

```
    --engine custom-sqlserver-se ^
    --engine-version 15.00.4322.2.v1 ^
    --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" ^
    --output table
```

출력에는 사용 가능한 대상 엔진 버전이 표시됩니다.

```
-------------------------
|DescribeDBEngineVersions|
+------------------------+
|      EngineVersion     |
+------------------------+
|  15.00.4410.1.v1       |
|  15.00.4415.2.v1       |
|  15.00.4430.1.v1       |
|  16.00.4165.4.v1       |
|  16.00.4175.1.v1       |
|  16.00.4185.3.v1       |
+------------------------+
```

대상 버전을 식별한 후 AWS Management Console을 사용하고 RDS Custom for SQL Server DB 인스턴스 수정의 지침을 따릅니다. 또는 `modify-db-instance` CLI 명령 또는 ModifyDBInstance RDS API 명령을 사용합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier DB_INSTANCE_IDENTIFIER \
    --engine-version ENGINE_VERSION \
    --allow-major-version-upgrade \
    --region Region \
    --no-apply-immediately
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier DB_INSTANCE_IDENTIFIER ^
    --engine-version ENGINE_VERSION ^
    --allow-major-version-upgrade ^
    --region Region ^
    --no-apply-immediately
```

> **Note**
>
> 메이저 버전 업그레이드를 수행하려면 `--allow-major-version-upgrade` 파라미터를 포함해야 합니다.

데이터베이스 호환성 수준

Microsoft SQL Server 데이터베이스 호환성 수준을 이용해 일부 데이터베이스 동작이 이전 버전의 SQL Server를 모방하도록 조정할 수 있습니다. 자세한 내용은 Microsoft 설명서의 호환성 수준을 참조하십시오.

DB 인스턴스를 업그레이드할 때 기존의 모든 데이터베이스는 원래 호환성 수준으로 유지됩니다. 예를 들어, SQL Server 2019에서 SQL Server 2022로 업그레이드할 경우 모든 기존 데이터베이스의 호환성 수준은 150입니다. 업그레이드 이후에 생성된 새 데이터베이스의 호환성 수준은 160입니다.

ALTER DATABASE 명령을 사용하여 데이터베이스의 호환성 수준을 변경할 수 있습니다. 예를 들어, `customeracct`라는 이름의 데이터베이스를 SQL Server 2022와 호환되도록 변경하려면 다음 명령을 실행합니다.

```
ALTER DATABASE customeracct SET COMPATIBILITY_LEVEL = 160
```

Amazon RDS Custom for SQL Server의 DB 문제 해결

RDS Custom의 공동 책임 모델은 OS 셸 수준 액세스 권한과 데이터베이스 관리자 액세스 권한을 제공합니다. RDS Custom은 시스템 계정에서 리소스를 실행하는 Amazon RDS와 달리 계정에서 리소스를 실행합니다. 접근 권한이 높을수록 책임도 커집니다. 다음 섹션에서는 Amazon RDS Custom for SQL Server DB 인스턴스의 문제를 해결하는 방법을 알아볼 수 있습니다.

> **Note**
>
> 이 섹션에서는 RDS Custom for SQL Server의 문제를 해결하는 방법을 설명합니다. RDS Custom for Oracle 관련 문제 해결을 보려면 Amazon RDS Custom for Oracle의 DB 문제 해결 섹션을 참조하세요.

주제

- RDS Custom 이벤트 보기
- RDS Custom 이벤트 구독
- RDS Custom for SQL Server의 CEV 오류 문제 해결
- RDS Custom for SQL Server에서 지원되지 않는 구성 문제 해결
- RDS Custom for SQL Server에서 Storage-Full 문제 해결
- RDS Custom for SQL Server에서 TDE 지원 데이터베이스의 PENDING_RECOVERY 상태 문제 해결

RDS Custom 이벤트 보기

RDS Custom 및 Amazon RDS DB 인스턴스의 이벤트 보기 절차는 동일합니다. 자세한 내용은 Amazon RDS 이벤트 보기 섹션을 참조하세요.

`describe-events` 명령을 사용하여 AWS CLI를 통해 RDS Custom 이벤트 알림을 볼 수 있습니다. RDS Custom에서는 몇 가지 새로운 이벤트가 도입되었습니다. 이벤트 카테고리는 Amazon RDS와 동일합니다. 이벤트 목록은 Amazon RDS 이벤트 카테고리 및 이벤트 메시지 섹션을 참조하세요.

다음 예제에서는 지정된 RDS Custom DB 인스턴스에 대해 발생한 이벤트의 세부 정보를 검색합니다.

```
aws rds describe-events \
    --source-identifier my-custom-instance \
```

```
    --source-type db-instance
```

RDS Custom 이벤트 구독

RDS Custom 및 Amazon RDS DB 인스턴스의 이벤트 구독 절차는 동일합니다. 자세한 내용은 Amazon RDS 이벤트 알림 구독 섹션을 참조하세요.

CLI를 사용하여 RDS Custom 이벤트 알림을 구독하려면 `create-event-subscription` 명령을 사용하면 되며, 다음 필수 파라미터를 포함합니다.

- `--subscription-name`
- `--sns-topic-arn`

다음 예제에서는 현재 AWS 계정에서 RDS Custom DB 인스턴스의 백업 및 복구 이벤트에 대한 구독을 생성합니다. 알림은 `--sns-topic-arn`에서 지정한 Amazon Simple Notification Service(Amazon SNS) 주제로 전송됩니다.

```
aws rds create-event-subscription \
    --subscription-name my-instance-events \
    --source-type db-instance \
    --event-categories '["backup","recovery"]' \
    --sns-topic-arn arn:aws:sns:us-east-1:123456789012:interesting-events
```

RDS Custom for SQL Server의 CEV 오류 문제 해결

CEV 생성이 실패할 수 있습니다. 이 경우 RDS Custom이 `RDS-EVENT-0198` 이벤트 메시지를 표시합니다. RDS 이벤트 보기에 대한 자세한 내용은 Amazon RDS 이벤트 카테고리 및 이벤트 메시지 단원을 참조하세요.

다음 정보를 사용하면 가능한 원인을 해결하는 데 도움이 됩니다.

메시지	문제 해결 제안		
`Custom Engine Version creation expected a Sysprep'd AMI. Retry creation using a Sysprep'd AMI.`	AMI에서 생성한 EC2 인스턴스에서 Sysprep을 실행합니다. Sysprep을 사용한 AMI 준비에 대한 자세한 내용은 Sysprep을 사용하여 표준화된 Amazon Machine Image(AMI) 생성을 참조하세요.		

메시지	문제 해결 제안		
EC2 Image permissio ns for image (AMI_ID) weren't found for customer (Customer_ID). Verify customer (Customer_ID) has valid permissions on the EC2 Image.	생성에 사용한 계정 및 프로 필에 선택한 AMI의 create EC2 Instance 및 Describe Images에 대한 필수 권한이 있는지 확인합니다.		
Failed to rebuild databases with server collation (collation name) due to missing setup.exe file for SQL Server.	C:\Program Files\Mic rosoft SQL Server\nnn \Setup Bootstrap\SQLnnnn\ setup.exe 에서 setup 파일을 사용할 수 있는지 확인합니다.		
Image (AMI_ID) doesn't exist in your account (ACCOUNT_ID). Verify (ACCOUNT_ID) is the owner of the EC2 image.	AMI가 동일한 고객 계정에 있는지 확 인합니다.		
Image id (AMI_ID) isn't valid. Specify a valid image id, and try again.	AMI 이름이 잘못되었습니다. 올바른 AMI ID가 제공되었는지 확인합니다.		

메시지	문제 해결 제안		
`Image (AMI_ID) operating system platform isn't supported. Specify a valid image, and try again.`	SQL Server Enterprise, Standard 또는 Web Edition을 포함하는 Windows Server가 있는 지원되는 AMI를 선택하세요. EC2 Marketplace에서 다음 사용 작업 코드 중 하나를 사용하여 AMI를 선택합니다. • RunInstances:0102 - SQL Server Enterprise가 설치된 Windows • RunInstances:0006 - SQL Server Standard가 설치된 Windows • RunInstances:0202 - SQL Server Web이 설치된 Windows		
`SQL Server Web Edition isn't supported for creating a Custom Engine Version using Bring Your Own Media. Specify a valid image, and try again.`	지원되는 에디션의 SQL Server가 포함된 AMI를 사용하세요. 자세한 내용은 RDS Custom for SQL Server CEV에 대한 버전 지원 섹션을 참조하세요.		
`The custom engine version can't be the same as the OEV engine version. Specify a valid CEV, and try again.`	클래식 RDS Custom for SQL Server 엔진 버전은 지원되지 않습니다. 예를 들어 15.00.4073.23.v1 버전은 지원되지 않습니다. 지원되는 버전 번호를 사용하세요.		
`The custom engine version isn't in an active state. Specify a valid CEV, and try again.`	작업을 완료하려면 CEV가 `AVAILABLE` 상태여야 합니다. CEV를 `INACTIVE`에서 `AVAILABLE` 로 수정합니다.		

메시지	문제 해결 제안		
The custom engine version isn't valid for an upgrade. Specify a valid CEV with an engine version greater or equal to (X), and try again.	대상 CEV가 유효하지 않습니다. 유효한 업그레이드 경로에 대한 요구 사항을 확인하세요.		
The custom engine version isn't valid. Names can include only lowercase letters (a-z), dashes (-), underscores (_), and periods (.). Specify a valid CEV, and try again.	필수 CEV 명명 규칙에 따르세요. 자세한 내용은 RDS Custom for SQL Server CEV 요구 사항 섹션을 참조하세요.		
The custom engine version isn't valid. Specify valid database engine version, and try again. Example: 15.00.4073.23-cev123.	지원되지 않는 DB 엔진 버전이 제공되었습니다. 지원되는 DB 엔진 버전을 사용하세요.		
The expected architecture is (X) for image (AMI_ID), but architecture (Y) was found.	x86_64 아키텍처를 기반으로 구축된 AMI를 사용하세요.		
The expected owner of image (AMI_ID) is customer account ID (ACCOUNT_ID), but owner (ACCOUNT_ID) was found.	권한이 있는 AMI에서 EC2 인스턴스를 생성하세요. EC2 인스턴스에서 Sysprep을 실행하여 기본 이미지를 생성하고 저장하세요.		
The expected platform is (X) for image (AMI_ID), but platform (Y) was found.	Windows 플랫폼으로 구축된 AMI를 사용하세요.		

메시지	문제 해결 제안		
`The expected root device type is (X) for image %s, but root device type (Y) was found.`	EBS 디바이스 유형을 사용하여 AMI를 생성하세요.		
`The expected SQL Server edition is (X), but (Y) was found.`	SQL Server Enterprise, Standard 또는 Web Edition을 포함하는 Windows Server가 있는 지원되는 AMI를 선택하세요. EC2 Marketplace에서 다음 사용 작업 코드 중 하나를 사용하여 AMI를 선택합니다. • RunInstances:0102 - SQL Server Enterprise가 설치된 Windows • RunInstances:0006 - SQL Server Standard가 설치된 Windows • RunInstances:0202 - SQL Server Web이 설치된 Windows		
`The expected state is (X) for image (AMI_ID), but the following state was found: (Y).`	AMI가 `AVAILABLE` 상태인지 확인하세요.		
`The provided Windows OS name (X) isn't valid. Make sure the OS is one of the following: (Y).`	지원되는 Windows OS를 사용하세요.		
`Unable to find bootstrap log file in path.`	`C:\Program Files\Microsoft SQL Server\nnn\Setup Bootstrap\Log\Summary.txt` 에서 로그 파일을 사용할 수 있는지 확인합니다.		

메시지	문제 해결 제안		
`RDS expected a Windows build version greater than or equal to (X), but found version (Y)..`	최소 OS 빌드 버전이 14393인 AMI를 사용하세요.		
`RDS expected a Windows major version greater than or equal to (X), but found version (Y)..`	최소 OS 메이저 버전이 10.0 이상인 AMI를 사용하세요.		

RDS Custom for SQL Server에서 지원되지 않는 구성 문제 해결

공유 책임 모델이므로 RDS Custom for SQL Server DB 인스턴스를 `unsupported-configuration` 상태로 전환시키는 구성 문제는 사용자가 해결해야 합니다. AWS 인프라와 관련된 문제인 경우 콘솔 또는 AWS CLI를 사용하여 해결할 수 있습니다. 운영 체제 또는 데이터베이스 구성에 문제가 있는 경우 호스트에 로그인하여 해결하면 됩니다.

> **Note**
>
> 이 섹션에서는 RDS Custom for SQL Server에서 지원되지 않는 구성 문제를 해결하는 방법을 알아봅니다. RDS Custom for Oracle에 대한 자세한 내용은 RDS Custom for Oracle에서 지원되지 않는 구성 문제 해결 섹션을 참조하세요.

다음 표에는 지원 경계에서 보내는 알림 이벤트와 이를 해결하는 방법에 대한 설명이 나와 있습니다. 이러한 알림과 지원 경계는 변경될 수 있습니다. 지원 경계의 배경은 RDS Custom 지원 범위 섹션을 참조하세요. 이벤트 설명은 Amazon RDS 이벤트 카테고리 및 이벤트 메시지 섹션을 참조하세요.

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스
`SP-S0000`	지원되지 않는 수동 구성	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성]으로 설정된 이유는 다음과 같습니다. X	이 문제를 해결하려면 지원 사례를 생성하세요.

AWS 리소스(인프라)

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스
SP-S1001	EC2 인스턴스 상태	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. 기본 EC2 인스턴스 %s이(가) RDS 인스턴스를 중단하지 않고 중지되었습니다. 기본 EC2 인스턴스를 시작하고 이진 볼륨 및 데이터 볼륨이 연결되어 있는지 확인하여 문제를 해결할 수 있습니다. RDS 인스턴스를 중지하려는 경우 먼저 기본 EC2 인스턴스가 AVAILABLE 상태인지 확인한 다음 RDS 콘솔 또는 CLI를 사용하여 RDS 인스턴스를 중지해야 합니다.	DB 인스턴스의 상태를 확인하려면 콘솔을 사용하거나 다음 AWS CLI 명령을 실행합니다. `aws rds describe-db-instances \ --db-instance-identifier db-instance-name |grep DBInstanceStatus`
SP-S1002	EC2 인스턴스 상태	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. RDS DB 인스턴스 상태는 STOPPED로 설정되었지만, 기본 EC2 인	다음 AWS CLI 명령을 사용하여 DB 인스턴스의 상태를 확인합니다. `aws rds describe-db-instances \ --db-instance-identifier *db-instance-name* | grep DBInstanceStatus`

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스
		스턴스 %s이(가) 시작되었습니다. 기본 EC2 인스턴스를 중지하여 문제를 해결할 수 있습니다. RDS 인스턴스를 시작하려는 경우 콘솔 또는 CLI를 사용하세요.	EC2 콘솔을 사용하여 EC2 인스턴스의 상태를 확인할 수도 있습니다. DB 인스턴스를 시작하려면 콘솔을 사용하거나 다음 AWS CLI 명령을 실행합니다. `aws rds start-db-instance \ --db-instance-identifier db-instance-name`
SP-S1003	EC2 인스턴스 클래스	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. EC2 호스트의 예상 DB 인스턴스 클래스와 구성된 DB 인스턴스 클래스 간에 불일치가 있습니다. DB 인스턴스 클래스를 원래 클래스 유형으로 수정하여 문제를 해결할 수 있습니다.	다음 CLI 명령을 사용하여 예상 DB 인스턴스 클래스를 확인합니다. `aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep DBInstanceClass`
SP-S1004	EBS 스토리지 볼륨에 액세스할 수 없음	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. EC2 인스턴스에 연결된 원래 EBS 스토리지 볼륨 %s에 현재 액세스할 수 없습니다.	

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스
SP-S1005	EBS 스토리지 볼륨이 분리됨	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. 원본 EBS 스토리지 볼륨 'volume-id'가 연결되지 않았습니다. EC2 인스턴스에 연결된 EBS 볼륨을 연결하여 문제를 해결할 수 있습니다.	EBS 볼륨을 다시 연결한 후 다음 CLI 명령을 사용하여 EBS 볼륨 'volume-id'가 RDS 인스턴스에 제대로 연결되어 있는지 확인합니다. `aws ec2 describe-volumes \ --volume-ids volume-id | grep InstanceId`

SP-S1006	EBS 스토리지 볼륨 크기	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. EBS 스토리지 볼륨 'volume-id'의 예상 설정과 구성된 설정이 일치하지 않습니다. EC2 수준에서 볼륨 크기가 원래 값인 [%s]에서 수동으로 변경되었습니다. 이 문제를 해결하려면 지원 사례를 생성하세요.	다음 CLI 명령을 사용하여 EBS 볼륨 'volume-id' 세부 정보와 RDS 인스턴스 세부 정보의 볼륨 크기를 비교합니다.
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep AllocatedStorage`
			다음 CLI 명령을 사용하여 실제 할당된 볼륨 크기를 확인합니다.
			`aws ec2 describe-volumes \ --volume-ids |grep Size`

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스

SP-S1007	EBS 스토리지 볼륨 구성	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. EBS 스토리지 볼륨 'volume-id'의 예상 설정과 구성된 설정이 일치하지 않습니다. EBS 스토리지 볼륨 구성 [IOPS, 처리량, 볼륨 유형]을 EC2 수준에서 원래 값인 [IOPS: %s, 처리량: %s, 볼륨 유형: %s]로 수정하여 문제를 해결할 수 있습니다. 향후 스토리지 수정에는 RDS 콘솔 또는 CLI를 사용하세요. EC2 수준에서 볼륨 크기도 원래 값인 [%s]에서 수동으로 변경되었습니다. 이 문제를 해결하려면 지원 사례를 생성하세요.	다음 CLI 명령을 사용하여 EBS 볼륨 'volume-id' 세부 정보의 볼륨 유형과 RDS 인스턴스 세부 정보를 비교합니다. EBS 수준의 값이 RDS 수준의 값과 일치하는지 확인하세요.
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep StorageType`
			RDS 수준에서 스토리지 처리량의 예상 값을 구하는 방법:
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep StorageThroughput`
			RDS 수준에서 볼륨 IOPS의 예상 값을 구하는 방법:
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep Iops`
			EC2 수준에서 현재 스토리지 유형을 가져오는 방법:
			`aws ec2 describe-volumes \ --volume-ids |grep VolumeType`

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스

			EC2 수준에서 스토리지 처리량의 현재 값을 구하는 방법:
			`aws ec2 describe-volumes \ --volume-ids |grep Throughput`
			EC2 수준에서 볼륨 IOPS의 현재 값을 구하는 방법:
			`aws ec2 describe-volumes \ --volume-ids |grep Iops`

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스

SP-S1008	EBS 스토리지 볼륨 크기 및 구성	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. EBS 스토리지 볼륨 'volume-id'의 예상 설정과 구성된 설정이 일치하지 않습니다. EBS 스토리지 볼륨 구성 [IOPS, 처리량, 볼륨 유형]을 EC2 수준에서 원래 값인 [IOPS: %s, 처리량: %s, 볼륨 유형: %s]로 수정하여 문제를 해결할 수 있습니다. 향후 스토리지 수정에는 RDS 콘솔 또는 CLI를 사용하세요. EC2 수준에서 볼륨 크기도 원래 값인 [%s]에서 수동으로 변경되었습니다. 이 문제를 해결하려면 지원 사례를 생성하세요.	다음 CLI 명령을 사용하여 EBS 볼륨 'volume-id' 세부 정보의 볼륨 유형과 RDS 인스턴스 세부 정보를 비교합니다. EBS 수준의 값이 RDS 수준의 값과 일치하는지 확인하세요.
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep StorageType`
			RDS 수준에서 스토리지 처리량의 예상 값을 구하는 방법:
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep StorageThroughput`
			RDS 수준에서 볼륨 IOPS의 예상 값을 구하는 방법:
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep Iops`
			EC2 수준에서 현재 스토리지 유형을 가져오는 방법:
			`aws ec2 describe-volumes \ --volume-ids |grep VolumeType`

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스

			EC2 수준에서 스토리지 처리량의 현재 값을 구하는 방법:
			`aws ec2 describe-volumes \ --volume-ids |grep Throughput`
			EC2 수준에서 볼륨 IOPS의 현재 값을 구하는 방법:
			`aws ec2 describe-volumes \ --volume-ids |grep Iops`
			예상 할당된 볼륨 크기를 구하는 방법:
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep AllocatedStorage`
			실제 할당된 볼륨 크기를 구하는 방법:
			`aws ec2 describe-volumes \ --volume-ids |grep Size`

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스
SP-S1009	SQS 권한	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. IAM 인스턴스 프로파일에 대한 Amazon Simple Queue Service(Amazon SQS) 권한이 누락되었습니다. 호스트와 연결된 IAM 프로파일에 ["SQS:SendMessage","SQS:ReceiveMessage","SQS:DeleteMessage","SQS:GetQueueUrl"] 권한이 있는지 확인하여 문제를 해결할 수 있습니다.	

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스
SP-S1010	SQS VPC 엔드포인트	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. VPC 엔드포인트 정책이 Amazon Simple Queue Service(Amazon SQS) 작업을 차단하고 있습니다. 필요한 SQS 작업을 허용하도록 VPC 엔드포인트 정책을 수정하여 문제를 해결할 수 있습니다.	

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스
SP-S1011	이벤트 버스 정책	RDS Custom DB 인스턴스 상태가 다음과 같은 이유로 [지원되지 않는 구성]으로 설정되었습니다. 이벤트 버스 arn:aws:events:region-1:123456789012:event-bus/default에 대한 리소스 기반 정책이 Amazon CloudWatch events:PutEvents 작업을 거부함. EventBus %s에 대한 events:PutEvents 작업을 허용하도록 리소스 기반 정책을 수정하여 이 문제를 해결하세요.	

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스
SP-S1012	CloudWatch VPC 권한	RDS Custom DB 인스턴스 상태가 다음과 같은 이유로 [지원되지 않는 구성]으로 설정되었습니다. VPC 엔드포인트 정책에 Amazon CloudWatch 이벤트에 액세스할 수 있는 권한이 누락됨. EventBus arn:aws:events:region-1:123456789012:event-bus/default에 대해 events:PutEvents를 허용하도록 VPC 엔드포인트 정책을 수정하여 이 문제를 해결하세요.	

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스
SP-S1013	서비스 제어 정책	RDS Custom DB 인스턴스 상태가 다음과 같은 이유로 [지원되지 않는 구성]으로 설정되었습니다. AWS Organizations의 서비스 제어 정책에 Amazon CloudWatch 이벤트에 액세스할 수 있는 권한이 누락됨. EventBus arn:aws:events:region-1:123456789012:event-bus/default에 대해 events:PutEvents를 허용하도록 서비스 제어 정책을 수정하여 이 문제를 해결하세요.	

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스
SP-S1014	IAM 인스턴스 프로파일	RDS Custom DB 인스턴스 상태가 다음과 같은 이유로 [지원되지 않는 구성]으로 설정되었습니다. IAM 인스턴스 프로파일 %s 권한이 Amazon CloudWatch 이벤트를 거부함. 인스턴스와 연결된 IAM 프로파일에서 ["events:PutEvents"]를 'Allow'로 설정하고 EventBus arn:aws:events:region-1:123456789012:event-bus/default에 대해 events:PutEvents를 허용하여 이 문제를 해결하세요.	

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스
SP-S1015	IAM 인스턴스 프로파일	RDS Custom DB 인스턴스 상태가 다음과 같은 이유로 [지원되지 않는 구성]으로 설정되었습니다. IAM 인스턴스 프로파일 %s에 Amazon CloudWatch 권한이 누락됨. 인스턴스와 연결된 IAM 프로파일에서 ["events:PutEvents"] 권한을 포함하고 EventBus arn:aws:events:region-1:123456789012:event-bus/default에 대해 events:PutEvents를 허용하여 이 문제를 해결하세요.	

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스
SP-S1016	IAM 권한 경계	RDS Custom DB 인스턴스 상태가 다음과 같은 이유로 [지원되지 않는 구성]으로 설정되었습니다. IAM 인스턴스 프로파일 %s 권한에 Amazon CloudWatch 이벤트를 거부하는 권한 경계가 있음. IAM 인스턴스 프로파일 권한 경계에서 EventBus arn:aws:events:region-1:123456789012:event-bus/default에 대해 ["events:PutEvents"]를 'Allow'로 설정하여 이 문제를 해결하세요.	

운영 체제

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스
SP-S2001	SQL 서비스 상태	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. SQL Server 서비스가 시작되지 않았습니다. 호스트에서 SQL Server 서비스	

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스
		를 다시 시작하여 문제를 해결할 수 있습니다. 이 DB 인스턴스가 다중 AZ DB 인스턴스이고 재시작이 실패할 경우 호스트를 중지하고 시작하여 장애 조치를 시작합니다.	

SP-S2002	RDS Custom 에이전트 상태	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. RDS Custom 에이전트 서비스가 설치되지 않았거나 시작할 수 없습니다. Windows 이벤트 로그를 검토하여 서비스가 시작되지 않는 이유를 확인하고 적절한 조치를 취하여 문제를 수정하면 해결할 수 있습니다. 추가 지원이 필요한 경우 지원 사례를 생성하세요.	호스트에 로그인하고 RDS Custom 에이전트가 실행 중인지 확인합니다.
			
			다음 명령을 사용하여 에이전트 상태를 볼 수 있습니다.
			
			`$name = "RDSCustomAgent"`
			`$service = Get-Service $name`
			`Write-Host $service.Status`
			
			상태가 Running이 아니면 다음 명령을 사용하여 서비스를 시작할 수 있습니다.
			
			`Start-Service $name`
			
			에이전트를 시작할 수 없는 경우 Windows 이벤트를 확인하여 에이전트를 시작할 수 없는 이유를 확인합니다. 에이전트를 사용하려면 Windows 사용자가 서비스를 시작해야 합니다. Windows 사용자가 존재하고 서비스를 실행할 권한이 있는지 확인하세요.

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스
SP-S2003	SSM 에이전트 상태	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. Amazon SSM 에이전트 서비스에 연결할 수 없습니다. `Get-Service AmazonSSMAgent` PowerShell 명령으로 서비스 상태를 확인하거나 `Start-Service AmazonSSMAgent` 로 서비스를 시작하여 문제를 해결할 수 있습니다. ssm, ssmmessages, ec2messages 리전 엔드포인트에 대한 HTTPS(포트 443) 아웃바운드 트래픽이 허용되는지 확인하세요.	자세한 내용은 SSM Agent 문제 해결을 참조하세요. SSM 엔드포인트 문제를 해결하려면 SSM 엔드포인트에 연결할 수 없음 및 ssm-cli를 사용하여 관리형 노드 가용성 문제 해결을 참조하세요.

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스
SP-S2004	RDS Custom 에이전트 로그인	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. SQL 로그인 "$HOSTNAME/RDSAgent" 에서 예상치 못한 문제가 발생했습니다. 이 문제를 해결하려면 지원 사례를 생성하세요.	
SP-S2005	시간대	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. Amazon EC2 인스턴스 [%s]의 시간대가 변경되었습니다. 인스턴스 생성 시 지정된 설정으로 시간대를 다시 수정하여 문제를 해결할 수 있습니다. 특정 시간대의 인스턴스를 만들려면 RDS Custom 설명서를 참조하세요.	Get-Timezone PowerShell 명령을 실행하여 시간대를 확인합니다. 자세한 내용은 RDS Custom for SQL Server DB 인스턴스의 현지 시간대 섹션을 참조하세요.

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스
SP-S2006	고가용성 소프트웨어 솔루션 버전	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. 현재 인스턴스의 고가용성 소프트웨어 솔루션이 예상 버전과 다릅니다. 이 문제를 해결하려면 지원 사례를 생성하세요.	
SP-S2007	고가용성 소프트웨어 솔루션 구성	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. 고가용성 소프트웨어 솔루션의 구성 설정이 %s 인스턴스에서 예상치 못한 값으로 수정되었습니다. 이 문제를 해결하려면 EC2 인스턴스를 재부팅합니다. EC2 인스턴스를 재부팅하면 설정이 고가용성 소프트웨어 솔루션에 필요한 구성으로 자동 업데이트됩니다.	

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스

SP-S2008	SQL Server 서비스	RDS Custom DB 인스턴스가 [지원되지 않는 구성](으)로 설정되었습니다. SQLServer (MSSQLServer) 서비스가 호스트에 없습니다. 이 문제를 해결하려면 지원 사례를 만드세요.	다음 명령을 사용하여 에이전트 상태를 볼 수 있습니다.
			`$name = "MSSQLServer"`
			`$service = Get-Service $name`
			`Write-Host $service.Status`

SP-2009	SSL 인증서	RDS 사용자 지정 DB 인스턴스는 RDS 중단을 유발하는 자체 서명되지 않은 SSL 인증서로 인해 [지원되지 않는 구성]으로 설정됩니다. 이 문제를 해결하려면 신뢰할 수 있는 루트 인증서 스토어에서 자체 서명되지 않은 인증서를 제거합니다.	다음 PowerShell 명령을 실행하여 자체 서명되지 않은 인증서를 검토합니다. `Get-ChildItem cert:\LocalMachine\root -Recurse | Where-Object {$_.Issuer -ne $_.Subject -and $_.Issuer -notlike "*RDSCustomAgentCA*"}` 자세한 내용은 HTTP Error 403.16 when you try to access a website that's hosted on IIS를 참조하시기 바랍니다.

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스
SP-2010	루트 볼륨 스토리지 상태	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성]으로 설정된 이유는 루트 볼륨 스토리지가 가득 찼기 때문입니다. 이 문제를 해결하려면 루트 EBS 볼륨 'volume-id'에서 최소 500MiB의 스토리지 공간을 확보하거나 볼륨 크기를 늘리고 EC2 인스턴스 'instance-id'에서 C 드라이브 크기를 조정합니다. EC2 인스턴스를 교체하면 루트 볼륨 크기 변경 사항이 유지되지 않습니다.	루트(C:) 볼륨에서 사용 가능한 스토리지를 보려면 다음 명령을 사용합니다. `(Get-PSDrive -Name C).Free / 1MB` EBS 루트 볼륨 수정에 대한 자세한 내용은 방법 섹션을 참조하세요.

데이터베이스

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스
SP-S3001	SQL Server 공유 메모리 프로토콜	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. SQL Server 공유 메모리 프로토콜이 비활성화되었습니다. SQL Server Configuration	SQL Server Configuration Manager > SQL Server 네트워크 구성 > MSSQLSERVER용 프로토콜 > 공유 메모리에서 활성화됨 상태를 확인하면 됩니다. 프로토콜을 활성화한 후 SQL Server 프로세스를 다시 시작합니다.

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스
		Manager에서 공유 메모리 프로토콜을 활성화하여 문제를 해결할 수 있습니다.	
SP-S3002	Service Master Key	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. RDS 자동화에서 새로운 Service Master Key(SMK) 생성의 일부로 SMK 백업을 수행할 수 없습니다. 이 문제를 해결하려면 지원 사례를 생성하세요.	
SP-S3003	Service Master Key	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. Service Master Key(SMK)와 관련된 메타데이터가 누락되었거나 불완전합니다. 이 문제를 해결하려면 지원 사례를 생성하세요.	

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스

SP-S3004	DB 엔진 버전 및 에디션	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. SQL Server의 필요한 버전 및 에디션과 설치된 버전 및 에디션 간에 불일치가 있습니다. RDS Custom for SQL Server에서 SQL Server 에디션 수정은 지원되지 않습니다. 또한 RDS Custom EC2 인스턴스에서 SQL Server 버전을 수동으로 변경하는 것도 지원되지 않습니다. 이 문제를 해결하려면 지원 사례를 생성하세요.	다음 쿼리를 실행하여 SQL 버전을 가져옵니다.
			`select @@version`
			다음 AWS CLI 명령을 실행하여 RDS SQL 엔진 버전 및 에디션을 가져옵니다.
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name |grep EngineVersion`
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name |grep Engine`
			자세한 내용은 RDS Custom for SQL Server DB 인스턴스 수정 및 DB 인스턴스 엔진 버전 업그레이드(을)를 참조하세요.

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스

SP-S3005	DB 엔진 에디션	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. 현재 SQL Server 에디션이 필요한 SQL Server 에디션인 [%s]과(와) 일치하지 않습니다. RDS Custom for SQL Server에서 SQL Server 에디션 수정은 지원되지 않습니다. 이 문제를 해결하려면 지원 사례를 생성하세요.	다음 쿼리를 실행하여 SQL 에디션을 가져옵니다.
			
			Example
			
			`select @@version`
			
			다음 AWS CLI 명령을 실행하여 RDS SQL 엔진 에디션을 가져옵니다.
			
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name |grep Engine`

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스

SP-S3006	DB 엔진 버전	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. 현재 SQL Server 에디션이 필요한 SQL Server 에디션인 [%s]과(와) 일치하지 않습니다. RDS Custom EC2 인스턴스에서는 SQL Server 버전을 수동으로 변경할 수 없습니다. 이 문제를 해결하려면 지원 사례를 생성하세요. 향후 SQL Server 버전을 수정할 경우 AWS RDS 콘솔에서 또는 modify-db-instance CLI 명령을 통해 인스턴스를 수정할 수 있습니다.	다음 쿼리를 실행하여 SQL 버전을 가져옵니다.
			
			Example
			
			`select @@version`
			
			다음 AWS CLI 명령을 실행하여 RDS SQL 엔진 버전을 가져옵니다.
			
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name |grep EngineVersion`
			
			자세한 내용은 RDS Custom for SQL Server DB 인스턴스 수정 및 DB 인스턴스 엔진 버전 업그레이드(을)를 참조하세요.

이벤트 코드	구성 영역	RDS 이벤트 메시지	검증 프로세스

SP-S3007	데이터베이스 파일 위치	RDS Custom DB 인스턴스 상태가 [지원되지 않는 구성](으)로 설정된 이유는 다음과 같습니다. 데이터베이스 파일이 D:\ 드라이브 외부에 구성되어 있습니다. ROW, LOG, FILESTREAM 등을 비롯한 모든 데이터베이스 파일이 D:\ 드라이브에 저장되어 있는지 확인하여 문제를 해결할 수 있습니다.	다음 쿼리를 실행하여 기본 경로에 없는 데이터베이스 파일의 위치를 나열합니다.
			`USE master;`
			`SELECT physical_name as files_not_in_default_path`
			`FROM sys.master_files`
			`WHERE SUBSTRING(physical_name,1,3)!='D:\';`

SP-S3008	데이터베이스 수 한도를 초과함	RDS Custom DB 인스턴스 상태가 다음과 같은 이유로 [지원되지 않는 구성]으로 설정되었습니다. DB 인스턴스의 총 데이터베이스 수가 최대 한도인 5,000개를 초과함. 이 문제를 해결하려면 데이터베이스 수를 지원되는 최대 한도 미만으로 줄이세요.	다음 명령을 사용하여 총 데이터베이스 수를 확인하세요.
			`SELECT COUNT(name) as databaseCount`
			`FROM sys.databases`
			`WHERE name not in ('tempdb','master','model','msdb','DWDiagnostics','DWConfiguration','DWQueue');`

RDS Custom for SQL Server에서 **`Storage-Full`** 문제 해결

RDS Custom은 루트(C:) 볼륨도 모니터링합니다. RDS Custom for SQL Server DB 인스턴스는 루트 볼륨에 사용 가능한 디스크 공간이 500MiB 미만일 때 `unsupported-configuration` 상태로 전환됩니다. (RDS Custom for SQL Server에서 지원되지 않는 구성 문제 해결의 `Event SP-S2010` 섹션 참조)

RDS Custom for SQL Server에서 TDE 지원 데이터베이스의 PENDING_RECOVERY 상태 문제 해결

자동 암호 해독에 문제가 발생해도 투명한 데이터 암호화(TDE)가 활성화된 SQL Server 데이터베이스는 `PENDING_RECOVERY` 상태를 유지할 수 있습니다. 이는 일반적으로 계정의 RDS Custom 관리형 S3 버킷에 저장된 소스 DB 인스턴스 Service Master Key(SMK) 백업 파일이 복원 완료 전에 삭제된 경우 DB 인스턴스 복원 후에 발생합니다.

자동 암호 해독을 활성화하고 TDE 지원 데이터베이스를 온라인으로 가져오려면 Database Master Key(DMK)를 암호로 열고 SMK를 사용하여 DMK를 암호화해야 합니다.

다음 SQL Server 명령을 참조용으로 사용하세요.

```
-- Identify PENDING_RECOVERY TDE databases
USE MASTER;
GO
SELECT name, is_encrypted, state_desc FROM sys.databases;
GO

-- Open DMK using password
OPEN MASTER KEY DECRYPTION BY PASSWORD = '<password>';
GO

-- Encrypt DMK using SMK
ALTER MASTER KEY ADD ENCRYPTION BY SERVICE MASTER KEY;
GO

-- Close SMK
CLOSE MASTER KEY;
GO

-- Bring the TDE databases online
ALTER DATABASE <database_name> SET ONLINE;
GO
```

```
-- Verify TDE databases are now in ONLINE state
SELECT name, is_encrypted, state_desc FROM sys.databases;
GO
```

AWS Outposts 기반 Amazon RDS

Amazon RDS on AWS Outposts는 RDS for SQL Server, RDS for MySQL 및 RDS for PostgreSQL 데이터베이스를 AWS Outposts Rack Gen1 및 Gen2 환경으로 확장합니다. AWS Outposts는 퍼블릭 AWS 리전과 동일한 하드웨어를 사용하여 AWS 서비스, 인프라 및 운영 모델을 온프레미스로 제공합니다. Outposts의 RDS를 사용하면 온프레미스로 실행해야 하는 비즈니스 애플리케이션과 가까운 곳에 관리형 DB 인스턴스를 프로비저닝할 수 있습니다. AWS Outposts에 대한 자세한 내용은 AWS Outposts 설명서 및 AWS Outposts 제품 페이지를 참조하세요.

AWS 클라우드에서 실행되는 RDS DB 인스턴스에서와 마찬가지로 동일한 AWS Management Console, AWS CLI 및 RDS API를 사용하여 Outposts 기반 RDS DB 인스턴스를 프로비저닝하고 관리할 수 있습니다. RDS on Outposts는 Amazon S3의 데이터베이스 프로비저닝, 운영 체제 및 데이터베이스 패치 적용, 백업 및 장기 보관과 같은 작업을 자동화합니다.

Outposts의 RDS는 DB 인스턴스의 자동 백업을 지원합니다. DB 인스턴스를 백업 및 복원하려면 Outpost와 AWS 리전 간 네트워크 연결이 필요합니다. Outpost의 모든 DB 스냅샷 및 트랜잭션 로그는 AWS 리전에 저장됩니다. AWS 리전으로부터 DB 스냅샷에서 다른 Outpost로 DB 인스턴스를 복원할 수 있습니다. 자세한 내용은 백업 소개 섹션을 참조하세요.

Outposts의 RDS는 DB 인스턴스의 자동 유지 관리 및 업그레이드를 지원합니다. 자세한 정보는 DB 인스턴스 유지 관리의 내용을 참조하세요.

RDS on Outposts는 AWS KMS key을(를) 사용하여 DB 인스턴스 및 DB 스냅샷에 대해 유휴 시 암호화를 사용합니다. 유휴 시 암호화에 대한 자세한 내용은 Amazon RDS 리소스 암호화 단원을 참조하세요.

기본적으로, Outposts 서브넷의 EC2 인스턴스는 Amazon Route 53 DNS 서비스를 사용하여 도메인 이름을 IP 주소로 확인할 수 있습니다. Outpost와 AWS 리전 간의 경로 대기 시간에 따라, Route 53에서 DNS 확인 시간이 길어질 수 있습니다. 이 경우, 온프레미스 환경에 로컬로 설치된 DNS 서버를 사용할 수 있습니다. 자세한 내용은 AWS Outposts 사용 설명서에서 DNS를 참조하세요.

AWS 리전에 대한 네트워크 연결을 사용할 수 없는 경우에도 DB 인스턴스는 로컬에서 계속 실행됩니다. 로컬 DNS 서버를 보조 서버로 구성하여 DNS 이름 확인을 사용해 DB 인스턴스에 계속 액세스할 수 있습니다. 그러나 새 DB 인스턴스를 생성하거나 기존 DB 인스턴스를 수정할 수 없습니다. 연결이 없을 때는 자동 백업이 수행되지 않습니다. DB 인스턴스 장애가 발생하면 연결이 복원될 때까지 DB 인스턴스가 자동으로 교체되지 않습니다. 가능한 한 빨리 네트워크 연결을 복원하는 것이 좋습니다.

주제

- Amazon RDS on AWS Outposts의 사전 조건

- Amazon RDS on AWS Outposts에서 Amazon RDS 기능 지원
- Amazon RDS on AWS Outposts에 대해 지원되는 DB 인스턴스 클래스
- Outposts의 RDS에 대한 고객 소유 IP 주소
- AWS Outposts의 Amazon RDS 다중 AZ 배포 작업
- Amazon RDS on AWS Outposts DB 인스턴스 생성
- AWS Outposts에서 Amazon RDS용 읽기 전용 복제본 생성
- AWS Outposts의 Amazon RDS에서 DB 인스턴스 복원 시 고려 사항

Amazon RDS on AWS Outposts의 사전 조건

다음은 Amazon RDS on AWS Outposts 사용을 위한 사전 조건입니다.

- 온프레미스 데이터 센터에 AWS Outposts을 설치합니다. 자세한 내용은 AWS Outposts Server installation guide의 Installing an AWS Outposts server를 참조하세요.
- Outposts의 RDS에 사용할 수 있는 서브넷이 하나 이상 있는지 확인합니다. 다른 워크로드에 동일한 서브넷을 사용할 수 있습니다.
- Outpost와 AWS 리전 간에 안정적인 네트워크 연결이 있어야 합니다.

Amazon RDS on AWS Outposts에서 Amazon RDS 기능 지원

다음 표에서는 Amazon RDS on AWS Outposts에서 지원하는 Amazon RDS 기능에 대해 설명합니다.

기능	지원	참고	추가 정보

DB 인스턴스 프로비저닝	예	RDS for SQL Server, RDS for MySQL 및 RDS for PostgreSQL DB 엔진에 대해서만 DB 인스턴스를 생성할 수 있습니다. 다음 버전이 지원됩니다.
		• Microsoft SQL Server:
		  • 15.00.4043.16.v1 이상 2019 버전
		  • 14.00.3294.2.v1 이상 2017 버전
		  • 13.00.5820.21.v1 이상 2016 버전
		• MySQL 8.0 및 8.4 버전
		• 모든 PostgreSQL 16, 15, 14, 13 버전, PostgreSQL 버전 12.5 이상의 PostgreSQL 12 버전
			Amazon RDS on AWS Outposts DB 인스턴스 생성

Microsoft SQL Server Management Studio를 사용하여 Microsoft SQL Server DB 인스턴스에 연결	예	일부 TLS 버전 및 암호화 암호는 안전하지 않을 수 있습니다. 암호화를 끄려면 ???에 나와 있는 지침을 따르세요.	Microsoft SQL Server DB 인스턴스에 연결
마스터 사용자 암호 수정	예	없음	Amazon RDS DB 인스턴스 수정

기능	지원	참고	추가 정보
DB 인스턴스 이름 변경	예	없음	Amazon RDS DB 인스턴스 수정
DB 인스턴스 재부팅	예	없음	DB 인스턴스 재부팅
DB 인스턴스 중지	예	없음	Amazon RDS DB 인스턴스의 일시적 중지
DB 인스턴스 시작	예	없음	이전에 중지된 Amazon RDS DB 인스턴스 시작
다중 AZ 배포	예	다중 AZ 배포는 MySQL 및 PostgreSQL DB 인스턴스에서 지원됩니다. 다중 AZ 배포는 직접 VPC 라우팅(DVR)을 지원하지 않습니다.	Amazon RDS on AWS Outposts DB 인스턴스 생성 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리
DB 파라미터 그룹	예	없음	Amazon RDS의 파라미터 그룹
읽기 전용 복제본	예	읽기 전용 복제본은 MySQL 및 PostgreSQL DB 인스턴스에서 지원됩니다. 읽기 전용 복제본은 직접 VPC 라우팅(DVR)을 지원하지 않습니다.	AWS Outposts에서 Amazon RDS용 읽기 전용 복제본 생성
저장 시 암호화	예	Outposts의 RDS는 암호화되지 않은 DB 인스턴스를 지원하지 않습니다.	Amazon RDS 리소스 암호화

기능	지원	참고	추가 정보
AWS Identity and Access Managemen t(IAM) 데이터베이스 인증	No	없음	MariaDB, MySQL 및 PostgreSQL IAM 데이터베이스 인증
IAM 역할을 DB 인스턴스와 연결	No	없음	add-role-to-db-instance AWS CLI 명령 AddRoleToDBInstance RDS API 작업
Kerberos 인증	No	없음	Kerberos 인증
Amazon RDS 리소스에 태그 지정	예	없음	Amazon RDS 리소스에 태그 지정
옵션 그룹 수	예	없음	옵션 그룹 작업
유지 관리 기간 변경	예	없음	DB 인스턴스 유지 관리
자동 마이너 버전 업그레이드	예	없음	마이너 엔진 버전 자동 업그레이드
백업 기간 수정	예	없음	백업 소개 Amazon RDS DB 인스턴스 수정
DB 인스턴스 클래스 변경	예	없음	Amazon RDS DB 인스턴스 수정
할당된 스토리지 변경	예	없음	Amazon RDS DB 인스턴스 수정

기능	지원	참고	추가 정보
Storage autoscaling(스토리지 Autoscaling)	예	없음	Amazon RDS 스토리지 Autoscaling을 사용한 용량 자동 관리
수동 및 자동 DB 인스턴스 스냅샷	예	자동 백업 및 수동 스냅샷을 AWS 리전에 저장할 수 있습니다. Outpost에 로컬로 저장할 수도 있습니다. 로컬 백업은 MySQL 및 PostgreSQL DB 인스턴스에서 지원됩니다. Outpost에 백업을 저장하려면 Outposts에서 Amazon S3가 구성되어 있는지 확인하세요. 다중 AZ 인스턴스 배포에는 로컬 백업이 지원되지 않습니다.	Amazon RDS on AWS Outposts DB 인스턴스 생성 Amazon S3 on Outposts Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성
DB 스냅샷에서 복원	예	복원된 DB 인스턴스에 대한 자동 백업 및 수동 스냅샷을 상위 AWS 리전 또는 Outpost에 로컬로 저장할 수 있습니다.	AWS Outposts의 Amazon RDS에서 DB 인스턴스 복원 시 고려 사항 DB 인스턴스 복원
Amazon S3에서 DB 인스턴스 복원	No	없음	MySQL DB 인스턴스로 백업 복원
Amazon S3에 스냅샷 데이터 내보내기	No	없음	Amazon S3 for Amazon RDS로 DB 스냅샷 데이터 내보내기

기능	지원	참고	추가 정보
시점 복구	예	복원된 DB 인스턴스에 대한 자동 백업 및 수동 스냅샷을 상위 AWS 리전 또는 Outpost에 로컬로 저장할 수 있습니다. 단, 한 가지 예외가 있습니다.	AWS Outposts의 Amazon RDS에서 DB 인스턴스 복원 시 고려 사항 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원
확장 모니터링	No	없음	Enhanced Monitoring을 사용하여 OS 지표 모니터링
Amazon CloudWatch 모니터링	예	AWS 리전의 데이터베이스에 사용할 수 있는 것과 동일한 지표 집합을 볼 수 있습니다.	Amazon CloudWatch로 Amazon RDS 지표 모니터링
CloudWatch Logs에 데이터베이스 엔진 로그 게시	예	없음	Amazon CloudWatch Logs에 데이터베이스 로그 게시
이벤트 알림	예	없음	Amazon RDS 이벤트 알림 작업
Amazon RDS Performance Insights	No	없음	성능 개선 도우미를 통한 Amazon RDS 모니터링

기능	지원	참고	추가 정보
데이터베이스 로그 보기 또는 다운로드	No	Outposts 기반 RDS는 콘솔을 사용하여 데이터베이스 로그를 보거나 AWS CLI 또는 RDS API를 사용하여 데이터베이스 로그를 설명하는 기능을 지원하지 않습니다. Outposts 기반 RDS는 콘솔을 사용하여 데이터베이스 로그를 다운로드하거나 AWS CLI 또는 RDS API를 사용하여 데이터베이스 로그를 다운로드하는 것을 지원하지 않습니다.	Amazon RDS 로그 파일 모니터링
Amazon RDS 프록시	No	없음	Amazon RDS Proxy
Amazon RDS for MySQL에 대한 저장된 프로시저	예	없음	RDS for MySQL 저장 프로시저 참조
RDS for MySQL에 대한 외부 데이터베이스를 사용한 복제	No	없음	외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성
Amazon RDS for Microsoft SQL Server에 대한 기본 백업 및 복원	예	없음	기본 백업 및 복원 기능을 사용하여 SQL Server 데이터베이스 가져오기 및 내보내기

Amazon RDS on AWS Outposts에 대해 지원되는 DB 인스턴스 클래스

Amazon RDS on AWS Outposts는 다음 DB 인스턴스 클래스를 지원합니다.

- 범용 DB 인스턴스 클래스
  - db.m7i.48xlarge
  - db.m7i.24xlarge
  - db.m7i.16xlarge
  - db.m7i.12xlarge
  - db.m7i.8xlarge
  - db.m7i.4xlarge
  - db.m7i.2xlarge
  - db.m7i.xlarge
  - db.m7i.large
  - db.m5.24xlarge
  - db.m5.16xlarge
  - db.m5.12xlarge
  - db.m5.8xlarge
  - db.m5.4xlarge
  - db.m5.2xlarge
  - db.m5.xlarge
  - db.m5.large
- 메모리 최적화 DB 인스턴스 클래스
  - db.r7i.48xlarge
  - db.r7i.24xlarge
  - db.r7i.16xlarge
  - db.r7i.12xlarge
  - db.r7i.8xlarge
  - db.r7i.4xlarge
  - db.r7i.2xlarge

- db.r7i.xlarge
- db.r7i.large
- db.r5.24xlarge
- db.r5.16xlarge
- db.r5.12xlarge
- db.r5.8xlarge
- db.r5.4xlarge
- db.r5.2xlarge
- db.r5.xlarge
- db.r5.large

Outpost를 구성한 방법에 따라 이러한 클래스 중 일부를 사용할 수 없을 수도 있습니다. 예를 들어 Outpost용 db.r5 클래스를 구입하지 않은 경우 Outposts의 RDS와 함께 사용할 수 없습니다.

Outposts의 RDS DB 인스턴스에는 범용 SSD 스토리지만 지원됩니다. DB 인스턴스 클래스에 대한 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요.

Amazon RDS은(는) DB 인스턴스의 유지 관리 및 복구를 관리하며, 이를 위해 Outpost에 활성 용량이 필요합니다. 프로덕션 환경의 각 DB 인스턴스 클래스에 대해 N+1 EC2 인스턴스를 구성하는 것이 좋습니다. RDS on Outposts는 이러한 EC2 인스턴스의 추가 용량을 사용하여 유지 관리 및 복구 작업을 수행할 수 있습니다. 예를 들어, 프로덕션 환경에 3개의 db.m5.large DB 인스턴스 클래스가 있고 5개의 db.r5.xlarge DB 인스턴스 클래스가 있는 경우, 최소 4개의 m5.large EC2 인스턴스와 6개의 r5.xlarge EC2 인스턴스를 사용하는 것이 좋습니다. 자세한 내용은 AWS Outposts 사용 설명서에서 AWS Outposts의 복원력을 참조하세요.

Outposts의 RDS에 대한 고객 소유 IP 주소

AWS Outposts 기반 Amazon RDS는 온프레미스 네트워크에 대해 제공한 정보를 사용하여 주소 풀을 생성합니다. 이 풀을 고객 소유 IP 주소 풀(CoIP 풀)이라고 합니다. 고객 소유 IP 주소(CoIP)는 온프레미스 네트워크를 통해 Outpost 서브넷의 리소스에 대한 로컬 또는 외부 연결을 제공합니다. CoIP에 대한 자세한 내용은 AWS Outposts 사용 설명서에서 고객 소유 IP 주소를 참조하세요.

각 Outposts의 RDS DB 인스턴스에는 Virtual Private Cloud(VPC) 내부의 트래픽에 사용할 프라이빗 IP 주소가 있습니다. 이 프라이빗 IP 주소는 공개적으로 액세스할 수 없습니다. 퍼블릭 옵션을 사용하여 DB 인스턴스에 프라이빗 IP 주소 외에 퍼블릭 IP 주소가 있는지 여부를 지정할 수 있습니다. 퍼블릭 IP 주소를 연결에 사용하면 인터넷을 통해 라우팅되며 경우에 따라 지연 시간이 길어질 수 있습니다.

Outposts의 RDS에서는 이러한 프라이빗 및 퍼블릭 IP 주소를 사용하는 대신, 서브넷을 통해 DB 인스턴스에 대한 CoIP를 활성화합니다. Outposts의 RDS DB 인스턴스에 대해 CoIP를 활성화하면 DB 인스턴스를 DB 인스턴스 엔드포인트에 연결합니다. Outposts의 RDS는 VPC 내부 및 외부의 모든 연결에 CoIP를 자동으로 사용합니다.

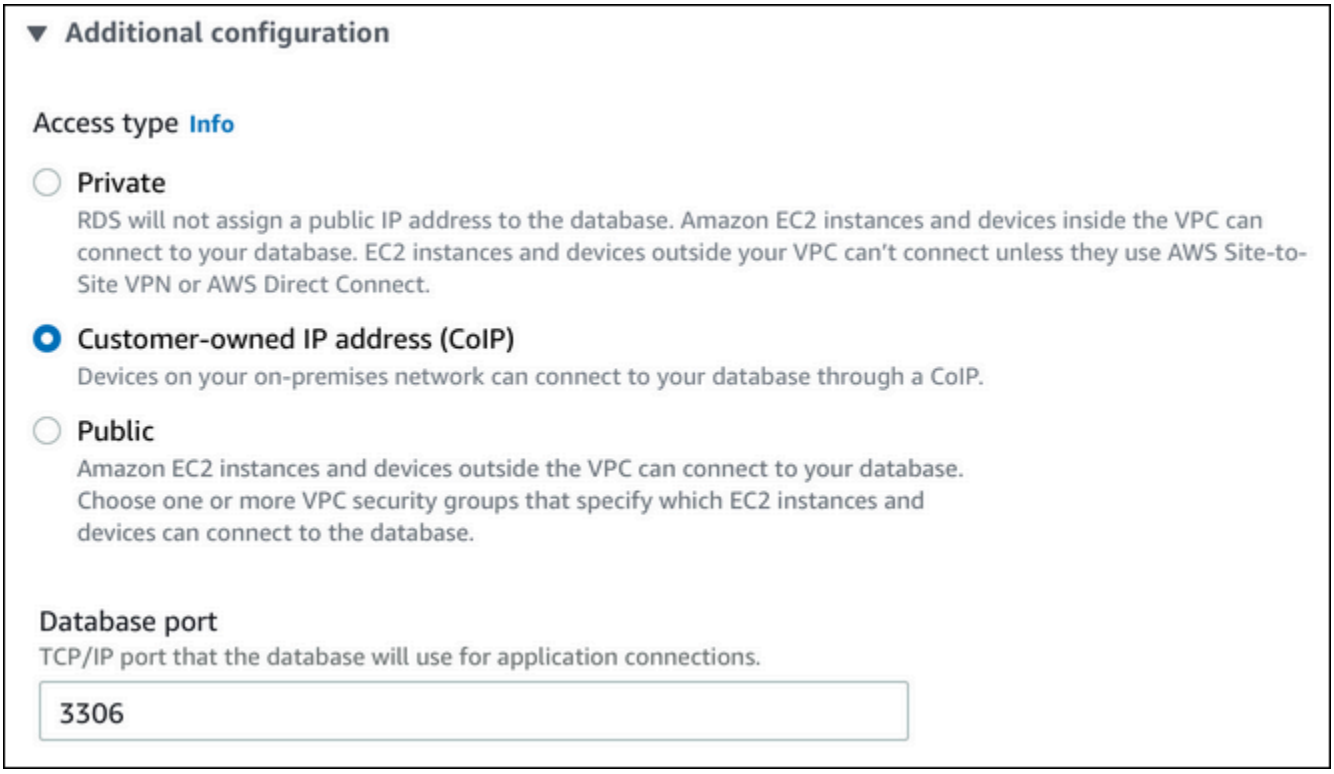
CoIP는 Outposts의 RDS DB 인스턴스에 다음과 같은 이점을 제공할 수 있습니다.

- 연결 지연 시간 단축
- 강화된 보안

COIP 사용

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 Outposts의 RDS DB 인스턴스에 대해 CoIP를 활성화하거나 비활성화할 수 있습니다.

- AWS Management Console에서 액세스 유형(Access type)의 고객 소유 IP 주소(Customer-owned IP address) (CoIP) 설정을 선택하여 CoIP를 활성화합니다. 다른 설정 중 하나를 선택하여 끕니다.



- AWS CLI에서는 `--enable-customer-owned-ip | --no-enable-customer-owned-ip` 옵션을 사용합니다.
- RDS API를 사용하는 경우 `EnableCustomerOwnedIp` 파라미터를 사용합니다.

다음 작업 중 하나를 수행할 때 CoIP를 켜거나 끌 수 있습니다.

- DB 인스턴스 생성

  자세한 내용은 Amazon RDS on AWS Outposts DB 인스턴스 생성 섹션을 참조하세요.
- DB 인스턴스 수정

  자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.
- 읽기 전용 복제본 생성

  자세한 내용은 AWS Outposts에서 Amazon RDS용 읽기 전용 복제본 생성 단원을 참조하십시오.
- 스냅샷에서 DB 인스턴스 복원

  자세한 내용은 DB 인스턴스 복원 섹션을 참조하세요.
- DB 인스턴스를 지정된 시간으로 복원

자세한 내용은 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 단원을 참조하십시오.

> **Note**
>
> 때로는 DB 인스턴스에 대해 CoIP를 활성화했지만 Amazon RDS가 해당 DB 인스턴스에 CoIP를 할당할 수 없는 경우가 있습니다. 이 경우 DB 인스턴스 상태가 incompatible-network로 변경됩니다. DB 인스턴스 상태에 대한 자세한 내용은 Amazon RDS DB 인스턴스 상태 보기 섹션을 참조하세요.

제한 사항

Outposts의 RDS DB 인스턴스에 대한 CoIP 지원에는 다음과 같은 제한이 적용됩니다.

- DB 인스턴스에 CoIP를 사용하는 경우 해당 DB 인스턴스에 대해 공개 액세스가 꺼져 있는지 확인하세요.
- VPC 보안 그룹에 대한 인바운드 규칙에 CoIP 주소 범위(CIDR 블록)가 포함되어 있는지 확인합니다. 보안 그룹 설정에 대한 자세한 내용은 보안 그룹을 생성하여 VPC 내부의 DB 인스턴스에 대한 액세스를 제공 섹션을 참조하세요.
- CoIP 풀에서 DB 인스턴스로 CoIP를 할당할 수 없습니다. DB 인스턴스에 대해 CoIP를 사용하면 Amazon RDS가 CoIP 풀의 CoIP를 DB 인스턴스에 자동으로 할당합니다.
- Outpost 리소스(소유자)를 소유하는 AWS 계정를 사용하거나 동일한 조직의 다른 AWS 계정(소비자)과 다음 리소스를 공유해야 합니다.
  - Outpost
  - DB 인스턴스의 VPC에 대한 로컬 게이트웨이(LGW) 라우팅 테이블
  - LGW 라우팅 테이블에 대한 CoIP 풀 또는 풀

  자세한 내용은 AWS Outposts 사용 설명서에서 공유 AWS Outposts 리소스 작업을 참조하세요.

AWS Outposts의 Amazon RDS 다중 AZ 배포 작업

다중 AZ 배포의 경우 Amazon RDS는 하나의 AWS Outpost에 기본 DB 인스턴스를 생성합니다. RDS는 다른 Outpost에 있는 대기 DB 인스턴스에 데이터를 동기식으로 복제합니다.

AWS Outposts의 다중 AZ 배포는 AWS 리전의 다중 AZ 배포처럼 작동하지만 다음과 같은 차이점이 있습니다.

- 두 개 이상의 Outposts 사이에 로컬 연결이 필요합니다.
- 고객 소유 IP(CoIP) 풀이 필요합니다. 자세한 내용은 Outposts의 RDS에 대한 고객 소유 IP 주소 단원을 참조하십시오.
- 로컬 네트워크에서 복제가 실행됩니다.

AWS Outposts의 다중 AZ는 Outposts의 RDS에 대해 지원되는 모든 버전의 MySQL 및 PostgreSQL에서 사용할 수 있습니다. 다중 AZ 배포에는 로컬 백업이 지원되지 않습니다. 자세한 내용은 Amazon RDS on AWS Outposts DB 인스턴스 생성 단원을 참조하십시오.

공동 책임 모델 작업

AWS는 고가용성을 위해 구성된 DB 인스턴스를 제공하기 위해 상업적으로 합당한 노력을 기울이지만 가용성은 공동 책임 모델을 사용합니다. Outposts의 RDS가 DB 인스턴스를 장애 조치 및 복구하는 기능을 사용하려면 각 Outposts가 B에 연결되어 있어야 합니다.

또한 Outposts의 RDS는 기본 DB 인스턴스를 호스팅하는 Outpost와 동기식 복제를 위해 대기 DB 인스턴스를 호스팅하는 Outpost 간에 연결해야 합니다. 이 연결에 영향을 주면 Outposts의 RDS가 장애 조치를 수행하지 못할 수 있습니다.

동기식 데이터 복제의 결과로 표준 DB 인스턴스 배포의 지연 시간이 길어질 수 있습니다. 기본 DB 인스턴스를 호스팅하는 Outpost와 대기 DB 인스턴스를 호스팅하는 Outpost 간의 연결 대역폭과 지연 시간은 지연 시간에 직접적인 영향을 미칩니다. 자세한 내용은 사전 조건 단원을 참조하십시오.

가용성 개선

가용성을 높이려면 다음 작업을 수행하는 것이 좋습니다.

- 기본 호스트 문제가 있는 경우 복구 및 장애 조치를 허용하도록 미션 크리티컬 애플리케이션에 충분한 추가 용량을 할당합니다. 이는 DB 서브넷 그룹의 서브넷을 포함하는 모든 Outposts에 적용됩니다. 자세한 내용은 AWS Outposts의 복원성을 참조하십시오.

- Outpost를 위한 중복 네트워크 연결을 제공합니다.
- 세 개 이상의 Outposts를 사용합니다. 세 개 이상의 Outposts를 사용하면 Amazon RDS는 DB 인스턴스를 복구할 수 있습니다. 현재 Outpost에 장애가 발생할 경우 RDS는 DB 인스턴스를 다른 Outpost로 이동시켜 복구합니다.
- Outpost를 위한 이중 전원 및 중복 네트워크 연결을 제공합니다.

로컬 네트워크에는 다음 사항을 권장합니다.

- 기본 DB 인스턴스를 호스팅하는 Outpost와 대기 DB 인스턴스를 호스팅하는 Outpost 간의 왕복 시간(RTT) 지연 시간은 쓰기 지연 시간에 직접적인 영향을 미칩니다. AWS Outposts 간의 RTT 대기 시간을 한 자릿수 밀리초로 낮게 유지하세요. 5밀리초를 넘지 않는 것이 좋지만 요구 사항은 다를 수 있습니다.

  `WriteLatency`에 대한 Amazon CloudWatch 지표에서 네트워크 지연 시간에 대한 순 영향을 찾을 수 있습니다. 자세한 내용은 Amazon RDS에 대한 Amazon CloudWatch 지표 단원을 참조하십시오.
- Outposts 간의 연결 가용성은 DB 인스턴스의 전체 가용성에 영향을 줍니다. Outposts 간에 중복 네트워크 연결이 있어야 합니다.

사전 조건

Outposts 기반 RDS의 다중 AZ 배포에는 다음과 같은 사전 요구 사항이 있습니다.

- 로컬 연결을 통해 연결되고 AWS 리전의 다른 가용 영역에 연결된 최소 2개의 Outpost가 있어야 합니다.
- DB 서브넷 그룹에 다음이 포함되어 있는지 확인합니다.
  - 지정된 AWS 리전의 2개 이상의 가용 영역에 있는 2개 이상의 서브넷
  - Outposts의 서브넷만 있어야 함
  - 동일한 Virtual Private Cloud(VPC) 내 최소 2개의 Outposts에 있는 최소 2개의 서브넷.
- DB 인스턴스의 VPC를 모든 로컬 게이트웨이 라우팅 테이블과 연결 복제가 Outposts의 로컬 게이트웨이를 사용하여 로컬 네트워크를 통해 실행되기 때문에 이 연결이 필요합니다.

  예를 들어 VPC에 Outpost-A의 서브넷 A와 Outpost-B의 서브넷 B가 포함되어 있다고 가정합니다. Outpost-A는 LocalGateway-A(LGW-A)를 사용하고 Outpost-B는 LocalGateway-B(LGW-B)를 사용합니다. LGW-A에는 RouteTable-A가 있고 LGW-B에는 RouteTable-B가 있습니다. 복제 트래픽에 RouteTable-A와 RouteTable-B를 모두 사용하려고 합니다. 모두 사용하려면 VPC를 RouteTable-A 및 RouteTable-B와 연결합니다.

연결을 생성하는 방법에 대한 자세한 내용은 Amazon EC2 create-local-gateway-route-table-vpc-association AWS CLI 명령을 참조하세요.

- Outposts에서 고객 소유 IP(CoIP) 라우팅을 사용하는지 확인합니다. 각 라우팅 테이블에는 각각 하나 이상의 주소 풀이 있어야 합니다. Amazon RDS는 데이터 동기화를 위해 기본 및 대기상태의 DB 인스턴스에 각각 추가 IP 주소를 할당합니다.
- RDS DB 인스턴스를 소유한 AWS 계정이 로컬 게이트웨이 라우팅 테이블 및 CoIP 풀을 소유하는지 확인합니다. 또는 로컬 게이트웨이 라우팅 테이블 및 CoIP 풀에 대한 액세스 권한이 있는 Resource Access Manager 공유의 일부인지 확인합니다.
- CoIP 풀의 IP 주소가 한 Outpost 로컬 게이트웨이에서 다른 게이트웨이로 라우팅될 수 있는지 확인합니다.
- VPC의 CIDR 블록(예: 10.0.0.0/4) 및 CoIP 풀 CIDR 블록에 클래스 E(240.0.0.0/4)의 IP 주소가 포함되어 있지 않은지 확인합니다. RDS는 이러한 IP 주소를 내부적으로 사용합니다.
- 아웃바운드 및 관련 인바운드 트래픽을 올바르게 설정했는지 확인합니다.

  Outposts 기반 RDS는 기본 DB 인스턴스와 대기 DB 인스턴스 간에 가상 사설 네트워크(VPN) 연결을 설정합니다. 이 기능이 올바르게 작동하려면 로컬 네트워크에서 인터넷 보안 연결 및 키 관리 프로토콜(ISAKMP)에 대한 아웃바운드 및 관련 인바운드 트래픽을 허용해야 합니다. UDP(User Datagram Protocol) 포트 500과 UDP 포트 4500을 사용하는 IP 보안(IPsec) NAT-T(Network Address Translation Traversal)를 사용하여 수행합니다.

CoIP에 대한 자세한 내용은 이 가이드의 Outposts의 RDS에 대한 고객 소유 IP 주소 섹션 및 AWS Outposts사용 설명서의 고객 소유 IP 주소를 참조하세요.

Amazon EC2 권한에 대한 API 작업 수행

AWS Outposts에서 DB 인스턴스에 CoIP를 사용하는지 여부에 관계없이 RDS는 CoIP 풀 리소스에 액세스해야 합니다. RDS는 다중 AZ 배포를 위해 사용자를 대신하여 CoIP에 대해 다음 EC2 권한 API 작업을 호출할 수 있습니다.

- `CreateCoipPoolPermission` - Outpost의 RDS에서 다중 AZ DB 인스턴스를 생성하는 경우
- `DeleteCoipPoolPermission` - Outpost의 RDS에서 다중 AZ DB 인스턴스를 삭제하는 경우

이러한 API 작업은 권한으로 지정된 CoIP 풀에서 탄력적 IP 주소를 할당할 수 있는 권한을 내부 RDS 계정에 부여하거나 제거합니다. `DescribeCoipPoolUsage` API 작업을 사용하여 이러한 IP 주소를

볼 수 있습니다. CoIP에 대한 자세한 내용은 AWS Outposts 사용 설명서의 Outposts의 RDS에 대한 고객 소유 IP 주소 및 고객 소유 IP 주소를 참조하세요.

RDS는 다중 AZ 배포를 위해 사용자를 대신하여 로컬 게이트웨이 라우팅 테이블에 대해 다음 EC2 권한 API 작업을 호출할 수도 있습니다.

- `CreateLocalGatewayRouteTablePermission` - Outpost의 RDS에서 다중 AZ DB 인스턴스를 생성하는 경우
- `DeleteLocalGatewayRouteTablePermission` - Outpost의 RDS에서 다중 AZ DB 인스턴스를 삭제하는 경우

이러한 API 작업은 내부 RDS VPC를 로컬 게이트웨이 라우팅 테이블과 연결할 수 있는 권한을 내부 RDS 계정에 부여하거나 제거합니다. API 작업을 사용하여 이러한 라우팅 테이블–VPC 연결을 볼 수 있습니다.

Amazon RDS on AWS Outposts DB 인스턴스 생성

Amazon RDS on AWS Outposts DB 인스턴스 생성은 AWS 클라우드에 Amazon RDS DB 인스턴스를 생성하는 것과 비슷합니다. 그러나 Outpost와 연결된 DB 서브넷 그룹을 지정해야 합니다.

Amazon VPC 서비스를 기반으로 하는 Virtual Private Cloud(VPC)는 AWS 리전의 모든 가용 영역에 걸쳐 있을 수 있습니다. Outpost 서브넷을 추가하여 AWS 리전의 모든 VPC를 Outpost로 확장할 수 있습니다. VPC에 Outpost 서브넷을 추가하려면 서브넷을 생성할 때 Outpost의 Amazon 리소스 이름(ARN)을 지정합니다.

Outposts의 RDS DB 인스턴스를 생성하기 전에 Outpost와 연결된 하나의 서브넷을 포함하는 DB 서브넷 그룹을 생성할 수 있습니다. Outposts의 RDS DB 인스턴스를 생성할 때 이 DB 서브넷 그룹을 지정하십시오. DB 인스턴스를 생성할 때 새 DB 서브넷 그룹을 생성하도록 선택할 수도 있습니다.

AWS Outposts 구성에 대한 자세한 내용은 AWS Outposts 사용 설명서를 참조하십시오.

콘솔

DB 서브넷 그룹 생성

Outpost와 연결된 하나의 서브넷으로 DB 서브넷 그룹을 만듭니다.

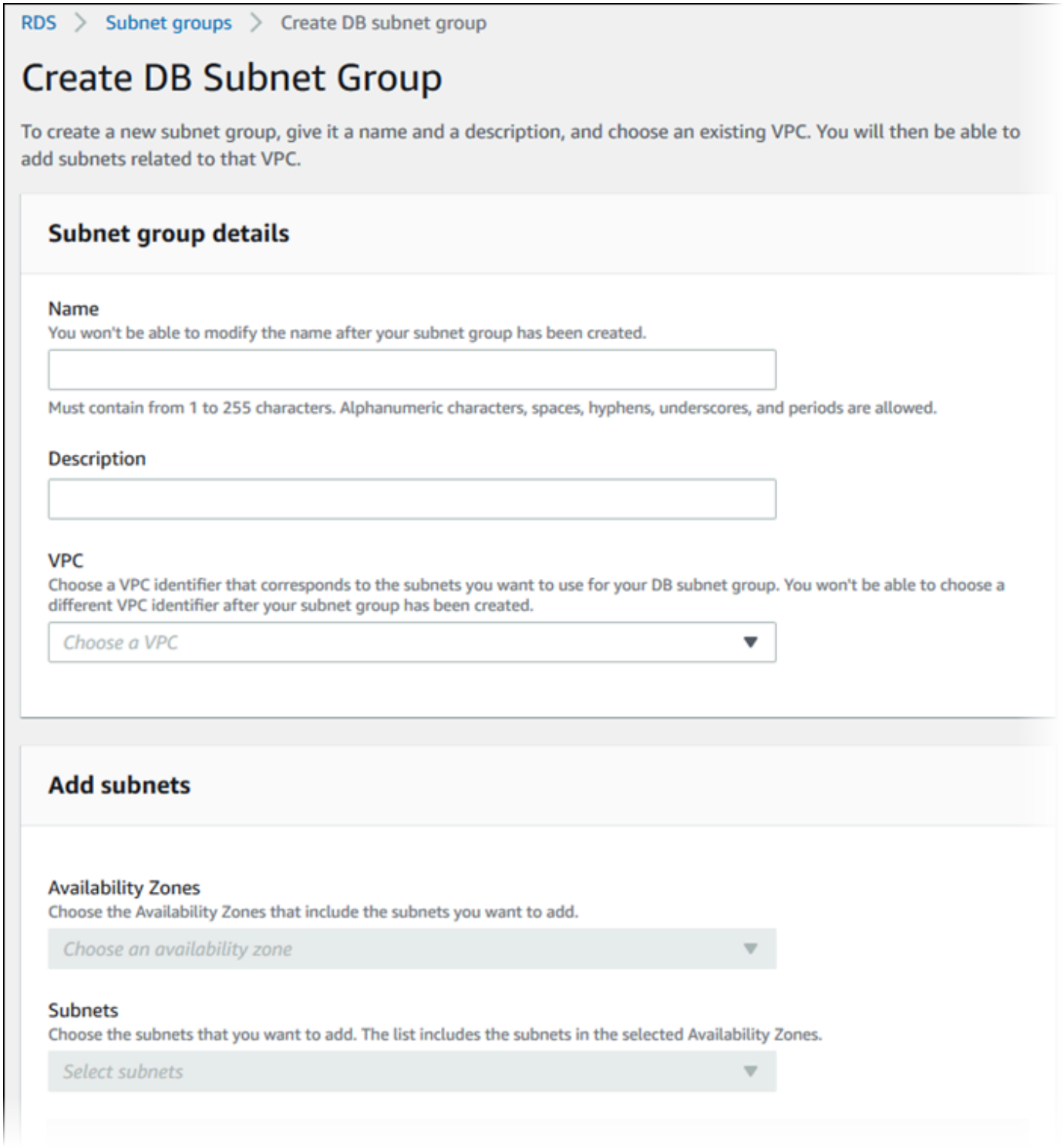
DB 인스턴스를 생성할 때 Outpost를 위한 새 DB 서브넷 그룹을 생성할 수도 있습니다. 이 경우 이 절차를 건너뜁니다.

> **Note**
>
> AWS 클라우드에 대한 DB 서브넷 그룹을 생성하려면 둘 이상의 서브넷을 지정하면 됩니다.

Outpost에 대한 DB 서브넷 그룹을 생성하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 콘솔의 오른쪽 상단에서 DB 서브넷 그룹을 생성하려는 AWS 리전을 선택합니다.
3. 서브넷 그룹을 선택한 후 DB 서브넷 그룹 생성을 선택합니다.

   [DB 서브넷 그룹 생성(Create DB subnet group)] 페이지가 나타납니다.



4. 이름(Name)의 경우 DB 서브넷 그룹의 이름을 선택합니다.
5. 설명(Description)의 경우 DB 서브넷 그룹에 대한 설명을 선택합니다.
6. VPC의 경우 DB 서브넷 그룹을 생성할 VPC를 선택합니다.
7. 가용 영역에서 Outpost의 가용 영역을 선택합니다.

8. 서브넷에서 Outposts의 RDS가 사용할 서브넷을 선택합니다.
9. 생성을 선택하여 DB 서브넷 그룹을 생성합니다.

Outposts 기반 RDS DB 인스턴스 생성

DB 인스턴스를 생성하고 DB 인스턴스의 Outpost를 선택합니다.

콘솔을 사용하여 Outposts의 RDS DB 인스턴스를 생성하려면

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 콘솔의 오른쪽 상단에서 DB 인스턴스를 생성하려는 AWS 리전을 선택합니다.
3. 탐색 창에서 데이터베이스를 선택합니다.
4. 데이터베이스 생성을 선택합니다.

   AWS Management Console은 사용자가 구성한 사용 가능한 Outposts를 탐지하고 데이터베이스 위치 섹션에 온프레미스 옵션을 제시합니다.

   > **Note**
   >
   > Outpost를 구성하지 않은 경우 데이터베이스 위치 섹션이 나타나지 않거나 온프레미스 생성 방법 선택 섹션에서 Outposts의 RDS 옵션을 사용할 수 없습니다.

5. 데이터베이스 위치(Database location)의 경우 온프레미스(On-premises)를 선택합니다.
6. 온프레미스 생성 방법(On-premises creation method)의 경우 Outposts 기반 RDS(RDS on Outposts)를 선택합니다.
7. Outposts 연결(Outposts Connectivity)에 대한 설정을 지정합니다. 이러한 설정은 DB 인스턴스에 대한 DB 서브넷 그룹이 있는 VPC를 사용하는 Outpost에 해당합니다. 여기서 VPC는 Amazon VPC 서비스를 기반으로 해야 합니다.

   a. Virtual Private Cloud(VPC)의 경우 DB 인스턴스의 DB 서브넷 그룹이 포함된 VPC를 선택합니다.
   b. VPC 보안 그룹의 경우 DB 인스턴스에 대한 Amazon VPC 보안 그룹을 선택합니다.
   c. DB 서브넷 그룹의 경우 DB 인스턴스에 대한 DB 서브넷 그룹을 선택합니다.

      예를 들어, DB 서브넷 그룹 생성에 나와 있는 절차를 수행한 경우 Outpost와 연결된 기존 DB 서브넷 그룹을 선택할 수 있습니다.

Outpost용 새 DB 서브넷 그룹을 생성할 수도 있습니다.

8. 다중 AZ 배포의 경우 대기 인스턴스 생성(프로덕션 사용에 권장)을 선택하여 다른 Outpost에 대기 DB 인스턴스를 생성합니다.

> **Note**
>
> Microsoft SQL 서버에는 이 옵션을 사용할 수 없습니다.
> 다중 AZ 배포를 생성하도록 선택하면 Outpost에 백업을 저장할 수 없습니다.

9. 백업(Backup)에서 다음을 수행합니다.

   a. 백업 대상으로 다음 중 하나를 선택합니다.

   - AWS 클라우드 - 자동 백업 및 수동 스냅샷을 상위 AWS 리전에 저장하려는 경우
   - Outposts(온프레미스) - 로컬 백업을 생성하려는 경우

> **Note**
>
> Outpost에 백업을 저장하려면 Outpost에 Amazon S3 기능이 있어야 합니다. 자세한 내용은 Outposts 기반 Amazon S3를 참조하세요.
> 다중 AZ 배포 또는 읽기 전용 복제본에는 로컬 백업이 지원되지 않습니다.

   b. DB 인스턴스의 특정 시점 스냅샷을 생성하려면 자동 백업 활성화(Enable automated backups)를 선택합니다.

   자동 백업을 켜면 백업 보존 기간(Backup retention period) 및 백업 기간(Backup window)을 선택하거나 기본값을 그대로 사용할 수 있습니다.

10. 필요에 따라 다른 DB 인스턴스 설정을 지정합니다.

    DB 인스턴스 생성 시 각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하십시오.

11. 데이터베이스 생성을 선택합니다.

    데이터베이스 페이지가 표시됩니다. 배너에서 DB 인스턴스가 생성 중임을 나타내며, 자격 증명 세부 정보 보기(View credential details) 버튼이 표시됩니다.

DB 인스턴스 세부 정보 보기

DB 인스턴스를 생성한 후 자격 증명 및 기타 세부 정보를 볼 수 있습니다.

DB 인스턴스 세부 정보를 확인하는 방법

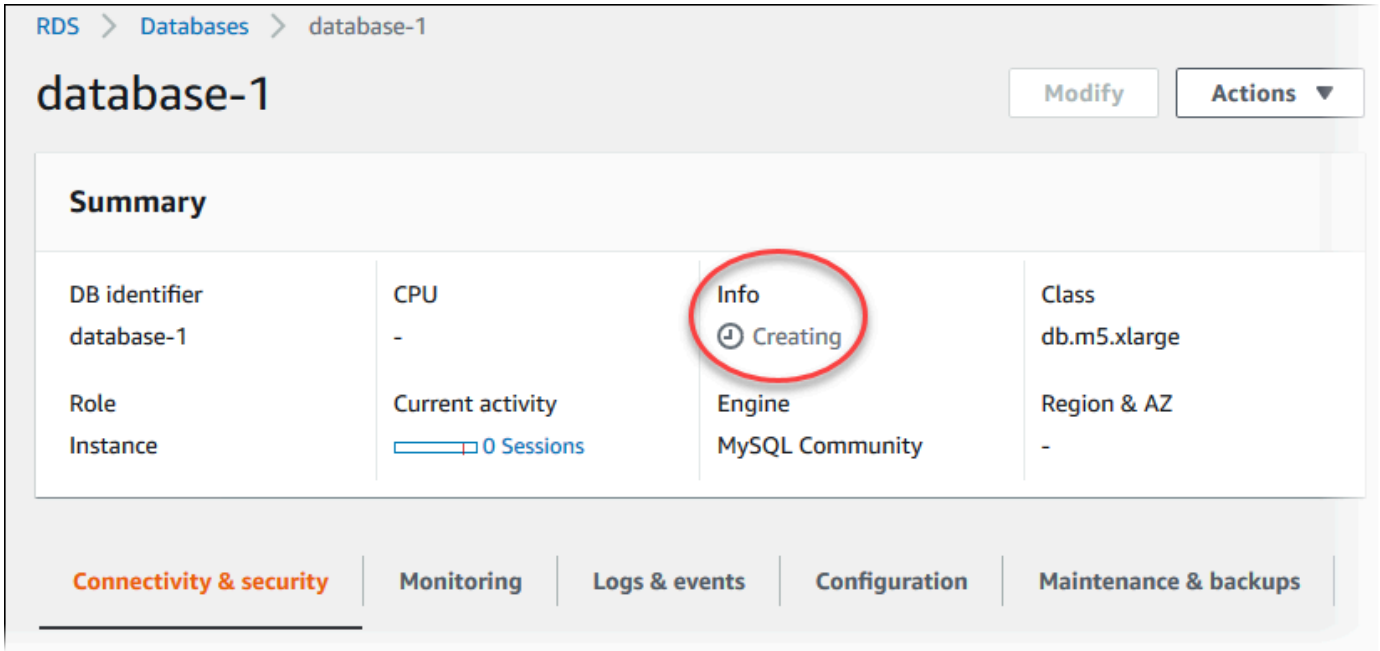
1. DB 인스턴스의 마스터 사용자 이름 및 암호를 보려면 데이터베이스 페이지에서 자격 증명 세부 정보 보기(View credential details)를 선택합니다.

   이 자격 증명을 사용하여 DB 인스턴스에 마스터 사용자로 연결할 수 있습니다.

   > **Important**
   >
   > 마스터 사용자 암호를 다시 볼 수는 없습니다. 따라서 기록을 해두지 않으면 이를 변경해야 합니다. DB 인스턴스가 사용 가능한 상태가 되고 난 후에 마스터 사용자 암호를 변경하려면 DB 인스턴스를 수정합니다. DB 인스턴스 수정에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

2. 데이터베이스 페이지에서 새 DB 인스턴스의 이름을 선택합니다.

   RDS 콘솔에 새 DB 인스턴스의 세부 정보가 표시됩니다. DB 인스턴스를 만들고 사용할 준비가 될 때까지 DB 인스턴스의 상태는 Creating입니다. 상태가 사용 가능으로 변경되면 DB 인스턴스에 연결할 수 있습니다. 할당된 DB 인스턴스 클래스와 스토리지에 따라 새 DB 인스턴스를 사용할 수 있을 때까지 몇 분 정도 걸릴 수 있습니다.



DB 인스턴스를 사용할 수 있게 되면 AWS 클라우드에서 RDS DB 인스턴스를 관리하는 것과 동일한 방식으로 관리할 수 있습니다.

AWS CLI

AWS CLI를 사용하여 Outpost에 새 DB 인스턴스를 생성하기 전에 Outposts 기반 RDS에서 사용할 DB 서브넷 그룹을 먼저 생성합니다.

Outpost에 대한 DB 서브넷 그룹을 생성하는 방법

- create-db-subnet-group 명령을 사용합니다. `--subnet-ids`의 경우 Outpost에 Outposts의 RDS에서 사용할 서브넷 그룹을 지정합니다,

  대상 LinuxmacOS, 또는Unix:

  ```
  aws rds create-db-subnet-group \
      --db-subnet-group-name myoutpostdbsubnetgr \
      --db-subnet-group-description "DB subnet group for RDS on Outposts" \
      --subnet-ids subnet-abc123
  ```

  Windows의 경우:

  ```
  aws rds create-db-subnet-group ^
      --db-subnet-group-name myoutpostdbsubnetgr ^
      --db-subnet-group-description "DB subnet group for RDS on Outposts" ^
      --subnet-ids subnet-abc123
  ```

AWS CLI를 사용하여 Outposts 기반 RDS DB 인스턴스를 생성하는 방법

- create-db-instance 명령을 사용합니다. Outpost의 가용 영역, Outpost와 연결된 Amazon VPC 보안 그룹 및 Outpost에 대해 생성한 DB 서브넷 그룹을 지정합니다. 다음 옵션을 포함할 수 있습니다.

  - `--db-instance-identifier`
  - `--db-instance-class`
  - `--engine` - 데이터베이스 엔진입니다. 다음 값 중 하나를 사용합니다.
    - MySQL - `mysql`을 지정합니다.

  - PostgreSQL - `postgres`를 지정합니다.
  - Microsoft SQL Server - `sqlserver-ee`, `sqlserver-se` 또는 `sqlserver-web`을 지정합니다.
- `--availability-zone`
- `--vpc-security-group-ids`
- `--db-subnet-group-name`
- `--allocated-storage`
- `--max-allocated-storage`
- `--master-username`
- `--master-user-password`
- `--multi-az | --no-multi-az` - (선택 사항) 다른 가용 영역에 대기 인스턴스를 생성할지 여부 기본값은 `--no-multi-az`입니다.

  SQL Server에는 `--multi-az` 옵션을 사용할 수 없습니다.
- `--backup-retention-period`
- `--backup-target` - (선택 사항) 자동 백업 및 수동 스냅샷을 저장할 위치입니다. 다음 값 중 하나를 사용합니다.
  - `outposts` - Outpost에 로컬로 저장합니다.
  - `region` - 상위 AWS 리전 리전에 저장합니다. 이것이 기본값입니다.

  `--multi-az` 옵션을 사용하면 `--backup-target`에 `outposts`를 사용할 수 없습니다. 또한 `--backup-target`에 `outposts`를 사용하는 경우 DB 인스턴스에는 읽기 전용 복제본이 있을 수 없습니다.
- `--storage-encrypted`
- `--kms-key-id`

Example

다음 예제에서는 Outpost에 저장된 백업과 함께 `myoutpostdbinstance`라는 이름의 MySQL DB 인스턴스를 생성합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance \
    --db-instance-identifier myoutpostdbinstance \
```

```
    --engine-version 8.0.17 \
    --db-instance-class db.m5.large \
    --engine mysql \
    --availability-zone us-east-1d \
    --vpc-security-group-ids outpost-sg \
    --db-subnet-group-name myoutpostdbsubnetgr \
    --allocated-storage 100 \
    --max-allocated-storage 1000 \
    --master-username masterawsuser \
    --manage-master-user-password \
    --backup-retention-period 3 \
    --backup-target outposts \
    --storage-encrypted \
    --kms-key-id mykey
```

Windows의 경우:

```
aws rds create-db-instance ^
    --db-instance-identifier myoutpostdbinstance ^
    --engine-version 8.0.17 ^
    --db-instance-class db.m5.large ^
    --engine mysql ^
    --availability-zone us-east-1d ^
    --vpc-security-group-ids outpost-sg ^
    --db-subnet-group-name myoutpostdbsubnetgr ^
    --allocated-storage 100 ^
    --max-allocated-storage 1000 ^
    --master-username masterawsuser ^
    --manage-master-user-password ^
    --backup-retention-period 3 ^
    --backup-target outposts ^
    --storage-encrypted ^
    --kms-key-id mykey
```

DB 인스턴스 생성 시 각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하십시오.

RDS API

RDS API를 사용하여 Outpost에 새 DB 인스턴스를 생성하려면 먼저 CreateDBSubnetGroup 작업을 호출하여 Outposts의 RDS에서 사용할 DB 서브넷 그룹을 생성합니다. `SubnetIds`의 경우 Outpost에 Outposts의 RDS에서 사용할 서브넷 그룹을 지정합니다,

다음으로 아래 파라미터를 사용하여 CreateDBInstance 작업을 호출합니다. Outpost의 가용 영역, Outpost와 연결된 Amazon VPC 보안 그룹 및 Outpost에 대해 생성한 DB 서브넷 그룹을 지정합니다.

- `AllocatedStorage`
- `AvailabilityZone`
- `BackupRetentionPeriod`
- `BackupTarget`

  다중 AZ DB 인스턴스 배포를 생성할 경우에는 `BackupTarget`에 `outposts`를 사용할 수 없습니다. 또한 `BackupTarget`에 `outposts`를 사용하는 경우 DB 인스턴스에는 읽기 전용 복제본이 있을 수 없습니다.
- `DBInstanceClass`
- `DBInstanceIdentifier`
- `VpcSecurityGroupIds`
- `DBSubnetGroupName`
- `Engine`
- `EngineVersion`
- `MasterUsername`
- `MasterUserPassword`
- `MaxAllocatedStorage` (선택 사항)
- `MultiAZ`(선택 사항)
- `StorageEncrypted`
- `KmsKeyID`

DB 인스턴스 생성 시 각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하십시오.

AWS Outposts에서 Amazon RDS용 읽기 전용 복제본 생성

Amazon RDS on AWS Outposts는 MySQL 및 PostgreSQL DB 엔진의 기본 복제 기능을 사용하여 원본 DB 인스턴스의 읽기 전용 복제본을 생성합니다. 원본 DB 인스턴스가 기본 DB 인스턴스가 됩니다. 기본 DB 인스턴스에 적용된 업데이트는 읽기 전용 복제본에 비동기식으로 복사됩니다. 애플리케이션에서 읽기 전용 복제본으로 읽기 쿼리를 라우팅하여 기본 DB 인스턴스의 로드를 줄일 수 있습니다. 읽기 전용 복제본을 사용하면 읽기 중심의 데이터베이스 워크로드에 대한 단일 DB 인스턴스의 용량 제한에서 벗어나 탄력적으로 늘릴 수 있습니다.

RDS on Outposts DB 인스턴스에서 읽기 전용 복제본을 생성하면 읽기 전용 복제본은 고객 소유 IP 주소(CoIP)를 사용합니다. 자세한 내용은 Outposts의 RDS에 대한 고객 소유 IP 주소 단원을 참조하십시오.

RDS on Outposts 기반 읽기 전용 복제본의 제한 사항은 다음과 같습니다.

- RDS on Outposts DB 인스턴스에서 RDS for SQL Server 읽기 전용 복제본을 생성할 수 없습니다.
- RDS on Outposts에서는 리전 간 읽기 전용 복제본이 지원되지 않습니다.
- RDS on Outposts에서는 계단식 읽기 전용 복제본이 지원되지 않습니다.
- 소스 RDS on Outposts DB 인스턴스에는 로컬 백업이 있을 수 없습니다. 소스 DB 인스턴스의 백업 대상은 사용자의 AWS 리전이어야 합니다.
- 읽기 전용 복제본에는 고객 소유 IP(CoIP) 풀이 필요합니다. 자세한 내용은 Outposts의 RDS에 대한 고객 소유 IP 주소 단원을 참조하십시오.
- RDS on Outposts에 있는 읽기 전용 복제본은 소스 DB 인스턴스와 동일한 Virtual Private Cloud(VPC)에서만 생성할 수 있습니다.
- RDS on Outposts의 읽기 전용 복제본은 소스 DB 인스턴스와 동일한 VPC에 있는 동일한 Outpost 또는 다른 Outpost에 위치할 수 있습니다.
- AWS KMS External Key Store (XKS)로 암호화된 DB 인스턴스에 대한 읽기 전용 복제본을 생성할 수 없습니다.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 RDS on Outpost DB 인스턴스에서 읽기 전용 복제본을 생성할 수 있습니다. 읽기 전용 복제본에 대한 자세한 내용은 DB 인스턴스 읽기 전용 복제본 작업 섹션을 참조하세요.

콘솔

소스 DB 인스턴스에서 읽기 복제본을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 읽기 전용 복제본의 소스로 사용할 DB 인스턴스를 선택합니다.
4. 작업에서 읽기 전용 복제본 만들기를 선택합니다.
5. DB 인스턴스 식별자에 읽기 전용 복제본의 이름을 입력합니다.
6. Outposts 연결(Outposts Connectivity)에 대한 설정을 지정합니다. 이러한 설정은 DB 인스턴스에 대한 DB 서브넷 그룹이 있는 Virtual Private Cloud(VPC)를 사용하는 Outpost에 해당합니다. 여기서 VPC는 Amazon VPC 서비스를 기반으로 해야 합니다.
7. DB 인스턴스 클래스를 선택합니다. 읽기 전용 복제본의 소스 DB 인스턴스보다 크거나 같은 DB 인스턴스 클래스와 스토리지 유형을 사용하는 것이 좋습니다.
8. 다중 AZ 배포의 경우 대기 인스턴스 생성(프로덕션 사용에 권장)을 선택하여 다른 가용 영역에 대기 DB 인스턴스를 생성합니다.

   읽기 전용 복제본을 다중 AZ DB 인스턴스로 생성하는 작업은 원본 데이터베이스가 다중 AZ DB 인스턴스인지 여부와는 무관합니다.
9. (선택 사항) Connectivity(연결)에서 Subnet Group(서브넷 그룹) 및 Availability Zone(가용 영역) 값을 설정합니다.

   서브넷 그룹과 가용 영역 모두에 값을 지정하면 DB 서브넷 그룹의 가용 영역과 연결된 Outpost에 읽기 전용 복제본이 생성됩니다.

   서브넷 그룹 값을 지정하고 가용 영역에 기본 설정 없음을 지정하면 DB 서브넷 그룹의 임의 Outpost에 읽기 전용 복제본이 생성됩니다.
10. AWS KMS key의 경우 KMS 키의 AWS KMS key 식별자를 선택합니다.

    읽기 전용 복제본은 암호화해야 합니다.
11. 필요에 따라 다른 옵션을 선택합니다.
12. 읽기 전용 복제본 생성을 선택합니다.

읽기 전용 복제본이 생성되면 RDS 콘솔의 [데이터베이스(Databases)] 페이지에서 확인할 수 있습니다. [역할(Role)] 열에 [복제본(Replica)]이 표시됩니다.

AWS CLI

원본 MySQL 또는 PostgreSQL DB 인스턴스에서 읽기 복제본을 생성하려면 AWS CLI 명령 create-db-instance-read-replica를 사용합니다.

`--db-subnet-group-name` 및 `--availability-zone` 옵션을 지정하여 읽기 전용 복제본이 생성되는 위치를 제어할 수 있습니다.

- `--db-subnet-group-name` 및 `--availability-zone` 옵션을 모두 지정하면 DB 서브넷 그룹의 가용 영역과 연결된 Outpost에 읽기 전용 복제본이 생성됩니다.
- `--db-subnet-group-name` 옵션을 지정하고 `--availability-zone` 옵션을 지정하지 않으면 DB 서브넷 그룹의 임의 Outpost에 읽기 전용 복제본이 생성됩니다.
- 두 옵션 모두 지정하지 않으면 소스 RDS on Outposts DB 인스턴스와 동일한 Outpost에 읽기 전용 복제본이 생성됩니다.

다음 예제에서는 복제본을 생성하고 `--db-subnet-group-name` 및 `--availability-zone` 옵션을 포함하여 읽기 전용 복제본의 위치를 지정합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance-read-replica \
    --db-instance-identifier myreadreplica \
    --source-db-instance-identifier mydbinstance \
    --availability-zone us-west-2a
```

Windows의 경우:

```
aws rds create-db-instance-read-replica ^
    --db-instance-identifier myreadreplica ^
    --source-db-instance-identifier mydbinstance ^
    --availability-zone us-west-2a
```

RDS API

소스 MySQL 또는 PostgreSQL DB 인스턴스에서 읽기 전용 복제본을 생성하려면 다음 필수 파라미터를 사용하여 Amazon RDS API CreateDBInstanceReadReplica 작업을 호출합니다.

- `DBInstanceIdentifier`
- `SourceDBInstanceIdentifier`

`DBSubnetGroupName` 및 `AvailabilityZone` 파라미터를 지정하여 읽기 전용 복제본이 생성되는 위치를 제어할 수 있습니다.

- `DBSubnetGroupName` 및 `AvailabilityZone` 파라미터를 모두 지정하면 DB 서브넷 그룹의 가용 영역과 연결된 Outpost에 읽기 전용 복제본이 생성됩니다.
- `DBSubnetGroupName` 파라미터를 지정하고 `AvailabilityZone` 파라미터를 지정하지 않으면 DB 서브넷 그룹의 임의 Outpost에 읽기 전용 복제본이 생성됩니다.
- 두 파라미터 모두 지정하지 않으면 소스 RDS on Outposts DB 인스턴스와 동일한 Outpost에 읽기 전용 복제본이 생성됩니다.

AWS Outposts의 Amazon RDS에서 DB 인스턴스 복원 시 고려 사항

AWS Outposts의 Amazon RDS에서 DB 인스턴스를 복원할 때 일반적으로 복원된 DB 인스턴스의 자동 백업 및 수동 스냅샷을 위한 저장 위치를 선택할 수 있습니다.

- 수동 DB 스냅샷에서 복원할 때 백업을 상위 AWS 리전 또는 Outpost에 로컬로 저장할 수 있습니다.
- 자동 백업(특정 시점으로 복구)에서 복원할 때 선택할 수 있는 옵션은 다음과 같습니다.
  - 상위 AWS 리전에서 복원하는 경우 AWS 리전 리전 또는 Outpost에 백업을 저장할 수 있습니다.
  - Outpost에서 복원할 경우 Outpost에만 백업을 저장할 수 있습니다.

Amazon RDS Proxy

Amazon RDS 프록시를 사용하면 애플리케이션이 데이터베이스 연결을 풀링하고 공유하도록 허용하여 확장 기능을 향상할 수 있습니다. RDS 프록시는 애플리케이션 연결을 유지하면서 예비 DB 인스턴스에 자동으로 연결하여 데이터베이스 장애에 대한 애플리케이션의 복원력을 높입니다. RDS 프록시를 사용하면 데이터베이스에 대해 AWS Identity and Access Management(IAM) 인증을 사용하고 AWS Secrets Manager에 자격 증명을 안전하게 저장합니다.

RDS 프록시를 사용하여 예기치 않은 데이터베이스 트래픽 급증을 처리할 수 있습니다. 급증을 처리하지 않으면 이러한 현상으로 인해 연결을 초과 구독하거나 새 연결이 빠른 속도로 생성되어 문제가 발생할 수 있습니다. RDS 프록시는 데이터베이스 연결 풀을 설정하고 이 풀에서 연결을 재사용합니다. 이 접근 방식은 매번 새 데이터베이스 연결을 여는 데서 오는 메모리 및 CPU 오버헤드 를 방지합니다. 과다 구독으로부터 데이터베이스를 보호하기 위해 생성되는 데이터베이스 연결 수를 제어할 수 있습니다.

RDS 프록시는 연결 풀에서 즉시 제공할 수 없는 애플리케이션 연결을 대기열에 추가하거나 제한합니다. 대기 시간이 증가할 수 있지만 애플리케이션은 갑작스러운 데이터베이스 장애 또는 압도 없이 계속 확장될 수 있습니다. 연결 요청이 지정된 한도를 초과하는 경우 RDS Proxy는 애플리케이션 연결을 거부합니다(즉, 부하 감소). 동시에 RDS가 사용 가능한 용량으로 제공할 수 있는 부하에 대해 예측 가능한 성능을 유지합니다.

자격 증명을 처리하고 각 새 연결에 대한 보안 연결을 설정하는 데 필요한 오버헤드를 줄일 수 있습니다. RDS 프록시는 데이터베이스를 대신하여 해당 작업 중 일부를 처리할 수 있습니다.

RDS 프록시는 지원하는 엔진 버전과 완전히 호환됩니다. 코드 변경 없이 대부분의 애플리케이션에 RDS 프록시를 활성화할 수 있습니다.

주제

- 리전 및 버전 사용 가능 여부
- RDS 프록시의 할당량 및 제한 사항
- RDS Proxy 사용 대상 계획
- RDS Proxy 개념 및 용어
- RDS 프록시 시작하기
- RDS 프록시 관리
- Amazon RDS 프록시 엔드포인트 작업

- Amazon CloudWatch를 사용한 RDS 프록시 지표 모니터링
- RDS 프록시 이벤트 작업
- RDS 프록시 문제 해결
- RDS Proxy를 AWS CloudFormation에서 사용

리전 및 버전 사용 가능 여부

기능 가용성 및 해당 지원은 각 데이터베이스 엔진의 특정 버전 및 AWS 리전에 따라 다릅니다. RDS Proxy에서 사용할 수 있는 Amazon RDS의 버전 및 지역에 대한 자세한 내용은 Amazon RDS 프록시를 지원하는 리전 및 DB 엔진 단원을 참조하십시오.

RDS 프록시의 할당량 및 제한 사항

RDS Proxy에는 다음과 같은 제한 사항이 적용됩니다.

- 각 AWS 계정 ID는 프록시 20개로 제한됩니다. 애플리케이션에 더 많은 프록시가 필요한 경우 AWS Management Console의 Service Quotas 페이지에서 증량을 요청하세요. Service Quotas 페이지에서 Amazon Relational Database Service(Amazon RDS)를 선택하고 프록시를 찾아 할당량 증량을 요청합니다. AWS는 자동으로 할당량을 늘리거나 지원의 요청에 대한 검토를 보류할 수 있습니다.
- 각 프록시는 최대 200개의 연결된 Secrets Manager 암호를 보유할 수 있습니다. 따라서 각 프록시는 지정된 시간에 최대 200개의 서로 다른 사용자 계정으로 연결할 수 있습니다.
- 각 프록시에는 기본 엔드포인트가 있습니다. 또한 각 프록시에 최대 20개의 프록시 엔드포인트를 추가할 수 있습니다. 이러한 엔드포인트를 생성, 조회, 수정 및 삭제할 수 있습니다.
- 복제 구성의 RDS DB 인스턴스의 경우, 프록시는 읽기 전용 복제본이 아닌 라이터 DB 인스턴스에만 연결할 수 있습니다.
- RDS Proxy는 데이터베이스와 동일한 Virtual Private Cloud(VPC)에 있어야 합니다. 데이터베이스에는 공개적으로 액세스할 수는 있지만 프록시에는 공개적으로 액세스할 수 없습니다. 예를 들어, 로컬 호스트에서 데이터베이스의 프로토타입을 생성하는 경우 프록시에 연결하는 데 필요한 네트워크 요구 사항을 설정하지 않는 한 프록시에 연결할 수 없습니다. 이는 로컬 호스트가 프록시의 VPC 외부에 있기 때문입니다.
- 테넌시가 `dedicated`로 설정된 VPC에서는 RDS 프록시를 사용할 수 없습니다.
- IAM 인증이 활성화된 RDS DB 인스턴스와 함께 RDS 프록시를 사용하는 경우 사용자 인증을 확인합니다. 프록시를 통해 연결하는 모든 사용자가 로그인 보안 인증 정보를 통해 인증되어야 합니다. Secrets Manager 및 RDS 프록시의 IAM 지원에 대한 자세한 내용은 AWS Secrets Manager에

서 RDS 프록시용 데이터베이스 자격 증명 설정 및 RDS 프록시에 대한 AWS Identity and Access Management(IAM) 정책 설정 섹션을 참조하세요.

- SSL 호스트 이름 검증을 사용할 경우, 사용자 지정 DNS와 함께 RDS 프록시를 사용할 수 없습니다.
- 각 프록시는 단일 대상 DB 인스턴스 와 연결될 수 있습니다. 그러나 여러 프록시를 동일한 DB 인스턴스 와 연결할 수 있습니다.
- 텍스트 크기가 16KB보다 큰 문을 사용하면 프록시가 세션을 현재 연결에 고정합니다.
- 특정 리전에는 프록시를 만들 때 고려해야 하는 가용 영역(AZ) 제한이 있습니다. 미국 동부(버지니아 북부) 리전은 `use1-az3` 가용 영역에서 RDS 프록시를 지원하지 않습니다. 미국 서부(캘리포니아 북부) 리전은 `usw1-az2` 가용 영역에서 RDS 프록시를 지원하지 않습니다. 프록시 생성 시 서브넷을 선택할 때는 위에서 언급한 가용 영역에서 서브넷을 선택하지 않도록 하세요.
- 현재 RDS 프록시는 모든 전역 조건 컨텍스트 키를 지원하지 않습니다.

  전역 조건 컨텍스트 키에 대한 자세한 내용은 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.
- RDS Custom for SQL Server에는 RDS 프록시를 사용할 수 없습니다.
- 프록시에 데이터베이스 파라미터 그룹 수정을 반영하려면 변경 사항을 즉시 적용하기로 선택한 경우에도 인스턴스 재부팅이 필요합니다. 클러스터 수준 파라미터의 경우 클러스터 전체 재부팅이 필요합니다.
- 프록시 대상을 등록하면 프록시가 `rdsproxyadmin` DB 사용자를 자동으로 만듭니다. 프록시 기능에 필수적인 보호 대상 사용자입니다. 어떤 방식으로든 `rdsproxyadmin` 사용자의 변조를 방지해야 합니다. `rdsproxyadmin` 사용자 또는 해당 권한을 삭제하거나 수정하면 애플리케이션에서 프록시를 전혀 사용하지 못하게 될 수 있습니다.

각 DB 엔진의 추가 제한 사항은 다음 섹션을 참조하세요.

- RDS for MariaDB 추가 제한 사항
- RDS for Microsoft SQL Server 추가 제한 사항
- RDS for MySQL 추가 제한 사항
- RDS for PostgreSQL 추가 제한 사항

RDS for MariaDB 추가 제한 사항

RDS for MariaDB 데이터베이스를 사용하는 RDS 프록시에는 다음과 같은 추가 제한 사항이 적용됩니다.

- 현재 모든 프록시는 MariaDB에 대한 포트 3306에서 수신합니다. 프록시는 여전히 데이터베이스 설정에서 지정한 포트를 사용하여 데이터베이스에 연결됩니다.
- Amazon EC2 인스턴스에서 실행되는 자체 관리형 MariaDB 데이터베이스에는 RDS 프록시를 사용할 수 없습니다.
- DB 파라미터 그룹의 `read_only` 파라미터가 1로 설정된 RDS for MariaDB DB 인스턴스에는 RDS 프록시를 사용할 수 없습니다.
- RDS 프록시는 MariaDB 압축 모드를 지원하지 않습니다. 예를 들어 `mysql` 명령의 `--compress` 또는 `-C` 옵션에서 사용하는 압축을 지원하지 않습니다.
- 일부 SQL 문 및 함수는 고정을 야기하지 않고 연결 상태를 변경할 수 있습니다. 최신 고정 동작은 RDS 프록시 고정 방지 단원을 참조하십시오.
- RDS 프록시는 MariaDB `auth_ed25519` 플러그인을 지원하지 않습니다.
- RDS 프록시는 MariaDB 데이터베이스용 전송 계층 보안(TLS) 버전 1.3을 지원하지 않습니다.
- RDS 프록시가 동일한 데이터베이스 연결을 재사용하여 다른 쿼리를 실행하면 `GET DIAGNOSTIC` 명령을 처리하는 데이터베이스 연결에서 부정확한 정보가 반환될 수 있습니다. 이는 RDS 프록시가 데이터베이스 연결을 멀티플렉싱할 때 발생할 수 있습니다. 자세한 내용은 RDS Proxy 개념 개요 섹션을 참조하세요.
- RDS Proxy는 현재 MariaDB용 `ClientPasswordAuthType`에 대해 `caching_sha2_password` 옵션을 지원하지 않습니다.

> **Important**
>
> MariaDB 데이터베이스와 관련된 프록시의 경우 초기화 쿼리에서 구성 파라미터 `sql_auto_is_null`을 `true`로 설정하거나 0이 아닌 값으로 설정하지 마시기 바랍니다. 이렇게 하면 잘못된 애플리케이션 동작이 발생할 수 있습니다.

RDS for Microsoft SQL Server 추가 제한 사항

RDS for Microsoft SQL Server 데이터베이스를 사용하는 RDS 프록시에는 다음과 같은 추가 제한 사항이 적용됩니다.

- 프록시에 대해 만들어야 하는 Secrets Manager 암호의 수는 DB 인스턴스에서 사용하는 콜레이션에 따라 달라집니다. 예를 들어 DB 인스턴스에서 대/소문자를 구분하는 콜레이션을 사용한다고 가정해 보겠습니다. 애플리케이션에서 'Admin'과 'admin'을 모두 허용하는 경우 프록시에는 두 개의 개별 암

호가 필요합니다. SQL Server의 콜레이션에 대한 자세한 내용은 Microsoft SQL Server 설명서를 참조하세요.

- RDS 프록시는 Active Directory를 사용하는 연결을 지원하지 않습니다.
- 토큰 속성을 지원하지 않는 클라이언트에는 IAM 인증을 사용할 수 없습니다. 자세한 내용은 Microsoft SQL Server를 사용하여 프록시에 연결할 때 고려할 사항 섹션을 참조하세요.
- `@@IDENTITY`, `@@ROWCOUNT` 및 `SCOPE_IDENTITY`의 결과가 항상 정확하지는 않습니다. 차선책은 동일한 세션 문에서 해당 값을 검색하여 올바른 정보를 반환하는지 확인하는 것입니다.
- 연결이 여러 개의 활성 결과 집합(MARS)을 사용하는 경우 RDS 프록시는 초기화 쿼리를 실행하지 않습니다. 사용자 이름에 대한 자세한 정보는 Microsoft SQL 설명서를 참조하세요.
- 현재 RDS 프록시는 메이저 버전 SQL Server 2022에서 실행되는 RDS for SQL Server DB 인스턴스를 지원하지 않습니다.
- RDS 프록시는 메이저 버전 SQL Server 2014에서 실행되는 RDS for SQL Server DB 인스턴스를 지원하지 않습니다.
- RDS 프록시는 하나의 TLS 레코드에서 여러 응답 메시지를 처리할 수 없는 클라이언트 애플리케이션을 지원하지 않습니다.

RDS for MySQL 추가 제한 사항

RDS for MySQL 데이터베이스를 사용하는 RDS 프록시에는 다음과 같은 추가 제한 사항이 적용됩니다.

- `caching_sha2_password` 인증에 대한 RDS Proxy 지원에는 보안(TLS) 연결이 필요합니다.
- `caching_sha2_password`에 대한 RDS Proxy 지원에는 특정 go-sql 드라이버 버전과의 호환성 문제가 있는 것으로 알려져 있습니다.
- MySQL 8.4 C 드라이버를 사용할 때 파라미터 수가 준비된 문에서 자리 표시자 수를 초과하는 경우 `mysql_stmt_bind_named_param` API가 잘못된 형식의 패킷을 형성할 수 있습니다. 이로 인해 잘못된 응답이 발생합니다. 자세한 내용은 MySQL bug report를 참조하세요.
- 현재 모든 프록시는 MySQL에 대한 포트 3306에서 수신합니다. 프록시는 여전히 데이터베이스 설정에서 지정한 포트를 사용하여 데이터베이스에 연결됩니다.
- EC2 인스턴스에서 실행되는 자체 관리형 MySQL 데이터베이스에는 RDS 프록시를 사용할 수 없습니다.
- DB 파라미터 그룹의 `read_only` 파라미터가 1로 설정된 MySQL DB 인스턴스에는 RDS 프록시를 사용할 수 없습니다.

- RDS 프록시는 MySQL 압축 모드를 지원하지 않습니다. 예를 들어 `mysql` 명령의 `--compress` 또는 `-C` 옵션에서 사용하는 압축을 지원하지 않습니다.
- RDS 프록시가 동일한 데이터베이스 연결을 재사용하여 다른 쿼리를 실행하면 `GET DIAGNOSTIC` 명령을 처리하는 데이터베이스 연결에서 부정확한 정보가 반환될 수 있습니다. 이는 RDS 프록시가 데이터베이스 연결을 멀티플렉싱할 때 발생할 수 있습니다.
- `SET LOCAL` 같은 일부 SQL 문 및 함수는 고정을 야기하지 않고 연결 상태를 변경할 수 있습니다. 최신 고정 동작은 RDS 프록시 고정 방지 단원을 참조하십시오.
- 다중 문 쿼리에서는 `ROW_COUNT()` 함수를 사용할 수 없습니다.
- RDS 프록시는 하나의 TLS 레코드에서 여러 응답 메시지를 처리할 수 없는 클라이언트 애플리케이션을 지원하지 않습니다.
- RDS Proxy는 MySQL 이중 암호를 지원하지 않습니다.

> **Important**
>
> MySQL 데이터베이스와 관련된 프록시의 경우 초기화 쿼리에서 구성 파라미터 `sql_auto_is_null`을 `true`로 설정하거나 0이 아닌 값으로 설정하지 마시기 바랍니다. 이렇게 하면 잘못된 애플리케이션 동작이 발생할 수 있습니다.

RDS for PostgreSQL 추가 제한 사항

RDS for PostgreSQL 데이터베이스를 사용하는 RDS 프록시에는 다음과 같은 추가 제한 사항이 적용됩니다.

- RDS 프록시는 PostgreSQL에 대한 세션 고정 필터를 지원하지 않습니다.
- 현재 모든 프록시는 PostgreSQL에 대한 포트 5432에서 수신합니다.
- PostgreSQL의 경우 RDS 프록시는 현재 `CancelRequest`를 실행하여 클라이언트에서 쿼리를 취소하는 것을 지원하지 않습니다. Ctrl+C를 사용하여 대화형 psql 세션에서 장기 실행 쿼리를 취소하는 경우를 예로 들 수 있습니다.
- PostgreSQL 함수 lastval의 결과가 항상 정확하지는 않습니다. 해결 방법으로 `RETURNING` 절과 함께 INSERT 문을 사용합니다.
- RDS 프록시는 현재 스트리밍 복제 모드를 지원하지 않습니다.
- RDS for PostgreSQL 16에서는 `scram_iterations` 값을 수정해도 프록시와 데이터베이스 간의 인증 프로세스에만 영향을 줍니다. 특히, `ClientPasswordAuthType`을 `scram-sha-256`으로 구

성한 경우 scram_iterations 값을 사용자 지정해도 클라이언트-프록시 암호 인증에는 영향을 주지 않습니다. 대신 클라이언트-프록시 암호 인증의 반복 값은 4096으로 고정됩니다.

- default 데이터베이스가 있어야 합니다.
- ALTER ROLE을 사용하여 SET ROLE로 사용자 역할을 변경하는 경우 해당 사용자와 프록시의 후속 연결에서 고정이 발생하는 경우 이 역할 설정이 사용되지 않을 수 있습니다. 이를 방지하려면 프록시를 사용할 때 프록시의 초기화 쿼리에 SET ROLE을 사용합니다. 자세한 내용은 Amazon RDS용 프록시 만들기에서 초기화 쿼리 섹션을 참조하세요.

> **Important**
>
> PostgreSQL 데이터베이스를 사용하는 기존 프록시의 경우, SCRAM만 사용하도록 데이터베이스 인증을 수정하면 최대 60초 동안 프록시를 사용할 수 없게 됩니다. 문제를 방지하려면 다음 중 하나를 수행합니다.
>
> - 데이터베이스에서 SCRAM 및 MD5 인증이 모두 허용되는지 확인합니다.
> - SCRAM 인증만 사용하려면 새 프록시를 생성하고, 애플리케이션 트래픽을 새 프록시로 마이그레이션한 다음 이전에 데이터베이스와 연결된 프록시를 삭제합니다.

RDS Proxy 사용 대상 계획

RDS Proxy를 사용하여 가장 많은 혜택을 누릴 수 있는 DB 인스턴스, 클러스터 및 애플리케이션을 결정할 수 있습니다. 이렇게 하려면 다음 요소를 고려하십시오.

- '연결이 너무 많음' 오류가 발생한 DB 인스턴스는 프록시와 연결하기에 좋은 후보입니다. 이는 종종 ConnectionAttempts CloudWatch 지표의 값이 높다는 특징이 있습니다. 프록시를 사용하면 애플리케이션은 많은 클라이언트 연결을 열 수 있고, 프록시는 DB 인스턴스에 대한 장기 연결을 더 적게 관리합니다.
- T2 또는 T3 같이 더 작은 규모의 AWS 인스턴스 클래스를 사용하는 DB 인스턴스의 경우 프록시를 사용하면 메모리 부족 상태를 방지할 수 있습니다. 또한 연결을 설정하는 데 필요한 CPU 오버헤드를 줄일 수 있습니다. 이러한 상태는 많은 수의 연결을 처리할 때 발생할 수 있습니다.
- 또한 특정 Amazon CloudWatch 지표를 모니터링하여 DB 인스턴스에서 특정 유형의 제한에 접근하는지 확인할 수 있습니다. 이러한 제한은 연결 수와 연결 관리와 관련된 메모리에 대한 것입니다. 또한 특정 CloudWatch 지표를 모니터링하여 DB 인스턴스에서 수명이 짧은 여러 연결을 처리하고 있는지 확인할 수 있습니다. 이러한 연결을 열고 닫으면 데이터베이스에 성능 오버헤드가 발생할 수 있

습니다. 모니터링할 지표에 대한 자세한 내용은 Amazon CloudWatch를 사용한 RDS 프록시 지표 모니터링 단원을 참조하십시오.

- AWS Lambda 함수도 프록시를 사용하기 위한 좋은 후보가 될 수 있습니다. 이러한 함수는 RDS Proxy가 제공하는 연결 풀링의 이점을 누릴 수 있는 단기 데이터베이스 연결을 자주 만듭니다. Lambda 애플리케이션 코드에서 데이터베이스 자격 증명을 관리하는 대신 Lambda 함수에 대해 이미 가지고 있는 IAM 인증을 활용할 수 있습니다.
- 일반적으로 많은 수의 데이터베이스 연결을 열고 닫으며 내장된 연결 풀링 메커니즘이 없는 애플리케이션이 프록시를 사용하기에 적합합니다.
- 오랜 기간 동안 많은 수의 연결을 열린 상태로 유지하는 애플리케이션은 일반적으로 프록시를 사용하는 것이 좋습니다. SaaS(Software as a Service) 또는 전자 상거래와 같은 업종의 애플리케이션은 연결을 열린 상태로 두고 데이터베이스 요청에 대한 지연 시간을 최소화하는 경우가 많습니다. RDS 프록시를 사용하면 애플리케이션은 DB 인스턴스에 직접 연결할 때보다 더 많은 연결을 열어 둘 수 있습니다.
- 모든 DB 인스턴스에 대해 이러한 인증을 설정하기는 복잡하기 때문에 IAM 인증 Secrets Manager를 채택하지 않았을 수 있습니다. 그렇다면 기존 인증 방법을 그대로 두고 인증을 프록시에 위임할 수 있습니다. 프록시는 특정 애플리케이션에 대한 클라이언트 연결에 인증 정책을 적용할 수 있습니다. Lambda 애플리케이션 코드에서 데이터베이스 자격 증명을 관리하는 대신 Lambda 함수에 대해 이미 가지고 있는 IAM 인증을 활용할 수 있습니다.
- RDS 프록시를 사용하면 데이터베이스 장애에 대해 애플리케이션 복원력이 높아지고 투명성이 높아집니다. RDS 프록시는 도메인 이름 시스템(DNS) 캐시를 우회하여 Aurora RDS 다중 AZ DB 인스턴스의 장애 조치 시간을 최대 66% 단축합니다. 또한 RDS 프록시는 애플리케이션 연결을 유지하면서 트래픽을 새 데이터베이스 인스턴스로 자동 라우팅합니다. 따라서 애플리케이션의 장애 조치 투명성이 높아집니다.

RDS Proxy 개념 및 용어

RDS 프록시를 사용하여 Amazon RDS DB 인스턴스에 대한 연결 관리를 간소화할 수 있습니다.

RDS Proxy는 클라이언트 애플리케이션과 데이터베이스 사이의 네트워크 트래픽을 처리합니다. 데이터베이스 프로토콜을 이해하여 능동적으로 수행합니다. 그런 다음 애플리케이션의 SQL 작업과 데이터베이스의 결과 집합을 기반으로 동작을 조정합니다.

RDS Proxy는 데이터베이스의 연결 관리를 위한 메모리 및 CPU 오버헤드를 줄입니다. 애플리케이션이 많은 연결을 동시에 열 때 데이터베이스에 필요한 메모리 및 CPU 리소스가 더 적습니다. 또한 오랫동안 유휴 상태를 유지하는 연결을 닫았다가 다시 여는 데 애플리케이션의 논리가 필요하지 않습니다.

마찬가지로, 데이터베이스 문제의 경우 연결을 다시 설정하는 데 더 적은 애플리케이션 논리가 필요합니다.

RDS Proxy의 인프라는 고가용성이고 여러 가용 영역(AZ)에 배포됩니다. RDS 프록시의 계산, 메모리 및 스토리지는 RDS DB 인스턴스와 별개입니다. 이러한 분리는 데이터베이스 서버의 오버헤드를 줄여 데이터베이스 워크로드를 처리하는 데 리소스가 투입될 수 있습니다. RDS Proxy 계산 리소스는 서버리스로 데이터베이스 워크로드에 따라 자동으로 조정됩니다.

주제

- RDS Proxy 개념 개요
- 연결 풀링
- RDS Proxy 보안
- 장애 조치
- 트랜잭션

RDS Proxy 개념 개요

RDS Proxy는 연결 풀링 및 다음 섹션에 설명된 다른 기능을 수행하기 위해 인프라를 처리합니다. RDS 콘솔의 프록시 페이지에서 표시된 프록시를 볼 수 있습니다.

각 프록시는 단일 RDS DB 인스턴스에 대한 연결을 처리합니다. RDS 다중 AZ DB 인스턴스의 경우 프록시가 현재 라이터 인스턴스를 자동으로 결정합니다.

프록시가 열린 상태를 유지하고 데이터베이스 애플리케이션이 사용할 수 있는 연결이 연결 풀을 구성합니다.

기본적으로 RDS Proxy는 세션에서 각 트랜잭션 후에 연결을 재사용할 수 있습니다. 이 트랜잭션 수준 재사용을 멀티플렉싱이라고 합니다. RDS Proxy가 일시적으로 연결 풀에서 연결을 제거하여 재사용할 경우 해당 작업을 연결 대여라고 합니다. 그렇게 하는 것이 안전하면 RDS Proxy는 연결 풀에 해당 연결을 반환합니다.

경우에 따라 RDS Proxy는 현재 세션 외부에서 데이터베이스 연결을 재사용하는 것이 안전한지 확신할 수 없습니다. 이러한 경우 세션이 끝날 때까지 동일한 연결에서 세션을 유지합니다. 이 대체 동작을 고정이라고 합니다.

프록시에는 기본 엔드포인트가 있습니다. Amazon RDS DB 인스턴스로 작업할 때 이 엔드포인트에 연결합니다. 인스턴스에 직접 연결하는 읽기/쓰기 엔드포인트에 연결하는 대신 이 작업을 수행합니다.

RDS DB 클러스터의 경우, 추가 읽기/쓰기 및 읽기 전용 엔드포인트를 생성할 수도 있습니다. 자세한 내용은 프록시 엔드포인트 개요 단원을 참조하십시오.

예를 들어 연결 풀링 없이 읽기/쓰기 연결을 위해 클러스터 엔드포인트에 계속 연결할 수 있습니다. 로드 밸런싱된 읽기 전용 연결을 위해 리더 엔드포인트에 계속 연결할 수 있습니다. 클러스터의 특정 DB 인스턴스에 대한 진단 및 문제 해결을 위해 인스턴스 엔드포인트에 계속 연결할 수 있습니다. AWS Lambda 같은 다른 AWS 서비스를 사용하여 RDS 데이터베이스에 연결하는 경우 프록시 엔드포인트를 사용하도록 연결 설정을 변경합니다. 예를 들어 RDS Proxy 기능을 활용하면서 Lambda 함수가 데이터베이스에 액세스할 수 있도록 프록시 엔드포인트을 지정합니다.

각 프록시는 대상 그룹을 포함합니다. 이 대상 그룹은 프록시가 연결할 수 있는 RDS DB 인스턴스를 구현합니다. 프록시와 연결된 RDS DB 인스턴스를 해당 프록시의 대상이라고 합니다. 편의를 위해 콘솔을 통해 프록시를 생성하면 RDS Proxy는 해당 대상 그룹을 생성하고 연결된 대상을 자동으로 등록합니다.

엔진 패밀리는 동일한 DB 프로토콜을 사용하는 관련 데이터베이스 엔진 집합입니다. 생성하는 각 프록시에 대해 엔진 패밀리를 선택합니다.

연결 풀링

각 프록시는 연결된 RDS 데이터베이스의 라이터와 리더 인스턴스에 대한 연결 풀링을 개별적으로 수행합니다. 연결 풀링은 연결을 열고 닫으며 많은 연결을 동시에 열린 상태로 유지하는 데 관련된 오버헤드를 줄이는 최적화 기능입니다. 이 오버헤드에는 각각의 새로운 연결을 처리하는 데 필요한 메모리가 포함됩니다. 또한 각 연결을 닫고 새 연결을 여는 데는 CPU 오버헤드가 수반됩니다. 예를 들어 Transport Layer Security/Secure Sockets Layer(TLS/SSL) 핸드셰이킹, 인증, 협상 기능 등이 포함됩니다. 연결 풀링은 애플리케이션 논리를 단순화합니다. 동시 연결 수를 최소화하기 위해 애플리케이션 코드를 작성할 필요가 없습니다.

또한 각 프록시는 연결 재사용이라고도 하는 연결 멀티플렉싱을 수행합니다. 멀티플렉싱을 사용하면 RDS 프록시가 하나의 기본 데이터베이스 연결을 사용하여 한 트랜잭션에 대한 모든 작업을 수행합니다. 그런 다음 RDS 는 다음 트랜잭션에 대해 다른 연결을 사용할 수 있습니다. 사용자는 프록시에 대해 많은 동시 연결을 열 수 있고, 프록시는 DB 인스턴스 또는 클러스터에 대해 더 적은 수의 연결을 열린 상태로 유지합니다. 이렇게 하면 데이터베이스 서버에서 연결에 대한 메모리 오버헤드가 최소화됩니다. 이 기술은 또한 “연결이 너무 많음” 오류의 가능성을 줄입니다.

RDS Proxy 보안

RDS Proxy는 TLS/SSL 및 AWS Identity and Access Management(IAM)와 같은 기존 RDS 보안 메커니즘을 사용합니다. 이러한 보안 기능에 대한 일반적인 내용은 Amazon RDS의 보안 단원을 참조하십시오. 또한 RDS가 인증, 권한 부여 및 기타 보안 영역에서 작동하는 방식을 숙지해야 합니다.

RDS Proxy는 클라이언트 애플리케이션과 기본 데이터베이스 간의 추가 보안 계층의 역할을 할 수 있습니다. 예를 들어 기본 DB 인스턴스가 이전 버전의 TLS를 지원하더라도 TLS 1.3을 사용하여 프록시에 연결할 수 있습니다. IAM 역할을 사용하여 프록시에 연결할 수 있습니다. 프록시가 기본 사용자 및 암호 인증 방법을 사용하여 데이터베이스에 연결하더라도 마찬가지입니다. 이 기술을 사용하면 많은 비용을 들여 DB 인스턴스 자체를 마이그레이션하지 않아도 데이터베이스 애플리케이션에 강력한 인증 요구 사항을 적용할 수 있습니다.

RDS Proxy가 사용하는 데이터베이스 자격 증명을 AWS Secrets Manager에 저장합니다. 프록시가 액세스하는 RDS DB 인스턴스의 각 데이터베이스 사용자는 Secrets Manager에 해당하는 보안 암호가 있어야 합니다. RDS Proxy의 사용자에 대한 IAM 인증을 설정할 수도 있습니다. 이렇게 하면 데이터베이스가 여전히 기본 암호 인증을 사용하는 경우에도 데이터베이스 액세스에 IAM 인증을 적용할 수 있습니다. 애플리케이션 코드에 데이터베이스 자격 증명을 포함하는 대신 이러한 보안 기능을 사용하는 것이 좋습니다.

RDS Proxy에서 TLS/SSL 사용

TLS/SSL 프로토콜을 사용하여 RDS Proxy에 연결할 수 있습니다.

> **Note**
>
> RDS Proxy는 AWS Certificate Manager(ACM)의 인증서를 사용합니다. RDS Proxy을 사용하는 경우 Amazon RDS 인증서를 다운로드하거나 RDS Proxy 연결을 사용하는 애플리케이션을 업데이트할 필요가 없습니다.

프록시와 데이터베이스 간의 모든 연결에 TLS를 적용하려면 AWS Management Console에서 프록시를 생성하거나 수정할 때 전송 계층 보안 필요 설정을 지정할 수 있습니다.

RDS Proxy를 사용하면 세션에서 클라이언트와 RDS Proxy 엔드포인트 간에 TLS/SSL을 사용하도록 할 수 있습니다. RDS Proxy가 그렇게 하도록 하려면 클라이언트 측에서 요구 사항을 지정합니다. SSL 세션 변수는 RDS Proxy를 사용하는 데이터베이스에 대한 SSL 연결에 대해 설정되지 않습니다.

- RDS for MySQL의 경우 `mysql` 명령을 실행할 때 `--ssl-mode` 파라미터를 사용하여 클라이언트 측에서 요구 사항을 지정합니다.
- Amazon RDS PostgreSQL의 경우 `psql` 명령을 실행할 때 `conninfo` 문자열의 일부로 `sslmode=require`를 지정합니다.

RDS Proxy는 TLS 프로토콜 버전 1.0, 1.1, 1.2 및 1.3을 지원합니다. 기본 데이터베이스에서 사용하는 것보다 더 높은 버전의 TLS를 사용하여 프록시에 연결할 수 있습니다.

기본적으로 클라이언트 프로그램은 RDS Proxy와의 암호화된 연결을 설정하며, `--ssl-mode` 옵션을 통해 추가 제어 기능도 제공합니다. 클라이언트 측에서 RDS Proxy는 모든 SSL 모드를 지원합니다.

클라이언트의 경우, SSL 모드는 다음과 같습니다.

PREFERRED

SSL이 최우선 선택 사항이지만 필수는 아닙니다.

비활성화됨

SSL이 허용되지 않습니다.

필수

SSL을 설정합니다.

VERIFY_CA

SSL을 설정하고 인증 기관(CA)을 확인합니다.

VERIFY_IDENTITY

SSL을 설정하고 CA 및 CA 호스트 이름을 확인합니다.

`--ssl-mode`, `VERIFY_CA` 또는 `VERIFY_IDENTITY`과 함께 클라이언트를 사용하는 경우 `--ssl-ca` 형식의 CA를 가리키는 `.pem` 옵션을 지정합니다. 사용할 `.pem` 파일의 경우 Amazon Trust Services에서 모든 루트 CA PEM을 다운로드하고 하나의 `.pem` 파일에 배치하세요.

RDS 프록시에서는 도메인과 하위 도메인 모두에 적용되는 와일드카드 인증서를 사용합니다. `mysql` 클라이언트를 사용하여 SSL 모드 `VERIFY_IDENTITY`로 연결하는 경우 현재는 MySQL 8.0 호환 `mysql` 명령을 사용해야 합니다.

장애 조치

장애 조치는 원래 인스턴스를 사용할 수 없게 될 때 데이터베이스 인스턴스를 다른 인스턴스로 대체하는 고가용성 기능입니다. 장애 조치는 데이터베이스 인스턴스 문제로 인해 발생할 수 있습니다. 또한 데이터베이스 업그레이드를 수행하는 경우와 같이 일반적인 유지 관리 절차의 일부로 발생할 수 있습니다. 장애 조치는 다중 AZ 구성의 RDS DB 인스턴스에 적용됩니다.

프록시를 통해 연결하면 애플리케이션이 데이터베이스 장애 조치를 보다 효율적으로 수행할 수 있습니다. 원래 DB 인스턴스를 사용할 수 없게 되면 RDS Proxy가 유휴 애플리케이션 연결을 끊지 않고 대기 데이터베이스에 연결합니다. 이렇게 하면 장애 조치 프로세스를 단축하고 간소화할 수 있습니다. 이

는 일반적인 재부팅 또는 데이터베이스 문제보다 애플리케이션에 대한 운영 중단이 발생할 확률이 적습니다.

RDS Proxy를 사용하지 않으면 장애 조치에 잠시 중단이 필요합니다. 중단 중에 장애 조치 시 데이터베이스에 대한 쓰기 작업을 수행할 수 없습니다. 모든 기존 데이터베이스 연결이 중단되며 애플리케이션이 해당 연결을 다시 열어야 합니다. 사용할 수 없는 인스턴스를 대체하여 읽기 전용 DB 인스턴스가 승격되면 새 연결 및 쓰기 작업에 데이터베이스를 사용할 수 있게 됩니다.

DB 장애 조치 중에 RDS Proxy는 계속해서 동일한 IP 주소에서 연결을 수락하고 자동으로 연결을 새 기본 DB 인스턴스로 전달합니다. RDS Proxy를 통해 연결하는 클라이언트는 다음 사항에 영향을 받지 않습니다.

- 장애 조치 시 Domain Name System(DNS) 전파가 지연됩니다.
- 로컬 DNS 캐싱.
- 연결 시간 초과.
- 현재 라이터인 DB 인스턴스가 확실하지 않습니다.
- 연결을 닫지 않은 상태에서 사용할 수 없게 된 이전 라이터의 쿼리 응답을 기다립니다.

자체 연결 풀을 유지 관리하는 애플리케이션의 경우 RDS Proxy를 통해 연결한다는 것은 장애 조치 또는 기타 중단 중에 대부분의 연결이 활성 상태를 유지함을 의미합니다. 트랜잭션 도중 또는 SQL 문의 중간에 있는 연결만 취소됩니다. RDS Proxy는 새 연결을 즉시 수락합니다. 데이터베이스 라이터를 사용할 수 없는 경우 RDS Proxy는 들어오는 요청을 대기열에 넣습니다.

자체 연결 풀을 유지 관리하지 않는 애플리케이션의 경우 RDS Proxy는 더 빠른 연결 속도와 더 많은 열린 연결을 제공합니다. 그러므로 데이터베이스에서 자주 다시 연결하기 위한 비용이 많이 드는 오버헤드가 감소합니다. 이를 위해 RDS Proxy 연결 풀에서 유지 관리되는 데이터베이스 연결을 재사용합니다. 이 접근 방식은 설정 비용이 상당한 TLS 연결에 특히 중요합니다.

트랜잭션

단일 트랜잭션 내의 모든 명령문은 항상 동일한 기본 데이터베이스 연결을 사용합니다. 트랜잭션이 종료하면 다른 세션에서 연결을 사용할 수 있게 됩니다. 트랜잭션을 세분화 단위로 사용하면 다음과 같은 결과가 발생합니다.

- RDS for MySQL `autocommit` 설정을 활성화하면 개별 문 종료 후 연결 재사용이 발생할 수 있습니다.
- 반대로, `autocommit` 설정을 비활성화하면 세션에서 실행하는 첫 번째 문이 새 트랜잭션을 시작합니다. 예를 들어,`SELECT`, `INSERT`, `UPDATE` 및 기타 데이터 조작 언어(DML) 문의 시퀀스를 입력한

다고 가정합시다. 이 경우, COMMIT, ROLLBACK을 실행하거나 기타 방법으로 트랜잭션을 종료할 때까지 연결을 재사용할 수 없습니다.

- DDL(데이터 정의 언어) 문을 입력하면 해당 문이 완료된 후 트랜잭션이 종료합니다.

RDS Proxy는 데이터베이스 클라이언트 애플리케이션에서 사용하는 네트워크 프로토콜을 통해 트랜잭션이 종료하는 시점을 감지합니다. 트랜잭션 감지는 SQL 문의 텍스트에 나타나는 COMMIT 또는 ROLLBACK 같은 키워드에 의존하지 않습니다.

경우에 따라 RDS Proxy에서 세션을 다른 연결로 이동하는 것이 비현실적인 데이터베이스 요청을 감지할 수 있습니다. 이러한 경우 세션의 나머지 부분에서 해당 연결에 대한 멀티플렉싱을 해제합니다. RDS Proxy가 세션에서 멀티플렉싱이 실용적이라는 것을 확신할 수 없는 경우에도 동일한 규칙이 적용됩니다. 이 작업을 고정이라고 합니다. 고정을 탐지하고 최소화하는 방법은 RDS 프록시 고정 방지 단원을 참조하십시오.

RDS 프록시 시작하기

다음 페이지의 정보를 통해 관련 보안 옵션을 설정하고 Amazon RDS Proxy를 관리하고 설정하세요. 보안 옵션은 각 프록시에 액세스할 수 있는 사용자 및 각 프록시가 DB 인스턴스에 연결하는 방법을 제어합니다.

RDS 프록시를 처음 사용하는 경우 제시된 순서대로 페이지를 따라가는 것이 좋습니다.

주제

- RDS 프록시에 대한 네트워크 사전 조건 설정
- AWS Secrets Manager에서 RDS 프록시용 데이터베이스 자격 증명 설정
- RDS 프록시에 대한 AWS Identity and Access Management(IAM) 정책 설정
- Amazon RDS용 프록시 만들기
- 프록시 보기
- RDS Proxy를 통해 데이터베이스에 연결

RDS 프록시에 대한 네트워크 사전 조건 설정

RDS 프록시를 사용하려면 RDS DB 인스턴스와 RDS 프록시 간에 공통의 Virtual Private Cloud(VPC)가 있어야 합니다. 이 VPC에는 서로 다른 가용 영역에 있는 최소 2개의 서브넷이 있어야 합니다. 계정은 이러한 서브넷을 소유하거나 다른 계정과 공유할 수 있습니다. VPC 공유에 대한 자세한 내용은 공유 VPC 작업을 참조하세요.

Amazon EC2, Lambda 또는 Amazon ECS와 같은 클라이언트 애플리케이션 리소스는 프록시와 동일한 VPC에 있을 수 있습니다. 또는 프록시와 별도의 VPC에 있을 수도 있습니다. RDS DB 인스턴스에 성공적으로 연결했다면 필요한 네트워크 리소스가 이미 있는 것입니다.

주제

- 서브넷에 대한 정보 가져오기
- IP 주소 용량 계획

서브넷에 대한 정보 가져오기

프록시를 만들려면 프록시가 작동하는 서브넷과 VPC를 제공해야 합니다. 다음 Linux 예에서는 AWS 계정에서 소유한 VPC 및 서브넷을 검사하는 AWS CLI 명령을 보여줍니다. 특히 CLI를 사용하여 프록시를 생성할 경우 서브넷 ID를 파라미터로 전달합니다.

```
aws ec2 describe-vpcs
aws ec2 describe-internet-gateways
aws ec2 describe-subnets --query '*[].[VpcId,SubnetId]' --output text | sort
```

다음 Linux 예시에서는 특정 RDS DB 인스턴스에 해당하는 서브넷 ID를 확인하는 AWS CLI 명령을 보여줍니다. DB 인스턴스의 VPC ID를 찾습니다. VPC를 검사하여 서브넷을 찾습니다. 다음 Linux 예제에서는 그 방법을 보여 줍니다.

```
$ #From the DB instance, trace through the DBSubnetGroup and Subnets to find the subnet
 IDs.
$ aws rds describe-db-instances --db-instance-identifier my_instance_id --query '*[].
[DBSubnetGroup]||[0]||[0]||[Subnets]||[0]||[*].SubnetIdentifier' --output text
```

```
subnet_id_1
subnet_id_2
subnet_id_3
...
```

```
$ #From the DB instance, find the VPC.
$ aws rds describe-db-instances --db-instance-identifier my_instance_id --query '*[].
[DBSubnetGroup]||[0]||[0].VpcId' --output text
```

```
my_vpc_id
```

```
$ aws ec2 describe-subnets --filters Name=vpc-id,Values=my_vpc_id --query '*[].
[SubnetId]' --output text
```

```
subnet_id_1
subnet_id_2
subnet_id_3
subnet_id_4
subnet_id_5
subnet_id_6
```

IP 주소 용량 계획

RDS 프록시는 등록된 DB 인스턴스의 크기 및 수를 기준으로 필요에 따라 용량을 자동으로 조정합니다. 등록된 데이터베이스 또는 내부 RDS 프록시 유지 관리 작업의 크기를 늘리는 것과 같은 특정 작업의 경우 프록시 용량을 늘려야 할 수 있습니다. 이러한 작업 중에 프록시에 추가 용량을 프로비저닝하기 위해 더 많은 IP 주소가 필요할 수 있습니다. 이러한 추가 주소를 사용하면 워크로드에 영향을 주지 않고 프록시를 확장할 수 있습니다. 서브넷에 사용 가능한 IP 주소가 부족하면 프록시가 스케일 업되지 않을 수 있습니다. 이로 인해 쿼리 지연 시간이 길어지거나 클라이언트 연결이 실패할 수 있습니다. RDS는 서브넷에 사용 가능한 IP 주소가 충분하지 않을 경우 RDS-EVENT-0243 이벤트를 통해 알려줍니다. 이 이벤트에 대한 자세한 내용은 RDS 프록시 이벤트 작업 섹션을 참조하세요.

DB 인스턴스 클래스 크기를 기준으로 프록시의 서브넷에 다음과 같은 최소 개수의 사용 가능한 IP 주소를 예약합니다.

DB 인스턴스 클래스	여유 IP 주소 최소 개수
db.*.xlarge 이하	10
db.*.2xlarge	15
db.*.4xlarge	25
db.*.8xlarge	45
db.*.12xlarge	60
db.*.16xlarge	75
db.*.24xlarge	110

이러한 권장 IP 주소 수는 기본 엔드포인트만 있는 프록시에 대한 예상 개수입니다. 추가 엔드포인트 또는 읽기 전용 복제본이 있는 프록시에는 여유 IP 주소가 더 필요할 수 있습니다. 각 추가 엔드포인트에 대해 IP 주소를 세 개 더 예약하는 것이 좋습니다. 각 읽기 전용 복제본에 대해 해당 읽기 전용 복제본의 크기를 기준으로 테이블에 지정된 대로 추가 IP 주소를 예약하는 것이 좋습니다.

> **Note**
>
> RDS Proxy는 하나의 VPC 내에서 IP 주소를 215개보다 많이 소비하지 않습니다.

AWS Secrets Manager에서 RDS 프록시용 데이터베이스 자격 증명 설정

Amazon RDS의 RDS 프록시는 AWS Secrets Manager를 사용하여 데이터베이스 자격 증명을 안전하게 저장하고 관리합니다. 애플리케이션에 임베딩 자격 증명을 포함하는 대신 프록시를 필요한 인증 세부 정보가 포함된 Secrets Manager 보안 암호와 연결합니다. 프록시가 RDS DB 인스턴스에서 연결하는 각 데이터베이스 사용자 계정에 대해 별도의 Secrets Manager 암호를 만듭니다.

주제

- RDS 프록시에 사용할 보안 암호 만들기

RDS 프록시에 사용할 보안 암호 만들기

프록시를 만들기 전에 데이터베이스 자격 증명을 저장하는 보안 암호를 최소 하나 이상 만들어야 합니다.

콘솔

보안 암호 생성

1. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. Store a new secret(새 보안 암호 저장)을 선택합니다.
3. Amazon RDS 데이터베이스에 대한 자격 증명을 선택합니다.
4. 사용자 이름과 암호를 입력합니다. 입력한 자격 증명은 연결된 RDS 데이터베이스에 있는 데이터베이스 사용자의 자격 증명과 일치해야 합니다. RDS 프록시는 이러한 자격 증명을 사용하여 애플리케이션 대신 데이터베이스에 대한 연결을 인증하고 설정합니다.

일치하지 않는 경우 데이터베이스 암호와 일치하도록 보안 암호를 업데이트할 수 있습니다. 보안 암호를 업데이트할 때까지는 해당 보안 암호를 사용하여 프록시를 통해 연결을 시도하는 작업이 실패할 것이지만, 다른 유효한 보안 암호를 사용하는 연결은 여전히 작동합니다.

> **Note**
>
> RDS for SQL Server의 경우 RDS 프록시에는 DB 인스턴스 데이터 정렬 설정에 관계없이 Secrets Manager에 대/소문자를 구분하는 보안 암호가 필요합니다. 애플리케이션에서 관리자('Admin' 및 'admin')와 같이 대문자가 다른 사용자 이름을 허용하는 경우 각각에 대해 별도의 보안 암호를 만들어야 합니다. RDS 프록시는 클라이언트와 프록시 간에 대/소문자를 구분하지 않는 사용자 이름 인증을 지원하지 않습니다.
> SQL Server의 콜레이션에 대한 자세한 내용은 Microsoft SQL Server 설명서를 참조하세요.

5. 데이터베이스에서 보안 암호가 액세스할 Amazon RDS 데이터베이스를 선택합니다.
6. 보안 암호의 다른 설정을 입력한 다음 저장을 선택합니다. 포괄적인 설명은 AWS Secrets Manager 사용 설명서에 나와 있는 Creating an AWS Secrets Manager secret를 참조하시기 바랍니다.

AWS CLI

AWS CLI를 통해 프록시를 만들 때 해당 보안 암호의 Amazon 리소스 이름(ARN)을 지정합니다. 프록시가 액세스할 수 있는 모든 DB 사용자 계정에 지정합니다. AWS Management Console에서는 설명하는 이름으로 비밀을 선택합니다.

- RDS 프록시와 함께 사용할 Secrets Manager 보안 암호를 만들려면 create-secret 명령을 사용합니다.

```
aws secretsmanager create-secret \
  --name "secret_name" \
  --description "secret_description" \
  --region region_name \
  --secret-string '{"username":"db_user","password":"db_user_password"}'
```

- 또한 사용자 지정 키를 생성하여 Secrets Manager 보안 암호를 암호화할 수 있습니다. 다음 명령은 예시 키를 생성합니다.

```
aws kms create-key --description "test-key" --policy '{
```

```
"Id":"kms-policy",
"Version":"2012-10-17",
"Statement":
  [
    {
      "Sid":"Enable IAM User Permissions",
      "Effect":"Allow",
      "Principal":{"AWS":"arn:aws:iam::account_id:root"},
      "Action":"kms:*","Resource":"*"
    },
    {
      "Sid":"Allow access for Key Administrators",
      "Effect":"Allow",
      "Principal":
        {
          "AWS":
            ["$USER_ARN","arn:aws:iam:account_id::role/Admin"]
        },
      "Action":
        [
          "kms:Create*",
          "kms:Describe*",
          "kms:Enable*",
          "kms:List*",
          "kms:Put*",
          "kms:Update*",
          "kms:Revoke*",
          "kms:Disable*",
          "kms:Get*",
          "kms:Delete*",
          "kms:TagResource",
          "kms:UntagResource",
          "kms:ScheduleKeyDeletion",
          "kms:CancelKeyDeletion"
        ],
      "Resource":"*"
    },
    {
      "Sid":"Allow use of the key",
      "Effect":"Allow",
      "Principal":{"AWS":"$ROLE_ARN"},
      "Action":["kms:Decrypt","kms:DescribeKey"],
      "Resource":"*"
    }
```

```
    ]
}'
```

예를 들어, 다음 명령은 두 데이터베이스 사용자에 대한 Secrets Manager 보안 암호를 생성합니다.

```
aws secretsmanager create-secret \
  --name secret_name_1 --description "db admin user" \
  --secret-string '{"username":"admin","password":"choose_your_own_password"}'

aws secretsmanager create-secret \
  --name secret_name_2 --description "application user" \
  --secret-string '{"username":"app-user","password":"choose_your_own_password"}'
```

사용자 지정 AWS KMS 키로 암호화된 이러한 보안 암호를 만들려면 다음 명령을 사용하세요.

```
aws secretsmanager create-secret \
  --name secret_name_1 --description "db admin user" \
  --secret-string '{"username":"admin","password":"choose_your_own_password"}' \
  --kms-key-id arn:aws:kms:us-east-2:account_id:key/key_id

aws secretsmanager create-secret \
  --name secret_name_2 --description "application user" \
  --secret-string '{"username":"app-user","password":"choose_your_own_password"}' \
  --kms-key-id arn:aws:kms:us-east-2:account_id:key/key_id
```

AWS 계정에서 소유한 보안 암호를 보려면 list-secrets 명령을 사용합니다.

```
aws secretsmanager list-secrets
```

CLI를 사용하여 프록시를 생성할 경우 하나 이상의 보안 정보에 대한 Amazon 리소스 이름(ARN)을 `--auth` 파라미터에 전달합니다. 다음 예제에서는 AWS 계정이 소유한 각 보안 암호의 이름과 ARN만 사용하여 보고서를 준비하는 방법을 보여 줍니다. 이 예에서는 `--output table` 버전 2에서 제공되는 AWS CLI 파라미터를 사용합니다. AWS CLI 버전 1을 사용하는 경우 `--output text`를 대신 사용합니다.

```
aws secretsmanager list-secrets --query '*[].[Name,ARN]' --output table
```

보안 암호에 올바른 자격 증명이 올바른 형식으로 포함되어 있는지 확인하려면 get-secret-value 명령을 사용합니다. *your_secret_name*을 보안 암호의 간단한 이름 또는 ARN으로 바꿉니다.

```
aws secretsmanager get-secret-value --secret-id your_secret_name
```

출력에는 다음과 유사한 JSON 인코딩 값이 있는 줄이 포함됩니다.

```
...
"SecretString": "{\"username\":\"your_username\",\"password\":\"your_password\"}",
...
```

RDS 프록시에 대한 AWS Identity and Access Management(IAM) 정책 설정

Secrets Manager에서 비밀을 만든 후 해당 비밀에 액세스할 수 있는 IAM 정책을 생성합니다. IAM 사용에 대한 일반적인 정보는 Amazon RDS의 자격 증명 및 액세스 관리 섹션을 참조하세요.

> **Tip**
>
> IAM 콘솔을 사용하는 경우 다음 절차가 적용됩니다. AWS Management Console for RDS를 사용하는 경우 RDS에서 자동으로 IAM 정책을 생성할 수 있습니다. 이 경우 다음 절차를 건너뛸 수 있습니다.

프록시와 함께 사용할 Secrets Manager 비밀에 액세스하는 IAM 정책을 생성하려면

1. IAM 콘솔에 로그인합니다. IAM 역할 생성에 설명된 대로 역할 생성 프로세스를 따르고, AWS 서비스에 대한 권한을 위임할 역할 생성을 선택합니다.

   신뢰할 수 있는 엔터티 유형에서 AWS 서비스를 선택합니다. 사용 사례 아래의 기타 AWS 서비스 사용 사례 드롭다운에서 RDS를 선택합니다. RDS – 데이터베이스에 역할 추가를 선택합니다.
2. 새 역할의 경우 인라인 정책 추가 단계를 수행합니다. IAM 정책 편집과 동일한 일반 절차를 사용합니다. 다음 JSON을 JSON 텍스트 상자에 붙여 넣습니다. 자신의 계정 ID를 대체합니다. AWS에 대한 `us-east-2` 리전을 대체합니다. 생성한 보안 암호에 대한 Amazon 리소스 이름(ARN)을 대체합니다. IAM 정책 설명에서 KMS 키 지정을 확인하세요. `kms:Decrypt` 작업을 수행하려면 기본 AWS KMS key 또는 자체 KMS 키를 대체하세요. 무엇을 사용할지는 Secrets Manager 암호를 암호화하는 데 사용할 때 무엇을 사용했는지에 따라 달라집니다.

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
   ```

```
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": [
                "arn:aws:secretsmanager:us-east-2:account_id:secret:secret_name_1",
                "arn:aws:secretsmanager:us-east-2:account_id:secret:secret_name_2"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "arn:aws:kms:us-east-2:account_id:key/key_id",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "secretsmanager.us-east-2.amazonaws.com"
                }
            }
        }
    ]
}
```

3. 이 IAM 역할에 대한 신뢰 정책을 편집합니다. 다음 JSON을 JSON 텍스트 상자에 붙여 넣습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

다음 명령은 AWS CLI를 통해 동일한 작업을 수행합니다.

```
PREFIX=my_identifier
USER_ARN=$(aws sts get-caller-identity --query "Arn" --output text)
```

```
aws iam create-role --role-name my_role_name \
  --assume-role-policy-document '{"Version":"2012-10-17","Statement":
[{"Effect":"Allow","Principal":{"Service":
["rds.amazonaws.com"]},"Action":"sts:AssumeRole"}]}'

ROLE_ARN=arn:aws:iam::account_id:role/my_role_name

aws iam put-role-policy --role-name my_role_name \
  --policy-name $PREFIX-secret-reader-policy --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": [
                "arn:aws:secretsmanager:us-east-2:account_id:secret:secret_name_1",
                "arn:aws:secretsmanager:us-east-2:account_id:secret:secret_name_2"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "arn:aws:kms:us-east-2:account_id:key/key_id",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "secretsmanager.us-east-2.amazonaws.com"
                }
            }
        }
    ]
}
```

Amazon RDS용 프록시 만들기

프록시를 RDS for MariaDB, RDS for Microsoft SQL Server, RDS for MySQL 또는 RDS for PostgreSQL DB 인스턴스와 연결할 수 있습니다.

콘솔

프록시를 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Proxies(프록시)를 선택합니다.
3. Create proxy(프록시 생성)를 선택합니다.
4. 프록시에 대한 다음 설정을 구성합니다.

설정	설명
엔진 패밀리.	프록시가 데이터베이스와 주고받는 네트워크 트래픽을 해석할 때 인식하는 데이터베이스 네트워크 프로토콜입니다. **Note** RDS for PostgreSQL을 사용하려면 인스턴스에서 `postgres` 데이터베이스를 유지해야 합니다. 삭제된 `postgres` 데이터베이스 문제 해결을(를) 참조하세요.
프록시 식별자	AWS 계정 ID와 현재 AWS 리전 내에서 고유한 이름입니다.
유휴 클라이언트 연결 시간 초과	프록시는 설정된 기간 동안 유휴 상태로 유지되면 클라이언트 연결을 닫습니다. 기본값은 1,800초(30분)입니다. 애플리케이션이 이전 요청을 완료한 후 지정된 시간 내에 새 요청을 제출하지 않으면 연결은 유휴 상태입니다. 프록시는 기본 데이터베이스 연결을 열어 두고 연결 풀로 반환하여 새 클라이언트 연결에 사용할 수 있도록 합니다. 기간 경과 연결을 사전에 제거하려면 유휴 클라이언트 연결 제한 시간을 줄이세요. 워크로드가 급증하는 동안 연결 비용을 최소화하려면 제한 시간을 늘립니다.
데이터베이스	이 프록시를 통해 액세스할 RDS DB 인스턴스입니다. 이 목록에는 호환되는 데이터베이스 엔진, 엔진 버전 및 기타 설정이 있는 DB 인스턴스 또는 클러스터만 포함됩니다. 목록이 비어 있으면

설정	설명
	RDS Proxy와 호환되는 새 DB 인스턴스 또는 클러스터를 생성합니다. 이 작업을 수행하려면 Amazon RDS DB 인스턴스 생성의 프로시저를 따르세요. 그런 다음 프록시를 다시 생성해 보세요.
연결 풀 최대 연결	RDS Proxy가 사용할 수 있는 `max_connections` 한도의 백분율을 정의하기 위한 1~100 사이의 값입니다. 이 DB 인스턴스 또는 클러스터에 하나의 프록시만 사용하려는 경우 이 값을 100으로 설정합니다. RDS Proxy가 이 설정을 사용하는 방법에 대한 자세한 내용은 the section called “MaxConnectionsPercent” 섹션을 참조하세요.
세션 고정 필터	RDS 프록시가 멀티플렉싱 연결에 대한 기본 안전 조치를 우회하는 감지된 특정 세션 상태를 고정하지 못하도록 합니다. 현재 PostgreSQL은 이 설정을 지원하지 않으며 사용 가능한 유일한 옵션은 `EXCLUDE_VARIABLE_SETS` 입니다. 이를 활성화하면 한 연결의 세션 변수가 다른 연결의 세션 변수에 영향을 줄 수 있으며, 쿼리가 현재 트랜잭션 외부에 설정된 세션 변수에 의존하는 경우 오류 또는 정확성 문제가 발생할 수 있습니다. 애플리케이션이 데이터베이스 연결을 안전하게 공유할 수 있는지 확인한 후에만 이 옵션을 사용합니다. 다음과 같은 패턴이 나타나면 안전한 상태로 간주될 수 있습니다. • 유효 세션 변수 값에 변경 사항이 없는 `SET` 문입니다. 즉, 세션 변수에 변경 사항이 없습니다. • 세션 변수 값을 변경하고 동일한 트랜잭션에서 명령문을 실행합니다. 자세한 내용은 RDS 프록시 고정 방지 섹션을 참조하세요.
연결 대여 시간 초과	프록시가 사용 가능한 모든 데이터베이스 연결을 사용할 것으로 예상되는 경우 제한 시간 오류가 반환되기 전에 대기 시간을 설정합니다. 최대 5분을 지정할 수 있습니다. 이 설정은 프록시가 최대 연결 수에 도달했으며 모두 사용 중인 경우에만 적용됩니다.

설정	설명
초기화 쿼리.	(선택 사항) 초기화 쿼리를 추가하거나 현재 쿼리를 수정합니다. 각 새 데이터베이스 접속을 열 때 실행할 프록시에 대한 하나 이상의 SQL 문을 지정할 수 있습니다. 이 설정은 일반적으로 각 접속에 동일한 설정이 있는지 확인하기 위해 SET 문과 함께 사용됩니다. 추가하는 쿼리가 유효한지 확인합니다. 단일 SET 문에 여러 변수를 포함하려면 쉼표 구분자를 사용합니다. 예시: `SET variable1 =value1, variable2 =value2` 여러 문의 경우 세미콜론을 구분 기호로 사용합니다. **Important** 대상 그룹 구성의 일부로 초기화 쿼리에 액세스할 수 있으므로 이는 인증 또는 암호화 방법으로 보호되지 않습니다. 프록시 대상 그룹 구성을 보거나 관리할 수 있는 액세스 권한이 있는 사람은 누구나 초기화 쿼리를 볼 수 있습니다. 암호 또는 수명이 긴 암호화 키와 같은 민감한 데이터를 이 옵션에 추가해서는 안 됩니다.
AWS Identity and Access Management(IAM) 역할	프록시가 사용할 수 있는 데이터베이스 사용자 계정의 자격 증명을 나타내는 Secrets Manager 보안 암호에 액세스할 수 있는 권한이 있는 IAM 역할입니다. 또는 AWS Management Console에서 IAM 역할을 만들 수 있습니다.
Secrets Manager 보안 암호	프록시가 RDS DB 인스턴스에 액세스할 수 있는 데이터베이스 사용자 보안 인증 정보를 포함하는 Secrets Manager 보안 암호를 하나 이상 선택합니다.
클라이언트 인증 유형	프록시가 클라이언트로부터의 연결에 사용하는 인증 유형입니다. 선택 사항은 이 프록시와 연결된 모든 Secrets Manager 비밀에 적용됩니다. 비밀마다 다른 클라이언트 인증 유형을 지정해야 하는 경우 AWS CLI 또는 API를 대신 사용하여 프록시를 생성합니다.

설정	설명
IAM 인증.	프록시 연결에 대해 IAM 인증을 요구할지, 허용할지 아니면 허용하지 않을지를 나타냅니다. 허용 옵션은 RDS for SQL Server 프록시에만 유효합니다. 선택 사항은 이 프록시와 연결된 모든 Secrets Manager 비밀에 적용됩니다. 암호마다 다른 IAM 인증을 지정해야 하는 경우 AWS CLI 또는 API를 대신 사용하여 프록시를 생성합니다.
전송 계층 보안 필요	모든 클라이언트 연결에 TLS/SSL을 적용합니다. 클라이언트 연결의 암호화 여부와는 상관없이 프록시는 기본 데이터베이스에 연결할 때 동일한 암호화 설정을 사용합니다.
서브넷	이 필드는 VPC와 연결된 모든 서브넷으로 미리 채워집니다. 프록시에 필요하지 않은 서브넷은 제거할 수 있지만 서브넷은 두 개 이상 남겨 두어야 합니다.

설정	설명
VPC 보안 그룹	기존 VPC 보안 그룹을 선택하거나 AWS Management Console에서 새 보안 그룹을 만듭니다. 애플리케이션이 프록시에 액세스할 수 있도록 인바운드 규칙을 구성하고 데이터베이스 대상의 트래픽을 허용하도록 아웃바운드 규칙을 구성합니다. **Note** 보안 그룹은 프록시에서 데이터베이스로의 연결을 허용해야 합니다. 보안 그룹은 애플리케이션에서 프록시로 수신하고 프록시에서 데이터베이스로 발신하는 데 모두 사용됩니다. 예를 들어 데이터베이스와 프록시 모두에 동일한 보안 그룹을 사용하는 경우, 해당 보안 그룹 내의 리소스가 서로 통신할 수 있는지 확인합니다. 공유 VPC를 사용하는 경우 VPC에 대한 기본 보안 그룹이나 다른 계정과 연결된 보안 그룹을 사용하지 마세요. 대신 본인 계정에 속한 보안 그룹을 선택합니다. VPC가 없다면 만드세요. 자세한 내용은 공유 VPC 작업을 참조하세요. RDS는 고가용성을 보장하기 위해 여러 가용 영역에 프록시를 배포합니다. AZ 간 통신을 활성화하려면 프록시 서브넷의 네트워크 액세스 제어 목록(ACL)에서 엔진 포트의 발신 및 모든 포트의 수신을 허용해야 합니다. 네트워크 ACL에 대한 자세한 내용은 네트워크 ACL을 사용하여 서브넷에 대한 트래픽 제어를 참조하세요. 프록시와 대상의 네트워크 ACL이 동일한 경우 소스가 VPC CIDR로 설정된 TCP 프로토콜 수신 규칙을 추가해야 합니다. 또한 대상이 VPC CIDR로 설정된 엔진 포트별 TCP 프로토콜 송신 규칙을 추가해야 합니다.

설정	설명
향상된 로깅 활성화	프록시 호환성 또는 성능 문제를 해결하려면 이 설정을 활성화합니다. 활성화된 경우 RDS 프록시는 SQL 동작 또는 프록시 연결 성능 및 확장성을 디버깅하는 데 도움이 되는 자세한 성능 정보를 로깅합니다. 디버깅에 대해서만 이 설정을 활성화하고 로그의 민감한 정보를 보호하기 위한 적절한 보안 조치가 마련되어 있는지 확인합니다. 오버헤드를 최소화하기 위해 RDS 프록시는 활성화 후 24시간이 지나면 이 설정을 자동으로 비활성화합니다. 특정 문제를 해결하기 위해 일시적으로 사용합니다.

5. Create proxy(프록시 생성)를 선택합니다.

AWS CLI

AWS CLI를 사용하여 프록시를 생성하려면 다음 필수 파라미터와 함께 create-db-proxy 명령을 호출합니다.

- `--db-proxy-name`
- `--engine-family`
- `--role-arn`
- `--auth`
- `--vpc-subnet-ids`

`--engine-family` 값은 대소문자를 구분합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-proxy \
    --db-proxy-name proxy_name \
    --engine-family { MYSQL | POSTGRESQL | SQLSERVER } \
    --auth ProxyAuthenticationConfig_JSON_string \
    --role-arn iam_role \
    --vpc-subnet-ids space_separated_list \
```

```
    [--vpc-security-group-ids space_separated_list] \
    [--require-tls | --no-require-tls] \
    [--idle-client-timeout value] \
    [--debug-logging | --no-debug-logging] \
    [--tags comma_separated_list]
```

Windows의 경우:

```
aws rds create-db-proxy ^
    --db-proxy-name proxy_name ^
    --engine-family { MYSQL | POSTGRESQL | SQLSERVER } ^
    --auth ProxyAuthenticationConfig_JSON_string ^
    --role-arn iam_role ^
    --vpc-subnet-ids space_separated_list ^
    [--vpc-security-group-ids space_separated_list] ^
    [--require-tls | --no-require-tls] ^
    [--idle-client-timeout value] ^
    [--debug-logging | --no-debug-logging] ^
    [--tags comma_separated_list]
```

다음은 `--auth` 옵션의 JSON 값 예시입니다. 이 예시는 각 보안 암호에 서로 다른 클라이언트 인증 유형을 적용합니다.

```
[
  {
    "Description": "proxy description 1",
    "AuthScheme": "SECRETS",
    "SecretArn": "arn:aws:secretsmanager:us-
west-2:123456789123:secret/1234abcd-12ab-34cd-56ef-1234567890ab",
    "IAMAuth": "DISABLED",
    "ClientPasswordAuthType": "POSTGRES_SCRAM_SHA_256"
  },

  {
    "Description": "proxy description 2",
    "AuthScheme": "SECRETS",
    "SecretArn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret/1234abcd-12ab-34cd-56ef-1234567890cd",
    "IAMAuth": "DISABLED",
    "ClientPasswordAuthType": "POSTGRES_MD5"

  },
```

```
  {
    "Description": "proxy description 3",
    "AuthScheme": "SECRETS",
    "SecretArn": "arn:aws:secretsmanager:us-
west-2:111122221111:secret/1234abcd-12ab-34cd-56ef-1234567890ef",
    "IAMAuth": "REQUIRED"
  }

]
```

> **Tip**
>
> `--vpc-subnet-ids` 파라미터에 사용할 서브넷 ID를 아직 모르는 경우, RDS 프록시에 대한 네트워크 사전 조건 설정에서 서브넷 ID를 찾는 방법의 예를 참조하세요.

> **Note**
>
> 보안 그룹은 프록시가 연결하는 데이터베이스에 대한 액세스를 허용해야 합니다. 동일한 보안 그룹이 애플리케이션에서 프록시로 수신하고 프록시에서 데이터베이스로 발신하는 데 사용됩니다. 예를 들어 데이터베이스와 프록시에 대해 동일한 보안 그룹을 사용한다고 가정합니다. 이 경우 해당 보안 그룹의 리소스가 동일한 보안 그룹의 다른 리소스와 통신할 수 있도록 지정해야 합니다.
>
> 공유 VPC를 사용하는 경우 VPC에 대한 기본 보안 그룹이나 다른 계정에 속한 보안 그룹을 사용할 수 없습니다. 본인 계정에 속한 보안 그룹을 선택합니다. 없으면 새로 생성합니다. 이 제한 사항에 대한 자세한 내용은 공유 VPC 작업을 참조하세요.

프록시에 적합한 연결을 생성하려면 register-db-proxy-targets 명령을 사용합니다. 대상 그룹 이름 `default`을 지정합니다. RDS Proxy는 각 프록시를 생성할 때 이 이름으로 대상 그룹을 자동으로 생성합니다.

```
aws rds register-db-proxy-targets
    --db-proxy-name value
    [--target-group-name target_group_name]
    [--db-instance-identifiers space_separated_list]  # rds db instances, or
    [--db-cluster-identifiers cluster_id]        # rds db cluster (all instances)
```

RDS API

RDS 프록시를 생성하려면 Amazon RDS API 작업 CreateDBProxy를 호출합니다. AuthConfig 데이터 구조와 함께 파라미터를 전달합니다.

RDS Proxy는 각 프록시를 생성할 때 `default`라는 대상 그룹을 자동으로 생성합니다. RegisterDBProxyTargets 함수를 호출하여 RDS DB 인스턴스를 대상 그룹에 연결합니다.

프록시 보기

하나 이상의 RDS 프록시를 생성한 후 AWS Management Console, AWS CLI 또는 RDS API에서 해당 프록시를 보고 관리할 수 있습니다. 구성 세부 정보를 검토하고, 성능을 모니터링하고, 필요에 따라 수정하거나 삭제할 프록시를 결정할 수 있습니다.

데이터베이스 애플리케이션이 프록시를 통해 트래픽을 라우팅하도록 하려면 연결 문자열에 프록시 엔드포인트를 지정해야 합니다.

콘솔

콘솔에서 프록시를 확인하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Proxies(프록시)를 선택합니다.
3. 세부 정보를 볼 프록시 이름을 선택합니다.
4. 세부 정보 페이지의 대상 그룹 섹션에 프록시가 특정 RDS DB 인스턴스에 어떻게 연결되는지 표시됩니다. 기본 대상 그룹 페이지로 이동하여 프록시를 만드는 중에 정의된 구성 설정을 포함하여 이 연결을 더 자세히 볼 수 있습니다. 이러한 설정에는 최대 연결 비율, 연결 대여 시간 초과, 엔진 패밀리, 세션 고정 필터 등이 포함됩니다.

CLI

CLI를 사용하여 프록시를 보려면 describe-db-proxy 명령을 사용합니다. 기본적으로 요청은 AWS 계정이 소유한 모든 프록시를 반환합니다. 단일 프록시에 대한 세부 정보를 보려면 `--db-proxy-name` 파라미터와 함께 해당 이름을 지정합니다.

```
aws rds describe-db-proxies [--db-proxy-name proxy_name]
```

프록시와 연결된 다른 정보를 보려면 다음 명령을 사용합니다.

```
aws rds describe-db-proxy-target-groups  --db-proxy-name proxy_name

aws rds describe-db-proxy-targets --db-proxy-name proxy_name
```

프록시와 연결된 항목에 대한 자세한 내용을 보려면 다음 명령 시퀀스를 사용합니다.

1. 프록시 목록을 얻으려면 describe-db-proxies를 실행합니다.
2. 프록시에서 사용할 수 있는 최대 연결 비율과 같은 연결 파라미터를 표시하려면 describe-db-proxy-target-groups `--db-proxy-name`을 실행합니다. 프록시의 이름을 파라미터 값으로 사용합니다.
3. 반환된 대상 그룹과 연결된 RDS DB 인스턴스의 세부 정보를 보려면 describe-db-proxy-targets를 실행합니다.

RDS API

RDS API를 사용하여 프록시를 보려면 DescribeDBProxies 작업을 사용합니다. 이 작업은 DBProxy 데이터 형식의 값을 반환합니다.

프록시 연결 설정의 세부 정보를 보려면 DescribeDBProxyTargetGroups 작업을 사용하여 이 반환 값의 프록시 식별자를 사용합니다. 이 작업은 DBProxyTargetGroup 데이터 형식의 값을 반환합니다.

프록시와 연결된 RDS 인스턴스 또는 Aurora DB 클러스터를 보려면 DescribeDBProxyTargets 작업을 사용합니다. 이 작업은 DBProxyTarget 데이터 형식의 값을 반환합니다.

RDS Proxy를 통해 데이터베이스에 연결

프록시를 통해 또는 데이터베이스에 연결하여 RDS DB 인스턴스에 연결하는 방법은 일반적으로 동일합니다. 자세한 내용은 프록시 엔드포인트 개요 섹션을 참조하세요.

주제

- 기본 인증을 사용하여 프록시에 연결
- IAM 인증을 사용하여 프록시에 연결
- Microsoft SQL Server를 사용하여 프록시에 연결할 때 고려할 사항
- PostgreSQL을 사용하여 프록시에 연결할 때 고려할 사항

기본 인증을 사용하여 프록시에 연결

기본 인증을 사용하여 프록시에 연결하려면 다음 단계를 사용합니다.

1. 프록시 엔드포인트를 찾습니다. AWS Management Console에서는 해당 프록시의 세부 정보 페이지에서 엔드포인트를 찾을 수 있습니다. AWS CLI에서는 describe-db-proxies 명령을 사용할 수 있습니다. 다음 예에서는 이 작업을 수행하는 방법을 보여줍니다.

```
# Add --output text to get output as a simple tab-separated list.
$ aws rds describe-db-proxies --query '*[*].
{DBProxyName:DBProxyName,Endpoint:Endpoint}'
[
    [
        {
            "Endpoint": "the-proxy.proxy-demo.us-east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy"
        },
        {
            "Endpoint": "the-proxy-other-secret.proxy-demo.us-
east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy-other-secret"
        },
        {
            "Endpoint": "the-proxy-rds-secret.proxy-demo.us-
east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy-rds-secret"
        },
        {
            "Endpoint": "the-proxy-t3.proxy-demo.us-east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy-t3"
        }
    ]
]
```

2. 클라이언트 애플리케이션의 연결 문자열에서 해당 엔드포인트를 호스트 파라미터로 지정합니다. 예를 들어 프록시 엔드포인트를 `mysql -h` 옵션 또는 `psql -h` 옵션의 값으로 지정합니다.
3. 평소와 동일한 데이터베이스 사용자 이름과 암호를 제공합니다.

IAM 인증을 사용하여 프록시에 연결

RDS Proxy에서 IAM 인증을 사용하는 경우 일반 사용자 이름 및 암호로 인증하도록 데이터베이스 사용자를 설정합니다. IAM 인증은 Secrets Manager에서 사용자 이름 및 암호 자격 증명을 검색하는 RDS Proxy에 적용됩니다. RDS 프록시에서 기본 데이터베이스로의 연결은 IAM을 거치지 않습니다.

IAM 인증을 사용하여 RDS 프록시에 연결하려면 IAM 인증을 통해 RDS DB 인스턴스에 연결하는 것과 동일한 일반 연결 프로시저를 따르면 됩니다. IAM 사용에 대한 일반적인 정보는 Amazon RDS의 보안 섹션을 참조하세요.

RDS Proxy에 대한 IAM 사용의 주요 차이점은 다음과 같습니다.

- 권한 부여 플러그인으로 각 개별 데이터베이스 사용자를 구성하지 않습니다. 데이터베이스 사용자는 여전히 데이터베이스 내에서 일반 사용자 이름과 암호를 가지고 있습니다. 이러한 사용자 이름과 암호를 포함하는 Secrets Manager 비밀을 설정하고 RDS Proxy가 Secrets Manager에서 자격 증명을 검색할 수 있는 권한을 부여합니다.

  IAM 인증은 클라이언트 프로그램과 프록시 간의 연결에 적용됩니다. 그런 다음 프록시는 Secrets Manager에서 검색된 사용자 이름 및 암호 자격 증명을 사용하여 데이터베이스에 대해 인증합니다.
- 인스턴스, 클러스터 또는 리더 엔드포인트 대신 프록시 엔드포인트를 지정합니다. 프록시 엔드포인트에 대한 자세한 내용은 IAM 인증을 사용하여 DB 인스턴스에 연결 단원을 참조하십시오.
- 직접 데이이터베이스 IAM 인증의 경우 데이터베이스 사용자를 선택적으로 선택하고 특수 인증 플러그인으로 식별되도록 구성합니다. 그런 다음 IAM 인증을 사용하여 해당 사용자에게 연결할 수 있습니다.

  프록시 사용 사례에서는 일부 사용자의 사용자 이름과 암호(기본 인증)가 포함된 암호를 프록시에 제공해야 합니다. 그런 다음 IAM 인증을 사용하여 프록시에 연결합니다. 여기서는 데이터베이스 엔드포인트가 아닌 프록시 엔드포인트로 인증 토큰을 생성하여 이를 수행합니다. 또한 제공한 암호의 사용자 이름 중 하나와 일치하는 사용자 이름을 사용합니다.
- IAM 인증을 사용하여 프록시에 연결할 때는 전송 계층 보안(TLS)/보안 소켓 계층(SSL)을 사용해야 합니다.

IAM 정책을 수정하여 특정 사용자에게 프록시에 대한 액세스 권한을 부여할 수 있습니다. 예를 들면 다음과 같습니다.

```
"Resource": "arn:aws:rds-db:us-east-2:1234567890:dbuser:prx-ABCDEFGHIJKL01234/db_user"
```

Microsoft SQL Server를 사용하여 프록시에 연결할 때 고려할 사항

IAM 인증을 사용하여 프록시에 연결하는 경우에는 암호 필드를 사용하지 않습니다. 대신 토큰 필드에 각 데이터베이스 드라이버 유형에 적합한 토큰 속성을 제공합니다. 예를 들어, JDBC의 경우 `accessToken` 속성을 사용하고 ODBC의 경우 `sql_copt_ss_access_token` 속성을 사용합니다. 또는 .NET.SqlClient 드라이버의 `AccessToken` 속성을 사용합니다. 토큰 속성을 지원하지 않는 클라이언트에는 IAM 인증을 사용할 수 없습니다.

일부 조건에서는 프록시가 데이터베이스 연결을 공유할 수 없고 대신 클라이언트 애플리케이션의 프록시 연결을 전용 데이터베이스 연결에 고정할 수 있습니다. 이러한 조건에 대한 자세한 내용은 RDS 프록시 고정 방지를 참조하세요.

PostgreSQL을 사용하여 프록시에 연결할 때 고려할 사항

RDS 프록시에 연결할 새 PostgreSQL 데이터베이스 사용자를 생성하는 경우 데이터베이스에 대해 사용자에게 CONNECT 권한을 부여해야 합니다. 이렇게 하지 않으면 사용자가 연결을 설정할 수 없습니다. 자세한 내용은 the section called “RDS 프록시 사용 시 PostgreSQL 데이터베이스에 새 데이터베이스 사용자 추가” 섹션을 참조하세요.

클라이언트가 PostgreSQL 데이터베이스에 대한 연결을 시작하면 시작 메시지가 전송됩니다. 이 메시지에는 파라미터 이름과 값 문자열의 쌍이 포함됩니다. 자세한 내용은 PostgreSQL 설명서에서 PostgreSQL 메시지 형식의 `StartupMessage`를 참조하십시오.

RDS 프록시를 통해 연결할 때 시작 메시지에는 현재 인식되는 다음과 같은 파라미터가 포함될 수 있습니다.

- `user`
- `database`

시작 메시지에는 다음과 같은 추가 런타임 파라미터가 포함될 수도 있습니다.

- `application_name`
- `client_encoding`
- `DateStyle`
- `TimeZone`
- `extra_float_digits`
- `search_path`

PostgreSQL 메시징에 대한 자세한 내용은 PostgreSQL 설명서의 프런트 엔드/백엔드 프로토콜을 참조하십시오.

PostgreSQL의 경우 JDBC를 사용한다면 고정을 피하기 위해 다음을 사용하는 것이 좋습니다.

- 고정을 방지하려면 JDBC 연결 파라미터 `assumeMinServerVersion`을 `9.0` 이상으로 설정합니다. 이렇게 하면 `SET extra_float_digits = 3`을 실행할 때 JDBC 드라이버가 연결 시작 중에 추가 왕복을 수행하지 못합니다.

- 고정을 방지하려면 JDBC 연결 파라미터 `ApplicationName`을 *any/your-application-name*으로 설정합니다. 이렇게 하면 `SET application_name = "PostgreSQL JDBC Driver"`을 실행할 때 JDBC 드라이버가 연결 시작 중에 추가 왕복을 수행하지 못합니다. JDBC 파라미터는 `ApplicationName`이지만 PostgreSQL `StartupMessage` 파라미터는 `application_name`입니다.

자세한 내용은 RDS 프록시 고정 방지 섹션을 참조하세요. JDBC를 사용한 연결에 대한 자세한 내용은 PostgreSQL 설명서의 데이터베이스에 연결을 참조하십시오.

RDS 프록시 관리

이 섹션에서는 RDS 프록시 작업 및 구성을 관리하는 방법에 대한 정보를 제공합니다. 이러한 절차는 애플리케이션이 데이터베이스 연결을 가장 효율적으로 사용하고 최대 연결 재사용을 달성하는 데 도움이 됩니다. 연결 재사용을 더 많이 활용할수록 CPU 및 메모리 오버헤드를 더 많이 절약할 수 있습니다. 이렇게 하면 애플리케이션의 대기 시간이 줄어들고 데이터베이스가 애플리케이션 요청을 처리하는 데 더 많은 리소스를 사용할 수 있습니다.

주제

- RDS 프록시 수정
- RDS 프록시 사용 시 새 데이터베이스 사용자 추가
- RDS 프록시 연결 고려 사항
- RDS 프록시 고정 방지
- RDS 프록시 삭제

RDS 프록시 수정

프록시를 생성한 후 프록시와 연결된 특정 설정을 변경할 수 있습니다. 프록시 자체, 연결된 대상 그룹 또는 둘 다 수정하면 됩니다. 각 프록시에는 연결된 대상 그룹이 있습니다.

AWS Management Console

> **Important**
>
> 클라이언트 인증 유형 및 IAM 인증 필드의 값은 이 프록시에 연결된 모든 Secrets Manager 보안 암호에 적용됩니다. 보안 암호마다 다른 값을 지정하려면 AWS CLI 또는 API를 대신 사용하여 프록시를 수정하세요.

프록시에 대한 설정을 수정하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Proxies(프록시)를 선택합니다.
3. 프록시 목록에서 설정을 수정하려는 프록시를 선택하거나 세부 정보 페이지로 이동합니다.
4. 작업에서 수정을 선택합니다.
5. 수정할 속성을 입력하거나 선택합니다. 다음을 수정할 수 있습니다.
   - 프록시 식별자 – 새 식별자를 입력하여 프록시 이름을 변경합니다.
   - 유휴 클라이언트 연결 시간 초과 - 유휴 클라이언트 연결 시간 초과의 기간을 입력합니다.
   - IAM 역할 – Secrets Manager에서 보안 암호를 검색하는 데 사용되는 IAM 역할을 변경합니다.
   - Secrets Manager 보안 암호 - Secrets Manager 보안 암호를 추가하거나 제거합니다. 이러한 비밀은 데이터베이스 사용자 이름 및 암호에 해당합니다.
   - 클라이언트 인증 유형 - (PostgreSQL만 해당) 프록시에 대한 클라이언트 연결의 인증 유형을 변경합니다.
   - IAM 인증 - 프록시 연결에 대해 IAM 인증을 요구하거나 허용하지 않습니다.
   - 전송 계층 보안 필요 - 전송 계층 보안(TLS) 요구 사항을 설정하거나 해제합니다.
   - VPC 보안 그룹 - 프록시가 사용할 VPC 보안 그룹을 추가하거나 제거합니다.
   - 향상된 로깅 활성화 - 고급 로깅을 사용하거나 사용 중지하도록 설정합니다.
6. 수정을 선택합니다.

변경할 설정을 찾지 못한 경우 다음 절차에 따라 프록시의 대상 그룹을 업데이트합니다. 프록시와 연결된 대상 그룹은 물리적 데이터베이스 연결과 관련된 설정을 제어합니다. 각 프록시에는 `default`라는 하나의 연결된 대상 그룹이 있으며, 이 그룹은 프록시와 함께 자동으로 생성됩니다. 기본 대상 그룹의 이름은 바꿀 수 없습니다.

대상 그룹은 프록시 세부 정보 페이지에서만 수정할 수 있으며 Proxies(프록시) 페이지의 목록에서는 수정할 수 없습니다.

프록시 대상 그룹에 대한 설정을 수정하려면

1. [프록시(Proxies)] 페이지에서 프록시의 세부 정보 페이지로 이동합니다.
2. 대상 그룹에서 `default` 링크를 선택합니다. 현재 모든 프록시에는 `default`라는 단일 대상 그룹이 있습니다.

3. 기본 대상 그룹에 대한 세부 정보 페이지에서 수정을 선택합니다.
4. 수정할 수 있는 속성의 새 설정을 선택합니다.

   - 데이터베이스 - 다른 RDS DB 인스턴스 또는 클러스터를 선택합니다.
   - 연결 풀 최대 연결 – 프록시에서 사용할 수 있는 최대 연결 비율을 조정합니다.
   - 세션 고정 필터 - (선택 사항) 세션 고정 필터를 선택합니다. 이렇게 하면 클라이언트 연결 간 데이터베이스 연결을 멀티플렉싱하기 위한 기본 안전 조치를 우회할 수 있습니다. 현재 PostgreSQL에서는 설정이 지원되지 않습니다. `EXCLUDE_VARIABLE_SETS`만 선택할 수 있습니다.

     이 설정을 활성화하면 어떤 연결의 세션 변수가 다른 연결에 영향을 줄 수 있습니다. 쿼리가 현재 트랜잭션 외부에 설정된 세션 변수 값에 의존하는 경우 이로 인해 오류나 정확성 문제가 발생할 수 있습니다. 애플리케이션이 클라이언트 연결 간에 데이터베이스 연결을 공유해도 안전한지 확인한 후 이 옵션을 사용하는 것이 좋습니다.

     다음과 같은 패턴이 나타나면 안전한 상태로 간주될 수 있습니다.

     - 유효 세션 변수 값에 변경 사항이 없는 `SET` 명령문이 있습니다(즉 세션 변수에 변경 사항이 없는 경우).
     - 세션 변수 값을 변경하고 동일한 트랜잭션에서 명령문을 실행합니다.

     자세한 내용은 RDS 프록시 고정 방지 단원을 참조하십시오.
   - 연결 대여 시간 초과 - 연결 대여 시간 초과 간격을 조정합니다. 이 설정은 최대 연결 수가 프록시에 이미 사용되고 있는 경우에 적용됩니다. 이 설정은 시간 초과 오류를 반환하기 전에 프록시가 연결을 사용할 수 있을 때까지 기다리는 시간을 지정합니다.
   - 초기화 쿼리. (선택 사항) 초기화 쿼리를 추가하거나 현재 쿼리를 수정합니다. 각 새 데이터베이스 접속을 열 때 실행할 프록시에 대한 하나 이상의 SQL 문을 지정할 수 있습니다. 이 설정은 일반적으로 각 접속에 동일한 설정이 있는지 확인하기 위해 `SET` 문과 함께 사용됩니다. 추가하는 쿼리가 유효한지 확인합니다. 단일 `SET` 문에 여러 변수를 포함하려면 쉼표 구분자를 사용합니다. 예시:

     ```
     SET variable1=value1, variable2=value2
     ```

     여러 문의 경우 세미콜론을 구분 기호로 사용합니다.

   대상 그룹 식별자, 데이터베이스 엔진과 같은 특정 속성은 변경할 수 없습니다.
5. Modify target group(대상 그룹 수정)을 선택합니다.

AWS CLI

AWS CLI를 사용하여 프록시를 수정하려면 modify-db-proxy, modify-db-proxy-target-group, deregister-db-proxy-targets 및 register-db-proxy-targets 명령을 사용합니다.

`modify-db-proxy` 명령을 사용하여 다음과 같은 속성을 변경할 수 있습니다.

- 프록시에서 사용하는 Secrets Manager 보안 암호 집합입니다.
- TLS가 필요한지 여부입니다.
- 유휴 클라이언트 시간 초과.
- 디버깅을 위해 SQL 문에서 추가 정보를 로깅할지 여부입니다.
- Secrets Manager 보안 암호를 검색하는 데 사용되는 IAM 역할입니다.
- 프록시에서 사용하는 보안 그룹입니다.

다음 예제에서는 기존 프록시의 이름을 바꾸는 방법을 보여 줍니다.

```
aws rds modify-db-proxy --db-proxy-name the-proxy --new-db-proxy-name the_new_name
```

연결 관련 설정을 수정하거나 대상 그룹의 이름을 변경하려면 `modify-db-proxy-target-group` 명령을 사용합니다. 현재 모든 프록시에는 `default`라는 단일 대상 그룹이 있습니다. 이 대상 그룹으로 작업하는 경우 프록시 이름을 지정하고 대상 그룹 이름에 `default`를 지정합니다. 기본 대상 그룹의 이름은 바꿀 수 없습니다.

다음 예제에서는 대상 그룹을 사용하여 프록시에 대한 `MaxIdleConnectionsPercent` 설정을 먼저 확인한 다음 변경하는 방법을 보여 줍니다.

```
aws rds describe-db-proxy-target-groups --db-proxy-name the-proxy

{
    "TargetGroups": [
        {
            "Status": "available",
            "UpdatedDate": "2019-11-30T16:49:30.342Z",
            "ConnectionPoolConfig": {
                "MaxIdleConnectionsPercent": 50,
                "ConnectionBorrowTimeout": 120,
                "MaxConnectionsPercent": 100,
                "SessionPinningFilters": []
            },
            "TargetGroupName": "default",
```

```
            "CreatedDate": "2019-11-30T16:49:27.940Z",
            "DBProxyName": "the-proxy",
            "IsDefault": true
        }
    ]
}

aws rds modify-db-proxy-target-group --db-proxy-name the-proxy --target-group-name
 default --connection-pool-config '
{ "MaxIdleConnectionsPercent": 75 }'

{
    "DBProxyTargetGroup": {
        "Status": "available",
        "UpdatedDate": "2019-12-02T04:09:50.420Z",
        "ConnectionPoolConfig": {
            "MaxIdleConnectionsPercent": 75,
            "ConnectionBorrowTimeout": 120,
            "MaxConnectionsPercent": 100,
            "SessionPinningFilters": []
        },
        "TargetGroupName": "default",
        "CreatedDate": "2019-11-30T16:49:27.940Z",
        "DBProxyName": "the-proxy",
        "IsDefault": true
    }
}
```

`deregister-db-proxy-targets` 및 `register-db-proxy-targets` 명령을 사용하여 대상 그룹을 통해 프록시가 연결된 RDS DB 인스턴스를 변경합니다. 현재 각 프록시는 하나의 RDS DB 인스턴스에 연결할 수 있습니다. 대상 그룹은 모든 다중 AZ 구성의 RDS DB 인스턴스에 대한 연결 세부 정보를 추적합니다.

다음 예제는 프록시가 `cluster-56-2020-02-25-1399`라는 Aurora MySQL 클러스터와 연결된 상태에서 시작합니다. 이 예제에서는 `provisioned-cluster`라는 다른 클러스터에 연결할 수 있도록 프록시를 변경하는 방법을 보여 줍니다.

RDS DB 인스턴스로 작업하는 경우 `--db-instance-identifier` 옵션을 지정합니다.

다음 예제에서는 Aurora MySQL 프록시를 수정합니다. Aurora PostgreSQL 프록시는 포트 5432가 있습니다.

```
aws rds describe-db-proxy-targets --db-proxy-name the-proxy
```

```
{
    "Targets": [
        {
            "Endpoint": "instance-9814.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-9814"
        },
        {
            "Endpoint": "instance-8898.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-8898"
        },
        {
            "Endpoint": "instance-1018.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-1018"
        },
        {
            "Type": "TRACKED_CLUSTER",
            "Port": 0,
            "RdsResourceId": "cluster-56-2020-02-25-1399"
        },
        {
            "Endpoint": "instance-4330.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-4330"
        }
    ]
}

aws rds deregister-db-proxy-targets --db-proxy-name the-proxy --db-cluster-identifier
 cluster-56-2020-02-25-1399

aws rds describe-db-proxy-targets --db-proxy-name the-proxy

{
    "Targets": []
}
```

```
aws rds register-db-proxy-targets --db-proxy-name the-proxy --db-cluster-identifier
 provisioned-cluster

{
    "DBProxyTargets": [
        {
            "Type": "TRACKED_CLUSTER",
            "Port": 0,
            "RdsResourceId": "provisioned-cluster"
        },
        {
            "Endpoint": "gkldje.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "gkldje"
        },
        {
            "Endpoint": "provisioned-1.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "provisioned-1"
        }
    ]
}
```

RDS API

RDS API를 사용하여 프록시를 수정하려면 ModifyDBProxy, ModifyDBProxyTargetGroup, DeregisterDBProxyTargets 및 RegisterDBProxyTargets 작업을 사용합니다.

`ModifyDBProxy`를 사용하면 다음과 같은 속성을 변경할 수 있습니다.

- 프록시에서 사용하는 Secrets Manager 보안 암호 집합입니다.
- TLS가 필요한지 여부입니다.
- 유휴 클라이언트 시간 초과.
- 디버깅을 위해 SQL 문에서 추가 정보를 로깅할지 여부입니다.
- Secrets Manager 보안 암호를 검색하는 데 사용되는 IAM 역할입니다.
- 프록시에서 사용하는 보안 그룹입니다.

ModifyDBProxyTargetGroup을 사용하여 연결 관련 설정을 수정할 수 있습니다. 현재 모든 프록시에는 default라는 단일 대상 그룹이 있습니다. 이 대상 그룹으로 작업하는 경우 프록시 이름을 지정하고 대상 그룹 이름에 default를 지정합니다. 기본 대상 그룹의 이름은 바꿀 수 없습니다.

DeregisterDBProxyTargets 및 RegisterDBProxyTargets를 사용하여 대상 그룹을 통해 프록시가 연결된 RDS DB 인스턴스를 변경합니다. 현재 각 프록시는 하나의 RDS DB 인스턴스에 연결할 수 있습니다. 대상 그룹은 다중 AZ 구성의 RDS DB 인스턴스에 대한 연결 세부 정보를 추적합니다.

RDS 프록시 사용 시 새 데이터베이스 사용자 추가

프록시와 연결된 RDS DB 인스턴스에 새 데이터베이스 사용자를 추가해야 할 경우가 있습니다. 그렇다면 Secrets Manager 비밀을 추가하거나 재사용하여 해당 사용자의 자격 증명을 저장합니다. 이를 수행하려면 다음 단계를 수행합니다.

1. AWS Secrets Manager에서 RDS 프록시용 데이터베이스 자격 증명 설정에 설명된 절차를 사용하여 새 Secrets Manager 비밀을 만듭니다.
2. IAM 역할을 업데이트하여 RDS Proxy에 새 Secrets Manager 비밀에 대한 액세스 권한을 부여합니다. 이렇게 하려면 IAM 역할 정책의 리소스 섹션을 업데이트합니다.
3. Secrets Manager 보안 암호 아래에서 RDS 프록시를 수정하여 새 Secrets Manager 보안 암호를 추가합니다.
4. 새 사용자가 기존 사용자를 대신하는 경우 기존 사용자의 프록시 Secrets Manager 비밀에 저장된 자격 증명을 업데이트합니다.

RDS 프록시 사용 시 PostgreSQL 데이터베이스에 새 데이터베이스 사용자 추가

새 사용자를 PostgreSQL 데이터베이스에 추가할 때 다음 명령을 실행한 경우:

```
REVOKE CONNECT ON DATABASE postgres FROM PUBLIC;
```

대상 데이터베이스의 연결을 모니터링할 수 있도록 rdsproxyadmin 사용자에게 CONNECT 권한을 부여합니다.

```
GRANT CONNECT ON DATABASE postgres TO rdsproxyadmin;
```

위 명령에서 rdsproxyadmin을 데이터베이스 사용자로 변경하여 다른 대상 데이터베이스 사용자가 상태 확인을 수행하도록 허용할 수도 있습니다.

RDS 프록시 사용 시 데이터베이스 사용자 암호 변경

프록시와 연결된 RDS DB 인스턴스에서 데이터베이스 사용자의 암호를 변경해야 할 경우가 있습니다. 그렇다면 해당 Secrets Manager 비밀을 새 암호로 업데이트합니다.

RDS 프록시 연결 고려 사항

연결 설정 구성

RDS 프록시의 연결 풀링을 조정하려면 다음 설정을 수정합니다.

- IdleClientTimeout
- MaxConnectionsPercent
- MaxIdleConnectionsPercent
- ConnectionBorrowTimeout

IdleClientTimeout

프록시에 의해 종료되기 전에 클라이언트 연결이 유휴 상태일 수 있는 시간을 지정합니다. 기본값은 1,800초(30분)입니다.

애플리케이션이 이전 요청이 완료된 후 지정된 시간 내에 새 요청을 제출하지 않으면 클라이언트 연결이 유휴 상태로 간주됩니다. 기본 데이터베이스 연결은 열린 상태를 유지하고 연결 풀로 반환됩니다. 따라서 새 클라이언트 연결에 다시 사용할 수 있습니다. 프록시가 기간 경과 연결을 사전에 제거하도록 하려면 유휴 클라이언트 연결 제한 시간을 줄이는 것이 좋습니다. 워크로드가 프록시와 자주 연결하는 경우 연결 설정 비용을 절약하기 위해 유휴 클라이언트 연결 제한 시간을 늘리세요.

이 설정은 RDS 콘솔의 유휴 클라이언트 연결 제한 시간(Idle client connection timeout) 필드와 AWS CLI 및 API의 `IdleClientTimeout` 설정으로 표시됩니다. RDS 콘솔에서 유휴 클라이언트 연결 제한 시간(Idle client connection timeout) 필드의 값을 변경하는 방법을 알아보려면 AWS Management Console 섹션을 참조하세요. `IdleClientTimeout` 설정의 값을 변경하는 방법을 알아보려면 CLI 명령 modify-db-proxy 또는 API 작업 ModifyDBProxy를 참조하세요.

MaxConnectionsPercent

RDS 프록시가 대상 데이터베이스와 설정할 수 있는 연결 수를 제한할 수 있습니다. 데이터베이스에 사용할 수 있는 최대 연결의 백분율로 제한을 지정합니다. 이 설정은 RDS 콘솔의 연결 풀 최대 연결(Connection pool maximum connections) 필드와 AWS CLI 또는 API의 `MaxConnectionsPercent` 설정으로 표시됩니다.

MaxConnectionsPercent 값은 대상 그룹에서 사용하는 RDS DB 인스턴스에 대한 max_connections 설정의 백분율로 표시됩니다. 프록시가 이러한 연결을 모두 미리 생성하지는 않습니다. 이 설정을 통해 프록시는 워크로드에 필요한 연결을 설정할 수 있습니다.

예를 들어 등록된 데이터베이스 대상이 max_connections가 1000으로 설정되어 있고 MaxConnectionsPercent가 95로 설정된 경우, RDS 프록시는 950개의 연결을 해당 데이터베이스 대상에 대한 동시 연결의 상한선으로 설정합니다.

워크로드가 허용된 최대 데이터베이스 연결 수에 도달할 경우 발생하는 일반적인 부작용은 전체 쿼리 지연 시간이 늘어나고, 그와 함께 DatabaseConnectionsBorrowLatency 지표도 증가한다는 점입니다. DatabaseConnections 및 MaxDatabaseConnectionsAllowed 지표를 비교하여 현재 사용된 데이터베이스 연결 수와 허용된 총 데이터베이스 연결 수를 모니터링할 수 있습니다.

이 파라미터를 설정할 때는 다음과 같은 모범 사례를 고려하세요.

- 워크로드 패턴의 변동에 대비하여 충분한 연결 여유 용량을 확보하세요. 파라미터를 최근에 모니터링한 최대 사용량보다 30% 이상 높게 설정하는 것이 좋습니다. RDS 프록시는 데이터베이스 연결 할당량을 여러 노드에 재분배하므로, 차용 지연 시간이 늘어나는 것을 방지하기 위해 내부 용량 변경 시 추가 연결을 위한 최소 30%의 여유 용량이 필요할 수 있습니다.
- RDS 프록시는 빠른 장애 조치, 트래픽 라우팅, 내부 작업을 지원하기 위해 활성 모니터링을 위한 특정 수의 연결을 예약합니다. MaxDatabaseConnectionsAllowed 지표에는 이러한 예약된 연결이 포함되지 않습니다. 이는 워크로드를 처리하는 데 사용 가능한 연결 수를 나타내며 MaxConnectionsPercent 설정에서 파생된 값보다 낮을 수 있습니다.

  최소 권장 MaxConnectionsPercent 값

  - db.t3.small: 30
  - db.t3.medium 이상: 20

RDS 콘솔에서 연결 풀 최대 연결(Connection pool maximum connections) 필드의 값을 변경하는 방법을 알아보려면 AWS Management Console 섹션을 참조하세요. MaxConnectionsPercent 설정의 값을 변경하는 방법을 알아보려면 CLI 명령 modify-db-proxy-target-group 또는 API 작업 ModifyDBProxyTargetGroup을 참조하세요.

데이터베이스 연결 한도에 대한 자세한 내용은 최대 데이터베이스 연결 수를 참조하세요.

MaxIdleConnectionsPercent

RDS 프록시가 연결 풀에서 유지할 수 있는 유휴 데이터베이스 연결 수를 제어할 수 있습니다. 기본적으로, RDS 프록시는 5분 동안 연결에 대한 활동이 없으면 풀의 데이터베이스 연결을 유휴 상태로 간주합니다.

`MaxIdleConnectionsPercent` 값은 RDS DB 인스턴스 대상 그룹에 대한 `max_connections` 설정의 백분율로 표시됩니다. 기본값은 `MaxConnectionsPercent`의 50%이고 상한은 `MaxConnectionsPercent` 값입니다. 예를 들어, `MaxConnectionsPercent`가 80인 경우 `MaxIdleConnectionsPercent`의 기본값은 40입니다. `MaxConnectionsPercent` 값이 지정되지 않은 경우 `MaxIdleConnectionsPercent`는 5(SQL Server)와 50(기타 모든 엔진)입니다.

값이 높으면 프록시가 유휴 데이터베이스 연결 비율을 높게 유지할 수 있습니다. 값이 낮으면 프록시가 높은 유휴 데이터베이스 연결 비율을 닫습니다. 워크로드를 예측할 수 없는 경우 `MaxIdleConnectionsPercent`에 높은 값을 설정하는 것을 고려해 보세요. 이렇게 하면 RDS 프록시가 새 데이터베이스 연결을 많이 열지 않고도 급증하는 활동을 수용할 수 있습니다.

이 설정은 AWS CLI 또는 API에서 `DBProxyTargetGroup`의 `MaxIdleConnectionsPercent` 설정으로 표시됩니다. `MaxIdleConnectionsPercent` 설정의 값을 변경하는 방법을 알아보려면 CLI 명령 modify-db-proxy-target-group 또는 API 작업 ModifyDBProxyTargetGroup을 참조하세요.

데이터베이스 연결 한도에 대한 자세한 내용은 최대 데이터베이스 연결 수를 참조하세요.

ConnectionBorrowTimeout

시간 초과 오류를 반환하기 전에 RDS 프록시가 연결 풀의 데이터베이스 연결을 사용할 수 있을 때까지 기다리는 시간을 선택할 수 있습니다. 기본값은 120초입니다. 이 설정은 연결 수가 최대일 때, 즉 연결 풀에서 사용할 수 있는 연결이 없을 때 적용됩니다. 이는 또한 장애 조치 작업이 진행 중인 경우와 같이 요청을 처리할 수 있는 적절한 데이터베이스 인스턴스가 없는 경우에도 적용됩니다. 이 설정을 사용하면 애플리케이션 코드에서 쿼리 시간 초과를 변경하지 않고도 애플리케이션에 가장 적합한 대기 기간을 설정할 수 있습니다.

이 설정은 RDS 콘솔의 연결 차용 제한 시간(Connection borrow timeout) 필드나 AWS CLI 또는 API의 `DBProxyTargetGroup`의 `ConnectionBorrowTimeout` 설정으로 표시됩니다. RDS 콘솔에서 연결 차용 제한 시간(Connection borrow timeout) 필드의 값을 변경하는 방법을 알아보려면 AWS Management Console 섹션을 참조하세요. `ConnectionBorrowTimeout` 설정의 값을 변경하는 방법을 알아보려면 CLI 명령 modify-db-proxy-target-group 또는 API 작업 ModifyDBProxyTargetGroup을 참조하세요.

클라이언트 및 데이터베이스 연결

애플리케이션에서 RDS 프록시로의 연결을 클라이언트 연결이라고 합니다. 프록시에서 데이터베이스로의 연결은 데이터베이스 연결입니다. RDS 프록시를 사용하는 경우 데이터베이스 연결은 RDS 프록시 내에서 관리되는 반면 클라이언트 연결은 프록시에서 종료됩니다.

애플리케이션 측 연결 풀링은 애플리케이션과 RDS 프록시 간의 반복적인 연결 설정을 줄이는 이점을 제공할 수 있습니다.

애플리케이션 측 연결 풀을 구현하기 전에 다음과 같은 구성 측면을 고려하세요.

- 클라이언트 연결 최대 수명: RDS 프록시는 클라이언트 연결의 최대 수명을 24시간으로 제한합니다. 이 값은 구성할 수 없습니다. 예상치 못한 클라이언트 연결 끊김을 방지하려면 최대 연결 수명을 24시간 미만으로 설정하여 풀을 구성하세요.
- 클라이언트 연결 유휴 제한 시간: RDS 프록시는 클라이언트 연결에 최대 유휴 시간을 적용합니다. 예상치 못한 연결 끊김을 방지하려면 RDS 프록시의 클라이언트 연결 유휴 제한 시간 설정보다 낮은 값으로 풀의 유휴 연결 제한 시간을 구성하세요.

애플리케이션 측 연결 풀에 구성된 최대 클라이언트 연결 수를 RDS 프록시의 max_connections 설정으로 제한할 필요는 없습니다.

클라이언트 연결 풀링을 사용하면 클라이언트 연결 수명이 길어집니다. 연결이 고정되는 경우 클라이언트 연결을 풀링하면 멀티플렉싱 효율성이 저하될 수 있습니다. 고정되어 있지만 애플리케이션 측 연결 풀에서 유휴 상태인 클라이언트 연결은 데이터베이스 연결을 계속 유지하고 다른 클라이언트 연결에서 데이터베이스 연결을 재사용하지 못하게 합니다. 프록시 로그를 검토하여 연결 고정이 발생하는지 확인하세요.

> **Note**
>
> RDS 프록시는 더 이상 사용되지 않을 경우 24시간 후에 데이터베이스 연결을 닫습니다. 프록시는 최대 유휴 연결 설정 값에 관계없이 이 작업을 수행합니다.

RDS 프록시 고정 방지

멀티플렉싱은 데이터베이스 요청이 이전 요청의 상태 정보에 의존하지 않을 때 더 효율적입니다. 이 경우 RDS Proxy는 각 트랜잭션이 완료될 때 연결을 다시 사용할 수 있습니다. 이러한 상태 정보의 예로는 SET 또는 SELECT 문을 통해 변경할 수 있는 대부분의 변수 및 구성 파라미터가 있습니다. 클라이언트 연결에 대한 SQL 트랜잭션은 기본적으로 기본 데이터베이스 연결 간에 멀티플렉싱할 수 있습니다.

프록시에 대한 연결은 고정이라는 상태로 들어갈 수 있습니다. 연결이 고정되면 이후의 각 트랜잭션은 세션이 끝날 때까지 동일한 기본 데이터베이스 연결을 사용합니다. 다른 클라이언트 연결도 세션이 끝날 때까지 해당 데이터베이스 연결을 다시 사용할 수 없습니다. 클라이언트 연결이 끊어지면 세션이 종료됩니다.

RDS Proxy는 다른 세션에 적합하지 않은 세션 상태 변경을 감지하면 클라이언트 연결을 특정 DB 연결에 자동으로 고정합니다. 고정은 연결 재사용의 효과를 줄입니다. 모든 또는 거의 모든 연결이 고정되는 경우 애플리케이션 코드 또는 워크로드를 수정하여 고정을 유발하는 조건을 줄이는 것이 좋습니다.

예를 들어, 애플리케이션이 세션 변수 또는 구성 파라미터를 변경한다고 가정하겠습니다. 이 경우 이후의 문이 효력을 발휘하기 위해 새 변수 또는 파라미터를 사용할 수 있습니다. 따라서 RDS Proxy는 세션 변수 또는 구성 설정을 변경하라는 요청을 처리할 때 해당 세션을 DB 연결에 고정합니다. 이렇게 하면 동일한 세션의 모든 이후 트랜잭션에 대해 세션 상태가 유효하게 유지됩니다.

일부 데이터베이스의 경우 이 규칙은 설정할 수 있는 모든 파라미터에 적용되지는 않습니다. RDS 프록시는 특정 문과 변수를 추적합니다. 따라서 RDS 프록시는 이들을 수정할 때 세션을 고정하지 않습니다. 이 경우 RDS 프록시는 해당 설정에 대해 동일한 값을 가진 다른 세션에 대해서만 연결을 재사용합니다. RDS 프록시가 데이터베이스 엔진에서 추적하는 항목에 대한 자세한 내용은 다음을 참조하세요.

- RDS 프록시가 RDS for SQL Server 데이터베이스에서 추적하는 대상
- RDS 프록시가 RDS for MariaDB 및 RDS for MySQL 데이터베이스에서 추적하는 대상

RDS 프록시가 RDS for SQL Server 데이터베이스에서 추적하는 대상

다음은 RDS 프록시가 추적하는 SQL Server 문입니다.

- `USE`
- `SET ANSI_NULLS`
- `SET ANSI_PADDING`
- `SET ANSI_WARNINGS`
- `SET ARITHABORT`
- `SET CONCAT_NULL_YIELDS_NULL`
- `SET CURSOR_CLOSE_ON_COMMIT`
- `SET DATEFIRST`
- `SET DATEFORMAT`
- `SET LANGUAGE`

- SET LOCK_TIMEOUT
- SET NUMERIC_ROUNDABORT
- SET QUOTED_IDENTIFIER
- SET TEXTSIZE
- SET TRANSACTION ISOLATION LEVEL

RDS 프록시가 RDS for MariaDB 및 RDS for MySQL 데이터베이스에서 추적하는 대상

다음은 RDS 프록시가 추적하는 MariaDB 및 MySQL 문입니다.

- DROP DATABASE
- DROP SCHEMA
- USE

다음은 RDS 프록시가 추적하는 MySQL 및 MariaDB 변수입니다.

- AUTOCOMMIT
- AUTO_INCREMENT_INCREMENT
- CHARACTER SET (or CHAR SET)
- CHARACTER_SET_CLIENT
- CHARACTER_SET_DATABASE
- CHARACTER_SET_FILESYSTEM
- CHARACTER_SET_CONNECTION
- CHARACTER_SET_RESULTS
- CHARACTER_SET_SERVER
- COLLATION_CONNECTION
- COLLATION_DATABASE
- COLLATION_SERVER
- INTERACTIVE_TIMEOUT
- NAMES
- NET_WRITE_TIMEOUT
- QUERY_CACHE_TYPE

- SESSION_TRACK_SCHEMA
- SQL_MODE
- TIME_ZONE
- TRANSACTION_ISOLATION (or TX_ISOLATION)
- TRANSACTION_READ_ONLY (or TX_READ_ONLY)
- WAIT_TIMEOUT

고정 최소화

RDS 프록시 성능 튜닝에는 고정을 최소화하여 트랜잭션 수준 연결 재사용(멀티플렉싱)을 최대화하려는 시도가 포함됩니다.

다음 작업을 수행하여 고정을 최소화할 수 있습니다.

- 고정을 야기할 수 있는 불필요한 데이터베이스 요청을 방지합니다.
- 모든 연결에서 변수와 구성 설정을 일관되게 설정합니다. 그렇게 하면 이후 세션에서 특정 설정이 있는 연결을 재사용할 가능성이 증가합니다.

  그러나 PostgreSQL의 경우 변수를 설정하면 세션이 고정됩니다.
- MySQL 엔진 패밀리 데이터베이스의 경우 프록시에 세션 고정 필터를 적용합니다. 이렇게 해도 애플리케이션의 올바른 작동에 영향을 주지 않음이 확인될 경우 특정 유형의 작업을 세션 고정에서 제외할 수 있습니다.
- Amazon CloudWatch 지표 중 DatabaseConnectionsCurrentlySessionPinned를 모니터링하여 고정이 얼마나 자주 발생하는지 확인합니다. 이 지표와 기타 CloudWatch 지표에 대한 자세한 내용은 Amazon CloudWatch를 사용한 RDS 프록시 지표 모니터링 단원을 참조하십시오.
- SET 문을 사용하여 각 클라이언트 연결에 대해 동일한 초기화를 수행하는 경우 트랜잭션 수준 멀티플렉싱을 유지하면서 그렇게 할 수 있습니다. 이 경우 초기 세션 상태를 설정하는 문을 프록시에서 사용하는 초기화 쿼리로 이동합니다. 이 속성은 세미콜론으로 구분된 하나 이상의 SQL 문을 포함하는 문자열입니다.

  예를 들어, 특정 구성 파라미터를 설정하는 프록시에 대한 초기화 질의를 정의할 수 있습니다. 그러면 RDS Proxy는 해당 프록시에 대해 새 연결을 설정할 때마다 해당 설정을 적용합니다. 애플리케이션 코드에서 해당 SET 문을 제거하여 트랜잭션 수준 멀티플렉싱을 방해하지 않도록 할 수 있습니다.

  특정 프록시에서 고정이 발생하는 빈도에 대한 지표를 보려면 Amazon CloudWatch를 사용한 RDS 프록시 지표 모니터링 단원을 참조하십시오.

모든 엔진 패밀리에 대해 고정을 유발하는 조건

다음과 같이 멀티플렉싱으로 예기치 않은 동작이 발생할 수 있는 상황에서 프록시는 세션을 현재 연결에 고정합니다.

- 텍스트 크기가 16KB보다 큰 명령문을 사용하면 프록시가 세션을 고정합니다.

RDS for Microsoft SQL Server에서 고정이 발생하는 조건

RDS for SQL Server의 경우 다음 상호 작용도 고정을 유발합니다.

- 여러 개의 활성 결과 집합(MARS) 사용 사용자 이름에 대한 자세한 정보는 SQL Server 설명서를 참조하세요.
- 분산 트랜잭션 코디네이터(DTC) 통신 사용
- 임시 테이블, 트랜잭션, 커서 또는 준비된 문 생성
- 다음 SET 문 사용
  - SET ANSI_DEFAULTS
  - SET ANSI_NULL_DFLT
  - SET ARITHIGNORE
  - SET DEADLOCK_PRIORITY
  - SET FIPS_FLAGGER
  - SET FMTONLY
  - SET FORCEPLAN
  - SET IDENTITY_INSERT
  - SET NOCOUNT
  - SET NOEXEC
  - SET OFFSETS
  - SET PARSEONLY
  - SET QUERY_GOVERNOR_COST_LIMIT
  - SET REMOTE_PROC_TRANSACTIONS
  - SET ROWCOUNT
  - SET SHOWPLAN_ALL, SHOWPLAN_TEXT 및 SHOWPLAN_XML
  - SET STATISTICS

- SET XACT_ABORT

RDS for MariaDB 및 RDS for MySQL에서 고정이 발생하는 조건

MariaDB 및 MySQL의 경우 다음 상호 작용으로 인한 고정도 발생합니다.

- 명시적 테이블 잠금 문인 LOCK TABLE, LOCK TABLES 또는 FLUSH TABLES WITH READ LOCK을 사용하면 프록시가 세션을 고정합니다.
- GET_LOCK을 사용하여 명명된 잠금을 만들면 프록시가 세션을 고정합니다.
- 사용자 변수 또는 시스템 변수(일부 예외)를 설정하면 프록시가 세션을 고정합니다. 이 경우 연결 재사용이 너무 줄어들면 SET 작업이 고정되지 않도록 선택합니다. 세션 고정 필터 속성을 설정하여 이를 수행하는 방법에 대한 자세한 내용은 Amazon RDS용 프록시 만들기 및 RDS 프록시 수정 섹션을 참조하세요.
- 임시 테이블을 생성하면 프록시가 세션을 고정합니다. 이렇게 하면 트랜잭션 경계에 관계없이 임시 테이블의 내용이 세션 전체에서 보존됩니다.
- 함수 ROW_COUNT, FOUND_ROWS 및 LAST_INSERT_ID를 호출하면 고정을 야기하는 경우가 있습니다.
- 준비된 문을 사용하면 프록시가 세션을 고정합니다. 이 규칙은 준비된 문이 SQL 텍스트를 사용하는지 이진 프로토콜을 사용하는지 여부를 적용합니다.
- SET LOCAL을 사용할 경우 RDS 프록시는 연결을 고정하지 않습니다.
- 저장 프로시저 및 저장 함수를 호출해도 고정이 발생하지 않습니다. RDS 프록시는 이러한 호출로 인한 세션 상태 변경을 감지하지 못합니다. 트랜잭션 전반에 걸쳐 세션 상태를 유지하려고 해당 세션 상태에 의존하는 경우 저장된 루틴 내에서 애플리케이션이 세션 상태를 변경하지 않는지 확인합니다. 예를 들어, RDS 프록시는 현재 모든 트랜잭션에 걸쳐 지속되는 임시 테이블을 생성하는 저장 프로시저와 호환되지 않습니다.

애플리케이션 동작에 대한 전문 지식이 있는 경우 특정 애플리케이션 문에서 고정 동작을 건너뛸 수 있습니다. 그러려면 프록시를 생성할 때 세션 고정 필터 옵션을 선택하면 됩니다. 현재 세션 변수 및 구성 설정을 설정하기 위해 세션 고정을 옵트아웃할 수 있습니다.

RDS for PostgreSQL에서 고정이 발생하는 조건

PostgreSQL의 경우 다음 상호 작용으로 인해 고정도 발생합니다.

- SET 명령 사용.
- PREPARE, DISCARD, DEALLOCATE 또는 EXECUTE 명령을 사용하여 준비된 문 관리.

- 임시 시퀀스, 테이블 또는 뷰 생성.
- 커서 선언.
- 세션 상태 무시.
- 알림 채널에서 수신.
- `auto_explain`과 같은 라이브러리 모듈 로드.
- `nextval` 및 `setval`과 같은 함수를 사용하여 시퀀스 조작.
- `pg_advisory_lock` 및 `pg_try_advisory_lock`과 같은 함수를 사용하여 잠금과 상호 작용.

> **Note**
>
> RDS 프록시는 트랜잭션 수준 권고 잠금(특히 `pg_advisory_xact_lock`, `pg_advisory_xact_lock_shared`, `pg_try_advisory_xact_lock`, `pg_try_advisory_xact_lock_shared`)을 고정하지 않습니다.

- 파라미터 설정 또는 파라미터를 기본값으로 재설정. 특히 `SET` 및 `set_config` 명령을 사용하여 세션 변수에 기본값 할당.
- 저장 프로시저 및 저장 함수를 호출해도 고정이 발생하지 않습니다. RDS 프록시는 이러한 호출로 인한 세션 상태 변경을 감지하지 못합니다. 트랜잭션 전반에 걸쳐 세션 상태를 유지하려고 해당 세션 상태에 의존하는 경우 저장된 루틴 내에서 애플리케이션이 세션 상태를 변경하지 않는지 확인합니다. 예를 들어, RDS 프록시는 현재 모든 트랜잭션에 걸쳐 지속되는 임시 테이블을 생성하는 저장 프로시저와 호환되지 않습니다.

RDS 프록시 삭제

더 이상 필요하지 않은 프록시는 삭제할 수 있습니다. 또는 프록시와 연결된 DB 인스턴스 또는 클러스터를 서비스 중단 상태로 전환하면 프록시를 삭제할 수 있습니다.

AWS Management Console

프록시를 삭제하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Proxies(프록시)를 선택합니다.
3. 목록에서 삭제할 프록시를 선택합니다.
4. Delete Proxy(프록시 삭제)를 선택합니다.

AWS CLI

DB 프록시를 삭제하려면 AWS CLI 명령 delete-db-proxy를 사용합니다. 관련 연결을 제거하려면 deregister-db-proxy-targets 명령도 사용합니다.

```
aws rds delete-db-proxy --name proxy_name
```

```
aws rds deregister-db-proxy-targets
    --db-proxy-name proxy_name
    [--target-group-name target_group_name]
    [--target-ids comma_separated_list]       # or
    [--db-instance-identifiers instance_id]       # or
    [--db-cluster-identifiers cluster_id]
```

RDS API

DB 프록시를 삭제하려면 Amazon RDS API 함수 DeleteDBProxy를 호출합니다. 관련 항목 및 연결을 삭제하려면 DeleteDBProxyTargetGroup 및 DeregisterDBProxyTargets 함수를 호출합니다.

Amazon RDS 프록시 엔드포인트 작업

RDS Proxy 엔드포인트는 데이터베이스 연결을 관리하는 유연하고 효율적인 방법을 제공하여 확장성, 가용성 및 보안을 개선합니다. 프록시 엔드포인트를 사용하면 다음을 수행할 수 있습니다.

- 모니터링 및 문제 해결 간소화 – 여러 엔드포인트를 사용하여 여러 애플리케이션의 연결을 독립적으로 모니터링하고 문제를 해결합니다.

주제

- 프록시 엔드포인트 개요
- 프록시 엔드포인트에 대한 제한 사항
- 다중 AZ DB 클러스터의 프록시 엔드포인트
- VPC 간에 RDS 데이터베이스 액세스
- 프록시 엔드포인트 생성
- 프록시 엔드포인트 보기
- 프록시 엔드포인트 수정
- 프록시 엔드포인트 삭제

프록시 엔드포인트 개요

RDS 프록시 엔드포인트 작업을 수행하려면 RDS 인스턴스 엔드포인트와 동일한 절차를 따릅니다. RDS 엔드포인트에 익숙하지 않은 경우 MySQL 데이터베이스 엔진 기반 DB 인스턴스에 연결하기 및 PostgreSQL 데이터베이스 엔진 기반 DB 인스턴스에 연결하기에서 자세한 내용을 참조하세요.

프록시 엔드포인트를 만들면 엔드포인트를 프록시의 VPC와 다른 가상 프라이빗 클라우드(VPC)와 연결할 수 있습니다. 이렇게 하면 조직 내 다른 애플리케이션에서 사용하는 VPC와 같은 다른 VPC에서 프록시에 연결할 수 있습니다.

프록시 엔드포인트과 관련한 제한 사항에 대한 자세한 내용은 프록시 엔드포인트에 대한 제한 사항 섹션을 참조하세요.

RDS Proxy 로그에서 각 항목에는 연결된 프록시 엔드포인트의 이름이 접두사로 붙습니다. 이 이름은 사용자 정의 엔드포인트에 대해 지정한 이름이거나, 프록시의 기본 읽기/쓰기 엔드포인트에 대한 특수한 이름 `default`일 수 있습니다.

각 프록시 엔드포인트에는 일련의 자체 CloudWatch 지표가 있습니다. 모든 프록시 엔드포인트, 특정 엔드포인트 또는 프록시의 모든 읽기/쓰기 또는 읽기 전용 엔드포인트에 대한 지표를 모니터링합니다. 자세한 내용은 Amazon CloudWatch를 사용한 RDS 프록시 지표 모니터링 섹션을 참조하세요.

프록시 엔드포인트는 연결된 프록시와 동일한 인증 메커니즘을 사용합니다. RDS Proxy는 연결된 프록시의 속성과 일치하는 사용자 정의 엔드포인트에 대한 권한 및 승인을 자동으로 설정합니다.

프록시 엔드포인트에 대한 제한 사항

RDS 프록시 엔드포인트에는 다음과 같은 제한 사항이 있습니다.

- RDS 프록시 기본 엔드포인트는 수정할 수 없습니다.
- 프록시의 사용자 정의 엔드포인트의 최대 개수는 20개입니다. 따라서 프록시는 최대 21개의 엔드포인트(기본 엔드포인트와 사용자가 생성하는 20개)를 가질 수 있습니다.
- 추가 엔드포인트를 프록시와 연결할 때 RDS Proxy는 클러스터에서 각 엔드포인트에 사용할 DB 인스턴스를 자동으로 결정합니다.

프록시를 생성하면 RDS는 애플리케이션과 데이터베이스 간의 보안 통신을 위해 VPC 엔드포인트를 자동으로 만듭니다. VPC 엔드포인트는 Amazon VPC 콘솔에서 보고 액세스할 수 있습니다.

새 프록시 엔드포인트를 추가하면 AWS PrivateLink 인터페이스 엔드포인트가 프로비저닝됩니다. 프록시에 엔드포인트를 하나 이상 추가하면 추가 요금이 발생합니다. 자세한 내용은 RDS Proxy Pricing을 참조하세요.

다중 AZ DB 클러스터의 프록시 엔드포인트

기본적으로, 다중 AZ DB 클러스터에 RDS 프록시를 사용할 때 연결하는 엔드포인트에는 읽기/쓰기 기능이 있습니다. 따라서 이 엔드포인트는 모든 요청을 클러스터의 라이터 인스턴스로 보냅니다. 이러한 모든 연결은 라이터 인스턴스의 `max_connections` 값에 포함됩니다. 프록시가 다중 AZ DB 클러스터에 연결되어 있는 경우, 해당 프록시에 대해 추가 읽기/쓰기 또는 읽기 전용 엔드포인트를 생성할 수 있습니다.

읽기 전용 쿼리를 위해 프록시와 함께 읽기 전용 엔드포인트를 사용할 수 있습니다. 다중 AZ DB 클러스터에 리더 엔드포인트를 사용하는 것과 동일한 방식으로 이를 수행합니다. 이렇게 하면 하나 이상의 리더 DB 인스턴스가 있는 다중 AZ DB 클러스터의 읽기 확장성을 활용할 수 있습니다. 읽기 전용 엔드포인트를 사용하고 필요에 따라 다중 AZ DB 클러스터에 리더 DB 인스턴스를 더 추가하여 동시 쿼리를 더 많이 실행하고 동시 연결을 더 많이 생성할 수 있습니다. 이러한 리더 엔드포인트는 쿼리 집약적 애플리케이션의 읽기 확장성을 개선하는 데 도움이 됩니다. 또한 리더 엔드포인트는 클러스터의 리더 DB 인스턴스를 사용할 수 없게 될 경우 연결 가용성을 높이는 데에도 도움이 됩니다.

다중 AZ DB 클러스터의 리더 엔드포인트

RDS Proxy에서 리더 엔드포인트를 생성하고 사용할 수 있습니다. 하지만 이러한 엔드포인트는 다중 AZ DB 클러스터에 연결된 프록시에 대해서만 작동합니다. RDS CLI 또는 API를 사용하는 경우 값이 `TargetRole`인 `READ_ONLY` 속성이 표시될 수 있습니다. 프록시의 대상을 RDS DB 인스턴스에서 다중 AZ DB 클러스터로 변경하여 이러한 프록시를 활용할 수 있습니다.

다중 AZ DB 클러스터에 RDS 프록시를 사용할 경우 리더 엔드포인트라는 읽기 전용 엔드포인트를 생성하고 연결할 수 있습니다.

리더 엔드포인트가 애플리케이션 가용성을 높이는 방법

경우에 따라 클러스터에 있는 리더 인스턴스를 사용할 수 없게 될 수 있습니다. 이 경우 DB 프록시의 리더 엔드포인트를 사용하는 연결은 다중 AZ DB 클러스터 리더 엔드포인트를 사용하는 연결보다 더 빠르게 복구됩니다. RDS 프록시는 클러스터에서 사용 가능한 리더 인스턴스로만 연결을 라우팅합니다. 인스턴스를 사용할 수 없게 될 때 DNS 캐싱으로 인해 지연이 발생하지 않습니다.

연결이 다중화된 경우 RDS 프록시는 애플리케이션을 중단하지 않고 후속 쿼리를 다른 리더 인스턴스로 보냅니다. 리더 인스턴스가 사용할 수 없는 상태인 경우 해당 인스턴스 엔드포인트에 대한 모든 클라이언트 연결이 닫힙니다.

연결이 고정된 경우 연결에 대한 다음 쿼리에서 오류가 반환됩니다. 그러나 애플리케이션은 동일한 프록시 엔드포인트에 즉시 다시 연결할 수 있습니다. RDS Proxy는 `available` 상태에 있는 다른 리더

DB 인스턴스로 연결을 라우팅합니다. 수동으로 다시 연결하면 RDS 프록시가 이전 리더 인스턴스와 새 리더 인스턴스 간의 복제 지연을 확인하지 않습니다.

다중 AZ DB 클러스터에 사용 가능한 리더 인스턴스가 없는 경우, RDS 프록시는 리더 엔드포인트가 사용 가능해지면 연결을 시도합니다. 연결 대여 제한 시간 내에 리더 인스턴스가 사용 가능한 상태가 되지 않으면 연결 시도가 실패합니다. 리더 인스턴스를 사용할 수 있게 되면 연결 시도가 성공합니다.

리더 엔드포인트가 쿼리 확장성을 높이는 방법

프록시용 리더 엔드포인트는 다음과 같은 방법으로 다중 AZ DB 클러스터 쿼리 확장성을 높입니다.

- 가능한 경우 RDS Proxy는 특정 리더 엔드포인트 연결을 사용하는 모든 쿼리 문제에 대해 동일한 리더 DB 인스턴스를 사용합니다. 이렇게 하면 동일한 테이블에 있는 일련의 관련 쿼리가 특정 DB 인스턴스에서 캐싱, 계획 최적화 등을 활용할 수 있습니다.
- Reader DB 인스턴스를 사용할 수 없게 되면 세션이 멀티플렉싱되었는지 고정되어 있는지에 따라 애플리케이션에 미치는 영향이 달라집니다. 세션이 멀티플렉싱된 경우 RDS Proxy는 후속 쿼리를 사용자의 작업 없이 다른 리더 DB 인스턴스로 라우팅합니다. 세션이 고정되어 있으면 애플리케이션에 오류가 발생하여 다시 연결해야 합니다. 리더 엔드포인트에 즉시 다시 연결할 수 있으며 RDS Proxy는 연결을 사용 가능한 리더 DB 인스턴스로 라우팅합니다. 프록시 세션의 멀티플렉싱 및 고정에 대한 자세한 내용은 RDS Proxy 개념 개요 섹션을 참조하세요.

VPC 간에 RDS 데이터베이스 액세스

기본적으로 RDS 기술 스택의 구성 요소는 모두 동일한 Amazon VPC에 있습니다. 예를 들어 Amazon EC2 인스턴스에서 실행되는 애플리케이션이 Amazon RDS DB 인스턴스에 연결한다고 가정합니다. 이 경우 애플리케이션 서버와 데이터베이스가 모두 동일한 VPC 내에 있어야 합니다.

RDS 프록시를 사용하면 EC2 인스턴스와 같은 다른 VPC의 리소스에서 한 VPC에 있는 Amazon RDS DB 인스턴스에 대한 액세스를 설정할 수 있습니다. 예를 들어 조직에 동일한 데이터베이스 리소스에 액세스하는 애플리케이션이 여러 개 있을 수 있습니다. 각 애플리케이션은 자체 VPC에 있을 수 있습니다.

VPC 간 액세스를 활성화하려면 프록시를 위한 새 엔드포인트를 생성합니다. 프록시 자체는 Amazon RDS DB 인스턴스와 동일한 VPC에 상주합니다. 하지만 VPC 간 엔드포인트는 EC2 인스턴스 등의 다른 리소스와 함께 다른 VPC에 상주합니다. VPC 간 엔드포인트는 EC2 및 기타 리소스와 동일한 VPC의 서브넷 및 보안 그룹과 연결됩니다. 이러한 연결을 사용하면 VPC 제한으로 인해 데이터베이스에 액세스할 수 없는 애플리케이션이 엔드포인트에 연결할 수 있습니다.

다음 단계에서는 RDS Proxy를 통해 VPC 간 엔드포인트를 생성하고 액세스하는 방법을 설명합니다.

1. VPC 2개를 생성하거나 RDS 작업에 이미 사용 중인 VPC 2개를 선택합니다. 각 VPC에는 인터넷 게이트웨이, 라우팅 테이블, 서브넷 및 보안 그룹과 같은 연결된 자체 네트워크 리소스가 있어야 합니다. VPC가 하나만 있는 경우 Amazon RDS 시작하기에서 RDS를 성공적으로 사용하기 위해 다른 VPC를 설정하는 단계를 참조하세요. 또한 Amazon EC2 콘솔에서 기존 VPC를 조사하여 함께 연결할 리소스의 종류를 확인할 수 있습니다.
2. 연결할 Amazon RDS DB 인스턴스에 연결된 DB 프록시를 생성합니다. Amazon RDS용 프록시 만들기의 절차를 따르십시오.
3. RDS 콘솔에서 프록시의 [세부 정보(Details)] 페이지에 있는 [프록시 엔드포인트(Proxy endpoints)] 섹션에서 [엔드포인트 생성(Create endpoint)]을 선택합니다. 프록시 엔드포인트 생성의 절차를 따르십시오.
4. VPC 간 엔드포인트를 읽기/쓰기와 읽기 전용 중 무엇으로 설정할지 선택합니다.
5. Amazon RDS DB 인스턴스와 동일한 VPC인 기본값을 사용하지 않고, 다른 VPC를 선택합니다. 이 VPC는 프록시가 상주하는 VPC와 동일한 AWS 리전에 있어야 합니다.
6. 이제 Amazon RDS DB 인스턴스와 동일한 VPC의 서브넷 및 보안 그룹에 대한 기본값을 사용하지 않고 새로 선택합니다. 선택한 VPC의 서브넷과 보안 그룹을 기준으로 선택합니다.
7. Secrets Manager 보안 정보에 대한 설정은 변경할 필요가 없습니다. 각 엔드포인트가 속한 VPC에 관계없이 프록시의 모든 엔드포인트에 대해 동일한 자격 증명이 작동합니다.
8. 새 엔드포인트가 [사용 가능(Available)] 상태로 바뀔 때까지 기다립니다.
9. 전체 엔드포인트 이름을 기록해 둡니다. 이 값은 데이터베이스 애플리케이션의 연결 문자열의 일부로 제공하는 *Region_name*`.rds.amazonaws.com`으로 끝납니다.
10. 엔드포인트와 동일한 VPC에 있는 리소스에서 새 엔드포인트에 액세스합니다. 이 프로세스를 테스트하는 간단한 방법은 이 VPC에 새 EC2 인스턴스를 생성하는 것입니다. 그런 다음 EC2 인스턴스에 로그인하고 `mysql` 또는 `psql` 명령을 실행하여 연결 문자열의 엔드포인트 값을 통해 연결합니다.

프록시 엔드포인트 생성

프록시 엔드포인트를 생성하려면 다음 지침을 따르세요.

콘솔

프록시 엔드포인트를 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.

2. 탐색 창에서 Proxies(프록시)를 선택합니다.
3. 새 엔드포인트를 생성하려는 프록시의 이름을 클릭합니다.

   해당 프록시에 대한 세부 정보 페이지가 나타납니다.
4. [프록시 엔드포인트(Proxy endpoints)] 섹션에서 [프록시 엔드포인트 생성(Create proxy endpoint)]을 선택합니다.

   [프록시 엔드포인트 생성(Create proxy endpoint)] 창이 나타납니다.
5. [프록시 엔드포인트 이름(Proxy endpoint name)]에 원하는 알기 쉬운 이름을 입력합니다.
6. [대상 역할(Target role)]에서 엔드포인트를 읽기/쓰기와 읽기 전용 중 무엇으로 설정할지 선택합니다.

   읽기/쓰기 엔드포인트를 사용하는 연결은 데이터 정의 언어(DDL) 문, 데이터 조작 언어(DML) 문, 쿼리 등 모든 종류의 작업을 수행할 수 있습니다. 이러한 엔드포인트는 항상 RDS DB 클러스터의 기본 인스턴스에 연결합니다. 애플리케이션에서 단일 엔드포인트만 사용하는 경우 일반 데이터베이스 작업에 읽기/쓰기 엔드포인트를 사용할 수 있습니다. 관리 작업, OLTP(온라인 트랜잭션 처리) 애플리케이션 및 ETL(추출 변환 로드) 작업에 읽기/쓰기 엔드포인트를 사용할 수도 있습니다.

   읽기 전용 엔드포인트를 사용하는 연결은 쿼리만 수행할 수 있습니다. RDS 프록시는 엔드포인트에 대한 각 연결마다 리더 인스턴스 중 하나를 사용할 수 있습니다. 이렇게 하면 쿼리 집약적인 애플리케이션이 다중 AZ DB 클러스터의 클러스터링 기능을 활용할 수 있습니다. 이러한 읽기 전용 연결은 클러스터의 기본 인스턴스에 오버헤드를 발생시키지 않습니다. 이렇게 하면 보고 및 분석 쿼리가 OLTP 애플리케이션의 쓰기 작업 속도를 저하시키지 않습니다.
7. Virtual Private Cloud(VPC)의 경우, 일반적으로 프록시 또는 연결된 데이터베이스에 액세스하는 데 사용하는 것과 동일한 EC2 인스턴스 또는 기타 리소스에서 엔드포인트에 액세스하려면 기본값을 선택합니다. 이 프록시에 대해 VPC 간 액세스를 설정하려면 기본값 이외의 VPC를 선택합니다. VPC 간 액세스에 대한 자세한 내용은 VPC 간에 RDS 데이터베이스 액세스 섹션을 참조하세요.
8. [서브넷(Subnets)]의 경우 RDS Proxy는 기본적으로 연결된 프록시와 동일한 서브넷을 채웁니다. VPC의 주소 범위 중 일부만 연결할 수 있도록 엔드포인트에 대한 액세스를 제한하려면 하나 이상의 서브넷을 제거합니다.
9. VPC 보안 그룹의 경우 기존 보안 그룹을 선택하거나 새 보안 그룹을 생성할 수 있습니다. RDS Proxy는 기본적으로 연결된 프록시와 동일한 보안 그룹 또는 그룹을 채웁니다. 프록시의 인바운드 및 아웃바운드 규칙이 이 엔드포인트에 적합한 경우, 기본 선택 항목을 그대로 두면 됩니다.

   새 보안 그룹을 생성하도록 선택한 경우 이 페이지에서 보안 그룹의 이름을 지정합니다. 그런 다음 EC2 콘솔에서 보안 그룹 설정을 편집합니다.
10. [프록시 엔드포인트 생성(Create proxy endpoint)]을 선택합니다.

AWS CLI

프록시 엔드포인트를 생성하려면 AWS CLI create-db-proxy-endpoint 명령을 사용합니다.

다음 필수 파라미터를 포함합니다.

- `--db-proxy-name` *value*
- `--db-proxy-endpoint-name` *value*
- `--vpc-subnet-ids` *list_of_ids*. 서브넷 ID를 공백으로 구분합니다. VPC 자체의 ID는 지정하지 않습니다.

다음과 같은 선택적 파라미터도 포함할 수 있습니다.

- `--target-role { READ_WRITE | READ_ONLY }`이 파라미터의 기본값은 입니다.`READ_WRITE` 프록시가 라이터 DB 인스턴스만 포함된 다중 AZ DB 클러스터에 연결되어 있는 경우 `READ_ONLY`를 지정할 수 없습니다. 다중 AZ DB 클러스터에서 읽기 전용 엔드포인트의 용도에 대한 자세한 내용은 다중 AZ DB 클러스터의 리더 엔드포인트 섹션을 참조하세요.
- `--vpc-security-group-ids` *value*. 보안 그룹 ID를 공백으로 구분합니다. 이 파라미터를 생략하면 RDS Proxy가 VPC에 기본 보안 그룹을 사용합니다. RDS Proxy는 `--vpc-subnet-ids` 파라미터에 대해 지정한 서브넷 ID를 기반으로 VPC를 확인합니다.

Example

다음 예에서는 `my-endpoint`라는 프록시 엔드포인트를 생성합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-proxy-endpoint \
  --db-proxy-name my-proxy \
  --db-proxy-endpoint-name my-endpoint \
  --vpc-subnet-ids subnet_id subnet_id subnet_id ... \
  --target-role READ_ONLY \
  --vpc-security-group-ids security_group_id ]
```

Windows의 경우:

```
aws rds create-db-proxy-endpoint ^
  --db-proxy-name my-proxy ^
```

```
  --db-proxy-endpoint-name my-endpoint ^
  --vpc-subnet-ids subnet_id_1 subnet_id_2 subnet_id_3 ... ^
  --target-role READ_ONLY ^
  --vpc-security-group-ids security_group_id
```

RDS API

프록시 엔드포인트를 생성하려면 RDS API CreateDBProxyEndpoint 작업을 사용합니다.

프록시 엔드포인트 보기

기존 프록시 엔드포인트를 보려면 다음 지침을 따르세요.

콘솔

프록시 엔드포인트에 대한 세부 정보를 보려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Proxies(프록시)를 선택합니다.
3. 목록에서 엔드포인트를 보려는 프록시를 선택합니다. 프록시 이름을 클릭하여 세부 정보 페이지를 봅니다.
4. [프록시 엔드포인트(Proxy endpoints)] 섹션에서 보려는 엔드포인트를 선택합니다. 세부 정보 페이지를 보려면 이름을 클릭합니다.
5. 값을 확인하고 싶은 파라미터를 조사합니다. 다음과 같은 속성을 확인할 수 있습니다.
   - 엔드포인트가 읽기/쓰기인지 아니면 읽기 전용인지 여부.
   - 데이터베이스 연결 문자열에 사용하는 엔드포인트 주소.
   - 엔드포인트와 연결된 VPC, 서브넷 및 보안 그룹.

AWS CLI

하나 이상의 프록시 엔드포인트를 보려면 AWS CLI describe-db-proxy-endpoints 명령을 사용합니다.

다음과 같은 선택적 파라미터를 포함할 수 있습니다.

- `--db-proxy-endpoint-name`
- `--db-proxy-name`

다음 예에서는 `my-endpoint` 프록시 엔드포인트를 설명합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-proxy-endpoints \
  --db-proxy-endpoint-name my-endpoint
```

Windows의 경우:

```
aws rds describe-db-proxy-endpoints ^
  --db-proxy-endpoint-name my-endpoint
```

RDS API

하나 이상의 프록시 엔드포인트를 설명하려면 RDS API DescribeDBProxyEndpoints 작업을 사용합니다.

프록시 엔드포인트 수정

프록시 엔드포인트를 수정하려면 다음 지침을 따르세요.

콘솔

하나 이상의 프록시 엔드포인트를 수정하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Proxies(프록시)를 선택합니다.
3. 목록에서 엔드포인트를 수정할 프록시를 선택합니다. 프록시 이름을 클릭하여 세부 정보 페이지를 봅니다.
4. [프록시 엔드포인트(Proxy endpoints)] 섹션에서 수정할 엔드포인트를 선택합니다. 목록에서 선택하거나 이름을 클릭하여 세부 정보 페이지를 볼 수 있습니다.
5. 프록시 세부 정보 페이지의 [프록시 엔드포인트(Proxy endpoints)] 섹션에서 [편집(Edit)]을 선택합니다. 또는 프록시 엔드포인트 세부 정보 페이지의 작업에서 편집을 선택합니다.
6. 수정할 파라미터의 값을 변경합니다.
7. Save changes(변경 사항 저장)를 선택합니다.

AWS CLI

프록시 엔드포인트를 수정하려면 AWS CLI modify-db-proxy-endpoint 명령을 다음 필수 파라미터와 함께 사용합니다.

- `--db-proxy-endpoint-name`

다음 파라미터 중 하나 이상을 사용하여 엔드포인트 속성에 대한 변경 사항을 지정합니다.

- `--new-db-proxy-endpoint-name`
- `--vpc-security-group-ids`. 보안 그룹 ID를 공백으로 구분합니다.

다음 예에서는 `my-endpoint` 프록시 엔드포인트의 이름을 `new-endpoint-name`으로 변경합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-proxy-endpoint \
  --db-proxy-endpoint-name my-endpoint \
  --new-db-proxy-endpoint-name new-endpoint-name
```

Windows의 경우:

```
aws rds modify-db-proxy-endpoint ^
  --db-proxy-endpoint-name my-endpoint ^
  --new-db-proxy-endpoint-name new-endpoint-name
```

RDS API

프록시 엔드포인트를 수정하려면 RDS API ModifyDBProxyEndpoint 작업을 사용합니다.

프록시 엔드포인트 삭제

프록시의 엔드포인트를 삭제하려면 다음 지침을 따르세요.

> **Note**
>
> RDS 프록시가 각 프록시에 대해 자동으로 생성하는 기본 프록시 엔드포인트는 삭제할 수 없습니다.

프록시를 삭제하면 RDS Proxy가 모든 연결된 엔드포인트를 자동으로 삭제합니다.

콘솔

AWS Management Console을 사용하여 프록시 엔드포인트를 삭제하려면

1. 탐색 창에서 Proxies(프록시)를 선택합니다.
2. 목록에서 엔드포인트를 삭제할 프록시를 선택합니다. 프록시 이름을 클릭하여 세부 정보 페이지를 봅니다.
3. [프록시 엔드포인트(Proxy endpoints)] 섹션에서 삭제할 엔드포인트를 선택합니다. 목록에서 하나 이상의 엔드포인트를 선택하거나, 단일 엔드포인트의 이름을 클릭하여 세부 정보 페이지를 볼 수 있습니다.
4. 프록시 세부 정보 페이지의 [프록시 엔드포인트(Proxy endpoints)] 섹션에서 [삭제(Delete)]를 선택합니다. 또는 프록시 엔드포인트 세부 정보 페이지의 작업에서 삭제를 선택합니다.

AWS CLI

프록시 엔드포인트를 삭제하려면 다음 필수 파라미터를 사용하여 delete-db-proxy-endpoint 명령을 실행합니다.

- `--db-proxy-endpoint-name`

다음 명령은 `my-endpoint`라는 프록시 엔드포인트를 삭제합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds delete-db-proxy-endpoint \
  --db-proxy-endpoint-name my-endpoint
```

Windows의 경우:

```
aws rds delete-db-proxy-endpoint ^
  --db-proxy-endpoint-name my-endpoint
```

RDS API

RDS API를 사용하여 프록시 엔드포인트를 삭제하려면 DeleteDBProxyEndpoint 작업을 실행합니다. `DBProxyEndpointName` 파라미터에 프록시 엔드포인트의 이름을 지정합니다.

Amazon CloudWatch를 사용한 RDS 프록시 지표 모니터링

Amazon CloudWatch를 사용하여 RDS Proxy를 모니터링할 수 있습니다. CloudWatch는 프록시에서 원시 데이터를 수집하여 읽기 쉬운 실시간 지표로 처리합니다. CloudWatch 콘솔에서 이러한 지표를 찾으려면 지표를 선택한 다음 RDS, Per-Proxy Metrics(프록시별 지표)를 차례로 선택합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch 지표 사용을 참조하세요.

> **Note**
>
> RDS는 프록시와 연결된 각 기본 Amazon EC2 인스턴스에 대해 이러한 지표를 게시합니다. 하나의 프록시가 둘 이상의 EC2 인스턴스에서 사용될 수 있습니다. CloudWatch 통계를 사용하여 연관된 모든 인스턴스에서 프록시 값을 집계합니다.
> 이러한 지표 중 일부는 프록시에 의한 첫 번째 연결이 성공하기 전까지 표시되지 않을 수 있습니다.

RDS Proxy 로그에서 각 항목에는 연결된 프록시 엔드포인트의 이름이 접두사로 붙습니다. 이 이름은 사용자 정의 엔드포인트에 대해 지정한 이름이거나 읽기/쓰기 요청을 수행하는 프록시의 기본 엔드포인트에 대한 특수 이름(`default`)일 수 있습니다.

모든 RDS Proxy 지표는 `proxy` 그룹에 있습니다.

각 프록시 엔드포인트에는 자체 CloudWatch 지표가 있습니다. 각 프록시 엔드포인트의 사용을 독립적으로 모니터링할 수 있습니다. 프록시 엔드포인트에 대한 자세한 내용은 Amazon RDS 프록시 엔드포인트 작업 섹션을 참조하세요.

다음 측정 기준 세트 중 하나를 사용하여 각 지표의 값을 집계할 수 있습니다. 예를 들어 `ProxyName` 측정 기준 세트를 사용하여 특정 프록시의 모든 트래픽을 분석할 수 있습니다. 다른 차원 세트를 사용하면 여러 가지 방식으로 지표를 분할할 수 있습니다. 각 프록시의 여러 엔드포인트 또는 대상 데이터베이스나, 읽기/쓰기 및 읽기 전용 트래픽을 기준으로 지표를 분할할 수 있습니다.

- 차원 세트 1: `ProxyName`
- 차원 세트 2: `ProxyName`, `EndpointName`
- 차원 세트 3: `ProxyName`, `TargetGroup`, `Target`
- 차원 세트 4: `ProxyName`, `TargetGroup`, `TargetRole`

측정치	설명	유효 기간	CloudWatch 측정 기준 세트
`AvailabilityPercentage`	차원이 나타내는 역할에서 대상 그룹을 사용할 수 있었던 시간의 비율입니다. 이 지표는 1분마다 보고됩니다. 이 지표에 가장 유용한 통계는 Sum입니다.	1분	Dimension set 4
`ClientConnections`	현재 클라이언트 연결 수입니다. 이 지표는 1분마다 보고됩니다. 이 지표에 가장 유용한 통계는 Sum입니다.	1분	Dimension set 1, Dimension set 2
`ClientConnectionsClosed`	닫은 클라이언트 연결 수입니다. 이 지표에 가장 유용한 통계는 Sum입니다.	1분 이상	Dimension set 1, Dimension set 2
`ClientConnectionsInSetup`	현재 열려 있지만 설정을 완료하지 않은 클라이언트 연결 수입니다. 이 지표는 1분마다 보고됩니다. 이 지표에 가장 유용한 통계는 합계입니다.	1분	Dimension set 1, Dimension set 2
`ClientConnectionsNoTLS`	TLS(전송 계층 보안)가 없는 현재 클라이언트 연결 수입니다. 이 지표는 1분마다 보고됩니다. 이 지표에 가장 유용한 통계는 Sum입니다.	1분	Dimension set 1, Dimension set 2

측정치	설명	유효 기간	CloudWatch 측정 기준 세트
ClientConnectionsReceived	수신된 클라이언트 연결 요청 수입니다. 이 지표에 가장 유용한 통계는 Sum입니다.	1분 이상	Dimension set 1, Dimension set 2
ClientConnectionsSetupFailedAuth	잘못 구성된 인증 또는 TLS로 인해 실패한 클라이언트 연결 시도 횟수입니다. 이 지표에 가장 유용한 통계는 Sum입니다.	1분 이상	Dimension set 1, Dimension set 2
ClientConnectionsSetupSucceeded	TLS를 사용하거나 사용하지 않는 인증 메커니즘을 통해 성공적으로 설정된 클라이언트 연결 수입니다. 이 지표에 가장 유용한 통계는 Sum입니다.	1분 이상	Dimension set 1, Dimension set 2
ClientConnectionsTLS	현재 TLS를 사용하는 클라이언트 연결 수입니다. 이 지표는 1분마다 보고됩니다. 이 지표에 가장 유용한 통계는 Sum입니다.	1분	Dimension set 1, Dimension set 2
DatabaseConnectionRequests	데이터베이스 연결을 생성하기 위한 요청 수입니다. 이 지표에 가장 유용한 통계는 Sum입니다.	1분 이상	Dimension set 1, Dimension set 3, Dimension set 4

측정치	설명	유효 기간	CloudWatch 측정 기준 세트
DatabaseConnectionRequestsWithTLS	TLS를 사용하는 데이터베이스 연결을 생성하기 위한 요청 수입니다. 이 지표에 가장 유용한 통계는 Sum입니다.	1분 이상	Dimension set 1, Dimension set 3, Dimension set 4
DatabaseConnections	현재 데이터베이스 연결 수입니다. 이 지표는 1분마다 보고됩니다. 이 지표에 가장 유용한 통계는 Sum입니다.	1분	Dimension set 1, Dimension set 3, Dimension set 4
DatabaseConnectionsBorrowLatency	모니터링되는 프록시가 데이터베이스 연결을 가져오는 데 걸리는 시간(마이크로초)입니다. 이 지표에 가장 유용한 통계는 Sum입니다.	1분 이상	Dimension set 1, Dimension set 2
DatabaseConnectionsCurrentlyBorrowed	현재 대여 상태인 데이터베이스 연결 수입니다. 이 지표는 1분마다 보고됩니다. 이 지표에 가장 유용한 통계는 Sum입니다.	1분	Dimension set 1, Dimension set 3, Dimension set 4

측정치	설명	유효 기간	CloudWatch 측정 기준 세트
DatabaseConnectionsCurrentlyInTransaction	현재 트랜잭션의 데이터베이스 연결 수입니다. 이 지표는 1분마다 보고됩니다. 이 지표에 가장 유용한 통계는 Sum입니다.	1분	Dimension set 1, Dimension set 3, Dimension set 4
DatabaseConnectionsCurrentlySessionPinned	세션 상태를 변경하는 클라이언트 요청의 작업으로 인해 현재 고정된 데이터베이스 연결 수입니다. 이 지표는 1분마다 보고됩니다. 이 지표에 가장 유용한 통계는 Sum입니다.	1분	Dimension set 1, Dimension set 3, Dimension set 4
DatabaseConnectionsSetupFailed	실패한 데이터베이스 연결 요청 수입니다. 이 지표에 가장 유용한 통계는 Sum입니다.	1분 이상	Dimension set 1, Dimension set 3, Dimension set 4
DatabaseConnectionsSetupSucceeded	TLS를 사용하거나 사용하지 않고 성공적으로 설정된 데이터베이스 연결 수입니다. 이 지표에 가장 유용한 통계는 Sum입니다.	1분 이상	Dimension set 1, Dimension set 3, Dimension set 4
DatabaseConnectionsWithTLS	현재 TLS를 사용하는 데이터베이스 연결 수입니다. 이 지표는 1분마다 보고됩니다. 이 지표에 가장 유용한 통계는 Sum입니다.	1분	Dimension set 1, Dimension set 3, Dimension set 4

측정치	설명	유효 기간	CloudWatch 측정 기준 세트
MaxDatabaseConnectionsAllowed	허용되는 최대 데이터베이스 연결 수입니다. 이 지표는 1분마다 보고됩니다. 이 지표에 가장 유용한 통계는 Sum입니다.	1분	Dimension set 1, Dimension set 3, Dimension set 4
QueryDatabaseResponseLatency	데이터베이스가 쿼리에 응답하는 데 걸린 시간(마이크로초)입니다. 이 지표에 가장 유용한 통계는 Average입니다.	1분 이상	Dimension set 1, Dimension set 2, Dimension set 3, Dimension set 4
QueryRequests	수신된 쿼리 수입니다. 여러 문을 포함하는 쿼리는 하나의 쿼리로 계산됩니다. 이 지표에 가장 유용한 통계는 Sum입니다.	1분 이상	Dimension set 1, Dimension set 2
QueryRequestsNoTLS	TLS 이외의 연결에서 받은 쿼리 수입니다. 여러 문을 포함하는 쿼리는 하나의 쿼리로 계산됩니다. 이 지표에 가장 유용한 통계는 Sum입니다.	1분 이상	Dimension set 1, Dimension set 2

측정치	설명	유효 기간	CloudWatch 측정 기준 세트
QueryRequestsTLS	TLS 연결에서 받은 쿼리 수입니다. 여러 문을 포함하는 쿼리는 하나의 쿼리로 계산됩니다. 이 지표에 가장 유용한 통계는 Sum입니다.	1분 이상	Dimension set 1, Dimension set 2
QueryResponseLatency	쿼리 요청을 받는 시간과 프록시가 응답하는 시간 사이의 시간(마이크로 초)입니다. 이 지표에 가장 유용한 통계는 Average입니다.	1분 이상	Dimension set 1, Dimension set 2

AWS Management Console의 CloudWatch 아래에서 RDS Proxy 활동 로그를 찾을 수 있습니다. 각 프록시는 로그 그룹 페이지에 항목이 있습니다.

> **Important**
>
> 이러한 로그는 프로그래밍 방식 액세스가 아니라 사용자가 문제 해결을 위해 사용하기 위한 로그입니다. 로그의 형식과 내용은 변경될 수 있습니다.
> 특히 이전 로그에는 각 요청의 엔드포인트를 나타내는 접두사가 포함되어 있지 않습니다. 최신 로그에서는 각 항목에 연결된 프록시 엔드포인트의 이름이 접두사로 붙습니다. 이 이름은 사용자 정의 엔드포인트에 대해 지정한 이름이거나, 프록시의 기본 엔드포인트를 사용한 요청을 위한 default라는 특수한 이름일 수 있습니다.

RDS 프록시 이벤트 작업

이벤트는 서비스형 소프트웨어(SaaS) 파트너로부터 AWS 환경, 서비스 또는 애플리케이션과 같은 환경의 변화를 나타냅니다. 또는 사용자 지정 애플리케이션이나 서비스 중 하나일 수 있습니다. 예를 들어 Amazon RDS에서는 RDS 프록시를 생성하거나 수정할 때 이벤트를 생성합니다. Amazon RDS는

거의 실시간으로 Amazon EventBridge에 이벤트를 전송합니다. 다음에서 구독할 수 있는 RDS 프록시 이벤트 목록과 RDS Proxy 이벤트의 예를 찾을 수 있습니다.

이벤트 작업에 대한 자세한 내용은 다음을 참조하세요.

- AWS Management Console, AWS CLI 또는 RDS API를 사용하여 이벤트를 보는 방법에 대한 지침은 Amazon RDS 이벤트 보기 섹션을 참조하세요.
- EventBridge로 이벤트를 전송하도록 Amazon RDS를 구성하는 방법을 알아보려면 Amazon RDS 이벤트에서 트리거되는 규칙 생성 섹션을 참조하세요.

RDS 프록시 이벤트

다음 표에서는 RDS 프록시가 소스 유형일 때 이벤트 범주와 이벤트 목록을 보여줍니다.

범주	RDS 이벤트 ID	메시지	참고
구성 변경	RDS-EVENT-0204	RDS가 DB 프록시(##)를 수정했습니다.	None
구성 변경	RDS-EVENT-0207	RDS가 DB 프록시(##)의 엔드포인트를 수정했습니다.	None
구성 변경	RDS-EVENT-0213	RDS가 DB 인스턴스 추가를 감지하여 DB 프록시(##)의 대상 그룹에 자동으로 추가했습니다.	None
구성 변경	RDS-EVENT-0214	RDS가 DB 인스턴스(##)의 삭제를 감지하여 DB 프록시(##)의 대상 그룹(##)에서 자동으로 제거했습니다.	None
구성 변경	RDS-EVENT-0215	RDS가 DB 클러스터(##)의 삭제를 감지하여 DB 프록시(##)의 대상 그룹(##)에서 자동으로 제거했습니다.	None

범주	RDS 이벤트 ID	메시지	참고
생성	RDS-EVENT-0203	RDS가 DB 프록시(##)를 생성했습니다.	None
생성	RDS-EVENT-0206	RDS가 DB 프록시(##)에 대한 엔드포인트(##)을 생성했습니다.	None
삭제	RDS-EVENT-0205	RDS가 DB 프록시(##)를 삭제했습니다.	None
삭제	RDS-EVENT-0208	RDS가 DB 프록시(##)에 대한 엔드포인트(##)을 삭제했습니다.	None
실패	RDS-EVENT-0243	서브넷(##)에서 사용할 수 있는 IP 주소가 충분하지 않기 때문에 RDS가 프록시(##)의 용량을 프로비저닝하지 못했습니다. 이러한 문제를 해결하려면 RDS 프록시 설명서의 권장 사항에 따라 서브넷에 최소한의 미사용 IP 주소 개수가 있는지 확인하세요.	권장 인스턴스 클래스 수를 확인하려면 IP 주소 용량 계획 섹션을 참조하세요.
실패	RDS-EVENT-0275	RDS가 DB 프록시 ##에 대한 일부 연결을 제한했습니다. 클라이언트에서 프록시로의 동시 연결 요청 수가 제한을 초과했습니다.	None

다음은 JSON 형식의 RDS 프록시 이벤트의 예입니다. 이벤트는 RDS가 `my-rds-proxy`라는 RDS 프록시의 `my-endpoint`라는 엔드포인트를 수정했음을 보여줍니다. 이벤트 ID는 RDS-EVENT-0207입니다.

```
{
  "version": "0",
  "id": "68f6e973-1a0c-d37b-f2f2-94a7f62ffd4e",
  "detail-type": "RDS DB Proxy Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-09-27T22:36:43Z",
  "region": "us-east-1",
  "resources": [
     "arn:aws:rds:us-east-1:123456789012:db-proxy:my-rds-proxy"
  ],
  "detail": {
    "EventCategories": [
      "configuration change"
    ],
    "SourceType": "DB_PROXY",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:db-proxy:my-rds-proxy",
    "Date": "2018-09-27T22:36:43.292Z",
    "Message": "RDS modified endpoint my-endpoint of DB Proxy my-rds-proxy.",
    "SourceIdentifier": "my-endpoint",
    "EventID": "RDS-EVENT-0207"
  }
}
```

RDS 프록시 문제 해결

다음에서 몇 가지 일반적인 RDS 프록시 문제에 대한 문제 해결 아이디어와 RDS 프록시에 대한 CloudWatch 로그에 대한 정보를 찾을 수 있습니다.

RDS 프록시 로그에서 각 항목에는 연결된 프록시 엔드포인트의 이름이 접두사로 붙습니다. 이 이름은 사용자 정의 엔드포인트에 지정한 이름일 수 있습니다. 아니면 읽기/쓰기 요청을 수행하는 프록시의 기본 엔드포인트에 대한 특수 이름(`default`)일 수 있습니다. 프록시 엔드포인트에 대한 자세한 내용은 Amazon RDS 프록시 엔드포인트 작업 섹션을 참조하세요.

주제

- 프록시에 대한 연결 확인
- 일반적인 문제 및 해결 방법
- RDS for MySQL 관련 RDS 프록시 문제 해결
- RDS for PostgreSQL 관련 RDS 프록시 문제 해결

프록시에 대한 연결 확인

다음 명령을 사용하여 연결 내의 모든 구성 요소(예: 프록시, 데이터베이스, 컴퓨팅 인스턴스)가 서로 통신할 수 있는지 확인할 수 있습니다.

describe-db-proxies 명령을 사용하여 프록시 자체를 검사합니다. 또한 describe-db-proxy-target-groups 명령을 사용하여 연결된 대상 그룹을 검사합니다. 대상의 세부 정보가 프록시와 연결하려는 RDS DB 인스턴스와 일치하는지 확인합니다. 다음과 같은 명령을 사용합니다.

```
aws rds describe-db-proxies --db-proxy-name $DB_PROXY_NAME
aws rds describe-db-proxy-target-groups --db-proxy-name $DB_PROXY_NAME
```

프록시가 기본 데이터베이스에 연결할 수 있는지 확인하려면 describe-db-proxy-targets 명령을 사용하여 대상 그룹에 지정된 대상을 검사합니다. 다음과 같은 명령을 사용합니다.

```
aws rds describe-db-proxy-targets --db-proxy-name $DB_PROXY_NAME
```

describe-db-proxy-targets 명령의 출력에는 `TargetHealth` 필드가 포함됩니다. `State` 내부의 `Reason`, `Description` 및 `TargetHealth` 필드를 검사하여 프록시가 기본 DB 인스턴스와 통신할 수 있는지 확인할 수 있습니다.

- 프록시가 DB 인스턴스에 연결할 수 있는 `State`의 `AVAILABLE` 값입니다.
- `State`의 `UNAVAILABLE` 값은 임시 또는 영구 연결 문제를 나타냅니다. 이 경우 `Reason` 및 `Description` 필드를 검사합니다. 예를 들어 `Reason`의 값이 `PENDING_PROXY_CAPACITY`인 경우 프록시가 조정 작업을 완료한 후 다시 연결해 보십시오. `Reason`의 값이 `UNREACHABLE`, `CONNECTION_FAILED` 또는 `AUTH_FAILURE`인 경우 `Description` 필드의 설명을 사용하여 문제를 진단합니다.
- `State` 또는 `REGISTERING`로 변경하기 전에 `AVAILABLE` 필드의 값이 잠시 동안 `UNAVAILABLE`일 수 있습니다.

다음 Netcat 명령(nc)이 성공하면 로그인한 EC2 인스턴스 또는 다른 시스템에서 프록시 엔드포인트에 액세스할 수 있습니다. 이 명령은 프록시 및 연결된 데이터베이스와 동일한 VPC에 있지 않은 경우 실패를 보고합니다. 동일한 VPC에 있지 않아도 데이터베이스에 직접 로그인할 수 있습니다. 그러나 동일한 VPC에 있지 않으면 프록시에 로그인할 수는 없습니다.

```
nc -zx MySQL_proxy_endpoint 3306
```

```
nc -zx PostgreSQL_proxy_endpoint 5432
```

다음 명령을 사용하여 EC2 인스턴스에 필수 속성이 있는지 확인할 수 있습니다. 특히 EC2 인스턴스의 VPC는 프록시가 연결하는 의 VPC와 동일해야 합니다.

```
aws ec2 describe-instances --instance-ids your_ec2_instance_id
```

프록시에 사용되는 Secrets Manager 비밀을 검사합니다.

```
aws secretsmanager list-secrets
aws secretsmanager get-secret-value --secret-id your_secret_id
```

`get-secret-value`에서 표시한 `SecretString` 필드가 `username` 및 `password` 필드를 포함하는 JSON 문자열로 인코딩되어 있는지 확인합니다. 다음 예제는 `SecretString` 필드의 형식을 보여 줍니다.

```
{
  "ARN": "some_arn",
  "Name": "some_name",
  "VersionId": "some_version_id",
  "SecretString": '{"username":"some_username","password":"some_password"}',
  "VersionStages": [ "some_stage" ],
  "CreatedDate": some_timestamp
}
```

일반적인 문제 및 해결 방법

이 섹션에서는 RDS 프록시를 사용할 때 발생하는 몇 가지 일반적인 문제와 잠재적 해결 방법에 대해 설명합니다.

`aws rds describe-db-proxy-targets` CLI 명령을 실행한 후 `TargetHealth` 설명에 `Proxy does not have any registered credentials`라고 표시되면 다음을 확인하세요.

- 사용자가 프록시에 액세스할 수 있도록 등록된 보안 인증 정보가 있습니다.
- 프록시에서 사용하는 Secrets Manager 보안 암호에 액세스하는 IAM 역할은 유효합니다.

DB 프록시를 생성하거나 DB 프록시에 연결하는 동안 다음과 같은 RDS 이벤트가 발생할 수 있습니다.

범주	RDS 이벤트 ID	설명
실패	RDS-EVENT-0243	서브넷에서 사용할 수 있는 IP 주소가 충분하지 않기 때문에 RDS가 프록시 용량을 프로비저닝할 수 없습니다. 이러한 문제를 해결하려면 서브넷에 최소한의 미사용 IP 주소가 있는지 확인합니다. 권장 인스턴스 클래스 수를 확인하려면 IP 주소 용량 계획 섹션을 참조하세요.
실패	RDS-EVENT-0275	RDS가 DB 프록시 ##에 대한 일부 연결을 제한했습니다. 클라이언트에서 프록시로의 동시 연결 요청 수가 제한을 초과했습니다.

새 프록시를 생성하거나 프록시에 연결하는 동안 다음과 같은 문제가 발생할 수 있습니다.

오류	원인 또는 해결 방법
`403: The security token included in the request is invalid`	새 IAM 역할을 생성하는 대신 기존 IAM 역할을 선택합니다.

RDS for MySQL 관련 RDS 프록시 문제 해결

MySQL 프록시에 연결하는 동안 다음과 같은 문제가 발생할 수 있습니다.

오류	원인 또는 해결 방법
`ERROR 1040 (HY000): Connections rate limit exceeded (limit_value )`	클라이언트에서 프록시로의 연결 요청 속도가 제한을 초과했습니다.
`ERROR 1040 (HY000): IAM authentication rate limit exceeded`	클라이언트에서 프록시로의 IAM 인증을 사용하는 동시 요청 수가 제한을 초과했습니다.
`ERROR 1040 (HY000): Number simultaneous connections exceeded (limit_value )`	클라이언트에서 프록시로의 동시 연결 요청 수가 제한을 초과했습니다.
`ERROR 1045 (28000): Access denied for user 'DB_USER'@'%' (using password: YES)`	프록시에서 사용하는 Secrets Manager 비밀이 기존 데이터베이스 사용자의 사용자 이름 및 암호와 일치하지 않습니다. Secrets Manager 비밀의 자격 증명을 업데이트하거나 데이터베이스 사용자가 존재하고 비밀과 동일한 암호를 가지고 있는지 확인합니다.
`ERROR 1105 (HY000): Unknown error`	알 수 없는 오류가 발생했습니다.
`ERROR 1231 (42000): Variable ''character_set_cl`	`character_set_client` 파라미터에 설정된 값이 유효하지 않습니다. 예를 들어 `ucs2` 값은 MySQL 서버를 충돌시킬 수 있으므로 유효하지 않습니다.

오류	원인 또는 해결 방법
`ient'' can't be set to the value of value`	
`ERROR 3159 (HY000): This RDS Proxy requires TLS connections.`	프록시에서 전송 계층 보안 필요 설정을 활성화했지만 연결에 MySQL 클라이언트의 파라미터 `ssl-mode=DISABLED` 가 포함되었습니다. 다음 중 하나를 수행하십시오. • 프록시에 대해 전송 계층 보안 필요 설정을 비활성화합니다. • PostgreSQL 클라이언트에서 `ssl-mode=REQUIRED` 의 최소 설정을 사용하여 데이터베이스에 연결합니다.
`ERROR 2026 (HY000): SSL connection error: Internal Server Error`	프록시에 대한 TLS 핸드셰이크가 실패했습니다. 몇 가지 가능한 이유는 다음과 같습니다. • SSL은 필요하지만 서버는 이를 지원하지 않습니다. • 내부 서버 오류가 발생했습니다. • 잘못된 핸드셰이크가 발생했습니다.
`ERROR 9501 (HY000): Timed-out waiting to acquire database connection`	데이터베이스 연결을 얻기 위해 대기 중인 프록시 시간이 초과되었습니다. 몇 가지 가능한 이유는 다음과 같습니다. • 최대 연결에 도달했기 때문에 프록시가 데이터베이스 연결을 설정할 수 없습니다. • 데이터베이스를 사용할 수 없으므로 프록시가 데이터베이스 연결을 설정할 수 없습니다.

RDS for PostgreSQL 관련 RDS 프록시 문제 해결

PostgreSQL 프록시에 연결하는 동안 다음과 같은 문제가 발생할 수 있습니다.

오류	원인	솔루션
`ERROR 28000: IAM authentication is`	사용자가 PostgreSQL 클라이언트에서 `sslmode=d`	사용자는 PostgreSQL 클라이언트에서 `sslmode=r`

오류	원인	솔루션
`allowed only with SSL connections.`	`isable` 설정이 있는 IAM 인증을 사용하여 데이터베이스에 연결하려고 했습니다.	`equire` 의 최소 설정을 사용하여 데이터베이스에 연결해야 합니다. 자세한 내용은 PostgreSQL SSL 지원 설명서를 참조하십시오.
`ERROR 28000: This RDS proxy has no credentials for the role` *`role_name`* `. Check the credentials for this role and try again.`	이 역할에 대한 Secrets Manager 암호가 없습니다.	이 역할에 대한 Secrets Manager 암호를 추가하세요. 자세한 내용은 RDS 프록시에 대한 AWS Identity and Access Management(IAM) 정책 설정 섹션을 참조하세요.
`ERROR 28000: RDS supports only IAM, MD5, or SCRAM authentication.`	프록시에 연결하는 데 사용되는 데이터베이스 클라이언트가 프록시에서 현재 지원하지 않는 인증 메커니즘을 사용하고 있습니다.	IAM 인증을 사용하지 않는 경우 MD5 또는 SCRAM 암호 인증을 사용하세요.
`ERROR 28000: A user name is missing from the connection startup packet. Provide a user name for this connection.`	연결을 설정하려고 할 때 프록시에 연결하는 데 사용되는 데이터베이스 클라이언트가 사용자 이름을 전송하지 않습니다.	선택한 PostgreSQL 클라이언트를 사용하여 프록시에 대한 연결을 설정할 때 사용자 이름을 정의해야 합니다.
`ERROR 28000: IAM is allowed only with SSL connections.`	클라이언트가 IAM 인증을 사용하여 연결을 시도했지만 SSL이 활성화되지 않았습니다.	PostgreSQL 클라이언트에서 SSL을 활성화하세요.

오류	원인	솔루션
`ERROR 28000: This RDS Proxy requires TLS connections.`	사용자가 전송 계층 보안 필요 설정을 활성화했지만 PostgreSQL 클라이언트에서 `sslmode=disable` 로 연결하려고 시도했습니다.	이 오류를 해결하려면 다음 중 하나를 수행합니다. • 프록시의 전송 계층 보안 필요 옵션을 비활성화합니다. • PostgreSQL 클라이언트에서 `sslmode=allow` 의 최소 설정을 사용하여 데이터베이스에 연결합니다.
`ERROR 28P01: IAM authentication failed for user` *`user_name`* `. Check the IAM token for this user and try again.`	이 오류는 다음과 같은 이유 때문일 수 있습니다. • 클라이언트가 잘못된 IAM 사용자 이름을 제공했습니다. • 클라이언트가 사용자에게 잘못된 IAM 권한 부여 토큰을 제공했습니다. • 클라이언트가 필요한 권한이 없는 IAM 정책을 사용하고 있습니다. • 클라이언트가 사용자에게 만료된 IAM 권한 부여 토큰을 제공했습니다.	이 오류를 해결하려면 다음을 수행하십시오. 1. 제공된 IAM 사용자가 있는지 확인합니다. 2. IAM 권한 부여 토큰이 제공된 IAM 사용자에 속하는지 확인합니다. 3. IAM 정책에 RDS에 대한 적절한 권한이 있는지 확인합니다. 4. 사용된 IAM 권한 부여 토큰의 유효성을 확인합니다.
`ERROR 28P01: The password that was provided for the role` *`role_name`* `is wrong.`	이 역할의 암호가 Secrets Manager 암호와 일치하지 않습니다.	Secrets Manager에서 이 역할의 암호를 확인하여 암호가 PostgreSQL 클라이언트에서 사용 중인 암호와 같은지 확인하세요.

오류	원인	솔루션
`ERROR 28P01: The IAM authentication failed for the role role_name . Check the IAM token for this role and try again.`	IAM 인증에 사용되는 IAM 토큰에 문제가 있습니다.	새 인증 토큰을 생성하여 새 연결에 사용하세요.
`ERROR 0A000: Feature not supported: RDS Proxy supports only version 3.0 of the PostgreSQL messaging protocol.`	프록시에 연결하는 데 사용되는 PostgreSQL 클라이언트는 3.0보다 오래된 프로토콜을 사용합니다.	3.0 메시징 프로토콜을 지원하는 최신 PostgreSQL 클라이언트를 사용하세요. PostgreSQL psql CLI를 사용하는 경우 7.4보다 크거나 같은 버전을 사용하세요.
`ERROR 0A000: Feature not supported: RDS Proxy currently doesn't support streaming replication mode.`	프록시에 연결하는 데 사용되는 PostgreSQL 클라이언트가 현재 RDS 프록시에서 지원되지 않는 스트리밍 복제 모드를 사용하려고 합니다.	연결하는 데 사용되는 PostgreSQL 클라이언트에서 스트리밍 복제 모드를 해제하세요.
`ERROR 0A000: Feature not supported: RDS Proxy currently doesn't support the option option_name .`	시작 메시지를 통해 프록시에 연결하는 데 사용되는 PostgreSQL 클라이언트가 현재 RDS 프록시에서 지원되지 않는 옵션을 요청하고 있습니다.	연결하는 데 사용되는 PostgreSQL 클라이언트에서 위의 메시지에서 지원되지 않는 것으로 표시되는 옵션을 해제하세요.
`ERROR 53300: The IAM authentication failed because of too many competing requests.`	클라이언트에서 프록시로의 IAM 인증을 사용하는 동시 요청 수가 제한을 초과했습니다.	PostgreSQL 클라이언트에서 IAM 인증을 사용하는 연결이 설정되는 속도를 줄이세요.

오류	원인	솔루션
`ERROR 53300: The maximum number of client connections to the proxy exceeded number_value .`	클라이언트에서 프록시로의 동시 연결 요청 수가 제한을 초과했습니다.	PostgreSQL 클라이언트에서 이 RDS 프록시로의 활성 연결 수를 줄이세요.
`ERROR 53300: Rate of connection to proxy exceeded number_value .`	클라이언트에서 프록시로의 연결 요청 속도가 제한을 초과했습니다.	PostgreSQL 클라이언트로부터의 연결이 설정되는 속도를 줄이세요.
`ERROR XX000: Unknown error.`	알 수 없는 오류가 발생했습니다.	문제를 조사할 수 있도록 AWS Support에 문의해 주세요.
`ERROR 08000: Timed-out waiting to acquire database connection.`	`ConnectionBorrowTimeout` 설정에 지정된 기간 내에 데이터베이스 연결을 획득하기 위해 대기 중인 프록시 제한 시간입니다. 몇 가지 가능한 이유는 다음과 같습니다. • 최대 연결에 도달했기 때문에 프록시가 데이터베이스 연결을 설정할 수 없습니다. • 데이터베이스를 사용할 수 없거나 데이터베이스에 대한 연결 설정이 구성된 `ConnectionBorrowTimeout` 보다 오래 걸리는 경우 프록시가 데이터베이스 연결을 설정할 수 없습니다.	가능한 해결책은 다음과 같습니다. • 프록시 연결을 고정하지 않도록 합니다. RDS 프록시 고정 방지을(를) 참조하세요. • `ConnectionBorrowTimeout` 및 `MaxConnectionsPercent` 설정을 검토합니다. RDS 프록시 연결 고려 사항을(를) 참조하세요. • 대상 가용성을 검토합니다. (Amazon CloudWatch를 사용한 RDS 프록시 지표 모니터링의 `AvailabilityPercentage` 섹션 참조)

오류	원인	솔루션
`ERROR XX000: Request returned an error: database_error .`	프록시에서 설정된 데이터베이스 연결이 오류를 반환했습니다.	솔루션은 특정 데이터베이스 오류에 따라 다릅니다. 예를 들면 다음과 같습니다. `Request returned an error: database "your-database-name" does not exist` 이는 지정된 데이터베이스 이름이 데이터베이스 서버에 존재하지 않음을 의미합니다. 또는 데이터베이스 이름으로 사용되는 사용자 이름(데이터베이스 이름을 지정하지 않은 경우)이 서버에 존재하지 않는다는 의미입니다.

삭제된 **postgres** 데이터베이스 문제 해결

실수로 인스턴스에서 `postgres` 데이터베이스를 삭제한 경우 데이터베이스를 복원하여 인스턴스에 대한 연결을 복원해야 합니다. DB 인스턴스 내에서 다음 명령을 실행합니다.

```
CREATE DATABASE postgres;
GRANT CONNECT ON DATABASE postgres TO rdsproxyadmin;
```

RDS Proxy를 AWS CloudFormation에서 사용

RDS Proxy를 AWS CloudFormation에서 사용할 수 있습니다. 이렇게 하면 관련 리소스 그룹을 생성하는 데 도움이 됩니다. 이러한 그룹에는 새로 생성된 Amazon RDS DB 인스턴스에 연결할 수 있는 프록시가 포함될 수 있습니다. AWS CloudFormation의 RDS Proxy 지원에는 두 가지 새로운 레지스트리 유형인 `DBProxy` 및 `DBProxyTargetGroup`이 포함됩니다

다음 목록은 RDS Proxy에 대한 샘플 AWS CloudFormation 템플릿을 보여줍니다.

```
Resources:
 DBProxy:
   Type: AWS::RDS::DBProxy
```

```
  Properties:
    DBProxyName: CanaryProxy
    EngineFamily: MYSQL
    RoleArn:
     Fn::ImportValue: SecretReaderRoleArn
    Auth:
      - {AuthScheme: SECRETS, SecretArn: !ImportValue ProxySecret, IAMAuth: DISABLED}
    VpcSubnetIds:
      Fn::Split: [",", "Fn::ImportValue": SubnetIds]

ProxyTargetGroup:
  Type: AWS::RDS::DBProxyTargetGroup
  Properties:
    DBProxyName: CanaryProxy
    TargetGroupName: default
    DBInstanceIdentifiers:
      - Fn::ImportValue: DBInstanceName
  DependsOn: DBProxy
```

이 샘플의 리소스에 대한 자세한 내용은 DBProxy 및 DBProxyTargetGroup을 참조하세요.

AWS CloudFormation을 사용하여 생성할 수 있는 리소스에 대한 자세한 내용은 RDS 리소스 유형 참조를 확인하세요.

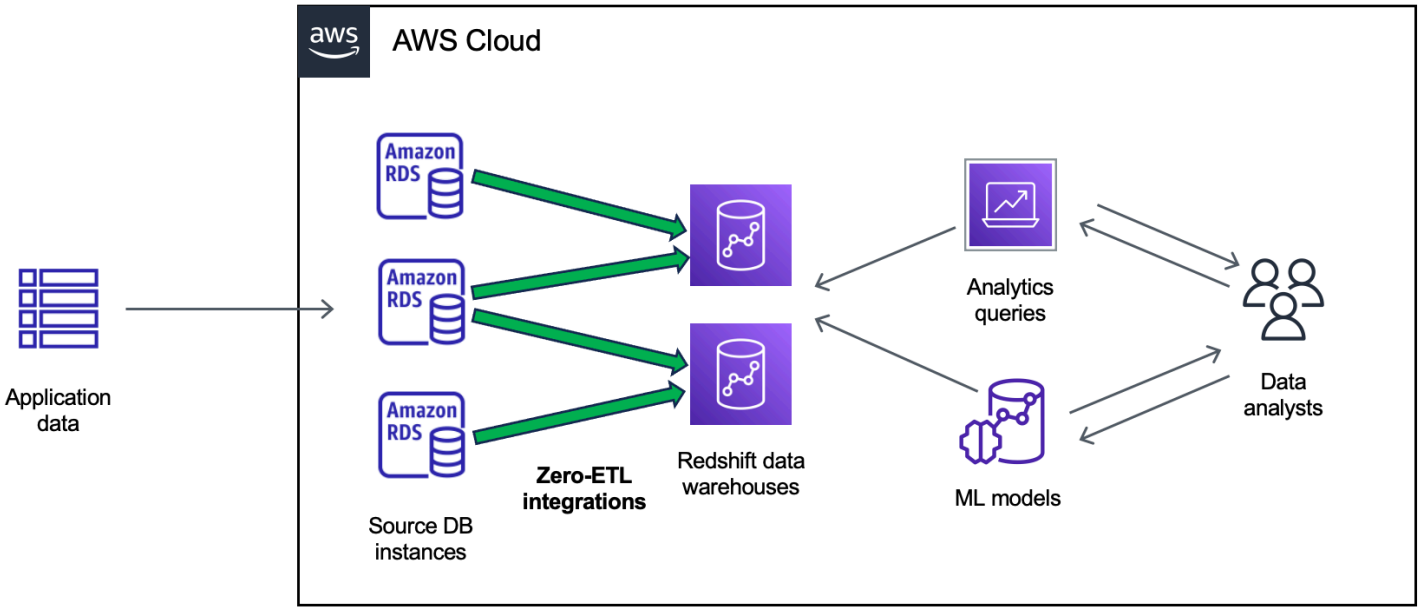
Amazon RDS 제로 ETL 통합

Amazon Redshift 및 Amazon SageMaker와의 Amazon RDS 제로 ETL 통합을 사용하면RDS의 데이터를 사용해 거의 실시간에 가까운 분석 및 기계 학습(ML)을 수행할 수 있습니다. 이 솔루션은 트랜잭션 데이터가 RDS 데이터베이스에 기록된 후 분석 대상에서 사용할 수 있도록 하기 위한 완전관리형 솔루션입니다. 추출, 전환, 적재(ETL)는 여러 소스의 데이터를 데이터 웨어하우스라는 대규모 중앙 리포지토리로 결합하는 프로세스입니다.

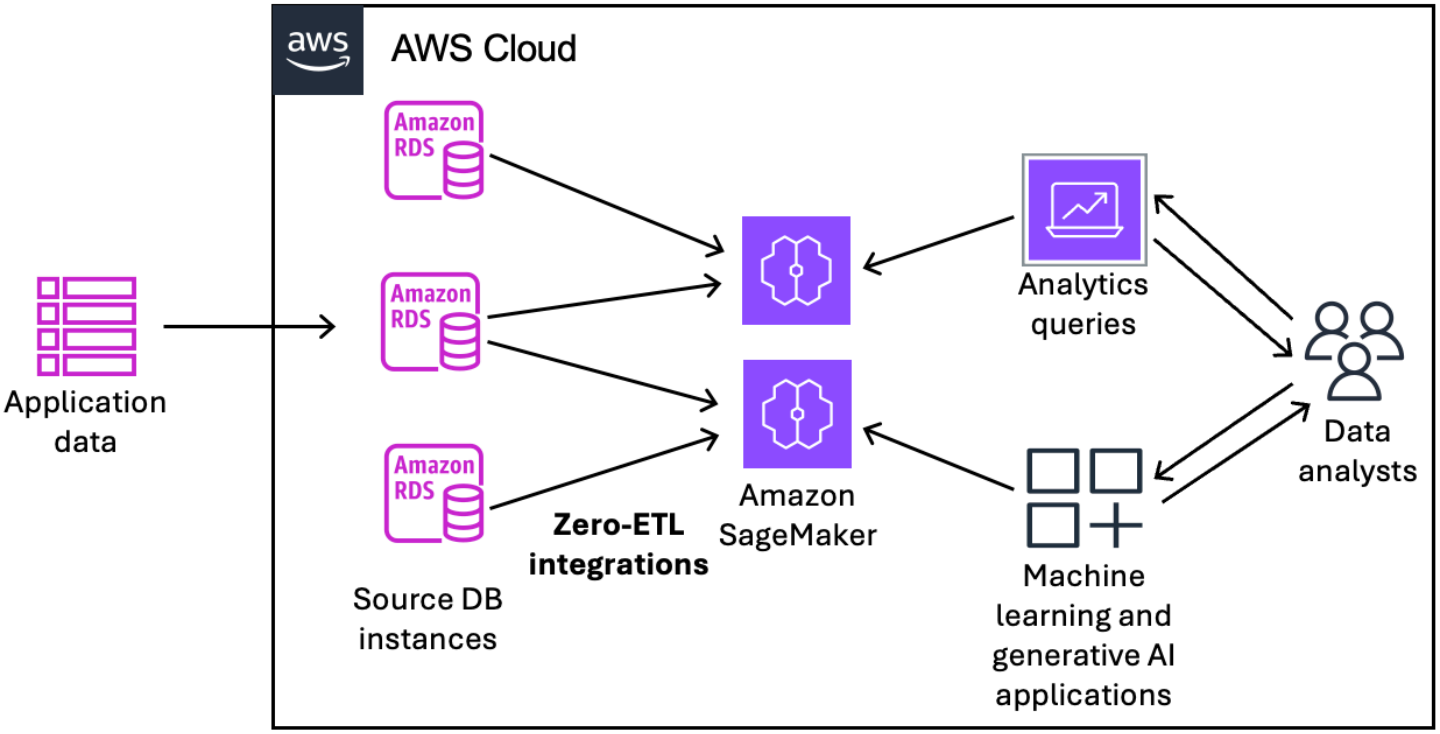
제로 ETL 통합을 통해 RDS 데이터베이스의 데이터를 거의 실시간으로 Amazon Redshift 또는 Amazon SageMaker 레이크하우스에서 사용할 수 있습니다. 데이터가 대상 데이터 웨어하우스 또는 데이터 레이크에 저장되면 기계 학습, 구체화된 뷰, 데이터 공유, 여러 데이터 저장소 및 데이터 레이크에 대한 페더레이션 액세스, Amazon SageMaker AI, QuickSight 및 기타 AWS 서비스와의 통합과 같은 기본 제공 기능을 사용하여 분석, ML 및 AI 워크로드를 강화할 수 있습니다.

제로 ETL 통합을 만들려면 RDS 데이터베이스를 소스로 지정하고 지원되는 데이터 웨어하우스 또는 레이크하우스를 대상으로 지정합니다. 통합은 소스 데이터베이스에서 대상 데이터 웨어하우스 또는 레이크하우스로 데이터를 복제합니다.

다음 다이어그램은 Amazon Redshift와의 제로 ETL 통합에서 이 기능을 보여줍니다.



다음 다이어그램은 Amazon SageMaker 레이크하우스와의 제로 ETL 통합에서 이 기능을 보여줍니다.



통합은 데이터 파이프라인의 상태를 모니터링하고 가능한 경우 문제로부터 복구합니다. 여러 RDS 데이터베이스를 단일 대상 데이터 웨어하우스 또는 레이크하우스로 통합하여 여러 애플리케이션에 걸쳐 인사이트를 도출할 수 있습니다.

주제

- 이점
- 주요 개념
- 제한 사항
- 할당량
- 지원되는 리전
- Amazon Redshift가 구성된 Amazon RDS 제로 ETL 통합 시작하기
- Amazon Redshift가 구성된 Amazon RDS 제로 ETL 통합 생성
- Amazon SageMaker 레이크하우스와 Amazon RDS 제로 ETL 통합 생성
- Amazon RDS 제로 ETL 통합의 데이터 필터링
- 소스 RDS 데이터베이스에 데이터 추가 및 Amazon Redshift에서 쿼리
- Amazon RDS 제로 ETL 통합 확인 및 모니터링
- Amazon RDS 제로 ETL 통합 수정

- Amazon RDS 제로 ETL 통합 삭제
- Amazon Redshift가 구성된 Amazon RDS 제로 ETL 통합 문제 해결

이점

RDS 제로 ETL 통합은 다음과 같은 주요 이점을 제공합니다.

- 여러 데이터 소스에서 총체적인 인사이트를 도출할 수 있도록 도와줍니다.
- 추출, 전환, 적재(ETL) 작업을 수행하는 복잡한 데이터 파이프라인을 구축하고 유지 관리할 필요가 없습니다. 제로 ETL 통합은 파이프라인을 프로비저닝하고 관리해 주므로 파이프라인을 구축하고 관리하는 데 따르는 어려움이 발생하지 않습니다.
- 운영 부담과 비용을 줄이고 애플리케이션 개선에 집중할 수 있습니다.
- 대상의 분석 및 ML 기능을 통해 트랜잭션 데이터 및 기타 데이터에서 인사이트를 도출하여 중요하고 시간에 민감한 이벤트에 효과적으로 대응할 수 있습니다.

주요 개념

제로 ETL 통합을 시작할 때는 다음 개념을 고려하세요.

통합

RDS 데이터베이스에서 데이터 웨어하우스 또는 카탈로그로 트랜잭션 데이터 및 스키마를 자동으로 복제하는 완전관리형 데이터 파이프라인입니다.

소스 데이터베이스

데이터가 복제되는 RDS 데이터베이스입니다. 단일 AZ나 다중 AZ DB 인스턴스 또는 다중 AZ DB 클러스터를 지정할 수 있습니다.

대상

데이터가 복제되는 데이터 웨어하우스 또는 레이크하우스입니다. 데이터 웨어하우스에는 프로비저닝된 클러스터 데이터 웨어하우스와 서버리스 데이터 웨어하우스라는 2가지 유형이 있습니다. 프로비저닝된 클러스터 데이터 웨어하우스는 노드라고 하는 컴퓨팅 리소스의 모음으로, 노드는 클러스터라고 하는 그룹을 구성합니다. 서버리스 데이터 웨어하우스는 컴퓨팅 리소스를 저장하는 작업 그룹과 데이터베이스 객체 및 사용자를 수용하는 네임스페이스로 구성됩니다. 두 데이터 웨어하우스 모두 분석 엔진을 실행하며 하나 이상의 데이터베이스를 포함합니다.

대상 레이크하우스는 카탈로그, 데이터베이스, 테이블 및 뷰로 구성됩니다. 레이크하우스 아키텍처에 대한 자세한 내용은 Amazon SageMaker Unified Studio 사용 설명서의 Amazon SageMaker Lakehouse components 섹션을 참조하세요.

여러 소스 데이터베이스가 동일한 대상에 쓸 수 있습니다.

자세한 내용은 Amazon Redshift 개발자 안내서의 데이터 웨어하우스 시스템 아키텍처를 참조하세요.

제한 사항

다음 제한 사항은 RDS 제로 ETL 통합에 적용됩니다.

주제

- 일반 제한 사항
- RDS for MySQL 제한 사항
- Amazon Redshift 제한 사항
- Amazon SageMaker 레이크하우스 제한 사항

일반 제한 사항

- 소스 데이터베이스는 대상과 동일한 리전에 있어야 합니다.
- 기존 통합이 있는 경우 데이터베이스의 이름을 변경할 수 없습니다.
- 동일한 소스 데이터베이스와 대상 데이터베이스 간에 여러 통합을 만들 수 없습니다.
- 기존 통합이 있는 데이터베이스는 삭제할 수 없습니다. 먼저 연결된 모든 통합을 삭제해야 합니다.
- 소스 데이터베이스를 중지하면 데이터베이스를 다시 시작할 때까지 마지막 몇 개의 트랜잭션이 복제되지 않을 수 있습니다.
- 소스 데이터베이스가 중지된 경우 통합을 삭제할 수 없습니다.
- 데이터베이스가 블루/그린 배포의 소스인 경우 블루 및 그린 환경은 전환 중에 기존의 제로 ETL 통합을 가질 수 없습니다. 먼저 통합을 삭제하고 전환한 다음 다시 만들어야 합니다.
- 다른 통합이 활발하게 생성되고 있는 소스 데이터베이스에 대해 통합을 생성할 수 없습니다.
- 처음에 통합을 생성하거나 테이블을 재동기화할 때는 소스 데이터베이스의 크기에 따라 소스에서 대상으로 데이터를 시드하는 데 20~25분 이상 걸릴 수 있습니다. 이러한 지연으로 인해 복제 지연이 증가할 수 있습니다.

- 일부 데이터 유형은 지원되지 않습니다. 자세한 내용은 the section called “데이터 형식 차이” 섹션을 참조하세요.
- 시스템 테이블, 임시 테이블 및 뷰는 대상 웨어하우스에 복제되지 않습니다.

RDS for MySQL 제한 사항

- 소스 데이터베이스에서 지원되는 RDS for MySQL 버전을 실행하고 있어야 합니다. 지원되는 버전 목록은 the section called “제로 ETL 통합 ” 섹션을 참조하세요.
- 제로 ETL 통합은 진행 중인 데이터 변경 사항을 캡처하기 위해 MySQL 바이너리 로깅(binlog)에 의존합니다. binlog 기반 데이터 필터링을 사용하면 소스 데이터베이스와 대상 데이터베이스 간에 데이터 불일치가 발생할 수 있으므로 사용하지 않는 것이 좋습니다.
- 제로 ETL 통합은 InnoDB 스토리지 엔진을 사용하도록 구성된 데이터베이스에만 지원됩니다.
- 사전 정의된 테이블 업데이트가 포함된 외래 키 참조는 지원되지 않습니다. 특히, `CASCADE`, `SET NULL`, `SET DEFAULT` 작업에서는 `ON DELETE` 및 `ON UPDATE` 규칙이 지원되지 않습니다. 다른 테이블에 대한 이러한 참조가 포함된 테이블을 만들거나 업데이트하려고 하면 테이블이 실패 상태가 됩니다.
- `ALTER TABLE` 파티션 작업에서는 RDS에서 대상 웨어하우스로 데이터를 다시 로드하기 위해 테이블이 다시 동기화됩니다. 테이블을 재동기화하는 동안에는 테이블을 쿼리할 수 없습니다. 자세한 내용은 the section called “Amazon Redshift 테이블 중 하나 이상을 재동기화해야 합니다” 섹션을 참조하세요.

Amazon Redshift 제한 사항

제로 ETL 통합과 관련된 Amazon Redshift 제한 사항 목록은 Amazon Redshift 관리 가이드의 Amazon Redshift와 제로 ETL 통합을 사용할 때 고려할 사항을 참조하세요.

Amazon SageMaker 레이크하우스 제한 사항

다음은 Amazon SageMaker 레이크하우스 제로 ETL 통합에 대한 제한 사항입니다.

- 카탈로그 이름의 길이는 19자로 제한됩니다.

할당량

계정에는 RDS 제로 ETL 통합과 관련된 다음과 같은 할당량이 있습니다. 각 할당량은 달리 지정되지 않는 한 리전별로 적용됩니다.

명칭	Default	설명
통합	100	AWS 계정 내 총 통합 수입니다.
대상당 통합	50	단일 대상 데이터 웨어하우스 또는 레이크하우스로 데이터를 보내는 통합 수입니다.
소스 인스턴스별 통합	5	단일 소스 DB 인스턴스에서 데이터를 보내는 통합 수입니다.

또한 대상 웨어하우스는 각 DB 인스턴스 또는 클러스터 노드에 허용되는 테이블 수에 구체적인 제한을 두고 있습니다. Amazon Redshift 할당량 및 제한에 대한 자세한 내용은 Amazon Redshift 관리 안내서의 Amazon Redshift의 할당량 및 제한을 참조하세요.

지원되는 리전

RDS 제로 ETL 통합은 일부 AWS 리전에서 사용할 수 있습니다. 지원되는 리전 목록은 the section called “제로 ETL 통합 ” 섹션을 참조하세요.

Amazon Redshift가 구성된 Amazon RDS 제로 ETL 통합 시작하기

> 이 문서는 미리 보기 릴리스 중이며 Amazon Redshift가 구성된 Amazon RDS 제로 ETL 통합에 대한 미리 보기 릴리스 설명서입니다. 설명서 및 기능은 모두 변경될 수 있습니다. 프로덕션 환경이 아닌 테스트 환경에서만 이 기능을 사용하는 것이 좋습니다. 미리 보기 이용 약관은 AWS 서비스 약관의 베타 및 미리 보기를 참조하세요.

Amazon Redshift가 구성된 제로 ETL 통합을 생성하기 전에 필수 파라미터와 권한을 사용하여 RDS 데이터베이스와 Amazon Redshift 데이터 웨어하우스를 구성합니다. 설정 중에 다음 단계를 완료해야 합니다.

1. 사용자 지정 DB 파라미터 그룹을 만듭니다.

2. 소스 데이터베이스를 생성합니다.
3. 대상 Amazon Redshift 데이터 웨어하우스를 만듭니다.

이러한 작업을 완료한 후 the section called “Amazon Redshift와 제로 ETL 통합 생성”으로 이동합니다.

1단계: 사용자 지정 DB 파라미터 그룹 생성

Amazon Redshift가 구성된 Amazon RDS 제로 ETL 통합은 이진 로깅(binlog)을 제어하는 DB 파라미터에 대한 특정 값을 필요로 합니다. 바이너리 로깅을 구성하려면 먼저 사용자 지정 DB 파라미터 그룹을 만든 다음 이를 소스 데이터베이스와 연결해야 합니다.

다음 설정을 사용하여 사용자 지정 DB 파라미터 그룹을 생성합니다. 파라미터 그룹을 만드는 방법에 대한 지침은 the section called “DB 파라미터 그룹” 섹션을 참조하세요.

- `binlog_format=ROW`
- `binlog_row_image=full`
- `binlog_checksum=NONE`

또한 `binlog_row_value_options` 파라미터가 `PARTIAL_JSON`으로 설정되어 있지 않아야 합니다.

2단계: 소스 데이터베이스 선택 또는 생성

사용자 지정 DB 파라미터 그룹을 생성한 후 RDS for MySQL 단일 AZ 또는 다중 AZ DB 인스턴스를 선택 또는 생성합니다. 이 데이터베이스는 Amazon Redshift로의 데이터 복제 소스가 됩니다.

데이터베이스는 RDS for MySQL 버전 8.0.32 이상을 실행해야 합니다. 단일 AZ 또는 다중 AZ DB 인스턴스를 만드는 방법에 대한 지침은 the section called “DB 인스턴스 생성” 섹션을 참조하세요.

추가 구성에서 기본 DB 파라미터 그룹을 이전 단계에서 생성한 사용자 지정 파라미터 그룹으로 변경합니다.

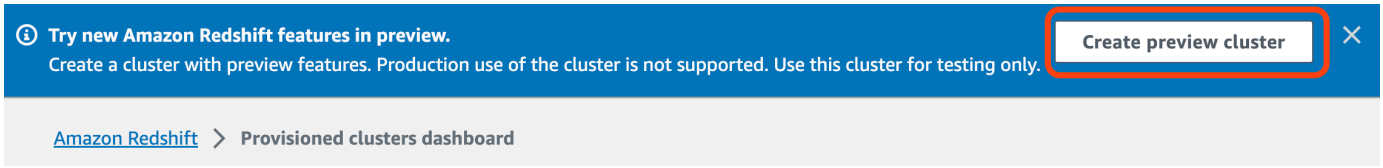
> **Note**
>
> , 데이터베이스가 이미 생성된 후에 데이터베이스와 파라미터 그룹을 연결한다면 제로 ETL 통합을 만들기 전에 데이터베이스를 재부팅하여 변경 사항을 적용해야 합니다. 지침은 the section called “DB 인스턴스 재부팅” 단원을 참조하십시오.

또한 데이터베이스에서 자동 백업이 활성화되어 있는지 확인하세요. 자세한 내용은 the section called "자동 백업 활성화" 단원을 참조하십시오.

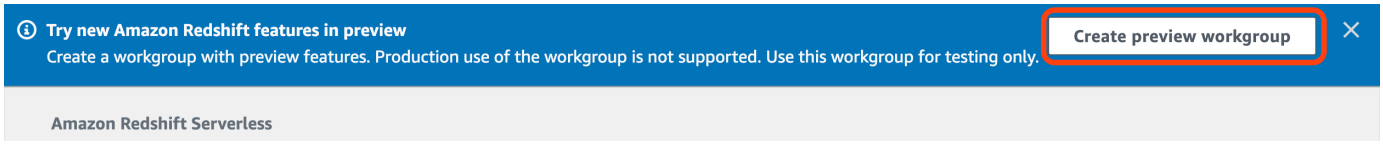
3단계: 대상 Amazon Redshift 데이터 웨어하우스 생성

소스 데이터베이스를 생성한 후에는 Amazon Redshift에서 대상 데이터 웨어하우스를 생성하고 구성해야 합니다. 데이터 웨어하우스는 다음 요구 사항을 충족해야 합니다.

- 미리 보기에서 생성되었습니다
  - 미리 보기에서 프로비저닝된 클러스터를 생성하려면 프로비저닝된 클러스터 대시보드의 배너에서 미리 보기 클러스터 생성을 선택합니다. 자세한 내용은 미리 보기 클러스터 생성 섹션을 참조하세요.



    클러스터를 생성할 때 미리 보기 트랙을 `preview_2023`으로 설정하세요.
  - 미리 보기에서 Redshift Serverless 작업 그룹을 만들려면 서버리스 대시보드의 배너에서 미리 보기 작업 그룹 생성을 선택합니다. 자세한 내용은 미리 보기 작업 그룹 생성 섹션을 참조하세요.



- 최소 2개 이상의 노드가 있는 RA3 노드 유형(`ra3.xlplus`, `ra3.4xlarge`, `ra3.16xlarge`) 또는 Redshift Serverless를 사용합니다.
- 암호화되어 있습니다(프로비저닝된 클러스터를 사용하는 경우). 자세한 내용은 Amazon Redshift 데이터베이스 암호화를 참조하세요.

데이터 웨어하우스를 만드는 방법에 대한 지침은 프로비전된 클러스터의 경우 클러스터 생성을, Redshift Serverless의 경우 네임스페이스가 있는 작업 그룹 생성을 참조하세요.

데이터 웨어하우스에서 대/소문자 구분 활성화

통합이 성공하려면 데이터 웨어하우스에서 대/소문자 구분 파라미터(`enable_case_sensitive_identifier`)를 활성화해야 합니다. 기본적으로 모든 프로비저닝된 클러스터와 Redshift Serverless 작업 그룹에서 대/소문자 구분이 비활성화되어 있습니다.

대/소문자 구분을 활성화하려면 데이터 웨어하우스 유형에 따라 다음 단계를 수행하세요.

- 프로비저닝된 클러스터 - 프로비저닝된 클러스터에서 대/소문자 구분을 활성화하려면 `enable_case_sensitive_identifier` 파라미터가 활성화된 사용자 지정 파라미터 그룹을 생성합니다. 그런 다음 이 파라미터 그룹을 클러스터와 연결합니다. 자세한 지침은 콘솔을 사용한 파라미터 그룹 관리 또는 AWS CLI를 사용한 파라미터 값 구성을 참조하세요.

  > **Note**
  >
  > 사용자 지정 파라미터 그룹을 연결한 후 클러스터를 재부팅해야 합니다.

- Serverless 작업 그룹 - Redshift Serverless 작업 그룹에서 대/소문자 구분을 활성화하려면 AWS CLI를 사용해야 합니다. Amazon Redshift 콘솔은 현재 Redshift Serverless 파라미터 값 수정을 지원하지 않습니다. 다음 update-workgroup 업데이트 요청을 보냅니다.

  ```
  aws redshift-serverless update-workgroup \
    --workgroup-name target-workgroup \
    --config-parameters
   parameterKey=enable_case_sensitive_identifier,parameterValue=true
  ```

  작업 그룹의 파라미터 값을 수정한 후 작업 그룹을 재부팅할 필요가 없습니다.

데이터 웨어하우스에 대한 권한 부여 구성

데이터 웨어하우스를 만든 후에는 소스 RDS 데이터베이스를 승인된 통합 소스로 구성해야 합니다. 자세한 지침은 Amazon Redshift 데이터 웨어하우스에 대한 권한 부여 구성을 참조하세요.

다음 단계

소스 RDS 데이터베이스와 Amazon Redshift 대상 데이터 웨어하우스가 있으므로, 이제 제로 ETL 통합을 생성하고 데이터를 복제할 수 있습니다. 지침은 the section called “Amazon Redshift와 제로 ETL 통합 생성” 단원을 참조하세요.

Amazon Redshift가 구성된 Amazon RDS 제로 ETL 통합 생성

Amazon RDS 제로 ETL 통합을 생성할 때는 소스 RDS 데이터베이스와 대상 Amazon Redshift 데이터 웨어하우스를 지정합니다. 암호화 설정을 사용자 지정하고 태그를 추가할 수도 있습니다. Amazon RDS는 소스 데이터베이스와 대상 간에 통합을 생성합니다. 통합이 활성화되면 소스 데이터베이스에 삽입한 모든 데이터가 구성된 Amazon Redshift 대상에 복제됩니다.

사전 조건

제로 ETL 통합을 생성하기 전에 소스 데이터베이스와 대상 Amazon Redshift 데이터 웨어하우스를 생성해야 합니다. 또한 데이터베이스를 인증된 통합 소스로 추가하여 데이터 웨어하우스로의 복제를 허용해야 합니다.

각 단계를 완료하기 위한 지침은 the section called “제로 ETL 통합 시작하기” 섹션을 참조하세요.

필수 권한

제로 ETL 통합을 생성하려면 특정 IAM 권한이 필요합니다. 최소한 다음 작업을 수행할 수 있는 권한이 필요합니다.

- 소스 RDS 데이터베이스를 위한 제로 ETL 통합을 생성합니다.
- 모든 제로 ETL 통합을 보고 삭제합니다.
- 대상 데이터 웨어하우스에 인바운드 통합을 생성합니다.

다음 샘플 정책은 통합을 생성하고 관리하는 데 필요한 최소 권한을 보여줍니다. 사용자 또는 역할에 더 광범위한 권한(예: `AdministratorAccess` 관리형 정책)이 있는 경우 이러한 정확한 권한이 필요하지 않을 수 있습니다.

> **Note**
>
> Redshift Amazon 리소스 이름(ARN)의 형식은 다음과 같습니다. 서버리스 네임스페이스 UUID 앞에는 콜론(`:`) 대신 슬래시(`/`)를 사용한다는 점에 유의하세요.
>
> - 프로비저닝된 클러스터 – `arn:aws:redshift:{region}:{account-id}:namespace:namespace-uuid`
> - 서버리스 – `arn:aws:redshift-serverless:{region}:{account-id}:namespace/namespace-uuid`

Redshift 대상에 대한 샘플 정책

JSON

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "rds:CreateIntegration"
        ],
        "Resource": [
            "arn:aws:rds:{region}:{account-id}:db:source-db",
            "arn:aws:rds:{region}:{account-id}:integration:*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "rds:DescribeIntegrations"
        ],
        "Resource": ["*"]
    },
    {
        "Effect": "Allow",
        "Action": [
            "rds:DeleteIntegration",
            "rds:ModifyIntegration"
        ],
        "Resource": [
            "arn:aws:rds:{region}:{account-id}:integration:*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "redshift:CreateInboundIntegration"
        ],
        "Resource": [
            "arn:aws:redshift:{region}:{account-id}:namespace:namespace-uuid"
        ]
    }]
}
```

다른 계정에서 대상 데이터 웨어하우스 선택

다른 AWS 계정에 있는 대상 Amazon Redshift 데이터 웨어하우스를 지정하려면 현재 계정의 사용자가 대상 계정의 리소스에 액세스할 수 있도록 허용하는 역할을 만들어야 합니다. 자세한 내용은 소유한 다른 AWS 계정의 IAM 사용자에 액세스 권한 제공을 참조하세요.

역할에는 사용자가 대상 계정에서 사용 가능한 Amazon Redshift 프로비저닝 클러스터와 Redshift Serverless 네임스페이스를 볼 수 있도록 허용하는 다음 권한이 있어야 합니다.

필수 권한 및 신뢰 정책

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "redshift:DescribeClusters",
            "redshift-serverless:ListNamespaces"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

역할에는 대상 계정 ID를 지정하는 다음과 같은 신뢰 정책이 있어야 합니다.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "AWS": "arn:aws:iam::{external-account-id}:root"
```

```
        },
        "Action":"sts:AssumeRole"
      }
   ]
}
```

역할을 생성하기 위한 지침은 사용자 지정 신뢰 정책을 사용하여 역할 생성을 참조하세요.

제로 ETL 통합 생성

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 제로 ETL 통합을 생성할 수 있습니다.

> **Important**
>
> 제로 ETL 통합은 새로 고침 또는 재동기화 작업을 지원하지 않습니다. 생성 후 통합에 문제가 발생하면 통합을 삭제하고 새 통합을 생성해야 합니다.

기본적으로 RDS for MySQL은 바이너리 로그 파일을 즉시 제거합니다. 제로 ETL 통합은 바이너리 로그를 사용하여 소스에서 대상으로 데이터를 복제하기 때문에 소스 데이터베이스의 보존 기간이 1시간 이상이어야 합니다. 통합을 생성하자마자 Amazon RDS는 선택한 소스 데이터베이스의 바이너리 로그 파일 보존 기간을 확인합니다. 현재 값이 0시간인 경우 Amazon RDS는 자동으로 값을 1시간으로 변경합니다. 0이 아니면 값이 동일하게 유지됩니다.

RDS 콘솔

제로 ETL 통합을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 제로 ETL 통합을 선택합니다.
3. 제로 ETL 통합 생성을 선택합니다.
4. 통합 식별자에 통합의 이름을 입력합니다. 이름은 최대 63자의 영숫자로 구성할 수 있으며, 하이픈을 포함할 수 있습니다.

> **Important**
>
> 카탈로그 이름의 길이는 19자로 제한됩니다. 통합 식별자가 카탈로그 이름으로 사용될 경우 이 요구 사항을 충족하는지 확인합니다.

5. 다음을 선택합니다.
6. 소스에서 데이터의 출처가 될 RDS 데이터베이스를 선택합니다.

> **Note**
>
> DB 파라미터가 올바르게 구성되지 않은 경우 RDS에서 알려줍니다. 이 메시지를 받으면 수정 요청을 선택하거나 수동으로 구성할 수 있습니다. 수동으로 수정하는 방법에 대한 지침은 the section called “1단계: 사용자 지정 DB 파라미터 그룹 생성” 섹션을 참조하세요. DB 파라미터를 수정하려면 재부팅해야 합니다. 통합을 생성하려면 먼저 재부팅을 완료하고 새 파라미터 값을 데이터베이스에 성공적으로 적용해야 합니다.

7. 소스 데이터베이스가 성공적으로 구성되면 다음을 선택합니다.
8. 대상에서 다음을 수행합니다.

   1. (선택 사항) Amazon Redshift 대상에 다른 AWS 계정를 사용하려면 다른 계정 지정을 선택합니다. 그런 다음 데이터 웨어하우스를 표시할 권한이 있는 IAM 역할의 ARN을 입력합니다. IAM 역할을 생성하는 방법에 대한 지침은 the section called “다른 계정에서 대상 데이터 웨어하우스 선택” 섹션을 참조하세요.
   2. Amazon Redshift 데이터 웨어하우스의 경우 소스 데이터베이스에서 복제된 데이터의 대상을 선택합니다. 프로비저닝된 Amazon Redshift 클러스터 또는 Redshift Serverless 네임스페이스를 대상으로 선택할 수 있습니다.

> **Note**
>
> 지정된 데이터 웨어하우스의 리소스 정책 또는 대/소문자 구분 설정이 올바르게 구성되지 않은 경우 RDS에서 알려줍니다. 이 메시지를 받으면 수정 요청을 선택하거나 수동으로 구성할 수 있습니다. 수동으로 수정하는 방법에 대한 지침은 Amazon Redshift 관리 가이드의 데이터 웨어하우스에 대/소문자 구분 기능 사용 설정 및 데이터 웨어하우스에 대한 권한 부여 구성을 참조하세요.

프로비저닝된 Redshift 클러스터의 대/소문자 구분을 수정하려면 재부팅해야 합니다. 통합을 생성하려면 먼저 재부팅을 완료하고 새 파라미터 값을 클러스터에 성공적으로 적용해야 합니다.
선택한 소스와 대상이 다른 AWS 계정에 있는 경우 Amazon RDS에서 이러한 설정을 수정할 수 없습니다. Amazon Redshift에서 다른 계정으로 이동하여 수동으로 수정해야 합니다.

9. 대상 데이터 웨어하우스가 올바르게 구성되었으면 다음을 선택합니다.
10. (선택 사항) 태그에서 통합에 하나 이상의 태그를 추가합니다. 자세한 내용은 the section called "RDS 리소스에 태그 지정" 섹션을 참조하세요.
11. 암호화에서 통합을 암호화할 방법을 지정합니다. 기본적으로 RDS는 AWS 소유 키를 사용하여 모든 통합을 암호화합니다. 고객 관리형 키를 대신 선택하려면 암호화 설정 사용자 지정을 활성화하고 암호화에 사용할 KMS 키를 선택합니다. 자세한 내용은 the section called "Amazon RDS 리소스 암호화" 섹션을 참조하세요.

    선택적으로 암호화 컨텍스트를 추가합니다. 자세한 내용은 AWS Key Management Service 개발자 가이드에서 암호화 컨텍스트를 참조하세요.

    > **Note**
    >
    > Amazon RDS는 추가한 모든 항목 외에도 다음과 같은 암호화 컨텍스트 쌍을 추가합니다.
    >
    > - `aws:redshift:integration:arn` - `IntegrationArn`
    > - `aws:servicename:id` - `Redshift`
    >
    > 이렇게 하면 추가할 수 있는 전체 쌍 수가 8개에서 6개로 줄어들고 권한 부여 제약 조건의 전체 문자 제한이 늘어납니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 권한 부여 제약 사용을 참조하세요.

12. 다음을 선택합니다.
13. 통합 설정을 검토하고 제로 ETL 통합 생성을 선택합니다.

    생성에 실패할 경우 the section called "제로 ETL 통합을 생성할 수 없습니다"에서 문제 해결 단계를 확인하세요.

통합 상태는 생성 중인 동안 `Creating`이며, 대상 Amazon Redshift 데이터 웨어하우스의 상태는 `Modifying`입니다. 이 기간 동안에는 데이터 웨어하우스를 쿼리하거나 구성을 변경할 수 없습니다.

통합이 성공적으로 생성되면 통합 상태와 대상 Amazon Redshift 데이터 웨어하우스가 모두 `Active`로 변경됩니다.

AWS CLI

AWS CLI를 사용하여 제로 ETL 통합을 생성하려면 다음 옵션과 함께 create-integration 명령을 사용하세요.

> **Note**
>
> 카탈로그 이름은 19자로 제한된다는 점에 유의하세요. 카탈로그 이름으로 사용할 경우 적절하게 통합 이름을 선택합니다.

- `--integration-name` - 통합 이름을 지정합니다.
- `--source-arn` - 통합의 소스가 될 RDS 데이터베이스의 ARN을 지정합니다.
- `--target-arn` - 통합의 대상이 될 Amazon Redshift 데이터 웨어하우스의 ARN을 지정합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-integration \
    --integration-name my-integration \
    --source-arn arn:aws:rds:{region}:{account-id}:my-db \
    --target-arn arn:aws:redshift:{region}:{account-id}:namespace:namespace-uuid
```

Windows의 경우:

```
aws rds create-integration ^
    --integration-name my-integration ^
    --source-arn arn:aws:rds:{region}:{account-id}:my-db ^
    --target-arn arn:aws:redshift:{region}:{account-id}:namespace:namespace-uuid
```

RDS API

Amazon RDS API를 사용하여 제로 ETL 통합을 생성하려면 다음 파라미터를 적용한 `CreateIntegration` 작업을 사용하세요.

> **Note**
>
> 카탈로그 이름은 19자로 제한됩니다. IntegrationName 파라미터가 카탈로그 이름으로 사용될 경우 이 요구 사항을 충족하는지 확인합니다.

- `IntegrationName` - 통합 이름을 지정합니다.
- `SourceArn` - 통합의 소스가 될 RDS 데이터베이스의 ARN을 지정합니다.
- `TargetArn` - 통합의 대상이 될 Amazon Redshift 데이터 웨어하우스의 ARN을 지정합니다.

고객 관리형 키를 사용하여 통합 암호화

통합을 생성할 때 AWS 소유 키가 아닌 사용자 지정 KMS 키를 설정하는 경우 키 정책은 Amazon Redshift 서비스 보안 주체에 `CreateGrant` 작업에 대한 액세스 권한을 제공해야 합니다. 또한 현재 사용자가 `DescribeKey` 및 `CreateGrant` 작업을 수행할 수 있도록 허용해야 합니다.

다음 샘플 정책은 키 정책에 필요한 권한을 제공하는 방법을 보여줍니다. 여기에는 권한 범위를 더 축소하기 위한 컨텍스트 키가 포함됩니다.

샘플 키 정책

JSON

```
{
    "Version": "2012-10-17",
    "Id": "Key policy",
    "Statement": [
        {
            "Sid": "Enables IAM user permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::{account-ID}:root"
            },
            "Action": "kms:*",
            "Resource": "*"
```

```
        },
        {
            "Sid": "Allows the Redshift service principal to add a grant to a KMS
key",
            "Effect": "Allow",
            "Principal": {
                "Service": "redshift.amazonaws.com"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}":"{context-value}"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
                    ]
                }
            }
        },
        {
            "Sid": "Allows the current user or role to add a grant to a KMS key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::{account-ID}:role/{role-name}"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}":"{context-value}",
                    "kms:ViaService": "rds.us-east-1.amazonaws.com"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
                    ]
                }
            }
```

```
        },
        {
            "Sid": "Allows the current uer or role to retrieve information about
 a KMS key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::{account-ID}:role/{role-name}"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*"
        }
    ]
}
```

자세한 내용은 AWS Key Management Service 개발자 안내서의 키 정책 생성을 참조하세요.

다음 단계

제로 ETL 통합을 성공적으로 생성한 후에는 대상 Amazon Redshift 클러스터 또는 작업 그룹 내에 대상 데이터베이스를 생성해야 합니다. 그런 다음 소스 RDS 데이터베이스에 데이터를 추가하고 Amazon Redshift에서 쿼리할 수 있습니다. 자세한 지침은 Amazon Redshift에서 대상 데이터베이스 생성을 참조하세요.

Amazon SageMaker 레이크하우스와 Amazon RDS 제로 ETL 통합 생성

Amazon SageMaker 레이크하우스와 Amazon RDS 제로 ETL 통합을 생성할 때 소스 RDS 데이터베이스와 대상 AWS Glue 관리형 카탈로그를 지정합니다. 암호화 설정을 사용자 지정하고 태그를 추가할 수도 있습니다. Amazon RDS는 소스 데이터베이스와 대상 간에 통합을 생성합니다. 통합이 활성화되면 소스 데이터베이스에 삽입한 모든 데이터가 구성된 대상에 복제됩니다.

사전 조건

Amazon SageMaker 레이크하우스와 제로 ETL 통합을 생성하기 전에 소스 데이터베이스와 대상 AWS Glue 관리형 카탈로그를 생성해야 합니다. 또한 데이터베이스를 인증된 통합 소스로 추가하여 카탈로그로의 복제를 허용해야 합니다.

각 단계를 완료하기 위한 지침은 the section called “제로 ETL 통합 시작하기” 섹션을 참조하세요.

필수 권한

Amazon SageMaker 레이크하우스와 제로 ETL 통합을 생성하려면 특정 IAM 권한이 필요합니다. 최소한 다음 작업을 수행할 수 있는 권한이 필요합니다.

- 소스 RDS 데이터베이스를 위한 제로 ETL 통합을 생성합니다.
- 모든 제로 ETL 통합을 보고 삭제합니다.
- 대상 AWS Glue 관리형 카탈로그에 인바운드 통합을 생성합니다.
- AWS Glue 관리형 카탈로그에서 사용하는 Amazon S3 버킷에 액세스합니다.
- 사용자 지정 암호화가 구성된 경우 암호화에 AWS KMS 키를 사용합니다.
- Lake Formation에 리소스를 등록합니다.
- AWS Glue 관리형 카탈로그에 리소스 정책을 적용하여 인바운드 통합을 승인합니다.

다음 샘플 정책은 Amazon SageMaker 레이크하우스와 통합을 생성하고 관리하는 데 필요한 최소 권한을 보여줍니다. 사용자 또는 역할에 더 광범위한 권한(예: `AdministratorAccess` 관리형 정책)이 있는 경우 이러한 정확한 권한이 필요하지 않을 수 있습니다.

또한 인바운드 통합을 승인하려면 대상 AWS Glue 관리형 카탈로그에서 리소스 정책을 구성해야 합니다. 다음 AWS CLI명령을 실행하여 리소스 정책을 적용합니다.

대상 카탈로그에서 인바운드 통합을 승인하는 샘플 AWS CLI 명령

```
aws glue put-resource-policy \
      --policy-in-json  '{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "glue.amazonaws.com"
        },
        "Action": [
            "glue:AuthorizeInboundIntegration"
        ],
        "Resource": ["arn:aws:glue:region:account_id:catalog/catalog_name"],
        "Condition": {
            "StringEquals": {
                "aws:SourceArn": "arn:aws:rds:region:account_id:db:source_name"
            }
```

```
        }
    },
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "account_id"
        },
        "Action": ["glue:CreateInboundIntegration"],
        "Resource": ["arn:aws:glue:region:account_id:catalog/catalog_name"]
    }
    ]
}' \
      --region region
```

> **Note**
>
> Glue 카탈로그 Amazon 리소스 이름(ARN) 형식은 다음과 같습니다.
>
> - Glue 카탈로그 – `arn:aws:glue:{region}:{account-id}:catalog/catalog-name`

다른 계정에서 대상 AWS Glue 관리형 카탈로그 선택

다른 AWS 계정에 있는 대상 AWS Glue 관리형 카탈로그를 지정하려면 현재 계정의 사용자가 대상 계정의 리소스에 액세스할 수 있도록 허용하는 역할을 만들어야 합니다. 자세한 내용은 소유한 다른 AWS 계정의 IAM 사용자에 액세스 권한 제공을 참조하세요.

역할에는 사용자가 대상 계정에서 사용 가능한 AWS Glue 카탈로그를 볼 수 있도록 허용하는 다음 권한이 있어야 합니다.

필수 권한 및 신뢰 정책

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "glue:GetCatalog"
         ],
         "Resource":[
```

```
            "*"
         ]
      }
   ]
}
```

역할에는 대상 계정 ID를 지정하는 다음과 같은 신뢰 정책이 있어야 합니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "AWS": "arn:aws:iam::{external-account-id}:root"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

역할을 생성하기 위한 지침은 사용자 지정 신뢰 정책을 사용하여 역할 생성을 참조하세요.

Amazon SageMaker 레이크하우스와 제로 ETL 통합 생성

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 Amazon SageMaker 레이크하우스와 제로 ETL 통합을 생성할 수 있습니다.

> **Important**
>
> Amazon SageMaker 레이크하우스와의 제로 ETL 통합은 새로 고침 또는 재동기화 작업을 지원하지 않습니다. 생성 후 통합에 문제가 발생하면 통합을 삭제하고 새 통합을 생성해야 합니다.

기본적으로 RDS for MySQL은 바이너리 로그 파일을 즉시 제거합니다. 제로 ETL 통합은 바이너리 로그를 사용하여 소스에서 대상으로 데이터를 복제하기 때문에 소스 데이터베이스의 보존 기간이 1시간 이상이어야 합니다. 통합을 생성하자마자 Amazon RDS는 선택한 소스 데이터베이스의 바이너리 로그 파일 보존 기간을 확인합니다. 현재 값이 0시간인 경우 Amazon RDS는 자동으로 값을 1시간으로 변경합니다. 0이 아니면 값이 동일하게 유지됩니다.

RDS 콘솔

Amazon SageMaker 레이크하우스와 제로 ETL 통합을 생성하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 제로 ETL 통합을 선택합니다.
3. 제로 ETL 통합 생성을 선택합니다.
4. 통합 식별자에 통합의 이름을 입력합니다. 이름은 최대 63자의 영숫자로 구성할 수 있으며, 하이픈을 포함할 수 있습니다.
5. 다음을 선택합니다.
6. 소스에서 데이터의 출처가 될 RDS 데이터베이스를 선택합니다.

> **Note**
>
> DB 파라미터가 올바르게 구성되지 않은 경우 RDS에서 알려줍니다. 이 메시지를 받으면 수정 요청을 선택하거나 수동으로 구성할 수 있습니다. 수동으로 수정하는 방법에 대한 지침은 the section called “1단계: 사용자 지정 DB 파라미터 그룹 생성” 섹션을 참조하세요. DB 파라미터를 수정하려면 재부팅해야 합니다. 통합을 생성하려면 먼저 재부팅을 완료하고 새 파라미터 값을 데이터베이스에 성공적으로 적용해야 합니다.

7. 소스 데이터베이스가 성공적으로 구성되면 다음을 선택합니다.
8. 대상에서 다음을 수행합니다.
    1. (선택 사항) Amazon SageMaker 레이크하우스 대상에 다른 AWS 계정을 사용하려면 다른 계정 지정을 선택합니다. 그런 다음 AWS Glue 카탈로그를 표시할 권한이 있는 IAM 역할의 ARN을 입력합니다. IAM 역할을 생성하는 방법에 대한 지침은 the section called “다른 계정에서 대상 AWS Glue 관리형 카탈로그 선택” 섹션을 참조하세요.
    2. AWS Glue 카탈로그의 경우 소스 데이터베이스에서 복제된 데이터의 대상을 선택합니다. 기존 AWS Glue 관리형 카탈로그를 대상으로 선택할 수 있습니다.
    3. 대상 IAM 역할에는 대상 카탈로그에 대한 설명 권한이 필요하며 다음 권한이 있어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
```

```
            "Effect": "Allow",
            "Action": "glue:GetCatalog",
            "Resource": [
                "arn:aws:glue:region:account-id:catalog/*",
                "arn:aws:glue:region:account-id:catalog"
            ]
        }
    ]
}
```

대상 IAM 역할은 다음과 같은 신뢰 관계를 맺고 있어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

4. ???에서 생성된 Lake Formation 관리자 역할을 사용하여 대상 AWS Glue 관리형 카탈로그에 대한 권한을 대상 IAM 역할에 부여해야 합니다.

> **Note**
>
> RDS는 지정된 AWS Glue 관리형 카탈로그의 리소스 정책 또는 구성 설정이 올바르게 구성되지 않은 경우 알려줍니다. 이 메시지를 받으면 수정 요청을 선택하거나 수동으로 구성할 수 있습니다.
> 선택한 소스와 대상이 다른 AWS 계정에 있는 경우 Amazon RDS에서 이러한 설정을 수정할 수 없습니다. SageMaker Unified Studio에서 다른 계정으로 이동하여 수동으로 수정해야 합니다.

9. 대상 AWS Glue 관리형 카탈로그가 올바르게 구성되면 다음을 선택합니다.
10. (선택 사항) 태그에서 통합에 하나 이상의 태그를 추가합니다. 자세한 내용은 the section called "RDS 리소스에 태그 지정" 섹션을 참조하세요.

11. 암호화에서 통합을 암호화할 방법을 지정합니다. 기본적으로 RDS는 AWS 소유 키를 사용하여 모든 통합을 암호화합니다. 고객 관리형 키를 대신 선택하려면 암호화 설정 사용자 지정을 활성화하고 암호화에 사용할 KMS 키를 선택합니다. 자세한 내용은 the section called “Amazon RDS 리소스 암호화” 섹션을 참조하세요.

    선택적으로 암호화 컨텍스트를 추가합니다. 자세한 내용은 AWS Key Management Service 개발자 가이드에서 암호화 컨텍스트를 참조하세요.

    > **Note**
    >
    > Amazon RDS는 추가한 모든 항목 외에도 다음과 같은 암호화 컨텍스트 쌍을 추가합니다.
    >
    > - `aws:glue:integration:arn` - `IntegrationArn`
    > - `aws:servicename:id` - `glue`
    >
    > 이렇게 하면 추가할 수 있는 전체 쌍 수가 8개에서 6개로 줄어들고 권한 부여 제약 조건의 전체 문자 제한이 늘어납니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 권한 부여 제약 사용을 참조하세요.

12. 다음을 선택합니다.
13. 통합 설정을 검토하고 제로 ETL 통합 생성을 선택합니다.

    생성에 실패할 경우 the section called “제로 ETL 통합 문제 해결”에서 문제 해결 단계를 확인하세요.

통합 상태는 생성 중인 동안 `Creating`이며, 대상 Amazon SageMaker 레이크하우스의 상태는 `Modifying`입니다. 이 기간 동안에는 카탈로그를 쿼리하거나 구성을 변경할 수 없습니다.

통합이 성공적으로 생성되면 통합과 대상 Amazon SageMaker 레이크하우스의 상태가 모두 `Active`로 변경됩니다.

AWS CLI

AWS CLI를 사용하여 제로 ETL 통합을 위한 대상 AWS Glue 관리형 카탈로그를 준비하려면 먼저 다음 옵션과 함께 create-integration-resource-property 명령을 사용해야 합니다.

- `--resource-arn` - 통합의 대상이 될 AWS Glue 관리형 카탈로그의 ARN을 지정합니다.

- `--target-processing-properties` - 대상 AWS Glue 관리형 카탈로그에 액세스할 IAM 역할의 ARN을 지정합니다.

```
aws glue create-integration-resource-property --region us-east-1
 --resource-arn arn:aws:glue:region:account_id:catalog/catalog_name \
 --target-processing-properties '{"RoleArn" : "arn:aws:iam::account_id:role/
TargetIamRole"}'
```

AWS CLI를 사용하여 Amazon SageMaker 레이크하우스와 제로 ETL 통합을 생성하려면 다음 옵션과 함께 create-integration 명령을 사용합니다.

- `--integration-name` - 통합 이름을 지정합니다.
- `--source-arn` - 통합의 소스가 될 RDS 데이터베이스의 ARN을 지정합니다.
- `--target-arn` - 통합의 대상이 될 AWS Glue 관리형 카탈로그의 ARN을 지정합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-integration \
    --integration-name my-sagemaker-integration \
    --source-arn arn:aws:rds:{region}:{account-id}:my-db \
    --target-arn arn:aws:glue:{region}:{account-id}:catalog/catalog-name
```

Windows의 경우:

```
aws rds create-integration ^
    --integration-name my-sagemaker-integration ^
    --source-arn arn:aws:rds:{region}:{account-id}:my-db ^
    --target-arn arn:aws:glue:{region}:{account-id}:catalog/catalog-name
```

RDS API

Amazon RDS API를 사용하여 Amazon SageMaker와 제로 ETL 통합을 생성하려면 다음 파라미터를 적용한 `CreateIntegration` 작업을 사용합니다.

> **Note**
>
> 카탈로그 이름은 19자로 제한됩니다. IntegrationName 파라미터가 카탈로그 이름으로 사용될 경우 이 요구 사항을 충족하는지 확인합니다.

- `IntegrationName` - 통합 이름을 지정합니다.
- `SourceArn` - 통합의 소스가 될 RDS 데이터베이스의 ARN을 지정합니다.
- `TargetArn` - 통합의 대상이 될 AWS Glue 관리형 카탈로그의 ARN을 지정합니다.

고객 관리형 키를 사용하여 통합 암호화

Amazon SageMaker와 통합을 생성할 때 AWS 소유 키가 아닌 사용자 지정 KMS 키를 지정하는 경우 키 정책은 SageMaker Unified Studio 서비스 위탁자에 `CreateGrant` 작업에 대한 액세스 권한을 제공해야 합니다. 또한 현재 사용자가 `DescribeKey` 및 `CreateGrant` 작업을 수행할 수 있도록 허용해야 합니다.

다음 샘플 정책은 키 정책에 필요한 권한을 제공하는 방법을 보여줍니다. 여기에는 권한 범위를 더 축소하기 위한 컨텍스트 키가 포함됩니다.

샘플 키 정책

```
{
    "Version": "2012-10-17",
    "Id": "Key policy",
    "Statement": [
        {
            "Sid": "Enables IAM user permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::{account-ID}:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Allows the Glue service principal to add a grant to an AWS KMS
 key",
            "Effect": "Allow",
            "Principal": {
```

```
                "Service": "glue.amazonaws.com"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}":"{context-value}"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
                    ]
                }
            }
        },
        {
            "Sid": "Allows the current user or role to add a grant to a KMS key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::{account-ID}:role/{role-name}"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}":"{context-value}",
                    "kms:ViaService": "rds.us-east-1.amazonaws.com"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
                    ]
                }
            }
        },
        {
            "Sid": "Allows the current uer or role to retrieve information about a KMS
key",
            "Effect": "Allow",
            "Principal": {
```

```
                "AWS": "arn:aws:iam::{account-ID}:role/{role-name}"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*"
        }
    ]
}
```

자세한 내용은 AWS Key Management Service 개발자 안내서의 키 정책 생성을 참조하세요.

다음 단계

Amazon SageMaker와 제로 ETL 통합을 성공적으로 생성한 후 소스 RDS 데이터베이스에 데이터를 추가하고 Amazon SageMaker 레이크하우스에서 쿼리를 시작할 수 있습니다. 데이터는 자동으로 복제되어 분석 및 기계 학습 워크로드에 사용할 수 있습니다.

Amazon RDS 제로 ETL 통합의 데이터 필터링

Amazon RDS 제로 ETL 통합은 데이터 필터링을 지원하므로 사용자는 소스 Amazon RDS 데이터베이스에서 대상 데이터 웨어하우스로 어떤 데이터가 복제될지 제어할 수 있습니다. 전체 데이터베이스를 복제하는 대신 하나 이상의 필터를 적용하여 특정 테이블을 선택적으로 포함하거나 제외할 수 있습니다. 이렇게 하면 관련 데이터만 전송되도록 하여 스토리지 및 쿼리 성능을 최적화할 수 있습니다. 현재 필터링 기능은 데이터베이스 및 테이블 수준으로 제한됩니다. 열 수준 및 행 수준 필터링은 지원되지 않습니다.

데이터 필터링은 다음과 같은 경우에 유용할 수 있습니다.

- 서로 다른 2개 이상의 소스 데이터베이스의 특정 테이블을 조인하며 데이터베이스 중의 전체 데이터가 필요하지 않은 경우
- 전체 데이터베이스 플릿 대신 테이블의 하위 집합만 사용해 분석을 수행하여 비용을 절감하려는 경우
- 특정 테이블에서 전화번호, 주소 또는 신용카드 세부 정보와 같은 민감한 정보를 필터링하는 경우

AWS Management Console, AWS Command Line Interface(AWS CLI) 또는 Amazon RDS API를 사용하여 제로 ETL 통합에 데이터 필터를 추가할 수 있습니다.

통합에서 프로비저닝된 클러스터를 대상으로 하는 경우 클러스터가 패치 180 이상이어야 데이터 필터링을 사용할 수 있습니다.

주제

- 데이터 필터의 형식
- "필터 로직"
- 필터 우선순위
- 예제
- 통합에 데이터 필터 추가
- 통합에서 데이터 필터 제거

데이터 필터의 형식

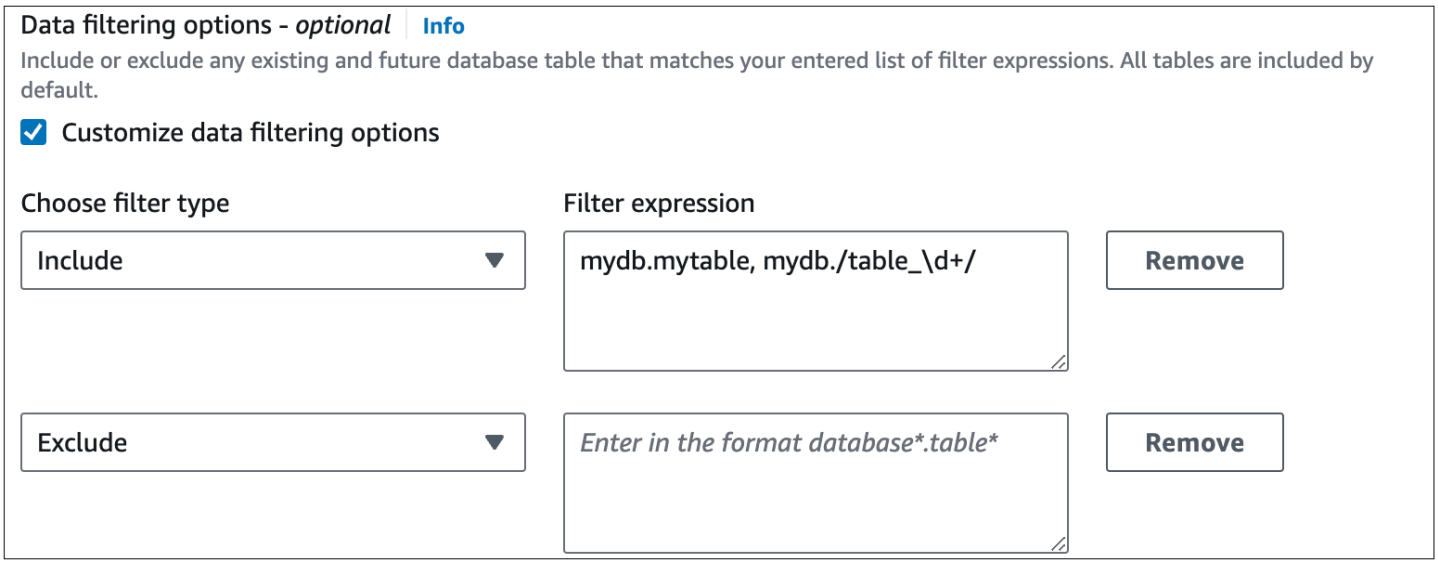
단일 통합에 대해 여러 필터를 정의할 수 있습니다. 각 필터는 필터 표현식의 패턴 중 하나와 일치하는 기존 및 미래 데이터베이스 테이블을 포함하거나 제외합니다. Amazon RDS 제로 ETL 통합은 데이터 필터링에 Maxwell 필터 구문을 사용합니다.

각 필터에는 다음 요소가 포함됩니다.

Element	설명
필터 유형	`Include` 필터 유형에는 필터 표현식의 패턴 중 하나와 일치하는 모든 테이블이 포함됩니다. `Exclude` 필터 유형에는 패턴 중 하나와 일치하는 모든 테이블이 제외됩니다.
필터 표현식	쉼표로 구분된 패턴 목록. 표현식은 Maxwell 필터 구문을 사용해야 합니다.
패턴	*`database`*`.`*`table`* 형식의 필터 패턴입니다. 리터럴 이름을 지정하거나 정규 표현식을 정의할 수 있습니다. 열 수준 필터 또는 거부 목록은 포함할 수 없습니다. 단일 통합은 총 99개까지 패턴을 보유할 수 있습니다. 콘솔에서는 단일 필터 표현식 내에 패턴을 입력하거나 여러 표현식에 패턴을 분산

Element	설명
	시킬 수 있습니다. 단일 패턴은 길이가 256자를 초과할 수 없습니다.

다음 이미지는 콘솔의 데이터 필터 구조를 보여줍니다.



> **Important**
>
> 개인 식별 정보, 기밀 정보 또는 민감한 정보를 필터 패턴에 포함하지 마세요.

AWS CLI의 데이터 필터

AWS CLI를 사용하여 데이터 필터를 추가하는 경우 콘솔의 구문과는 약간 다릅니다. 필터 유형(`Include` 또는 `Exclude`)을 각 패턴에 개별적으로 할당해야 하나의 필터 유형으로 여러 패턴을 그룹화할 수 있습니다.

예를 들어 콘솔에서는 쉼표로 구분된 다음과 같은 패턴을 단일 `Include` 스테이트먼트 내에 그룹화할 수 있습니다.

```
mydb.mytable, mydb./table_\d+/
```

하지만 AWS CLI를 사용할 때는 동일한 데이터 필터가 다음 형식이어야 합니다.

```
'include: mydb.mytable, include: mydb./table_\d+/'
```

"필터 로직"

통합에서 데이터 필터를 지정하지 않는 경우 Amazon RDS는 기본 필터(`include:*.*`)를 가정하고 모든 테이블을 대상 데이터 웨어하우스에 복제합니다. 그러나 최소 하나 이상의 필터를 추가하면 기본 로직이 `exclude:*.*`로 전환되어 기본적으로 모든 테이블이 제외됩니다. 이를 통해 복제에 포함할 데이터베이스와 테이블을 명시적으로 정의할 수 있습니다.

예를 들어 다음을 필터를 정의하는 경우:

```
'include: db.table1, include: db.table2'
```

Amazon RDS는 다음과 같이 필터를 평가합니다.

```
'exclude:*.*, include: db.table1, include: db.table2'
```

따라서 Amazon RDS는 db라는 데이터베이스의 `table1` 및 `table2`만 대상 데이터 웨어하우스에 복제합니다.

필터 우선순위

Amazon RDS는 지정된 순서대로 데이터 필터를 평가합니다. AWS Management Console에서는 필터 표현식을 왼쪽에서 오른쪽으로, 위에서 아래로 처리합니다. 두 번째 필터 또는 첫 번째 뒤에 오는 개별 패턴은 이를 재정의할 수 있습니다.

예를 들어 첫 번째 필터가 `Include books.stephenking`인 경우 `books` 데이터베이스의 `stephenking` 테이블만 포함됩니다. 하지만 두 번째 필터(`Exclude books.*`)를 추가하면 첫 번째 필터가 재정의됩니다. 이렇게 하면 `books` 인덱스의 테이블이 대상 데이터 웨어하우스로 복제되지 않습니다.

최소 1개의 필터를 지정하는 경우 로직은 가정된 `exclude:*.*`로 시작되며, 이는 모든 테이블이 복제에서 자동으로 제외된다는 의미입니다. 가장 광범위한 필터부터 가장 구체적인 필터 순으로 정의하는 것이 좋습니다. 하나 이상의 `Include` 스테이트먼트로 시작하여 복제할 데이터를 지정한 다음 `Exclude` 필터를 추가하여 특정 테이블을 선택적으로 제거합니다.

AWS CLI를 사용하여 정의하는 필터에도 동일한 원칙이 적용됩니다. Amazon RDS는 이러한 필터 패턴을 지정된 순서대로 평가하므로 앞서 지정된 패턴을 재정의할 수 있습니다.

예제

다음 예제에서는 제로 ETL 통합에서 데이터 필터링이 작동하는 방식을 보여줍니다.

- 모든 데이터베이스와 모든 테이블 포함:

```
'include: *.*'
```

- books 데이터베이스 내의 모든 테이블 포함:

```
'include: books.*'
```

- 이름이 mystery인 모든 테이블 제외:

```
'include: *.*, exclude: *.mystery'
```

- books 데이터베이스 내에서 두 개의 특정 테이블 포함:

```
'include: books.stephen_king, include: books.carolyn_keene'
```

- mystery 하위 문자열이 포함된 테이블을 제외한 모든 테이블을 books 데이터베이스에 포함:

```
'include: books.*, exclude: books./.*mystery.*/'
```

- mystery로 시작하는 테이블을 제외한 모든 테이블을 books 데이터베이스에 포함:

```
'include: books.*, exclude: books./mystery.*/'
```

- mystery로 끝나는 테이블을 제외한 모든 테이블을 books 데이터베이스에 포함:

```
'include: books.*, exclude: books./.*mystery/'
```

- books 데이터베이스 내에서 table_stephen_king을 제외하고 이름이 table_로 시작하는 모든 테이블 포함: 예를 들어 table_movies 또는 table_books는 복제되지만 table_stephen_king은 복제되지 않습니다.

```
'include: books./table_.*/, exclude: books.table_stephen_king'
```

통합에 데이터 필터 추가

AWS Management Console, AWS CLI 또는 Amazon RDS API를 사용하여 데이터 필터링을 구성할 수 있습니다.

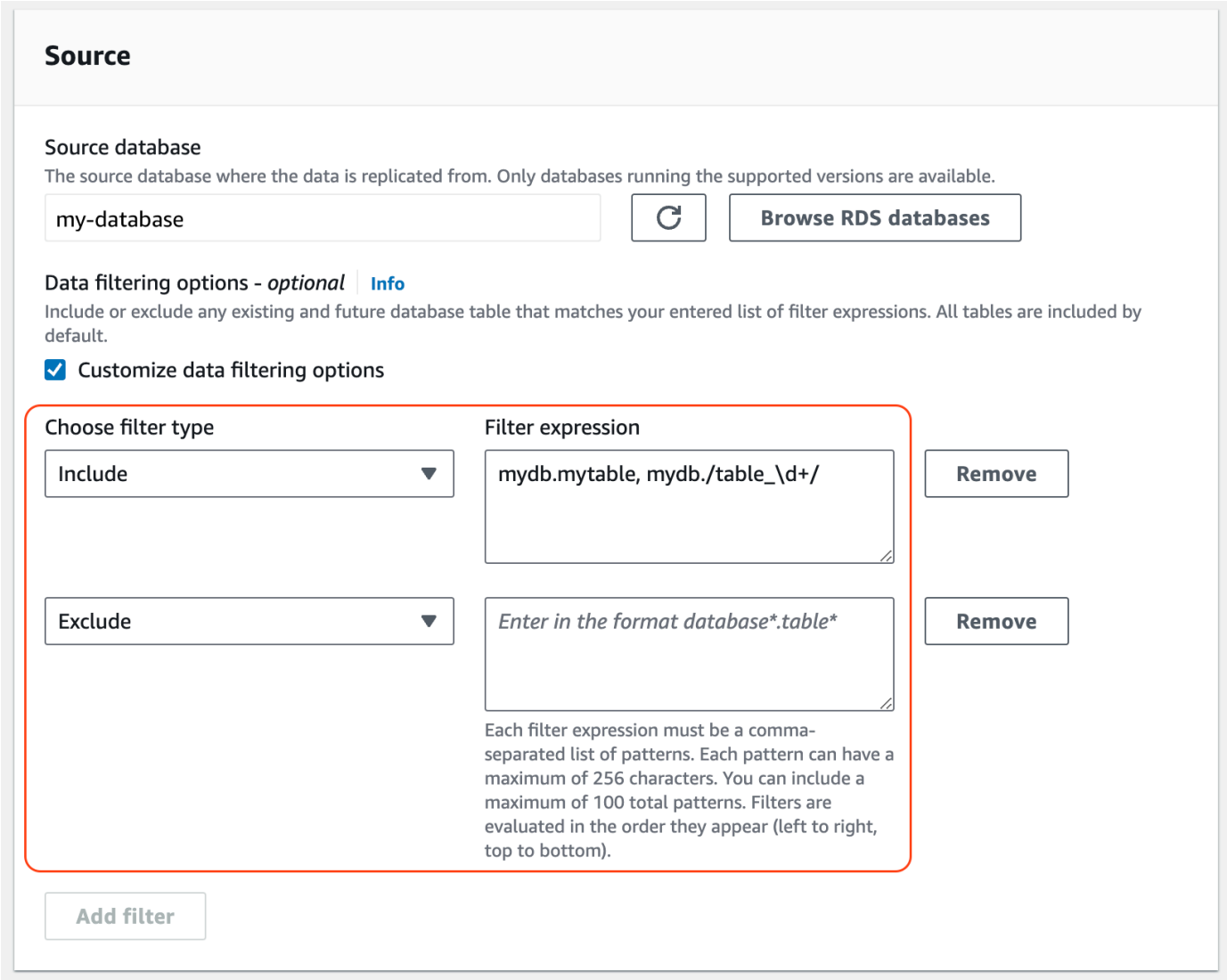
> **Important**
>
> 통합을 만든 후 필터를 추가하면 Amazon RDS가 필터가 항상 존재했던 것처럼 이를 처리합니다. 대상 데이터 웨어하우스에 있는 데이터 중 새로운 필터링 기준과 일치하지 않는 데이터를 제거하고 모든 영향받는 테이블을 재동기화합니다.

RDS 콘솔

제로 ETL 통합에 데이터 필터를 추가하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 제로 ETL 통합을 선택합니다. 데이터 필터를 추가할 통합을 선택한 다음 수정을 선택합니다.
3. 소스에서 하나 이상의 `Include` 및 `Exclude` 문을 추가합니다.

   다음 이미지는 MySQL 통합의 데이터 필터 예시를 보여줍니다.



4. 변경 사항에 만족하면 계속 및 변경 사항 저장을 선택합니다.

AWS CLI

AWS CLI를 사용하여 제로 ETL 통합에 데이터 필터를 추가하려면 modify-integration 명령을 직접 호출하세요. 통합 식별자 외에도 쉼표로 구분된 `Include` 및 `Exclude` Maxwell 필터 목록을 사용하여 `--data-filter` 파라미터를 지정하세요.

Example

다음 예시에서는 `my-integration`에 필터 패턴을 추가합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-integration \
```

```
    --integration-identifier my-integration \
    --data-filter 'include: foodb.*, exclude: foodb.tbl, exclude: foodb./table_\d+/'
```

Windows의 경우:

```
aws rds modify-integration ^
    --integration-identifier my-integration ^
    --data-filter 'include: foodb.*, exclude: foodb.tbl, exclude: foodb./table_\d+/'
```

RDS API

RDS API를 사용하여 제로 ETL 통합을 수정하려면 ModifyIntegration을 호출하세요. 통합 식별자를 지정하고 쉼표로 구분된 필터 패턴 목록을 제공하세요.

통합에서 데이터 필터 제거

통합에서 데이터 필터를 제거하면 Amazon RDS는 제거된 필터가 존재하지 않았던 것처럼 나머지 필터를 재평가합니다. 그런 다음, 이전에 제외되었지만 이제 기준을 충족하는 데이터를 대상 데이터 웨어하우스에 복제합니다. 그러면 모든 영향받는 테이블의 재동기화가 트리거됩니다.

소스 RDS 데이터베이스에 데이터 추가 및 Amazon Redshift에서 쿼리

> 이 문서는 미리 보기 릴리스 중이며 Amazon Redshift가 구성된 Amazon RDS 제로 ETL 통합에 대한 미리 보기 릴리스 설명서입니다. 설명서 및 기능은 모두 변경될 수 있습니다. 프로덕션 환경이 아닌 테스트 환경에서만 이 기능을 사용하는 것이 좋습니다. 미리 보기 이용 약관은 AWS 서비스 약관의 베타 및 미리 보기를 참조하세요.

Amazon RDS에서 Amazon Redshift로 데이터를 복제하는 제로 ETL 통합 생성을 완료하려면 Amazon Redshift에서 대상 데이터베이스를 생성해야 합니다.

먼저 Amazon Redshift 클러스터 또는 작업 그룹에 연결하고 통합 식별자에 대한 참조가 있는 데이터베이스를 생성합니다. 그런 다음 소스 RDS 데이터베이스에 데이터를 추가하고 Amazon Redshift에서 복제된 것을 확인할 수 있습니다.

주제

- Amazon Redshift에서 대상 데이터베이스 생성
- 소스 데이터베이스에 데이터 추가
- Amazon Redshift에서 Amazon RDS 데이터 쿼리
- RDS와 Amazon Redshift 데이터베이스 간의 데이터 유형 차이

Amazon Redshift에서 대상 데이터베이스 생성

통합을 생성한 후에 Amazon Redshift로 데이터 복제를 시작하려면 먼저 대상 데이터 웨어하우스에 대상 데이터베이스를 만들어야 합니다. 이 대상 데이터베이스에는 통합 식별자에 대한 참조가 포함되어야 합니다. Amazon Redshift 콘솔 또는 쿼리 편집기 v2를 사용하여 데이터베이스를 생성할 수 있습니다.

대상 데이터베이스를 생성하는 방법에 대한 지침은 Amazon Redshift에서 대상 데이터베이스 생성을 참조하세요.

소스 데이터베이스에 데이터 추가

통합을 구성한 후, Amazon Redshift 데이터 웨어하우스로 복제하려는 일부 데이터를 RDS 데이터베이스에 추가합니다.

> **Note**
>
> Amazon RDS와 Amazon Redshift의 데이터 유형 간에는 차이가 있습니다. 데이터 유형 매핑 표는 the section called “데이터 형식 차이” 섹션을 참조하세요.

먼저, 선택한 MySQL 클라이언트를 사용하여 소스 데이터베이스에 연결합니다. 지침은 the section called “MySQL 기반 DB 인스턴스에 연결” 단원을 참조하십시오.

그런 다음 테이블을 만들고 샘플 데이터 행을 삽입합니다.

> **Important**
>
> 테이블에 프라이머리 키가 있는지 확인하세요. 없는 경우 대상 데이터 웨어하우스에 복제할 수 없습니다.

다음 예제에서는 MySQL Workbench 유틸리티를 사용합니다.

```
CREATE DATABASE my_db;

USE my_db;

CREATE TABLE books_table (ID int NOT NULL, Title VARCHAR(50) NOT NULL, Author
 VARCHAR(50) NOT NULL,
Copyright INT NOT NULL, Genre VARCHAR(50) NOT NULL, PRIMARY KEY (ID));

INSERT INTO books_table VALUES (1, 'The Shining', 'Stephen King', 1977, 'Supernatural
 fiction');
```

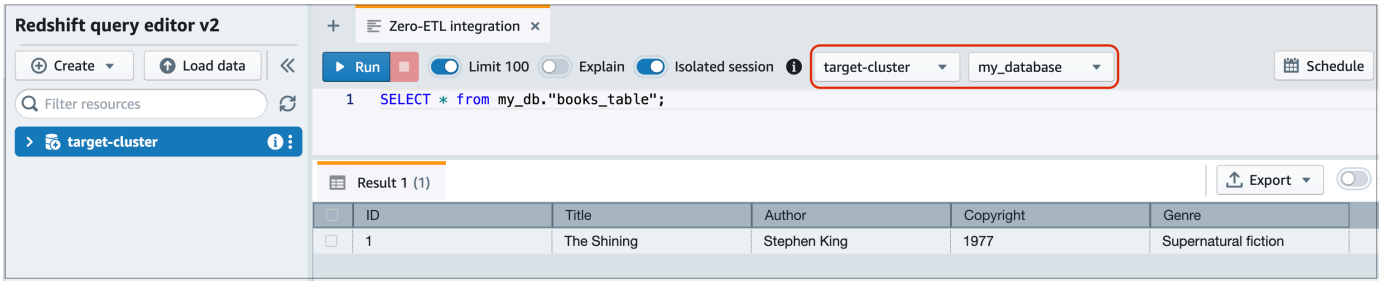
Amazon Redshift에서 Amazon RDS 데이터 쿼리

RDS 데이터베이스에 데이터를 추가하면 데이터가 Amazon Redshift에 복제되어 쿼리할 준비가 됩니다.

복제된 데이터를 쿼리하려면

1. Amazon Redshift 콘솔로 이동한 다음 왼쪽 탐색 창에서 쿼리 편집기 v2를 선택합니다.
2. 클러스터 또는 작업 그룹에 연결하고 드롭다운 메뉴(이 예시에서는 destination_database)에서 대상 데이터베이스(통합에서 생성한 데이터베이스)를 선택합니다. 대상 데이터베이스를 생성하는 방법에 대한 지침은 Amazon Redshift에서 대상 데이터베이스 생성을 참조하세요.
3. SELECT 문을 사용하여 데이터를 쿼리합니다. 이 예제에서는 다음 명령을 실행하여 소스 RDS 데이터베이스에서 생성한 테이블의 모든 데이터를 선택합니다.

```
SELECT * from my_db."books_table";
```



- *my_db*는 RDS 데이터베이스 스키마 이름입니다.
- *books_table*은 RDS 테이블 이름입니다.

명령줄 클라이언트를 사용하여 데이터를 쿼리할 수도 있습니다. 예:

```
destination_database=# select * from my_db."books_table";

 ID |       Title |        Author |   Copyright |                  Genre |  txn_seq |
 txn_id
----+-------------+---------------+-------------+------------------------+----------
+--------+
  1 | The Shining |  Stephen King |        1977 |   Supernatural fiction |        2 |
 12192
```

> **Note**
>
> 대/소문자를 구분하려면 스키마, 테이블 및 열 이름에 큰따옴표(" ")를 사용합니다. 자세한 내용은 enable_case_sensitive_identifier 단원을 참조하세요.

RDS와 Amazon Redshift 데이터베이스 간의 데이터 유형 차이

다음 표에는 해당하는 Amazon Redshift 데이터 유형에 대응하는 RDS for MySQL 데이터 유형의 매핑이 나와 있습니다. Amazon RDS는 현재 제로 ETL 통합에 이러한 데이터 유형만 지원합니다.

소스 데이터베이스의 테이블에 지원되지 않는 데이터 유형이 포함된 경우 테이블이 동기화되지 않아 Amazon Redshift 대상에서 사용할 수 없습니다. 소스에서 대상으로의 스트리밍은 계속되지만 지원되지 않는 데이터 유형이 있는 테이블은 사용할 수 없습니다. 테이블을 수정하고 Amazon Redshift에서 사용할 수 있게 하려면 주요 변경 사항을 수동으로 되돌린 다음 `ALTER DATABASE...INTEGRATION REFRESH`를 실행하여 통합을 새로 고쳐야 합니다.

RDS for MySQL

RDS for MySQL 데이터 유형	Amazon Redshift 데이터 형식	설명	제한 사항
INT	INTEGER	4바이트 부호화 정수	
SMALLINT	SMALLINT	2바이트 부호화 정수	

RDS for MySQL 데이터 유형	Amazon Redshift 데이터 형식	설명	제한 사항
TINYINT	SMALLINT	2바이트 부호화 정수	
MEDIUMINT	INTEGER	4바이트 부호화 정수	
BIGINT	BIGINT	8바이트 부호화 정수	
INT UNSIGNED	BIGINT	8바이트 부호화 정수	
TINYINT UNSIGNED	SMALLINT	2바이트 부호화 정수	
MEDIUMINT UNSIGNED	INTEGER	4바이트 부호화 정수	
BIGINT UNSIGNED	DECIMAL(20,0)	정밀도를 선택할 수 있는 정확한 숫자	
DECIMAL(p,s) = NUMERIC(p,s)	DECIMAL (p,s)	정밀도를 선택할 수 있는 정확한 숫자	38보다 큰 정밀도 및 37보다 큰 확장은 지원되지 않음
DECIMAL(p,s) UNSIGNED = NUMERIC(p,s) UNSIGNED	DECIMAL (p,s)	정밀도를 선택할 수 있는 정확한 숫자	38보다 큰 정밀도 및 37보다 큰 확장은 지원되지 않음
FLOAT4/REAL	REAL	단정밀도 부동 소수점 수	

RDS for MySQL 데이터 유형	Amazon Redshift 데이터 형식	설명	제한 사항
FLOAT4/REAL UNSIGNED	REAL	단정밀도 부동 소수점 수	
DOUBLE/REAL/FLOAT8	DOUBLE PRECISION	배정밀도 부동 소수점 수	
DOUBLE/REAL/FLOAT8 UNSIGNED	DOUBLE PRECISION	배정밀도 부동 소수점 수	
BIT(n)	VARBYTE(8)	가변 길이 이진 값	
BINARY(n)	VARBYTE(n)	가변 길이 이진 값	
VARBINARY(n)	VARBYTE(n)	가변 길이 이진 값	
CHAR(n)	VARCHAR(n)	길이가 가변적인 문자열 값	
VARCHAR(n)	VARCHAR(n)	길이가 가변적인 문자열 값	
TEXT	VARCHAR(65,535)	최대 65,535바이트의 길이가 가변적인 문자열 값	
TINYTEXT	VARCHAR(255)	최대 255바이트의 길이가 가변적인 문자열 값	
ENUM	VARCHAR(1,020)	최대 1,020바이트의 길이가 가변적인 문자열 값	

RDS for MySQL 데이터 유형	Amazon Redshift 데이터 형식	설명	제한 사항
SET	VARCHAR(1,020)	최대 1,020바이트의 길이가 가변적인 문자열 값	
날짜	날짜	날짜(년, 월, 일)	
DATETIME	TIMESTAMP	날짜/시간(시간대 제외)	
TIMESTAMP(p)	TIMESTAMP	날짜/시간(시간대 제외)	
TIME	VARCHAR(18)	최대 18바이트의 길이가 가변적인 문자열 값	
YEAR	VARCHAR(4)	최대 4바이트의 길이가 가변적인 문자열 값	
JSON	SUPER	반정형 데이터 또는 문서 값	

Amazon RDS 제로 ETL 통합 확인 및 모니터링

Amazon RDS 제로 ETL 통합의 세부 정보를 보고 구성 정보와 현재 상태를 확인할 수 있습니다. Amazon Redshift에서 특정 시스템 보기를 쿼리하여 통합 상태를 모니터링할 수도 있습니다. 또한 Amazon Redshift는 특정 통합 관련 지표를 Amazon CloudWatch에 게시하며, 이를 Amazon Redshift 콘솔에서 확인할 수 있습니다.

주제

- 통합 보기
- Amazon Redshift의 시스템 테이블을 사용한 통합 모니터링
- Amazon EventBridge로 Amazon Redshift에 대한 통합 모니터링

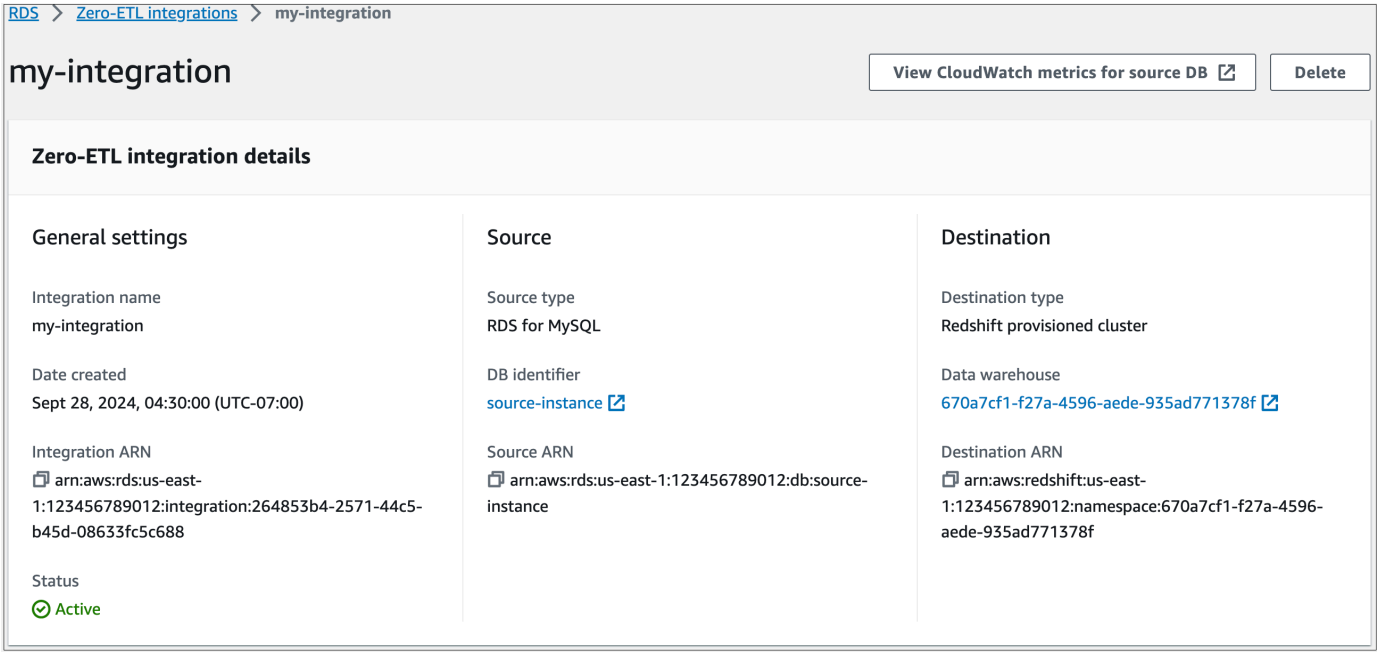
통합 보기

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 Amazon RDS 제로 ETL 통합을 볼 수 있습니다.

콘솔

제로 ETL 통합 세부 정보를 보려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 제로 ETL 통합을 선택합니다.
3. 통합을 선택하면 소스 데이터베이스 및 대상 데이터 웨어하우스와 같은 통합 세부 정보를 볼 수 있습니다.



통합에는 다음과 같은 상태가 있을 수 있습니다.

- `Creating` - 통합이 생성 중입니다.
- `Active` - 통합이 트랜잭션 데이터를 대상 데이터 웨어하우스로 전송하고 있습니다.
- `Syncing` - 통합에 복구 가능한 오류가 발생하여 데이터를 다시 시드하고 있습니다. 영향을 받는 테이블은 재동기화가 완료될 때까지 쿼리할 수 없습니다.
- `Needs attention` - 통합에 수동 개입이 필요한 이벤트 또는 오류가 발생하여 이를 해결해야 합니다. 문제를 해결하기 위해 통합 세부 정보 페이지에 있는 오류 메시지의 지침을 따릅니다.

- `Failed` - 통합에서 복구할 수 없는 이벤트 또는 수정할 수 없는 오류가 발생했습니다. 통합을 삭제하고 다시 만들어야 합니다.
- `Deleting` - 통합이 삭제 중입니다.

AWS CLI

AWS CLI를 사용하여 현재 계정의 모든 제로 ETL 통합을 보려면 describe-integration 명령을 사용하고 `--integration-identifier` 옵션을 지정하세요.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-integrations \
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

Windows의 경우:

```
aws rds describe-integrations ^
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

RDS API

Amazon RDS API를 사용하여 제로 ETL 통합을 보려면 `IntegrationIdentifier` 파라미터와 함께 `DescribeIntegrations` 작업을 사용하세요.

Amazon Redshift의 시스템 테이블을 사용한 통합 모니터링

Amazon Redshift에는 시스템 작동 방식에 대한 정보가 저장되어 있는 시스템 테이블 및 보기가 있습니다. 이러한 시스템 테이블 및 보기 역시 다른 데이터베이스 테이블에 대한 쿼리와 동일한 방법으로 쿼리를 실행할 수 있습니다. Amazon Redshift의 시스템 테이블 및 보기에 대한 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 시스템 테이블 및 뷰 참조를 참조하세요.

다음 시스템 뷰와 테이블을 쿼리하여 제로 ETL 통합에 대한 정보를 얻을 수 있습니다.

- SVV_INTEGRATION – 통합에 대한 구성 세부 정보를 제공합니다.
- SVV_INTEGRATION_TABLE_STATE – 통합 내 각 테이블의 상태를 설명합니다.
- SYS_INTEGRATION_TABLE_STATE_CHANGE - 통합에 대한 테이블 상태 변경 로그를 표시합니다.

- SYS_INTEGRATION_ACTIVITY – 완료된 통합 실행에 대한 정보를 제공합니다.

모든 통합 관련 Amazon CloudWatch 지표의 출처는 Amazon Redshift입니다. 자세한 내용은 Amazon Redshift 관리 안내서의 제로 ETL 통합의 지표를 참조하세요. 현재 Amazon RDS는 Amazon CloudWatch에 통합 지표를 게시하지 않습니다.

Amazon EventBridge로 Amazon Redshift에 대한 통합 모니터링

Amazon Redshift는 통합 관련 이벤트를 Amazon EventBridge로 보냅니다. 이벤트 및 해당 이벤트 ID의 목록은 Amazon Redshift 관리 가이드의 Amazon EventBridge를 사용한 제로 ETL 통합 이벤트 알림을 참조하세요.

Amazon RDS 제로 ETL 통합 수정

지원되는 데이터 웨어하우스에서 제로 ETL 통합의 이름, 설명 및 데이터 필터링 옵션만 수정할 수 있습니다. 통합을 암호화하는 데 사용된 AWS KMS 키나 소스 또는 대상 데이터베이스는 수정할 수 없습니다.

기존 통합에 데이터 필터를 추가하면 Amazon RDS가 필터가 항상 존재했던 것처럼 필터를 재평가합니다. 현재 대상 데이터 웨어하우스에 있는 데이터 중 새로운 필터링 기준과 일치하지 않는 모든 데이터를 제거합니다. 통합에서 데이터 필터를 제거하면 이전에는 필터링 기준과 일치하지 않았던(단, 지금은 일치) 모든 데이터가 대상 데이터 웨어하우스에 복제됩니다. 자세한 내용은 the section called “제로 ETL 통합의 데이터 필터링” 섹션을 참조하세요.

AWS Management Console, AWS CLI 또는 Amazon RDS API를 사용하여 제로 ETL 통합을 수정할 수 있습니다.

RDS 콘솔

제로 ETL 통합을 수정하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 제로 ETL 통합을 선택한 다음 수정하려는 통합을 선택합니다.
3. 수정을 선택하고 사용 가능한 설정을 수정합니다.
4. 원하는 대로 모두 변경되었으면 수정을 선택합니다.

AWS CLI

AWS CLI를 사용하여 제로 ETL 통합을 수정하려면 modify-integration 명령을 직접 호출하세요. `--integration-identifier`와 함께 다음 옵션 중 하나를 지정하세요.

- `--integration-name` - 통합의 새로운 이름을 지정합니다.
- `--description` - 통합의 새로운 설명을 지정합니다.
- `--data-filter` - 통합의 데이터 필터링 옵션을 지정합니다. 자세한 내용은 the section called “제로 ETL 통합의 데이터 필터링” 섹션을 참조하세요.

Example

다음 요청은 기존 통합을 수정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-integration \
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374 \
    --integration-name my-renamed-integration
```

Windows의 경우:

```
aws rds modify-integration ^
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374 ^
    --integration-name my-renamed-integration
```

RDS API

RDS API를 사용하여 제로 ETL 통합을 수정하려면 ModifyIntegration을 호출하세요. 통합 식별자 및 수정하려는 파라미터를 지정합니다.

Amazon RDS 제로 ETL 통합 삭제

제로 ETL 통합을 삭제할 때 Amazon RDS가 소스 데이터베이스에서 해당 통합을 제거합니다. 트랜잭션 데이터는 Amazon RDS나 분석 대상에서 삭제되지 않지만, Amazon RDS는 Amazon Redshift 또는 Amazon SageMaker로 새 데이터를 전송하지 않습니다.

상태가 `Active`, `Failed`, `Syncing` 또는 `Needs attention`인 통합만 삭제할 수 있습니다.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 제로 ETL 통합을 삭제할 수 있습니다.

콘솔

제로 ETL 통합을 삭제하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 제로 ETL 통합을 선택합니다.
3. 삭제하려는 제로 ETL 통합을 선택합니다.
4. 작업, 삭제를 차례로 선택한 후 삭제를 확인합니다.

AWS CLI

제로 ETL 통합을 삭제하려면 delete-integration 명령을 사용하고 `--integration-identifier` 옵션을 지정합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds delete-integration \
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

Windows의 경우:

```
aws rds delete-integration ^
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

RDS API

Amazon RDS API를 사용하여 제로 ETL 통합을 삭제하려면 `IntegrationIdentifier` 파라미터와 함께 `DeleteIntegration` 작업을 사용합니다.

Amazon Redshift가 구성된 Amazon RDS 제로 ETL 통합 문제 해결

이 문서는 미리 보기 릴리스 중이며 Amazon Redshift가 구성된 Amazon RDS 제로 ETL 통합에 대한 미리 보기 릴리스 설명서입니다. 설명서 및 기능은 모두 변경될 수 있습니다. 프로덕션 환경이 아닌 테스트 환경에서만 이 기능을 사용하는 것이 좋습니다. 미리 보기 이용 약관은 AWS 서비스 약관의 베타 및 미리 보기를 참조하세요.

Amazon Redshift에서 SVV_INTEGRATION 시스템 테이블을 쿼리하여 제로 ETL 통합의 상태를 확인할 수 있습니다. `state` 열의 값이 `ErrorState`면 문제가 있다는 뜻입니다. 자세한 내용은 the section called “시스템 테이블을 사용한 모니터링” 단원을 참조하십시오.

다음 정보를 사용하여 Amazon Redshift가 구성된 Amazon RDS 제로 ETL 통합과 관련된 일반적인 문제를 해결하세요.

주제

- 제로 ETL 통합을 생성할 수 없습니다
- 내 통합이 Syncing 상태에서 멈췄습니다.
- 내 테이블이 Amazon Redshift에 복제되지 않는 경우
- Amazon Redshift 테이블 중 하나 이상을 재동기화해야 합니다

제로 ETL 통합을 생성할 수 없습니다

제로 ETL 통합을 생성할 수 없는 경우 소스 DB 인스턴스에 다음 사항이 올바른지 확인하세요.

- 소스 데이터베이스는 RDS for MySQL 버전 8.0.32 이상을 실행해야 합니다. 엔진 버전을 확인하려면 데이터베이스의 구성 탭을 선택하여 엔진 버전을 확인하세요.
- DB 파라미터를 올바르게 구성했습니다. 필수 파라미터가 잘못 설정되었거나 DB 인스턴스와 연결되지 않은 경우 생성에 실패합니다. the section called “1단계: 사용자 지정 DB 파라미터 그룹 생성” 섹션을 참조하세요.

또한 대상 데이터 웨어하우스에 대해 다음 사항이 올바른지 확인하세요.

- 대소문자 구분이 활성화되어 있습니다. 데이터 웨어하우스에 대소문자 구분 기능 사용 설정을 참조하세요.

- 올바른 권한 있는 보안 주체 및 통합 소스를 추가했습니다. Amazon Redshift 데이터 웨어하우스에 대한 권한 구성을 참조하세요.
- 데이터 웨어하우스는 암호화되어 있습니다(프로비저닝된 클러스터인 경우). Amazon Redshift 데이터베이스 암호화를 참조하세요.

내 통합이 **Syncing** 상태에서 멈췄습니다.

필수 DB 클러스터 파라미터 중 하나의 값을 변경하면 통합에서 Syncing 상태가 일관되게 표시될 수 있습니다.

이 문제를 해결하려면 소스 데이터베이스와 연결된 파라미터 그룹의 파라미터 값을 살펴보고, 필수 값과 일치하는지 확인하세요. 자세한 내용은 the section called “1단계: 사용자 지정 DB 파라미터 그룹 생성” 단원을 참조하십시오.

파라미터를 수정할 경우 데이터베이스를 재부팅하여 변경 사항을 적용해야 합니다.

내 테이블이 Amazon Redshift에 복제되지 않는 경우

Amazon Redshift에 하나 이상의 표가 반영되지 않는 경우 다음 명령을 실행하여 표를 다시 동기화할 수 있습니다.

```
ALTER DATABASE dbname INTEGRATION REFRESH TABLES table1, table2;
```

자세한 내용은 Amazon Redshift SQL 참조의 ALTER DATABASE를 참조하세요.

하나 이상의 원본 테이블에 프라이머리 키가 없어 데이터가 복제되지 않을 수 있습니다. Amazon Redshift의 모니터링 대시보드에는 이러한 테이블의 상태가 Failed로 표시되고 전체 제로 ETL 통합 상태는 Needs attention으로 변경됩니다. 이 문제를 해결하려면 테이블에서 프라이머리 키가 될 수 있는 기존 키를 식별하거나 가상 프라이머리 키를 추가할 수 있습니다. 자세한 해결 방법은 Amazon Redshift를 사용해 Amazon Aurora MySQL 또는 Amazon RDS for MySQL 제로 ETL 통합을 생성하는 동안 프라이머리 키가 없는 테이블 처리를 참조하세요.

Amazon Redshift 테이블 중 하나 이상을 재동기화해야 합니다

소스 DB 인스턴스에서 특정 명령을 실행하려면 테이블을 재동기화해야 할 수 있습니다. 이러한 경우 SVV_INTEGRATION_TABLE_STATE 시스템 뷰에 ResyncRequired의 table_state가 표시됩니다. MySQL의 해당 테이블에서 Amazon Redshift로 데이터를 완전히 다시 로드해야 한다는 의미입니다.

테이블이 재동기화되기 시작하면 Syncing 상태가 됩니다. 테이블을 재동기화하기 위해 수동 작업을 수행할 필요가 없습니다. 테이블 데이터가 재동기화되는 동안에는 Amazon Redshift에서 액세스하지 못할 수도 있습니다.

다음은 테이블을 ResyncRequired 상태로 전환할 수 있는 몇 가지 예제 작업과 고려할 수 있는 대안입니다.

Operation	예	대안
특정 위치에 열 추가	`ALTER TABLE table_name ADD COLUMN column_name INTEGER NOT NULL first;`	Amazon Redshift는 first 또는 after 키워드를 사용하여 특정 위치에 열을 추가하는 것을 지원하지 않습니다. 대상 테이블의 열 순서가 중요하지 않은 경우 다음과 같은 간단한 명령을 사용하여 테이블 끝에 열을 추가합니다. `ALTER TABLE table_name ADD COLUMN column_name column_type ;`
기본 값 CURRENT_T	`ALTER TABLE table_name ADD COLUMN column_name TIMESTAMP`	기존 표 행의 CURRENT_T

Operation	예	대안
IMESTAMP 로 타임스탬프 열 추가	`NOT NULL DEFAULT CURRENT_TIMESTAMP;`	IMESTAMP 값은 RDS for MySQL에서 계산되며, 전체 표 데이터를 재동기화하지 않고는 Amazon Redshift에서 시뮬레이션할 수 없습니다. 가능하면 기본값을 리터럴 상수(예: `2023-01-01 00:00:15`)로 전환하여 테이블 가용성에 지연 시간이 생기지 않도록 하세요.

단일 명령 내에서 여러 열 작업 수행	`ALTER TABLE table_name`
	`  ADD COLUMN column_1,`
	`  RENAME COLUMN column_2 TO column_3;`
		명령을 2개의 개별 작업(ADD, RENAME)으로 분할하여 재동기화할 필요가 없도록 하는 것이 좋습니다.

Amazon RDS for Db2

Amazon RDS는 다음 IBM Db2 에디션을 실행하는 DB 인스턴스를 지원합니다.

- Db2 Standard Edition
- Db2 Advanced Edition

Amazon RDS는 다음 버전의 Db2를 실행하는 DB 인스턴스를 지원합니다.

- Db2 11.5

마이너 버전 지원에 대한 자세한 내용은 Amazon RDS의 Db2 버전 단원을 참조하세요.

DB 인스턴스를 생성하기 전에 이 사용 설명서의 Amazon RDS 환경 설정 섹션에 있는 단계를 완료해야 합니다. 마스터 사용자를 이용하여 DB 인스턴스를 생성하면 사용자에게 몇 가지 제한이 적용된 DBADM 권한이 부여됩니다. 추가 데이터베이스 계정 생성과 같은 관리 작업에 이 사용자를 사용합니다. SYSADM, SYSCTRL, SYSMAINT 인스턴스 수준 권한 또는 SECADM 데이터베이스 수준 권한은 사용할 수 없습니다.

다음을 생성할 수 있습니다.

- DB 인스턴스
- DB 스냅샷
- 특정 시점 복원
- 자동 스토리지 백업
- 수동 스토리지 백업

Virtual Private Cloud(VPC)에서 Db2를 실행하는 DB 인스턴스를 사용할 수 있습니다. 또한 다양한 옵션을 활성화하여 Amazon RDS for Db2 DB 인스턴스에 기능을 추가할 수 있습니다. Amazon RDS는 고가용성 장애 조치 솔루션으로서 RDS for Db2용 다중 AZ 배포를 지원합니다.

> **Important**
>
> 관리형 서비스 환경을 제공하기 위해 Amazon RDS는 DB 인스턴스에 대해 shell 액세스를 제공하지 않습니다. 또한 고급 권한이 필요한 특정 시스템 절차와 표에 대한 액세스를 제한합니다. IBM Db2 CLP와 같은 표준 SQL 클라이언트를 사용하여 데이터베이스에 액세스할 수 있습

니다. 그러나 Telnet 또는 SSH(Secure Shell)를 사용하여 호스트에 직접 액세스할 수는 없습니다.

주제

- Amazon RDS의 Db2 개요
- Amazon RDS for Db2 DB 인스턴스를 생성하기 위한 사전 조건
- 단일 Amazon RDS for Db2 DB 인스턴스의 여러 데이터베이스
- DB 인스턴스에 연결
- Amazon RDS for Db2 DB 인스턴스 연결 보안
- Amazon RDS for Db2 DB 인스턴스 관리
- Amazon RDS for Db2 DB 인스턴스와 Amazon S3 통합
- Amazon RDS for Db2로 데이터 마이그레이션
- Amazon RDS for Db2 페더레이션
- Amazon RDS for Db2의 복제본 작업
- Amazon RDS for Db2 DB 인스턴스 옵션
- Amazon RDS for Db2를 위한 외부 저장 프로시저
- Amazon RDS for Db2에 대해 알려진 문제 및 제한
- Amazon RDS for Db2 저장 프로시저 참조
- Amazon RDS for Db2 사용자 정의 함수 참조
- Amazon RDS for Db2 문제 해결

Amazon RDS의 Db2 개요

다음 섹션을 살펴보면 Amazon RDS의 Db2에 대한 개요를 파악할 수 있습니다.

주제

- Amazon RDS for Db2 기능
- Amazon RDS의 Db2 버전
- Amazon RDS for Db2 라이선스 옵션
- Amazon RDS for Db2 인스턴스 클래스
- Amazon RDS for Db2 기본 역할

- Amazon RDS for Db2 파라미터
- Amazon RDS에서 Db2 데이터베이스를 위한 EBCDIC 데이터 정렬
- Amazon RDS for Db2 DB 인스턴스의 현지 시간대

Amazon RDS for Db2 기능

Amazon RDS for Db2는 IBM Db2 데이터베이스의 특성과 기능을 대부분 지원합니다. 일부 기능에는 제한된 지원 또는 제한된 권한이 있을 수 있습니다. 특정 Db2 버전의 Db2 데이터베이스 기능에 대한 자세한 내용은 IBM Db2 설명서를 참조하세요.

데이터베이스 관련 새로운 소식 페이지에서 새로운 Amazon RDS 기능을 필터링할 수 있습니다. [제품(Products)]에서 [Amazon RDS]를 선택합니다. 그런 다음 **`Db2 2023`**과 같은 키워드를 사용하여 검색할 수 있습니다.

> **Note**
>
> 다음 목록은 완전하지 않습니다.

주제

- RDS for Db2에서 지원되는 기능
- RDS for Db2에서 지원되지 않는 기능

RDS for Db2에서 지원되는 기능

RDS for Db2는 IBM Db2의 기본 기능과 Amazon RDS의 핵심 기능을 포함하는 기능을 지원합니다.

IBM Db2의 기본 기능

RDS for Db2는 다음과 같은 Db2 데이터베이스 기능을 지원합니다.

- 고객이 정의한 코드 세트, 데이터 정렬, 페이지 크기 및 지역을 사용하는 표준 데이터베이스 생성 기능을 제공합니다. Amazon RDS rdsadmin.create_database 저장 프로시저를 사용합니다.
- 로컬 사용자 및 그룹 추가, 삭제 또는 수정 기능을 제공합니다. 권한 부여 및 취소에 대한 Amazon RDS 저장 프로시저를 사용합니다.
- Amazon RDS rdsadmin.create_role 저장 프로시저를 사용한 역할 생성 기능을 제공합니다.
- 표준 행 구성 표를 지원합니다.

- 열로 구성된 표의 분석 워크로드를 지원합니다.
- Oracle 및 MySQL과 같은 DB2 호환성 기능을 정의할 수 있습니다.
- Java 기반 외부 저장 프로시저 지원
- SSL/TLS를 사용하여 전송 중 데이터 암호화를 지원합니다.
- 데이터베이스 상태 모니터링(`ALIVE`, `DOWN`, `STORAGE_FULL`, `UNKNOWN`, `STANDBY_CONNECTABLE`) 기능을 제공합니다.
- 고객이 제공한 오프라인 또는 온라인 Linux (LE) 데이터베이스 복원 기능을 제공합니다. 데이터베이스 관리에 대한 Amazon RDS 저장 프로시저를 사용합니다.
- 고객이 제공한 Db2 아카이브 로그를 적용하여 데이터베이스를 자체 관리형 Db2 데이터베이스와 동기화한 상태를 유지합니다. 데이터베이스 관리에 대한 Amazon RDS 저장 프로시저를 사용합니다.
- Db2 인스턴스 수준 및 데이터베이스 수준 감사를 지원합니다.
- 동종 페더레이션을 지원합니다.
- Amazon Simple Storage Service(S3) 내 데이터 파일에서 표를 로드할 수 있습니다.
- 사용자, 그룹 또는 역할에 부여한 `CONNECT`, `SYSMON`, `ACCESSCTRL`, `DATAACCESS`, `SQLADM`, `WLMADM`, `EXPLAIN`, `LOAD`, `IMPLICIT_SCHEMA` 등의 권한
- 여러 데이터베이스 만들기.

> **Note**
>
> RDS for Db2 DB 인스턴스에는 최대 50개의 데이터베이스가 포함될 수 있습니다. 자세한 내용은 the section called “여러 Db2 데이터베이스” 섹션을 참조하세요.

Amazon RDS의 핵심 기능

RDS for Db2는 다음과 같은 Amazon RDS의 핵심 기능을 지원합니다.

- DB 인스턴스에 할당할 사용자 지정 파라미터 그룹
- DB 인스턴스 생성, 수정 및 삭제
- 자체 관리형 Db2 오프라인 또는 온라인 Linux (LE) 데이터베이스 백업 복원
- gp3, io2 및 io1 스토리지 유형 지원
- Kerberos 인증을 위한 AWS Managed Microsoft AD 사용 및 RDS for Db2를 위한 LDAP 그룹 인증
- 기존 Db2 인스턴스의 보안 그룹, 포트, 인스턴스 유형, 스토리지, 백업 보존 기간 및 기타 설정 수정
- DB 인스턴스를 위한 삭제 방지

- 리전 간 특정 시점 복구(PITR)
- 스토리지 암호화 및 저장 시 암호화에 AWS Key Management Service(AWS KMS) 사용
- 고가용성을 위한 예비 복제본이 하나인 다중 AZ DB 인스턴스
- DB 인스턴스 재부팅
- 마스터 암호 업데이트
- DB 인스턴스를 특정 시점으로 복원
- 스토리지 수준 백업을 사용한 DB 인스턴스의 백업 및 복원
- DB 인스턴스 시작 및 중지
- DB 인스턴스 유지 관리

RDS for Db2에서 지원되지 않는 기능

RDS for Db2는 다음과 같은 Db2 데이터베이스 기능을 지원하지 않습니다.

- 마스터 사용자의 `SYSADM`, `SECADM`, `SYSMAINT` 액세스
- C, C++ 또는 Cobol로 작성된 외부 저장 프로시저
- 단일 호스트의 여러 Db2 DB 인스턴스
- 인증을 위한 외부 GSS-API 플러그인
- Db2 데이터베이스 백업 또는 복원을 위한 외부 타사 플러그인
- IBM Db2 Warehouse 등의 다중 노드 대량 병렬 처리(MPP)
- IBM Db2 pureScale.
- RDS for Db2에 대한 고가용성 재해 복구(HADR)의 수동 설정입니다.

> **Note**
>
> Amazon RDS는 복제본을 통해 RDS for Db2에 대한 HADR을 지원하고 관리합니다. 자세한 내용은 Amazon RDS for Db2의 복제본 작업 섹션을 참조하세요.
> RDS for Db2는 다중 AZ 배포, 교차 리전 자동 백업 및 복제를 지원합니다. 자세한 내용은 Amazon RDS에 대한 다중 AZ 인스턴스 배포 및 다른 AWS 리전에 자동 백업 복제(을)를 참조하세요.

- 기본 데이터베이스 암호화
- Informix, Sybase 및 Teradata에 대한 이기종 페더레이션. 자세한 내용은 the section called “연동” 섹션을 참조하세요.

- 암호화된 백업의 리전 간 시점 복구(PITR)
- 데이터 백업 및 복원을 통한 펜싱되지 않은 루틴 생성 및 펜싱되지 않은 기존 루틴 마이그레이션. 자세한 내용은 울타리가 없는 루틴 섹션을 참조하세요.
- 자동이 아닌 새 스토리지 테이블스페이스 생성 자세한 내용은 마이그레이션 중 자동이 아닌 스토리지 테이블스페이스 섹션을 참조하세요.
- 외부 테이블.

Amazon RDS의 Db2 버전

Db2의 경우 버전 번호는 major.minor.build.revision의 형식을 취합니다(예: 11.5.9.0.sb00000000.r1). 버전 구현은 Db2의 버전 구현과 일치합니다.

메이저

버전 번호의 정수 부분과 첫 번째 소수 부분 모두가 메이저 버전 번호입니다(예: 11.5). 메이저 버전 번호가 변경되는 경우(예: 버전 11.5에서 12.1로 변경되는 경우) 메이저 버전 변경으로 간주합니다.

마이너

마이너 버전 번호는 버전 번호의 세 번째와 네 번째 부분 모두입니다(예: 11.5.9.0의 9.0). 세 번째 부분은 Db2 모드팩을 나타냅니다(예: 9.0의 9). 네 번째 부분은 Db2 픽스팩을 나타냅니다(예: 9.0의 0). Db2 모드팩 또는 Db2 픽스팩이 변경될 경우(예: 버전 11.5.9.0에서 11.5.9.1로 또는 11.5.9.0에서 11.5.10.0으로 변경되는 경우) 마이너 버전 변경으로 간주합니다. 단, 카탈로그 표 업데이트는 예외입니다. (Amazon RDS는 이러한 예외를 처리합니다.)

build

빌드 번호는 버전 번호의 다섯 번째 부분입니다. 예를 들어, 11.5.9.0.sb00000000의 sb00000000이 이에 해당합니다. 숫자 부분이 모두 0인 빌드 번호는 표준 빌드를 나타냅니다. 숫자 부분이 모두 0이 아닌 빌드 번호는 특수 빌드를 나타냅니다. 기존 Db2 버전의 보안 수정 사항이나 특수 빌드가 있는 경우 빌드 번호가 변경됩니다. 빌드 번호 변경은 Amazon RDS에서 새 마이너 버전을 자동으로 적용했음을 나타내기도 합니다.

개정

수정 번호는 버전 번호의 여섯 번째 부분입니다(예: 11.5.9.0.sb00000000.r1의 r1). 수정 버전은 기존 Db2 릴리스에 대한 Amazon RDS 개정판입니다. 수정 번호 변경은 Amazon RDS에서 새 마이너 버전을 자동으로 적용했음을 나타냅니다.

주제

- Amazon RDS에서 지원되는 Db2 마이너 버전
- Amazon RDS에서 지원되는 Db2 메이저 버전

Amazon RDS에서 지원되는 Db2 마이너 버전

다음 표는 현재 Amazon RDS가 지원하는 Db2 11.5의 마이너 버전을 보여줍니다.

> **Note**
>
> 월과 연도만 있는 날짜는 대략적인 날짜이며 알 수 있는 정확한 날짜로 업데이트됩니다.

Db2 엔진 버전	IBM 릴리스 날짜	RDS 릴리스 날짜	
11.5.9.0	2023년 11월 15일	2023년 11월 27일	

새 DB 인스턴스를 생성할 때는 현재 지원되는 모든 Db2 버전을 지정할 수 있습니다. 메이저 버전(예: Db2 11.5) 및 지정된 메이저 버전에 대해 지원되는 모든 마이너 버전을 지정할 수 있습니다. 버전이 지정되지 않은 경우 Amazon RDS는 지원되는 버전(보통 최신 버전)을 기본값으로 설정합니다. 메이저 버전이 지정되었지만 마이너 버전이 지정되지 않은 경우, Amazon RDS는 고객이 지정한 메이저 버전의 최근 릴리스를 기본값으로 설정합니다. 지원되는 버전 목록과 새로 만든 DB 인스턴스의 기본값을 보려면 describe-db-engine-versions AWS Command Line Interface(AWS CLI) 명령을 사용합니다.

예를 들어, 지원되는 Amazon RDS for Db2 엔진 버전 목록을 보려면 다음 AWS CLI 명령을 실행합니다. *##*을 사용자의 AWS 리전으로 바꿉니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-engine-versions \
    --filters Name=engine,Values=db2-ae,db2-se \
    --query "DBEngineVersions[].{Engine:Engine, EngineVersion:EngineVersion,
 DBParameterGroupFamily:DBParameterGroupFamily}" \
    --region region
```

Windows의 경우:

```
aws rds describe-db-engine-versions ^
    --filters Name=engine,Values=db2-ae,db2-se ^
```

```
    --query "DBEngineVersions[].{Engine:Engine, EngineVersion:EngineVersion,
 DBParameterGroupFamily:DBParameterGroupFamily}" ^
    --region region
```

다음 예제와 비슷한 출력이 생성됩니다.

```
[
    {
    "Engine": "db2-ae",
    "EngineVersion": "11.5.9.0.sb00000000.r1",
    "DBParameterGroupFamily": "db2-ae-11.5"
    },
    {
    "Engine": "db2-se",
    "EngineVersion": "11.5.9.0.sb00000000.r1",
    "DBParameterGroupFamily": "db2-se-11.5"
    }
]
```

기본 Db2 버전은 AWS 리전에 따라 다를 수 있습니다. 특정 마이너 버전으로 DB 인스턴스를 생성하려면 DB 인스턴스 생성 중에 마이너 버전을 지정합니다. db2-ae 및 db2-se 데이터베이스 엔진에 대한 AWS 리전의 기본 버전은 describe-db-engine-versions 명령을 실행하여 확인할 수 있습니다. 다음 예제에서는 미국 동부(버지니아 북부)의 db2-ae에 대한 기본 버전을 반환합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-engine-versions \
    --default-only --engine db2-ae \
    --query "DBEngineVersions[].{Engine:Engine, EngineVersion:EngineVersion,
 DBParameterGroupFamily:DBParameterGroupFamily}" \
    --region us-east-1
```

Windows의 경우:

```
aws rds describe-db-engine-versions ^
    --default-only --engine db2-ae ^
    --query "DBEngineVersions[].{Engine:Engine, EngineVersion:EngineVersion,
 DBParameterGroupFamily:DBParameterGroupFamily}" ^
    --region us-east-1
```

다음 예제와 비슷한 출력이 생성됩니다.

```
[
    {
    "Engine": "db2-ae",
    "EngineVersion": "11.5.9.0.sb00000000.r1",
    "DBParameterGroupFamily": "db2-ae-11.5"
    }
]
```

Amazon RDS를 통해 사용자는 Db2 인스턴스를 Amazon RDS가 지원하는 새 메이저 버전으로 언제 업그레이드할지를 제어합니다. 특정 Db2 버전과의 호환성을 유지하고, 프로덕션 환경에 배포하기 전에 애플리케이션으로 새 버전을 테스트하고, 가장 원하는 일정에 맞춰 메이저 버전 업그레이드를 수행할 수 있습니다.

자동 마이너 버전 업그레이드를 사용하면 Amazon RDS가 지원하는 DB 인스턴스를 새 Db2 마이너 버전으로 자동 업그레이드합니다. 이 패치는 예약 유지보수 중에 발생합니다. DB 인스턴스를 수정하여 자동 마이너 버전 업그레이드를 활성화 또는 비활성화할 수 있습니다.

Db2 버전 11.5.9.1 및 11.5.10.0을 제외하고 새 Db2 마이너 버전으로의 자동 업그레이드에는 새 빌드 및 수정본으로의 자동 업그레이드가 포함됩니다. 11.5.9.1 및 11.5.10.0의 경우 마이너 버전을 수동으로 업그레이드하세요.

자동으로 예약된 업그레이드를 사용하지 않기로 선택한 경우, 사용자는 메이저 버전 업데이트를 위해 선택한 것과 동일한 프로시저에 따라 지원되는 마이너 버전 릴리스로 수동으로 업그레이드할 수 있습니다. 자세한 내용은 DB 인스턴스 엔진 버전 업그레이드 섹션을 참조하세요.

Amazon RDS에서 지원되는 Db2 메이저 버전

RDS for Db2 메이저 버전은 해당 IBM 버전에 대해 IBM 지원(기본)이 종료될 때까지 표준 지원에 따라 사용할 수 있습니다. 다음 표에는 테스트 및 업그레이드 주기를 계획하는 데 참고할 수 있는 날짜가 나와 있습니다. Amazon이 RDS for Db2 버전에 대한 지원을 원래 명시일보다 오래 연장할 경우, 이 표를 이후 날짜를 반영하도록 업데이트할 계획입니다.

다음 날짜를 사용하여 테스트 및 업그레이드 주기를 계획할 수 있습니다.

> **Note**
>
> 월과 연도만 있는 날짜는 대략적인 날짜이며 알 수 있는 정확한 날짜로 업데이트됩니다. describe-db-major-engine-versions AWS CLI 명령을 실행하거나 DescribeDBMajorEngineVersions RDS API 작업을 사용하여 Db2 데이터베이스의 메이저 버전을 볼 수 있습니다.

Db2 메이저 버전	IBM 릴리스 날짜	RDS 릴리스 날짜	IBM 지원 종료(Standard 및 Advanced Edition)	IBM 지원 종료(연장)	
Db2 11.5	2019년 6월 27일	2023년 11월 27일	IBM에서 아직 발표하지 않음[1]	지원 종료 후 4년	

[1] 최신 정보는 IBM Db2 설명서의 Db2 product end-of-support dates를 참조하세요.

Amazon RDS for Db2 라이선스 옵션

Amazon RDS for Db2에는 기존 보유 라이선스 사용(BYOL) 라이선스 옵션과 AWS Marketplace를 통한 Db2 라이선스 옵션이 있습니다.

주제

- Db2에 기존 보유 라이선스 사용
- AWS Marketplace를 통한 Db2 라이선스
- Db2 라이선스 간 전환

Db2에 기존 보유 라이선스 사용

BYOL 모델에서는 기존의 Db2 데이터베이스 라이선스를 사용하여 Amazon RDS에서 데이터베이스를 배포합니다. 실행할 DB 인스턴스 클래스와 Db2 데이터베이스 에디션에 적합한 Db2 데이터베이스 라이선스가 있어야 합니다. 또한 클라우드 컴퓨팅 환경에서 IBM 데이터베이스 소프트웨어 라이선스 IBM 정책을 따라야 합니다.

> **Note**
>
> 다중 AZ DB 인스턴스는 Db2 데이터베이스가 설치되었으나 실행되지 않기 때문에 콜드 스탠바이 상태입니다. 스탠바이 인스턴스는 요청을 읽거나 실행 또는 처리할 수 없습니다. 자세한 내용은 IBM 웹 사이트의 IBM Db2 라이선스 정보를 참조하세요.

이 모델에서는 활성 IBM 지원 계정을 계속 사용합니다. Db2 데이터베이스 서비스 요청은 IBM에 직접 문의하세요. 지원 계정에 사례 지원이 있는 경우 Amazon RDS 관련 문제는 지원에 문의합니다. Amazon Web Services 및 IBM에는 두 조직의 지원이 필요한 사례에 대해 다중 벤더 지원 프로세스가 있습니다.

Amazon RDS는 Db2 Standard Edition 및 Db2 Advanced Edition에 대한 BYOL 모델을 지원합니다.

주제

- Db2에 기존 보유 라이선스 사용을 위한 IBM ID
- RDS for Db2 DB 인스턴스의 파라미터 그룹에 IBM ID 추가
- AWS License Manager과 통합

Db2에 기존 보유 라이선스 사용을 위한 IBM ID

BYOL 모델에서는 RDS for Db2 DB 인스턴스를 생성, 수정 또는 복원하려면 IBM Customer ID 및 IBM Site ID가 필요합니다. RDS for Db2 DB 인스턴스를 생성하기 전에 IBM Customer ID 및 IBM Site ID를 사용하여 사용자 지정 파라미터 그룹을 만들어야 합니다. 자세한 내용은 RDS for Db2 DB 인스턴스의 파라미터 그룹에 IBM ID 추가 단원을 참조하십시오. 동일한 AWS 계정 또는 AWS 리전에서 서로 다른 IBM Customer IDs와 IBM Site IDs를 가진 여러 RDS for Db2 DB 인스턴스를 실행할 수 있습니다.

> **Important**
>
> IBM Customer ID 및 IBM Site ID로 라이선스를 확인할 수 없는 경우 이러한 확인되지 않은 라이선스로 실행 중인 DB 인스턴스를 종료할 수 있습니다.

신규 IBM Db2 고객인 경우 먼저 Db2 소프트웨어 라이선스를 IBM에서 구매해야 합니다. Db2 소프트웨어 라이선스를 구매한 후에 IBM에서 IBM Customer ID와 IBM Site ID가 나와 있는 자격 증명을 받게 됩니다.

기존 IBM Db2 고객인 경우 IBM에서 제공하는 자격 증명 인증서에서 IBM Customer ID와 IBM Site ID를 찾아볼 수 있습니다.

IBM Passport Advantage Online 계정에서도 IBM Customer ID 및 IBM Site ID를 찾을 수 있습니다. 로그인한 후 기본 페이지 또는 소프트웨어 다운로드 페이지에서 두 ID를 모두 볼 수 있습니다.

RDS for Db2 DB 인스턴스의 파라미터 그룹에 IBM ID 추가

기본 파라미터 그룹을 수정할 수 없으므로, 사용자 지정 파라미터 그룹을 만든 다음 IBM Customer ID 및 IBM Site ID 값을 포함하도록 수정해야 합니다. 파라미터 그룹에 대한 자세한 내용은 Amazon RDS DB 인스턴스용 DB 파라미터 그룹 단원을 참조하세요.

> **Important**
>
> RDS for Db2 DB 인스턴스를 생성하기 전에 IBM Customer ID 및 IBM Site ID를 사용하여 사용자 지정 파라미터 그룹을 만들어야 합니다.

다음 표에 있는 파라미터 설정을 사용합니다.

파라미터	값
`rds.ibm_customer_id`	`<your IBM Customer ID>`
`rds.ibm_site_id`	`<your IBM Site ID>`
`ApplyMethod`	`immediate` , `pending-reboot`

파라미터는 동적이므로, 파라미터에 대한 모든 변경 사항이 즉시 적용되기 때문에 DB 인스턴스를 재부팅할 필요가 없습니다. 변경 사항을 즉시 적용하지 않으려면 `ApplyMethod`를 `pending-reboot`로 설정하여 유지 관리 기간에 변경을 예약할 수 있습니다.

AWS Management Console, AWS CLI 또는 Amazon RDS API를 통해 사용자 지정 파라미터 그룹을 생성하고 수정할 수 있습니다.

콘솔

IBM Customer ID 및 IBM Site ID를 파라미터 그룹에 추가하려면

1. 새 DB 파라미터 그룹을 만듭니다. DB 파라미터 그룹을 생성하는 것에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹 생성 단원을 참조하십시오.
2. 생성한 파라미터 그룹을 수정합니다. 파라미터 그룹 수정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요.

AWS CLI

IBM Customer ID 및 IBM Site ID를 파라미터 그룹에 추가하려면

1. create-db-parameter-group 명령을 실행하여 사용자 지정 파라미터 그룹을 생성합니다.

   다음 필수 옵션을 포함합니다.

   - `--db-parameter-group-name` – 생성하려는 파라미터 그룹의 이름입니다.
   - `--db-parameter-group-family` – Db2 엔진 에디션 및 메이저 버전입니다. 유효한 값: `db2-se-11.5`, `db2-ae-11.5`.
   - `--description` – 이 파라미터 그룹에 대한 설명입니다.

   DB 파라미터 그룹을 생성하는 것에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹 생성 단원을 참조하십시오.
2. modify-db-parameter-group 명령을 실행하여 만든 사용자 지정 파라미터 그룹의 파라미터를 수정합니다.

   다음 필수 옵션을 포함합니다.

   - `--db-parameter-group-name` – 생성한 파라미터 그룹의 이름입니다.
   - `--parameters` – 파라미터 업데이트를 위한 파라미터 이름, 값, 응용 방법으로 구성된 배열입니다.

   파라미터 그룹 수정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요.

RDS API

IBM Customer ID 및 IBM Site ID를 파라미터 그룹에 추가하려면

1. Amazon RDS API CreateDBParameterGroup 작업을 사용하여 DB 파라미터 그룹을 생성합니다.

   다음 필수 파라미터를 포함합니다.

   - `DBParameterGroupName`
   - `DBParameterGroupFamily`
   - `Description`

DB 파라미터 그룹을 생성하는 것에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹 생성 단원을 참조하십시오.

2. RDS API ModifyDBParameterGroup 작업을 사용하여 생성한 사용자 지정 파라미터 그룹의 파라미터를 수정합니다.

   다음 필수 파라미터를 포함합니다.

   - `DBParameterGroupName`
   - `Parameters`

   파라미터 그룹 수정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요.

이제 DB 인스턴스를 만들고 사용자 지정 파라미터 그룹을 DB 인스턴스에 연결할 준비가 되었습니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 및 Amazon RDS의 DB 인스턴스에 DB 파라미터 그룹 연결 단원을 참조하세요.

AWS License Manager과 통합

BYOL 모델에서 RDS for Db2 라이선스 사용을 모니터링하는 데 도움이 되도록 AWS License Manager는 RDS for Db2와 통합됩니다. License Manager는 Oracle 가상 CPU(vCPU)를 기반으로 RDS for Db2 엔진 버전 추적을 지원합니다. 또한 License Manager를 AWS Organizations와 함께 사용하여 모든 조직 계정을 중앙에서 관리할 수도 있습니다.

RDS for Db2 DB 인스턴스의 라이선스 사용량을 추적하려면 자체 관리형 라이선스를 만들어야 합니다. AWS Management Console, AWS License Manager CLI 및 AWS License Manager API를 사용하여 자체 관리형 라이선스를 만들 수 있습니다. 또는 AWS CloudFormation 및 Terraform 템플릿을 사용하여 자체 관리형 라이선스 만들기를 자동화할 수 있습니다.

제품 정보 필터와 일치하는 RDS for Db2 리소스가 자체 관리형 라이선스와 자동으로 연결됩니다. RDS for Db2 DB 인스턴스 검색에는 최대 24시간이 소요될 수 있습니다.

다음 표에는 RDS for Db2용 엔진 에디션 제품 정보 필터에 사용 가능한 값이 나와 있습니다.

값	설명
`db2-se`	Db2 Standard Edition

값	설명
db2-ae	Db2 Advanced Edition

주제

- 용어
- AWS License Manager에서 자체 관리형 라이선스 생성
- 템플릿을 사용하여 AWS License Manager에서 자체 관리형 라이선스 만들기 자동화
- 자체 관리형 라이선스를 만들기 위한 설정

용어

이 페이지에서는 Amazon RDS와 AWS License Manager의 통합을 이야기할 때 다음 용어를 사용합니다.

자체 관리형 라이선스

> 자체 관리형 라이선스는 AWS License Manager에서 사용되는 용어입니다. Amazon RDS 콘솔은 라이선스를 AWS License Manager 구성이라고 합니다. 자체 관리형 라이선스에는 기업 계약 조건에 기반한 라이선스 규칙이 포함됩니다. 생성하는 규칙은 AWS에서 라이선스를 사용하는 명령을 처리하는 방법을 결정합니다. 자체 관리형 라이선스를 만드는 동안 조직의 규정 준수 팀과 긴밀히 협력하여 기업 계약을 검토하세요. 자세한 내용은 License Manager의 자체 관리형 라이선스를 참조하세요.

AWS License Manager에서 자체 관리형 라이선스 생성

AWS Management Console, AWS License Manager CLI 및 AWS License Manager API를 사용하여 자체 관리형 라이선스를 만들 수 있습니다.

> **Note**
>
> AWS Management Console을 사용하여 RDS for Db2 DB 인스턴스를 생성하는 경우 라이선스의 이름을 입력하여 자체 관리형 라이선스를 생성합니다. 그런 다음 Amazon RDS는 DB 인스턴스를 이 라이선스와 연결합니다. (Amazon RDS 콘솔에서 이 라이선스를 AWS License Manager 구성이라고 합니다.) AWS License Manager CLI 또는 AWS License Manager API를 사용하여 RDS for Db2 DB 인스턴스를 생성하려면 먼저 다음 단계를 통해 자체 관리형 라이선

스를 생성해야 합니다. RDS for Db2 DB 인스턴스를 특정 시점으로 복원하거나 스냅샷에서 복원하는 경우에도 동일한 상황이 적용됩니다.

콘솔

RDS for Db2 DB 인스턴스의 라이선스 사용량을 추적하기 위한 자체 관리형 라이선스를 생성하는 방법

1. https://console.aws.amazon.com/license-manager/로 이동합니다.
2. 자체 관리형 라이선스를 생성합니다.

   자세한 내용은 AWS License Manager 사용 설명서의 Create a self-managed license를 참조하세요.

   제품 정보 패널에서 RDS Product Information Filter(RDS 제품 정보 필터)에 대한 규칙을 추가합니다.

   자세한 내용은AWS License Manager API 참조의 ProductInformation을 참조하십시오.

AWS License Manager CLI

> **Note**
>
> 이 절차에서는 AWS License Manager CLI 명령을 사용합니다.

AWS CLI를 사용하여 자체 관리형 라이선스를 만들려면 AWS License Manager create-license-configuration 명령을 실행합니다. `--cli-input-json` 또는 `--cli-input-yaml` 옵션을 사용하여 옵션을 명령에 전달합니다.

자세한 내용은 the section called “자체 관리형 라이선스를 만들기 위한 설정” 단원을 참조하십시오.

다음 명령은 Db2 Standard Edition에 대한 자체 관리형 라이선스를 만듭니다.

```
aws license-manager create-license-configuration --cli-input-json file://rds-db2-se.json
```

이전 명령의 `rds-db2-se.json` 파일에도 다음 JSON 내용이 있습니다.

```
{
    "Name": "rds-db2-se",
    "Description": "RDS Db2 Standard Edition",
    "LicenseCountingType": "vCPU",
    "LicenseCountHardLimit": false,
    "ProductInformationList": [
        {
            "ResourceType": "RDS",
            "ProductInformationFilterList": [
                {
                    "ProductInformationFilterName": "Engine Edition",
                    "ProductInformationFilterValue": ["db2-se"],
                    "ProductInformationFilterComparator": "EQUALS"
                }
            ]
        }
    ]
}
```

제품 정보에 대한 자세한 내용은 AWS License Manager 사용 설명서의 리소스 인벤토리 자동 검색을 참조하십시오.

`--cli-input` 파라미터에 대한 자세한 내용은 AWS CLI 사용 설명서의 JSON 또는 YAML 입력 파일에서 AWS CLI 스켈레톤 및 입력 파라미터 생성을 참조하세요.

AWS License Manager API

> **Note**
>
> 이 절차에서는 AWS License Manager API 명령을 사용합니다.

자체 관리형 라이선스를 생성하려면 다음과 같은 필수 파라미터로 `CreateLicenseConfiguration` AWS License Manager API 작업을 사용합니다.

- `Name`
- `LicenseCountingType`
- `ProductInformationList`
- `ResourceType`
- `ProductInformationFilterList`

- `ProductInformationFilterName`
- `ProductInformationFilterValue`
- `ProductInformationFilterComparator`

파라미터에 대한 자세한 내용은 the section called “자체 관리형 라이선스를 만들기 위한 설정” 섹션을 참조하세요.

템플릿을 사용하여 AWS License Manager에서 자체 관리형 라이선스 만들기 자동화

AWS CloudFormation 및 Terraform 템플릿을 사용하여 자체 관리형 라이선스 만들기를 자동화할 수 있습니다.

다음 예시 AWS CloudFormation 템플릿은 RDS for Db2의 Db2 Standard Edition에 대한 자체 관리형 라이선스를 만듭니다. Db2 Advanced Edition용 템플릿의 경우 `Name`, `Description` 및 `ProductInformationFilter`의 값을 업데이트합니다.

```
AWSTemplateFormatVersion: "2010-09-09"
Description: CloudFormation template to create a License Configuration for Db2 Standard
 Edition on RDS for Db2.

Resources:
  Db2LicenseConfiguration:
    Type: "AWS::LicenseManager::LicenseConfiguration"
    Properties:
      Name: "rds-db2-se"
      Description: "Db2 Standard Edition on RDS for Db2"
      LicenseCountingType: "vCPU"
      LicenseCountHardLimit: false
      ProductInformationList:
        - ResourceType: "RDS"
          ProductInformationFilterList:
            - ProductInformationFilterName: "Engine Edition"
              ProductInformationFilterValue:
                - "db2-se"
              ProductInformationFilterComparator: "EQUALS"
```

Amazon RDS에서 AWS CloudFormation를 사용하는 방법에 대한 자세한 내용은 AWS CloudFormation을 사용하여 Amazon RDS 리소스 생성 단원을 참조하십시오.

다음 예시 Terraform 템플릿은 RDS for Db2의 Db2 Standard Edition에 대한 자체 관리형 라이선스를 만듭니다. *us-east-1*을 해당 AWS 리전으로 대체합니다. Db2 Advanced Edition용 템플릿의 경우 `name`, `description` 및 `product_information_filter`의 값을 업데이트합니다.

```
provider "aws" {
  region = "us-east-1"
}

resource "aws_licensemanager_license_configuration" "rds_db2_license_config" {
  name                     = "rds-db2-se"
  description              = "Db2 Standard Edition on RDS for Db2
  license_counting_type    = "vCPU"
  license_count_hard_limit = false

  product_information_list {
    resource_type = "RDS"

    product_information_filter {
      name       = "Engine Edition"
      comparator = "EQUALS"
      value      = ["db2-se"]
    }
  }
}
```

Terraform 및 Amazon RDS 사용에 대한 자세한 내용은 AWS 권장 가이드의 Using Terraform as an IaC tool for the AWS 클라우드 및 Best practices for using the Terraform AWS Provider를 참조하세요.

자체 관리형 라이선스를 만들기 위한 설정

다음 표에서는 AWS License Manager CLI, AWS License Manager API, AWS CloudFormation 템플릿, Terraform 템플릿을 사용하여 자체 관리형 라이선스를 만들기 위한 설정에 대한 세부 정보를 찾을 수 있습니다. 다음 표의 파라미터 이름은 AWS License Manager API 및 AWS CloudFormation 템플릿에서 사용되는 이름의 형식으로 표시됩니다.

파라미터 이름	데이터 형식	필수	설명
명칭	문자열	예	구성 설정의 이름입니다.

파라미터 이름	데이터 형식	필수	설명
설명	문자열	No	라이선스 구성에 대한 설명입니다.
LicenseCountingType	문자열	예	라이선스 인벤토리를 추적하는 데 사용되는 차원입니다. 유효한 값: vCPU.
LicenseCountHardLimit	boolean	No	하드 또는 소프트 라이선스 적용이 사용되는지 여부를 나타냅니다. 하드 한도를 초과하면 새 인스턴스의 시작이 차단됩니다.
ProductInformationList	객체 배열	예	라이선스 구성에 대한 제품 정보 목록입니다.
ResourceType	문자열	예	리소스 유형. 유효한 값: RDS.
ProductInformationFilterList	객체 배열	예	라이선스 구성에 대한 제품 정보 필터 목록입니다.
ProductInformationFilterName	문자열	예	선언할 필터 유형의 이름입니다. 유효한 값: Engine Edition.
ProductInformationFilterValue	문자열 배열	예	필터링할 값입니다. 값을 하나만 지정할 수 있습니다. 유효한 값: db2-se 또는 db2-ae

파라미터 이름	데이터 형식	필수	설명
ProductInformationFilterComparator	문자열	예	`ProductInformationFilterName` 의 논리적 연산자입니다. 유효한 값: `EQUALS`.

AWS Marketplace를 통한 Db2 라이선스

AWS Marketplace를 통한 Db2 라이선스 모델에서는 시간당 요금을 지불하고 Db2 라이선스를 구독할 수 있습니다. 이 모델을 사용하면 라이선스를 구매할 필요 없이 RDS for Db2를 빠르게 시작할 수 있습니다.

AWS Marketplace를 통한 Db2 라이선스를 사용하려면 사용하려는 특정 IBM Db2 에디션에 대한 활성 AWS Marketplace 구독이 필요합니다. 아직 구독이 없는 경우 해당 IBM Db2 에디션에 대해 AWS Marketplace를 구독합니다.

Amazon RDS는 IBM Db2 Standard Edition과 IBM Db2 Advanced Edition에 AWS Marketplace를 통한 Db2 라이선스를 지원합니다.

주제

- 용어
- 결제 및 청구
- Db2 Marketplace 리스팅 구독 및 IBM으로 등록
- 비공개 제안 받기

용어

이 페이지에서는 Amazon RDS와 AWS Marketplace의 통합을 이야기할 때 다음 용어를 사용합니다.

SaaS 구독

> AWS Marketplace에서는 사용한 만큼만 지불하는 라이선스 모델과 같은 서비스형 소프트웨어(SaaS) 제품이 사용량 기반 구독 모델을 채택합니다. Db2의 소프트웨어 판매자인 IBM이 사용량을 추적하고, 고객은 사용한 만큼 요금을 지불합니다.

공개 제안

공개 제안을 통해 AWS Management Console에서 직접 AWS Marketplace 제품을 구매할 수 있습니다.

비공개 제안

비공개 제안은 판매자와 구매자가 AWS Marketplace에서 구매에 대한 사용자 지정 가격과 최종 사용자 라이선스 계약(EULA) 조건을 협상할 수 있는 구매 프로그램입니다.

Db2 Marketplace 요금

Db2 소프트웨어 라이선스 사용에 대해 IBM에서 청구하는 요금. 이러한 서비스 요금은 AWS Marketplace를 통해 측정되며 AWS 청구서에서 AWS Marketplace 섹션에 표시됩니다.

Amazon RDS 수수료

AWS에서 RDS for Db2 서비스에 부과하는 요금으로, AWS Marketplace for Db2 라이선스를 사용할 때 라이선스가 제외됩니다. 요금은 사용 중인 Amazon RDS를 통해 측정되며 AWS 청구서에 표시됩니다.

결제 및 청구

RDS for Db2는 AWS Marketplace와 통합되어 Db2에 대해 사용한 만큼만 시간당 요금을 지불하는 라이선스를 제공합니다. Db2 Marketplace 요금은 Db2 소프트웨어의 라이선스 비용을 포함하며, Amazon RDS 요금은 RDS for Db2 DB 인스턴스 사용에 대한 비용을 포함합니다. 요금에 대한 자세한 내용은 Amazon RDS for Db2 요금을 참조하세요.

이러한 요금이 부과되지 않도록 하려면 RDS for Db2 DB 인스턴스를 모두 삭제해야 합니다. 또한 AWS Marketplace for Db2 라이선스 구독을 제거할 수 있습니다. DB 인스턴스를 삭제하지 않고 구독을 제거하면 Amazon RDS에서 계속해서 DB 인스턴스 사용에 대한 요금을 청구합니다. 자세한 내용은 the section called “DB 인스턴스 삭제” 단원을 참조하십시오.

AWS Billing 콘솔에서 AWS Marketplace를 통한 Db2 라이선스를 사용하는 RDS for Db2 DB 인스턴스의 청구서를 보고 결제를 관리할 수 있습니다. 청구서에는 두 가지 요금이 포함됩니다. 하나는 AWS Marketplace를 통한 Db2 라이선스 사용에 대한 요금이고 다른 하나는 Amazon RDS 사용에 대한 요금입니다. 청구에 대한 자세한 내용은 AWS 결제 및 비용 관리 사용 설명서에서 Viewing your bill을 참조하세요.

Db2 Marketplace 리스팅 구독 및 IBM으로 등록

AWS Marketplace를 통한 Db2 라이선스를 사용하려면 AWS Management Console을 사용하여 다음 두 작업을 완료해야 합니다. AWS CLI 또는 RDS API를 통해서는 이러한 작업을 완료할 수 없습니다.

> **Note**
>
> AWS CLI 또는 RDS API를 사용하여 DB 인스턴스를 만들려면 먼저 이 두 작업을 완료해야 합니다.

주제

- 작업 1: AWS Marketplace에서 Db2를 구독합니다.
- 작업 2: IBM으로 구독 등록

작업 1: AWS Marketplace에서 Db2를 구독합니다.

AWS Marketplace를 통해 Db2 라이선스를 사용하려면 Db2에 대한 활성 AWS Marketplace 구독이 있어야 합니다. 구독은 특정 IBM Db2 에디션과 연결되어 있으므로 사용하려는 Db2의 각 에디션(IBM Db2 Advanced Edition, IBM Db2 Standard Edition)에 대해 AWS Marketplace에서 Db2를 구독해야 합니다. AWS Marketplace 구독에 대한 자세한 내용은 AWS Marketplace 구매자 안내서의 Saas usage-based subscriptions를 참조하세요.

DB 인스턴스 생성을 시작하기 전에 AWS Marketplace에서 Db2를 구독하는 것이 좋습니다.

작업 2: IBM으로 구독 등록

AWS Marketplace에서 Db2를 구독한 후 선택한 선택한 Db2 구독 유형에 대한 AWS Marketplace 페이지에서 IBM 주문 등록을 완료합니다. AWS Marketplace 페이지에서 구매 옵션 보기를 선택한 다음 계정 설정을 선택합니다. 기존 IBM 계정으로 등록하거나 무료 IBM 계정을 만들어 등록할 수 있습니다.

비공개 제안 받기

IBM에서 Db2에 대한 AWS Marketplace 비공개 제안을 요청할 수 있습니다. 자세한 내용은 AWS Marketplace 구매자 설명서의 비공개 제안을 참조하세요.

> **Note**
>
> AWS Organizations 사용자로서 지급인 및 회원 계정에 발행된 비공개 제안을 받은 경우 아래 절차에 따라 조직의 각 계정에서 Db2를 직접 구독하세요.

Db2 비공개 제안을 받으려면

1. 비공개 제안이 발행되면 AWS Marketplace 콘솔에 로그인합니다.

2. Db2 비공개 제안 링크가 포함된 이메일을 엽니다.
3. 링크를 따라 비공개 제안에 직접 액세스합니다.

> **Note**
>
> 올바른 계정에 로그인하지 않고 이 링크를 따라가면 Page not found(404) 오류가 발생합니다.

4. 이용 약관을 검토합니다.
5. 약관에 동의를 선택합니다.

> **Note**
>
> AWS Marketplace 비공개 제안이 수락되지 않는 경우에도 AWS Marketplace의 Db2 서비스 수수료는 계속해서 공개 시간당 요율로 청구됩니다.

6. 제안 세부 정보를 확인하려면 제품 목록에서 세부 정보 표시를 선택합니다.

절차를 완료한 후 the section called “DB 인스턴스 생성”의 단계에 따라 DB 인스턴스를 만들 수 있습니다. AWS Management Console의 라이선스에서 AWS Marketplace를 통해를 선택했는지 확인하세요.

Db2 라이선스 간 전환

RDS for Db2에서 Db2 라이선스 간에 전환할 수 있습니다. 예를 들어 기존 보유 라이선스 사용으로 시작한 다음 AWS Marketplace를 통한 Db2 라이선스로 전환할 수 있습니다.

> **Important**
>
> AWS Marketplace를 통한 Db2 라이선스로 전환하려면 사용하려는 IBM Db2 에디션에 대한 AWS Marketplace 구독이 활성화되어 있는지 확인하세요. 활성화되어 있지 않으면 먼저 해당 Db2 에디션에 대해 AWS Marketplace에서 Db2를 구독한 다음 복원 절차를 완료하세요.

콘솔

Db2 라이선스 간에 전환하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.

2. 탐색 창에서 자동 백업(Automated backups)을 선택합니다.

   자동 백업은 현재 리전(Current Region) 탭에 표시됩니다.
3. 복원하려는 DB 인스턴스를 선택합니다.
4. 작업에서 특정 시점으로 복구를 선택합니다.

   특정 시점으로 복구 창이 나타납니다.
5. 최근 복원 가능 시간을 선택하여 가능한 최근 시간으로 복원하거나, 사용자 지정을 선택하여 시간을 선택합니다.

   사용자 지정을 선택한 경우 인스턴스를 복원하려는 날짜와 시간을 입력합니다.

   > **Note**
   >
   > 시간은 현지 시간대로 표시됩니다. 즉, 협정 세계시(UTC)에서 오프셋으로 표시됩니다. 예를 들어 UTC-5는 동부 표준시/하절기 중부 표준시입니다.
6. DB 엔진에서 사용하려는 Db2 라이선스를 선택합니다.
7. DB 인스턴스 식별자에 대상 복원된 DB 인스턴스의 이름을 입력합니다. 이름은 고유해야 합니다.
8. 필요에 따라 DB 인스턴스 클래스, 스토리지, 스토리지 자동 크기 조정 사용 여부 등의 다른 옵션을 선택합니다.

   각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하세요.
9. 특정 시점으로 복구를 선택합니다.

자세한 내용은 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 단원을 참조하십시오.

AWS CLI

Db2 라이선스 간에 전환하려면 restore-db-instance-to-point-in-time 명령을 실행합니다. 다음 예시에서는 최신 특정 시점 버전으로 복원하고, DB 엔진을 IBM Db2 Advanced Edition으로 설정하고, 라이선스 모델을 AWS Marketplace를 통한 Db2 라이선스로 설정합니다.

다른 설정을 지정할 수 있습니다. 각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하세요.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds restore-db-instance-to-point-in-time \
    --source-db-instance-identifier my_source_db_instance \
    --target-db-instance-identifier my_target_db_instance \
    --use-latest-restorable-time \
    --engine db2-ae \
    --license-model marketplace-license
```

Windows의 경우:

```
aws rds restore-db-instance-to-point-in-time ^
    --source-db-instance-identifier my_source_db_instance ^
    --target-db-instance-identifier my_target_db_instance ^
    --use-latest-restorable-time ^
    --engine db2-ae ^
    --license-model marketplace-license
```

자세한 내용은 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 단원을 참조하십시오.

RDS API

Db2 라이선스 간에 전환하려면, Amazon RDS API `RestoreDBInstanceToPointInTime` 작업을 다음 파라미터와 함께 직접 호출합니다.

- `SourceDBInstanceIdentifier`
- `TargetDBInstanceIdentifier`
- `RestoreTime`
- `Engine`
- `LicenseModel`

자세한 내용은 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 단원을 참조하십시오.

Amazon RDS for Db2 인스턴스 클래스

DB 인스턴스의 계산 및 메모리 용량은 해당 DB 인스턴스 클래스에 의해 결정됩니다. 필요한 DB 인스턴스 클래스는 DB 인스턴스의 처리력 및 메모리 요구 사항에 따라 다릅니다.

지원되는 RDS for Db2 인스턴스 클래스

지원되는 Amazon RDS for Db2 인스턴스 클래스는 Amazon RDS DB 인스턴스 클래스의 하위 집합입니다. Amazon RDS 인스턴스 클래스의 전체 목록은 DB 인스턴스 클래스 섹션을 참조하세요.

주제

- Db2 Standard Edition에 대해 지원되는 RDS for Db2 인스턴스 클래스
- Db2 Advanced Edition에 대해 지원되는 RDS for Db2 인스턴스 클래스

Db2 Standard Edition에 대해 지원되는 RDS for Db2 인스턴스 클래스

다음 표에는 Db2 데이터베이스 버전 11.5.9.0의 Db2 Standard Edition에 대해 지원되는 모든 인스턴스 클래스가 나열되어 있습니다. 이러한 인스턴스 클래스는 기존 보유 라이선스 사용(BYOL) 및 AWS Marketplace를 통한 Db2 라이선스 모두에 사용할 수 있습니다.

인스턴스 클래스 유형	인스턴스 클래스
3세대 Intel Xeon Scalable 프로세서, SSD 스토리지 및 네트워크 최적화 기능이 갖춰진 범용 인스턴스 클래스	db.m6idn.large–db.m6idn.8xlarge
3세대 Intel Xeon Scalable 프로세서로 구동되는 범용 인스턴스 클래스	db.m6in.large–db.m6in.8xlarge
범용 인스턴스 클래스	db.m7i.large–db.m7i.8xlarge db.m6i.large–db.m6i.8xlarge
3세대 Intel Xeon Scalable 프로세서로 구동되는 로컬 NVMe 기반 SSD가 갖춰진 메모리 최적화 인스턴스 클래스	db.x2iedn.xlarge
3세대 Intel Xeon Scalable 프로세서로 구동되는 메모리 최적화 인스턴스 클래스	db.r6idn.large–db.r6idn.4xlarge db.r6in.large–db.r6in.4xlarge
메모리 최적화 인스턴스 클래스	db.r7i.large–db.r7i.8xlarge db.r6i.large–db.r6i.4xlarge

인스턴스 클래스 유형	인스턴스 클래스
버스트 가능한 성능 인스턴스 클래스	db.t3.small–db.t3.2xlarge

Db2 Advanced Edition에 대해 지원되는 RDS for Db2 인스턴스 클래스

다음 표에는 Db2 데이터베이스 버전 11.5.9.0의 Db2 Advanced Edition에 대해 지원되는 모든 인스턴스 클래스가 나열되어 있습니다. 이러한 인스턴스 클래스는 기존 보유 라이선스 사용(BYOL) 및 AWS Marketplace를 통한 Db2 라이선스 모두에 사용할 수 있습니다.

인스턴스 클래스 유형	인스턴스 클래스
3세대 Intel Xeon Scalable 프로세서, SSD 스토리지 및 네트워크 최적화 기능이 갖춰진 범용 인스턴스 클래스	db.m6idn.large–db.m6idn.32xlarge
3세대 Intel Xeon Scalable 프로세서로 구동되는 범용 인스턴스 클래스	db.m6in.large–db.m6in.32xlarge
범용 인스턴스 클래스	db.m6i.large–db.m7i.48xlarge db.m7i.large–db.m7i.48xlarge
3세대 Intel Xeon Scalable 프로세서로 구동되는 로컬 NVMe 기반 SSD가 갖춰진 메모리 최적화 인스턴스 클래스	db.x2iedn.xlarge–db.x2iedn.32xlarge
3세대 Intel Xeon Scalable 프로세서로 구동되는 메모리 최적화 인스턴스 클래스	db.r6idn.large–db.r6idn.32xlarge db.r6in.8xlarge–db.r6in.32xlarge
메모리 최적화 인스턴스 클래스	db.r6i.large–db.r7i.48xlarge db.r7i.large–db.r7i.48xlarge
버스트 가능한 성능 인스턴스 클래스	db.t3.small–db.t3.2xlarge

Amazon RDS for Db2 기본 역할

RDS for Db2는 다음 6가지 역할을 추가하고 ADMIN 옵션을 통해 해당 역할을 master_user_role에 부여합니다. 데이터베이스가 프로비저닝되면 RDS for Db2가 마스터 사용자에게 master_user_role을 부여합니다. 그러면 마스터 사용자는 데이터베이스에 연결하여 네이티브 GRANT 문으로 다른 사용자, 그룹 또는 역할에 이러한 역할을 부여할 수 있습니다.

- DBA – RDS for Db2는 DATAACCESS 권한으로 이 빈 역할을 생성합니다. 마스터 사용자는 이 역할에 더 많은 인증 또는 권한을 추가한 다음 다른 사용자, 그룹 또는 역할에 해당 역할을 부여할 수 있습니다.
- DBA_RESTRICTED – RDS for Db2는 이 빈 역할을 생성합니다. 마스터 사용자는 이 역할에 더 많은 권한을 추가한 다음 다른 사용자, 그룹 또는 역할에 해당 역할을 부여할 수 있습니다.
- DEVELOPER – RDS for Db2는 DATAACCESS 권한으로 이 빈 역할을 생성합니다. 마스터 사용자는 이 역할에 더 많은 인증 또는 권한을 추가한 다음 다른 사용자, 그룹 또는 역할에 해당 역할을 부여할 수 있습니다.
- ROLE_NULLID_PACKAGES – RDS for Db2는 CREATE DATABASE 실행 당시 Db2에 바인딩된 ALL NULLID 패키지에서 이 역할에 EXECUTE 권한을 부여합니다.
- ROLE_PROCEDURES – RDS for Db2는 모든 SYSIBM 절차에서 이 역할에 EXECUTE 권한을 부여합니다.
- ROLE_TABLESPACES – RDS for Db2는 CREATE DATABASE 명령으로 생성된 테이블스페이스에서 USAGE 권한을 부여합니다.

Amazon RDS for Db2 파라미터

Amazon RDS for Db2는 데이터베이스 관리자 구성 파라미터, 레지스트리 변수, 데이터베이스 구성 파라미터 등 세 가지 유형의 파라미터를 사용합니다. 처음 두 유형은 파라미터 그룹을 통해 관리하고 마지막 유형은 rdsadmin.update_db_param 저장 절차를 통해 관리할 수 있습니다.

기본적으로, RDS for Db2 DB 인스턴스는 Db2 데이터베이스 및 DB 인스턴스에 고유한 DB 파라미터 그룹을 사용합니다. 이 파라미터 그룹에는 IBM Db2 데이터베이스 엔진에 대한 파라미터가 포함되어 있습니다. 특히 데이터베이스 관리자 구성 파라미터와 레지스트리 변수가 포함되어 있습니다. 파라미터 그룹 작업에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하세요.

RDS for Db2 데이터베이스 구성 파라미터는 사용자가 선택한 스토리지 엔진의 기본값으로 설정됩니다. Db2 파라미터에 대한 자세한 내용은 IBM Db2 설명서의 Db2 데이터베이스 구성 파라미터를 참조하세요.

주제

- 파라미터 그룹의 파라미터 보기
- Db2 명령으로 모든 파라미터 보기
- 파라미터 그룹의 파라미터를 수정합니다.
- Db2 명령을 사용하여 데이터베이스 구성 파라미터 수정

파라미터 그룹의 파라미터 보기

데이터베이스 관리자 구성 파라미터와 레지스트리 변수는 파라미터 그룹에서 설정됩니다. AWS Management Console, AWS CLI 또는 RDS API를 사용하여 특정 Db2 버전의 데이터베이스 관리자 구성 파라미터와 레지스트리 변수를 볼 수 있습니다.

콘솔

DB 파라미터 그룹의 파라미터 값을 보려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.

   DB 파라미터 그룹이 목록에 나타납니다.
3. 파라미터 그룹의 이름을 선택하여 파라미터 목록을 봅니다.

AWS CLI

describe-engine-default-parameters 명령을 실행하여 Db2 버전의 데이터베이스 관리자 구성 파라미터와 레지스트리 변수를 볼 수 있습니다. `--db-parameter-group-family` 옵션에 대해 다음 값 중 하나를 지정할 수 있습니다.

- `db2-ae-11.5`
- `db2-se-11.5`

예를 들어, Db2 Standard Edition 11.5에 대한 파라미터를 보려면 다음 명령을 실행합니다.

```
aws rds describe-engine-default-parameters --db-parameter-group-family db2-se-11.5
```

다음 예제와 비슷한 출력이 생성됩니다.

```
{
    "EngineDefaults": {
        "Parameters": [
            {
                "ParameterName": "agent_stack_sz",
                "ParameterValue": "1024",
                "Description": "You can use this parameter to determine the amount of
 memory that is allocated by Db2 for each agent thread stack.",
                "Source": "engine-default",
                "ApplyType": "static",
                "DataType": "integer",
                "AllowedValues": "256-32768",
                "IsModifiable": false
            },
            {
                "ParameterName": "agentpri",
                "ParameterValue": "-1",
                "Description": "This parameter controls the priority given to all
 agents and to other database manager instance processes and threads by the operating
 system scheduler. This priority determines how CPU time is allocated to the database
 manager processes, agents, and threads relative to other processes and threads running
 on the machine.",
                "Source": "engine-default",
                "ApplyType": "static",
                "DataType": "integer",
                "AllowedValues": "1-99",
                "IsModifiable": false
            },
            ...
        ]
    }
}
```

Db2 Standard Edition 11.5에 대한 수정 가능 파라미터만 나열하려면 다음 명령을 실행합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-engine-default-parameters \
    --db-parameter-group-family db2-se-11.5 \
    --query 'EngineDefaults.Parameters[?IsModifiable==`true`].
{ParameterName:ParameterName, DefaultValue:ParameterValue}'
```

Windows의 경우:

```
aws rds describe-engine-default-parameters ^
    --db-parameter-group-family db2-se-11.5 ^
    --query 'EngineDefaults.Parameters[?IsModifiable==`true`].
{ParameterName:ParameterName, DefaultValue:ParameterValue}'
```

RDS API

DB 파라미터 그룹의 파라미터 값을 보려면 DescribeDBParameters 작업을 다음 필수 파라미터와 함께 사용합니다.

- DBParameterGroupName

Db2 명령으로 모든 파라미터 보기

Db2 명령을 사용하여 데이터베이스 관리자 구성 파라미터, 데이터베이스 구성 파라미터 및 레지스트리 변수에 대한 설정을 볼 수 있습니다.

설정을 확인하려면

1. Db2 데이터베이스에 연결합니다. 다음 예제에서 *database_name*, *master_username*, *master_password*를 사용자 정보로 바꿉니다.

   ```
   db2 "connect to database_name user master_username using master_password"
   ```

2. 지원되는 Db2 버전을 찾습니다.

   ```
   db2 "select service_level, fixpack_num from table(sysproc.env_get_inst_info()) as
    instanceinfo"
   ```

3. 특정 Db2 버전의 파라미터를 확인합니다.

   - 다음 명령을 실행하여 데이터베이스 관리자 구성 파라미터를 확인합니다.

     ```
     db2 "select cast(substr(name,1,24) as varchar(24)) as name, case
         when value_flags = 'NONE' then '' else value_flags end flags,
         cast(substr(value,1,64) as varchar(64)) as current_value
         from sysibmadm.dbmcfg
         order by name asc with UR"
     ```

- 다음 명령을 실행하여 모든 데이터베이스 구성 파라미터를 확인합니다.

```
db2 "select cast(substr(name,1,24) as varchar(24)) as name, case
    when value_flags = 'NONE' then '' else value_flags end flags,
    cast(substr(value,1,64) as varchar(64)) as current_value
    from table(db_get_cfg(null)) order by name asc, member asc with UR"
```

- 다음 명령을 실행하여 현재 설정된 레지스트리 변수를 확인합니다.

```
db2 "select cast(substr(reg_var_name,1,50) as varchar(50)) as reg_var_name,
    cast(substr(reg_var_value,1,50) as varchar(50)) as reg_var_value,
    level from table(env_get_reg_variables(null))
    order by reg_var_name,member with UR"
```

파라미터 그룹의 파라미터를 수정합니다.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 사용자 지정 파라미터 그룹의 데이터베이스 관리자 구성 파라미터와 레지스트리 변수를 수정할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스용 DB 파라미터 그룹 단원을 참조하십시오.

콘솔

데이터베이스 관리자 구성 파라미터와 레지스트리 변수를 수정하려면

1. 사용자 지정 DB 파라미터 그룹을 만듭니다. DB 파라미터 그룹을 생성하는 것에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹 생성 단원을 참조하십시오.
2. 사용자 지정 파라미터 그룹의 파라미터를 수정하는 방법 파라미터 그룹 수정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요.

AWS CLI

데이터베이스 관리자 구성 파라미터와 레지스트리 변수를 수정하려면

1. create-db-parameter-group 명령을 실행하여 사용자 지정 파라미터 그룹을 생성합니다.

   다음 필수 옵션을 포함합니다.

   - `--db-parameter-group-name` – 생성하려는 파라미터 그룹의 이름입니다.

- `--db-parameter-group-family` – Db2 엔진 에디션 및 메이저 버전입니다. 유효한 값: `db2-se-11.5`, `db2-ae-11.5`.
- `--description` – 이 파라미터 그룹에 대한 설명입니다.

DB 파라미터 그룹을 생성하는 것에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹 생성 단원을 참조하십시오.

2. modify-db-parameter-group 명령을 실행하여 만든 사용자 지정 파라미터 그룹의 파라미터를 수정합니다.

   다음 필수 옵션을 포함합니다.

   - `--db-parameter-group-name` – 생성한 파라미터 그룹의 이름입니다.
   - `--parameters` – 파라미터 업데이트를 위한 파라미터 이름, 값, 응용 방법으로 구성된 배열입니다.

   파라미터 그룹 수정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요.

RDS API

데이터베이스 관리자 구성 파라미터와 레지스트리 변수를 수정하려면

1. CreateDBParameterGroup 작업을 사용하여 사용자 지정 DB 파라미터 그룹을 생성합니다.

   다음 필수 파라미터를 포함합니다.

   - `DBParameterGroupName`
   - `DBParameterGroupFamily`
   - `Description`

   DB 파라미터 그룹을 생성하는 것에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹 생성 단원을 참조하십시오.

2. ModifyDBParameterGroup 작업을 사용하여 생성한 사용자 지정 파라미터 그룹의 파라미터를 수정합니다.

   다음 필수 파라미터를 포함합니다.

- DBParameterGroupName
- Parameters

파라미터 그룹 수정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요.

Db2 명령을 사용하여 데이터베이스 구성 파라미터 수정

Db2 명령으로 데이터베이스 구성 파라미터를 수정할 수 있습니다.

데이터베이스 구성 파라미터를 수정하려면

1. rdsadmin 데이터베이스에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 정보로 대체합니다.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. rdsadmin.update_db_param 저장 절차를 직접적으로 호출하여 데이터베이스 구성 파라미터를 변경합니다. 자세한 내용은 rdsadmin.update_db_param 단원을 참조하십시오.

```
db2 "call rdsadmin.update_db_param(
    'database_name',
    'parameter_to_modify',
    'changed_value')"
```

Amazon RDS에서 Db2 데이터베이스를 위한 EBCDIC 데이터 정렬

Amazon RDS for Db2는 Db2 데이터베이스에 대한 EBCDIC 데이터 정렬을 지원합니다. Amazon RDS the section called “rdsadmin.create_database” 저장 프로시저를 사용하여 데이터베이스를 생성할 때만 데이터베이스에 대해 EBCDIC 데이터 정렬 시퀀스를 지정할 수 있습니다.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 RDS for Db2 DB 인스턴스를 생성할 때 데이터베이스 이름을 지정할 수 있습니다. 데이터베이스 이름을 지정하는 경우 Amazon RDS는 기본 데이터 정렬이 SYSTEM인 데이터베이스를 생성합니다. EBCDIC 데이터 정렬을 사용하는 데이터베이스를 생성해야 하는 경우 DB 인스턴스를 생성할 때 데이터베이스 이름을 지정하지 마세요.

RDS for Db2의 데이터베이스 데이터 정렬은 생성 시점에 설정되고 변경할 수 없습니다.

EBCDIC 데이터 정렬을 사용하는 Db2 데이터베이스를 만들려면

1. RDS for Db2 DB 인스턴스가 없는 경우 DB 인스턴스를 만들되 데이터베이스 이름을 지정하지 마세요. AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB 인스턴스를 만들 수 있습니다. 자세한 내용은 DB 인스턴스 생성 단원을 참조하십시오.
2. Db2 데이터베이스를 만들고 `rdsadmin.create_database` 저장 프로시저를 호출하여 데이터 정렬 옵션을 EBCDIC 값으로 설정합니다. 자세한 내용은 rdsadmin.create_database 단원을 참조하십시오.

> **Important**
>
> 저장 프로시저를 사용하여 데이터베이스를 만든 후에는 데이터 정렬 시퀀스를 변경할 수 없습니다. 데이터베이스에서 다른 데이터 정렬 시퀀스를 사용하려면 the section called "rdsadmin.drop_database" 저장 프로시저를 호출하여 데이터베이스를 삭제하세요. 그런 다음 필요한 데이터 정렬 시퀀스로 데이터베이스를 생성합니다.

Amazon RDS for Db2 DB 인스턴스의 현지 시간대

Db2를 실행 중인 Amazon RDS DB 인스턴스의 시간대가 기본적으로 설정되어 있습니다. 기본값은 협정 세계 표준시(UTC)입니다. 애플리케이션의 시간대와 일치하도록 대신 DB 인스턴스의 시간대를 현지 시간대로 설정할 수 있습니다.

DB 인스턴스를 처음 만들 때 시간대를 설정합니다. AWS Management Console, RDS API, AWS CLI를 사용하여 DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 DB 인스턴스 생성 단원을 참조하십시오.

DB 인스턴스가 다중 AZ 배포의 일부인 경우 장애 조치 중에 시간대가 설정된 현지 시간대로 유지됩니다.

DB 인스턴스를 지정한 시점으로 복원할 수 있습니다. 시간은 현지 시간대로 표시됩니다. 자세한 내용은 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 단원을 참조하십시오.

DB 인스턴스에 대해 현지 시간대를 설정할 때 다음 제한 사항이 적용됩니다.

- 기존 Amazon RDS for Db2 DB 인스턴스의 시간대를 수정할 수 없습니다.
- DB 인스턴스의 스냅샷을 다른 시간대의 DB 인스턴스로 복원할 수 없습니다.

- 한 표준 시간대의 백업 파일을 다른 표준 시간대로 복원하지 않는 것이 좋습니다. 한 표준 시간대의 백업 파일을 다른 표준 시간대로 복원하는 경우 쿼리와 애플리케이션을 감사하여 표준 시간대 변경의 영향을 확인해야 합니다.

사용 가능한 시간대

시간대 설정에 다음 값을 사용할 수 있습니다.

영역	시간대
아프리카	Africa/Cairo, Africa/Casablanca, Africa/Harare, Africa/Lagos, Africa/Luanda, Africa/Monrovia, Africa/Nairobi, Africa/Tripoli, Africa/Windhoek
아메리카	America/Araguaina, America/Argentina/Buenos_Aires, America/Asuncion, America/Bogota, America/Caracas, America/Chicago, America/Chihuahua, America/Cuiaba, America/Denver, America/Detroit, America/Fortaleza, America/Godthab, America/Guatemala, America/Halifax, America/Lima, America/Los_Angeles, America/Manaus, America/Matamoros, America/Mexico_City, America/Monterrey, America/Montevideo, America/New_York, America/Phoenix, America/Santiago, America/Sao_Paulo, America/Tijuana, America/Toronto
아시아	Asia/Amman, Asia/Ashgabat, Asia/Baghdad, Asia/Baku, Asia/Bangkok, Asia/Beirut, Asia/Calcutta, Asia/Damascus, Asia/Dhaka, Asia/Hong_Kong, Asia/Irkutsk, Asia/Jakarta, Asia/Jerusalem, Asia/Kabul, Asia/Karachi, Asia/Kathmandu, Asia/Kolkata, Asia/Krasnoyarsk, Asia/Magadan, Asia/Manila, Asia/Muscat, Asia/Novosibirsk, Asia/Rangoon, Asia/Riyadh, Asia/Seoul, Asia/Shanghai, Asia/Singapore, Asia/Taipei, Asia/Tehran, Asia/Tokyo, Asia/Ulaanbaatar, Asia/Vladivostok, Asia/Yakutsk, Asia/Yerevan
대서양	Atlantic/Azores, Atlantic/Cape_Verde
호주	Australia/Adelaide, Australia/Brisbane, Australia/Darwin, Australia/Eucla, Australia/Hobart, Australia/Lord_Howe, Australia/Perth, Australia/Sydney
브라질	Brazil/DeNoronha, Brazil/East
캐나다	Canada/Newfoundland, Canada/Saskatchewan

영역	시간대
기타	Etc/GMT-3
유럽	Europe/Amsterdam, Europe/Athens, Europe/Berlin, Europe/Dublin, Europe/Helsinki, Europe/Kaliningrad, Europe/London, Europe/Madrid, Europe/Moscow, Europe/Paris, Europe/Prague, Europe/Rome, Europe/Sarajevo, Europe/Stockholm
태평양	Pacific/Apia, Pacific/Auckland, Pacific/Chatham, Pacific/Fiji, Pacific/Guam, Pacific/Honolulu, Pacific/Kiritimati, Pacific/Marquesas, Pacific/Samoa, Pacific/Tongatapu, Pacific/Wake
US	US/Alaska, US/Central, US/East-Indiana, US/Eastern, US/Pacific
UTC	UTC

Amazon RDS for Db2 DB 인스턴스를 생성하기 위한 사전 조건

다음 항목은 DB 인스턴스를 생성하기 전에 필요한 사전 조건입니다.

주제

- 관리자 계정
- 추가 고려 사항

관리자 계정

DB 인스턴스를 생성할 때 해당 인스턴스의 관리자 계정을 지정해야 합니다. Amazon RDS는 이 로컬 데이터베이스 관리자 계정에 DBADM 권한을 부여합니다.

관리자 계정에는 다음과 같은 특징과 기능이 있으며 제한이 적용됩니다.

- 로컬 사용자이지만, AWS 계정는 아닙니다.
- SYSADM, SYSMAINT 또는 SYSCTRL 등의 Db2 인스턴스 수준 권한이 없습니다.
- Db2 인스턴스를 중지하거나 시작할 수 없습니다.
- DB 인스턴스를 생성할 때 이름을 지정한 경우 Db2 데이터베이스를 삭제할 수 없습니다.
- 카탈로그 표 및 뷰를 포함하여 Db2 데이터베이스에 대한 전체 액세스 권한을 가집니다.
- Amazon RDS 저장 프로시저를 사용하여 로컬 사용자 및 그룹을 생성할 수 있습니다.
- 자격 및 권한을 부여하거나 취소할 수 있습니다.

관리자 계정은 다음 작업을 수행할 수 있습니다.

- DB 인스턴스를 생성, 수정 또는 삭제합니다.
- DB 스냅샷을 만듭니다.
- 특정 시점으로 복원을 시작합니다.
- DB 스냅샷의 자동 백업을 생성합니다.
- DB 스냅샷의 수동 백업을 생성합니다.
- Amazon RDS의 기타 기능을 사용합니다.

추가 고려 사항

DB 인스턴스를 생성하기 전에 다음 항목을 고려하세요.

- 각 Amazon RDS for Db2 DB 인스턴스는 최대 50개의 Db2 데이터베이스를 호스팅할 수 있습니다. 자세한 내용은 단일 Amazon RDS for Db2 DB 인스턴스의 여러 데이터베이스 단원을 참조하십시오.
- 초기 데이터베이스 이름
  - DB 인스턴스를 생성할 때 데이터베이스 이름을 입력하지 않으면 Amazon RDS는 데이터베이스를 생성하지 않습니다.
  - 다음과 같은 상황에서는 데이터베이스 이름을 입력하지 마세요.
    - `db2_compatibility_vector` 파라미터를 수정하려고 합니다. 자세한 내용은 db2_compatibility_vector 파라미터 설정 단원을 참조하십시오.
- 기존 보유 라이선스 사용(BYOL) 모델에서는 먼저 IBM Customer ID와 IBM Site ID를 포함하는 사용자 지정 파라미터 그룹을 만들어야 합니다. 자세한 내용은 Db2에 기존 보유 라이선스 사용 단원을 참조하십시오.
- AWS Marketplace를 통한 Db2 라이선스 모델에서는 사용하려는 특정 IBM Db2 에디션에 대한 활성 AWS Marketplace 구독이 필요합니다. 아직 구독이 없는 경우 사용하려는 IBM Db2 에디션에 대해 AWS Marketplace에서 Db2를 구독합니다. 자세한 내용은 AWS Marketplace를 통한 Db2 라이선스 단원을 참조하십시오.

단일 Amazon RDS for Db2 DB 인스턴스의 여러 데이터베이스

rdsadmin.create_database 저장 프로시저를 직접 호출하여 단일 RDS for Db2 DB 인스턴스에 여러 데이터베이스를 만들 수 있습니다. 단일 RDS for Db2 DB 인스턴스는 데이터베이스 50개로 제한됩니다. 이 개수에는 활성화된 상태와 비활성화된 상태의 데이터베이스가 모두 포함됩니다.

> **Note**
>
> 2024년 11월 15일 전에 만들어진 RDS for Db2 DB 인스턴스에 여러 데이터베이스를 만드는 경우 DB 인스턴스를 재부팅하여 여러 데이터베이스에 대한 지원을 활성화해야 합니다.

기본적으로 Amazon RDS는 데이터베이스를 만들 때 데이터베이스를 활성화합니다. 메모리 리소스를 최적화하기 위해 자주 사용하지 않는 데이터베이스를 비활성화한 다음 나중에 필요할 때 활성화할 수 있습니다. 자세한 내용은 the section called “데이터베이스 비활성화” 및 the section called “데이터베이스 활성화” 단원을 참조하세요.

DB 인스턴스에서 활성화된 데이터베이스의 수는 서버에서 사용 가능한 메모리 리소스에 따라 달라집니다. 메모리 리소스는 DB 인스턴스 클래스와 데이터베이스에 대해 구성된 메모리의 양에 따라 다릅니다. DB 인스턴스 클래스에 대한 자세한 내용은 the section called “DB 인스턴스 클래스” 섹션을 참조하세요. RDS for Db2 데이터베이스의 메모리를 업데이트하는 방법에 대한 자세한 내용은 the section called “rdsadmin.update_db_param” 섹션을 참조하세요.

일반적인 데이터베이스 작업, 운영 체제 요구 사항 및 백업과 같은 기타 Amazon RDS 자동화 작업을 위해 2GB의 메모리가 있는 DB 인스턴스 클래스를 선택하는 것이 좋습니다. DB 인스턴스 클래스 변경에 대한 자세한 내용은 the section called “DB 인스턴스 수정” 섹션을 참조하세요.

또한 IBM은 각 활성 데이터베이스에 대해 최소 1GB의 메모리를 권장합니다. 자세한 내용은 IBM 설명서의 Disk and memory requirements를 참조하세요.

다음 공식을 사용하여 DB 인스턴스가 가질 수 있는 최대 활성 데이터베이스 수를 계산할 수 있습니다.

```
Active database limit = (total server memory - 2 GB) / 1 GB
```

다음 예시에서는 db.m6i.xlarge DB 인스턴스 클래스를 사용하는 DB 인스턴스의 최대 활성 데이터베이스 수를 보여줍니다.

```
Active database limit = (total server memory - 2 GB) / 1 GB
                      = (16 GB - 2 GB) / 1 GB
```

```
                    = 14 databases
```

Amazon RDS는 충돌 후 데이터베이스를 복구할 때 이전에 활성 상태였던 경우 데이터베이스를 활성화합니다. DB 인스턴스 클래스를 더 낮은 메모리 구성으로 수정하는 경우와 같은 특정 상황에서는 DB 인스턴스의 모든 데이터베이스를 활성화할 수 있는 메모리가 부족할 수 있습니다. 이러한 경우 Amazon RDS는 데이터베이스가 만들어진 순서대로 데이터베이스를 활성화합니다.

> **Note**
>
> 메모리 부족으로 인해 Amazon RDS가 활성화할 수 없는 모든 데이터베이스는 비활성화된 상태로 유지됩니다.

DB 인스턴스에 연결

Amazon RDS가 Amazon RDS for Db2 DB 인스턴스를 프로비저닝한 후에는 표준 SQL 클라이언트 애플리케이션을 사용해 DB 인스턴스에 연결할 수 있습니다. Amazon RDS는 관리형 서비스이므로, `SYSADM` `SYSCTRL`, `SECADM`, 또는 `SYSMAINT`로 로그인할 수 없습니다.

IBM Db2 CLP, IBM CLPPlus, DBeaver, 또는 IBM Db2 Data Management Console을 사용하여 IBM Db2 데이터베이스 엔진을 실행하는 DB 인스턴스에 연결할 수 있습니다.

> **Note**
>
> RDS for Db2 DB 인스턴스에 메모리가 충분하지 않은 경우 Db2 데이터베이스에 연결하지 못할 수 있습니다. 자세한 내용은 the section called “데이터베이스 연결 오류” 단원을 참조하십시오.

주제

- Amazon RDS for Db2 DB 인스턴스의 엔드포인트 찾기
- IBM Db2 CLP을 사용하여 Amazon RDS for Db2 DB 인스턴스에 연결
- IBM CLPPlus을 사용하여 Amazon RDS for Db2 DB 인스턴스에 연결
- DBeaver을 사용하여 Amazon RDS for Db2 DB 인스턴스에 연결
- IBM Db2 Data Management Console을 사용하여 Amazon RDS for Db2 DB 인스턴스에 연결
- Amazon RDS for Db2를 사용하는 보안 그룹에 대한 고려 사항

Amazon RDS for Db2 DB 인스턴스의 엔드포인트 찾기

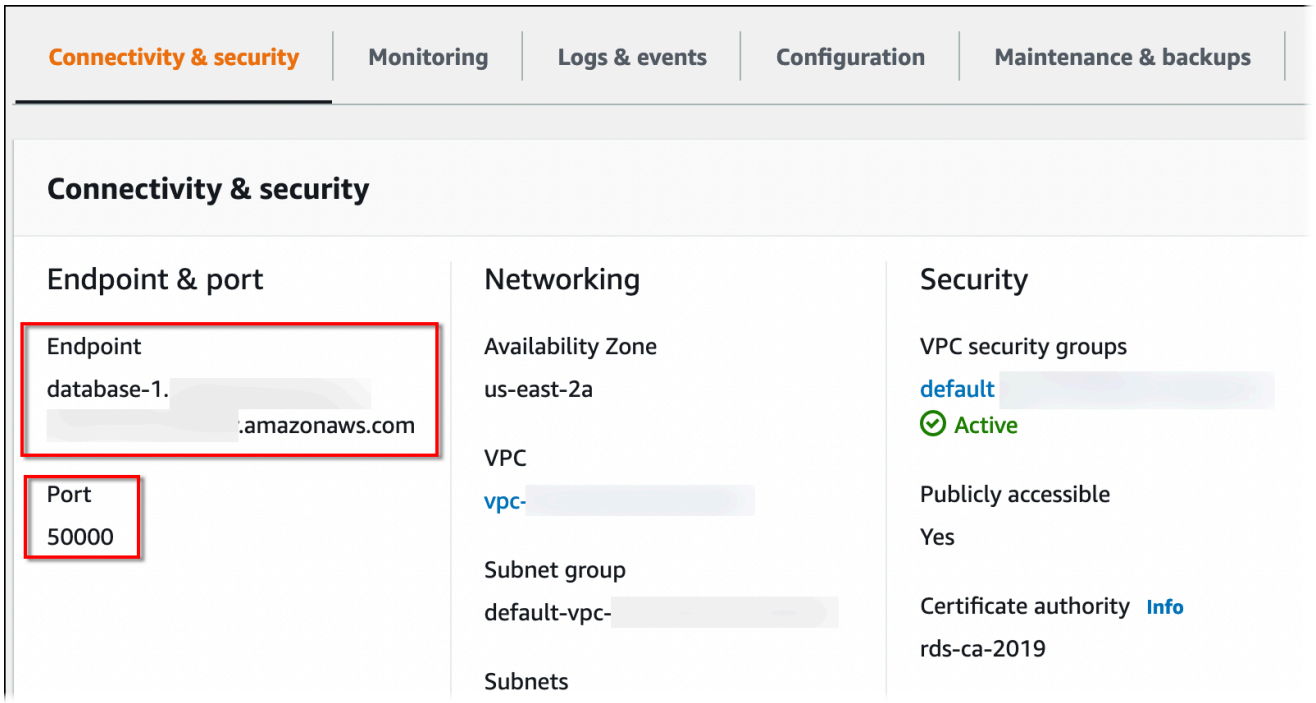
각 Amazon RDS DB 인스턴스에는 엔드포인트가 있으며, 각 엔드포인트에는 DB 인스턴스의 DNS 이름과 포트 번호가 있습니다. SQL 클라이언트 애플리케이션을 사용하여 Amazon RDS for Db2 DB 인스턴스에 연결하려면 DB 인스턴스에 연결할 수 있는 DNS 이름과 포트 번호가 필요합니다.

AWS Management Console 또는 AWS CLI를 사용하여 DB 인스턴스의 엔드포인트를 찾을 수 있습니다.

콘솔

RDS for Db2 DB 인스턴스의 엔드포인트를 찾으려면

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 콘솔의 오른쪽 상단에서 DB 인스턴스의 AWS 리전을 선택합니다.
3. RDS for Db2 DB 인스턴스에 대한 DNS 이름과 포트 번호를 찾습니다.
   a. DB 인스턴스 목록을 표시할 인스턴스를 선택합니다.
   b. 인스턴스 세부 정보를 표시할 RDS for Db2 DB 인스턴스 이름을 선택합니다.
   c. Connectivity & security(연결 및 보안) 탭에서 엔드포인트를 복사합니다. 또한 포트 번호를 적어 둡니다. DB 인스턴스에 연결하려면 엔드포인트와 포트 번호가 모두 필요합니다.



AWS CLI

RDS for Db2 DB 인스턴스의 엔드포인트를 찾으려면 describe-db-instances 명령을 실행합니다. 다음 예제에서 *database-1*을 DB 인스턴스 이름으로 바꿉니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-instances \
```

```
    --db-instance-identifier database-1 \
    --query 'DBInstances[].
{DBInstanceIdentifier:DBInstanceIdentifier,DBName:DBName,Endpoint:Endpoint}' \
    --output json
```

Windows의 경우:

```
aws rds describe-db-instances ^
    --db-instance-identifier database-1 ^
    --query 'DBInstances[].
{DBInstanceIdentifier:DBInstanceIdentifier,DBName:DBName,Endpoint:Endpoint}' ^
    --output json
```

다음 예제와 비슷한 출력이 생성됩니다. DNS 이름은 출력의 `Address` 라인에 포함되어 있습니다.

```
[
    {
        "DBInstanceIdentifier": "database-1",
        "DBName": "DB2DB",
        "Endpoint": {
            "Address": "database-1.123456789012.us-east-2.amazonaws.com",
            "Port": 50000,
            "HostedZoneId": "Z2OC4A7DETW6VH"
        }
    }
]
```

IBM Db2 CLP을 사용하여 Amazon RDS for Db2 DB 인스턴스에 연결

IBM Db2 CLP와 같은 명령줄 유틸리티를 사용하여 Amazon RDS for Db2 DB 인스턴스에 연결할 수 있습니다. 이 유틸리티는 IBM Data Server Runtime Client의 일부입니다. IBM Fix Central에서 클라이언트를 다운로드하려면 IBM Support의 IBM 데이터 서버 클라이언트 패키지 버전 11.5 Mod 8 수정 팩 0을 참조하세요.

주제

- 용어
- 클라이언트 설치
- DB 인스턴스에 연결
- RDS for Db2 DB 인스턴스에 대한 연결 문제 해결

용어

다음 용어는 RDS for Db2 DB 인스턴스에 연결할 때 사용되는 명령을 설명하는 데 도움이 됩니다.

catalog tcpip node

이 명령은 로컬 Db2 클라이언트에 원격 데이터베이스 노드를 등록하여 노드를 클라이언트 애플리케이션에 액세스할 수 있도록 합니다. 노드를 카탈로그화하려면 서버의 호스트 이름, 포트 번호, 통신 프로토콜과 같은 정보를 제공합니다. 그러면 카탈로그화된 노드는 하나 이상의 원격 데이터베이스가 있는 대상 서버를 나타냅니다. 자세한 내용은 IBM Db2 설명서의 CATALOG TCPIP/TCPIP4/TCPIP6 NODE 명령을 참조하세요.

catalog database

이 명령은 로컬 Db2 클라이언트에 원격 데이터베이스를 등록하여 데이터베이스를 클라이언트 애플리케이션에 액세스할 수 있도록 합니다. 데이터베이스를 카탈로그화하려면 데이터베이스의 별칭, 데이터베이스가 있는 노드, 데이터베이스 연결에 필요한 인증 유형 등의 정보를 제공합니다. 자세한 내용은 IBM Db2 설명서의 CATALOG DATABASE 명령을 참조하세요.

클라이언트 설치

downloading the package for Linux 후에 루트 또는 관리자 권한을 사용하여 클라이언트를 설치합니다.

> **Note**
>
> AIX 또는 Windows에 클라이언트를 설치하려면 동일한 절차를 따르되, 운영 체제에 맞게 명령을 수정하세요.

Linux에 클라이언트를 설치하려면

1. **`./db2_install -f sysreq`**를 실행하고 **`yes`**를 선택하여 라이선스를 수락합니다.
2. 클라이언트를 설치할 위치를 선택합니다.
3. **`clientInstallDir/instance/db2icrt -s client`** ***`instance_name`***을 실행합니다. ***`instance_name`***을 Linux에서 올바른 운영 체제 사용자로 바꿉니다. Linux에서 Db2 DB 인스턴스 이름은 운영 체제 사용자 이름에 연결됩니다.

   이 명령은 Linux에서 지정된 사용자의 홈 디렉터리에 **`sqllib`** 디렉터리를 만듭니다.

DB 인스턴스에 연결

RDS for Db2 DB 인스턴스에 연결하려면 인스턴스의 DNS 이름과 포트 번호가 필요합니다. 찾는 방법에 대한 자세한 내용은 엔드포인트 찾기 섹션을 참조하세요. 또한 RDS for Db2 DB 인스턴스를 만들 때 정의한 데이터베이스 이름, 마스터 사용자 이름, 마스터 암호도 알아야 합니다. 찾는 방법에 대한 자세한 내용은 DB 인스턴스 생성 섹션을 참조하세요.

IBM Db2 CLP를 사용하여 RDS for Db2 DB 인스턴스에 연결하려면

1. IBM Db2 CLP 클라이언트 설치 중에 지정한 사용자 이름으로 로그인합니다.
2. RDS for Db2 DB 인스턴스를 카탈로그화합니다. 다음 예제에서 *node_name*, *dns_name*, *port*를 로컬 카탈로그의 노드 이름, DB 인스턴스의 DNS 이름, 포트 번호로 대체합니다.

   ```
   db2 catalog TCPIP node node_name remote dns_name server port
   ```

   예

   ```
   db2 catalog TCPIP node remnode remote database-1.123456789012.us-
   east-1.amazonaws.com server 50000
   ```

3. `rdsadmin` 데이터베이스와 데이터베이스를 카탈로그화합니다. 그러면 Amazon RDS 저장 프로시저로 `rdsadmin` 데이터베이스에 연결하여 일부 관리 작업을 수행할 수 있습니다. 자세한 내용은 RDS for Db2 DB 인스턴스 관리 단원을 참조하십시오.

   다음 예제에서 *database_alias*, *node_name*, *database_name*을 이 데이터베이스의 별칭, 이전 단계에서 정의한 노드 이름, 데이터베이스 이름으로 바꿉니다. `server_encrypt`에서는 네트워크를 통해 사용자 이름과 암호를 암호화합니다.

   ```
   db2 catalog database rdsadmin [ as database_alias ] at node node_name
    authentication server_encrypt

   db2 catalog database database_name [ as database_alias ] at node node_name
    authentication server_encrypt
   ```

   예

   ```
   db2 catalog database rdsadmin at node remnode authentication server_encrypt

   db2 catalog database testdb as rdsdb2 at node remnode authentication server_encrypt
   ```

4. RDS for Db2 데이터베이스에 연결합니다. 다음 예제에서 *rds_database_alias*, *master_username*, *master_password*를 RDS for Db2 DB 인스턴스의 마스터 암호, 마스터 사용자 이름, 데이터베이스 이름으로 바꿉니다.

```
db2 connect to rds_database_alias user master_username using master_password
```

다음 예제와 비슷한 출력이 생성됩니다.

```
Database Connection Information

    Database server        = DB2/LINUXX8664 11.5.9.0
    SQL authorization ID   = ADMIN
    Local database alias   = TESTDB
```

5. 쿼리를 실행하고 결과를 확인합니다. 다음 예제는 생성한 데이터베이스를 선택하는 SQL 문을 보여줍니다.

```
db2 "select current server from sysibm.dual"
```

다음 예제와 비슷한 출력이 생성됩니다.

```
1
    ------------------
    TESTDB

    1 record(s) selected.
```

RDS for Db2 DB 인스턴스에 대한 연결 문제 해결

다음 NULLID 오류는 보통 클라이언트와 RDS for Db2 서버 버전이 일치하지 않아 발생합니다. 지원되는 Db2 클라이언트 버전은 IBM Db2 설명서의 지원되는 클라이언트, 드라이버 및 서버 수준 조합을 참조하세요.

```
db2 "select * from syscat.tables"
SQL0805N Package "NULLID.SQLC2O29 0X4141414141454A69" was not found.
SQLSTATE=51002
```

이 오류가 발생한 후에는 이전 Db2 클라이언트의 패키지를 RDS for Db2에서 지원하는 Db2 서버 버전에 바인딩해야 합니다.

이전 Db2 클라이언트의 패키지를 최신 Db2 서버에 바인딩하려면

1. 클라이언트 머신에서 바인딩할 파일을 찾습니다. 일반적으로 이러한 파일은 Db2 클라이언트 설치 경로의 bnd 디렉터리에 있으며 확장자는 .bnd입니다.
2. Db2 서버에 연결합니다. 다음 예제에서 *database_name*을 Db2 서버 이름으로 바꿉니다. *master_username* 및 *master_password*를 사용자 정보로 대체합니다. 이 사용자에게는 DBADM 권한이 있습니다.

   ```
   db2 connect to database_name user master_username using master_password
   ```

3. 패키지를 바인딩하려면 bind 명령을 실행합니다.

   a. 클라이언트 머신에서 바인딩 파일이 있는 디렉터리로 이동합니다.

   b. 각 파일에 대해 bind 명령을 실행합니다.

      다음 옵션이 필요합니다.

      - `blocking all` - 단일 데이터베이스 요청으로 바인딩할 파일의 모든 패키지를 바인딩합니다.
      - `grant public` - 패키지를 실행할 public 권한을 부여합니다.
      - `sqlerror continue` - 오류가 발생해도 bind 프로세스가 계속되도록 지정합니다.

      bind 명령에 대한 자세한 내용은 IBM Db2 설명서의 BIND 명령을 참조하세요.
4. `syscat.package` 카탈로그 뷰를 쿼리하거나 bind 명령 이후에 반환되는 메시지를 확인하여 바인딩이 성공했는지 확인합니다.

자세한 내용은 IBM Support의 DB2 v11.5 바인딩 파일 및 패키지 이름 목록을 참조하세요.

IBM CLPPlus을 사용하여 Amazon RDS for Db2 DB 인스턴스에 연결

IBM CLPPlus와 같은 유틸리티를 사용하여 Amazon RDS for Db2 DB 인스턴스에 연결할 수 있습니다. 이 유틸리티는 IBM Data Server Runtime Client의 일부입니다. IBM Fix Central에서 클라이언트를 다운로드하려면 IBM Support의 IBM 데이터 서버 클라이언트 패키지 버전 11.5 Mod 8 수정 팩 0을 참조하세요.

> **Important**
>
> 데스크톱과 함께 macOS, Windows, Linux 등 그래픽 사용자 인터페이스를 지원하는 운영 체제에서 IBM CLPPlus를 실행하는 것이 좋습니다. 헤드리스 Linux를 실행하는 경우 CLPPlus 명령과 함께 switch -nw를 사용하세요.

주제

- 클라이언트 설치
- DB 인스턴스에 연결

클라이언트 설치

Linux의 패키지를 다운로드한 후 클라이언트를 설치합니다.

> **Note**
>
> AIX 또는 Windows에 클라이언트를 설치하려면 동일한 절차를 따르되, 운영 체제에 맞게 명령을 수정하세요.

Linux에 클라이언트를 설치하려면

1. **`./db2_install`**를 실행합니다.
2. **`clientInstallDir/instance/db2icrt -s client`** ***`instance_name`***을 실행합니다. ***`instance_name`***을 Linux에서 올바른 운영 체제 사용자로 바꿉니다. Linux에서 Db2 DB 인스턴스 이름은 운영 체제 사용자 이름에 연결됩니다.

   이 명령은 Linux에서 지정된 사용자의 홈 디렉터리에 **`sqllib`** 디렉터리를 만듭니다.

DB 인스턴스에 연결

RDS for Db2 DB 인스턴스에 연결하려면 인스턴스의 DNS 이름과 포트 번호가 필요합니다. 찾는 방법에 대한 자세한 내용은 엔드포인트 찾기 섹션을 참조하세요. 또한 RDS for Db2 DB 인스턴스를 만들 때 정의한 데이터베이스 이름, 마스터 사용자 이름, 마스터 암호도 알아야 합니다. 찾는 방법에 대한 자세한 내용은 DB 인스턴스 생성 섹션을 참조하세요.

IBM CLPPlus를 사용하여 RDS for Db2 DB 인스턴스에 연결하려면

1. 명령 구문을 검토합니다. 다음 예제에서 *clientDir*을 클라이언트가 설치된 위치로 바꿉니다.

```
cd clientDir/bin
    ./clpplus -h
```

2. Db2 서버를 구성합니다. 다음 예제에서 *dsn_name*, *database_name*, *endpoint*, *port*를 RDS for Db2 DB 인스턴스의 DSN 이름, 데이터베이스 이름, 엔드포인트 및 포트로 대체합니다. 자세한 내용은 Amazon RDS for Db2 DB 인스턴스의 엔드포인트 찾기 단원을 참조하십시오.

```
db2cli writecfg add -dsn dsn_name -database database_name -host endpoint -port port
 -parameter "Authentication=SERVER_ENCRYPT"
```

3. RDS for Db2 DB 인스턴스에 연결합니다. 다음 예제에서 *master_username* 및 *dsn_name*을 마스터 사용자 이름과 DSN 이름으로 대체합니다.

```
./clpplus -nw master_username@dsn_name
```

4. Java Shell 창이 열립니다. RDS for Db2 DB 인스턴스의 마스터 암호를 입력합니다.

> **Note**
>
> Java Shell 창이 열리지 않는 경우 **./clpplus -nw**를 실행하여 동일한 명령줄 창을 사용합니다.

```
Enter password: *********
```

연결이 이루어지면 다음 예제와 비슷한 출력이 생성됩니다.

```
Database Connection Information :
---------------------------------
Hostname = database-1.abcdefghij.us-east-1.rds.amazonaws.com
Database server = DB2/LINUXX8664  SQL110590
SQL authorization ID = admin
Local database alias = DB2DB
Port = 50000
```

5. 쿼리를 실행하고 결과를 확인합니다. 다음 예제는 생성한 데이터베이스를 선택하는 SQL 문을 보여줍니다.

```
SQL > select current server from sysibm.dual;
```

다음 예제와 비슷한 출력이 생성됩니다.

```
1
    --------------------
    DB2DB
    SQL>
```

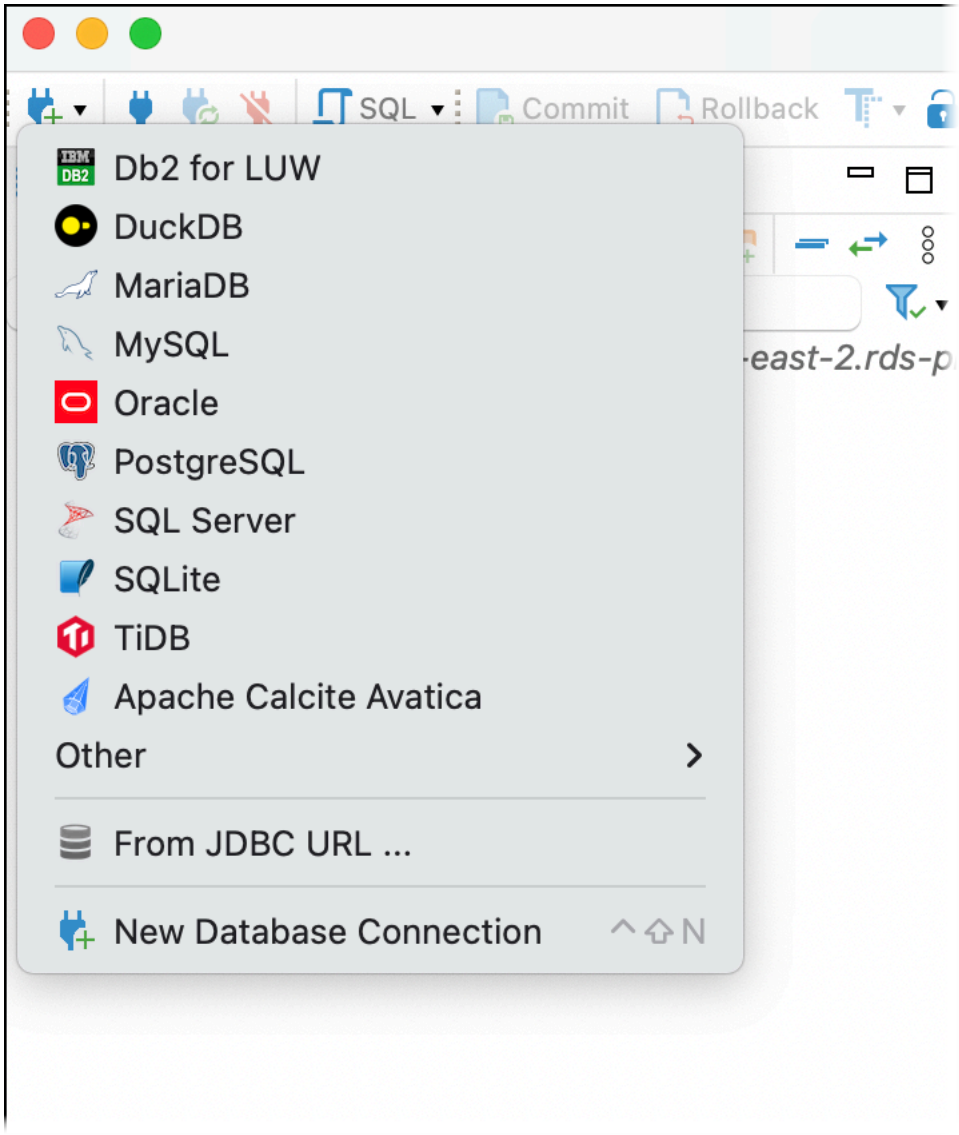
DBeaver을 사용하여 Amazon RDS for Db2 DB 인스턴스에 연결

DBeaver와 같은 타사 도구를 사용하여 Amazon RDS for Db2 DB 인스턴스에 연결할 수 있습니다. 이 유틸리티를 다운로드하려면 DBeaver 커뮤니티를 참조하세요.

RDS for Db2 DB 인스턴스에 연결하려면 인스턴스의 DNS 이름과 포트 번호가 필요합니다. 찾는 방법에 대한 자세한 내용은 엔드포인트 찾기 섹션을 참조하세요. 또한 RDS for Db2 DB 인스턴스를 만들 때 정의한 데이터베이스 이름, 마스터 사용자 이름, 마스터 암호도 알아야 합니다. 찾는 방법에 대한 자세한 내용은 DB 인스턴스 생성 섹션을 참조하세요.

DBeaver를 사용하여 RDS for Db2 DB 인스턴스에 연결하려면

1. Start DBeaver.
2. 도구 모음에서 새 연결 아이콘을 선택한 다음 Db2 for LUW를 선택합니다.



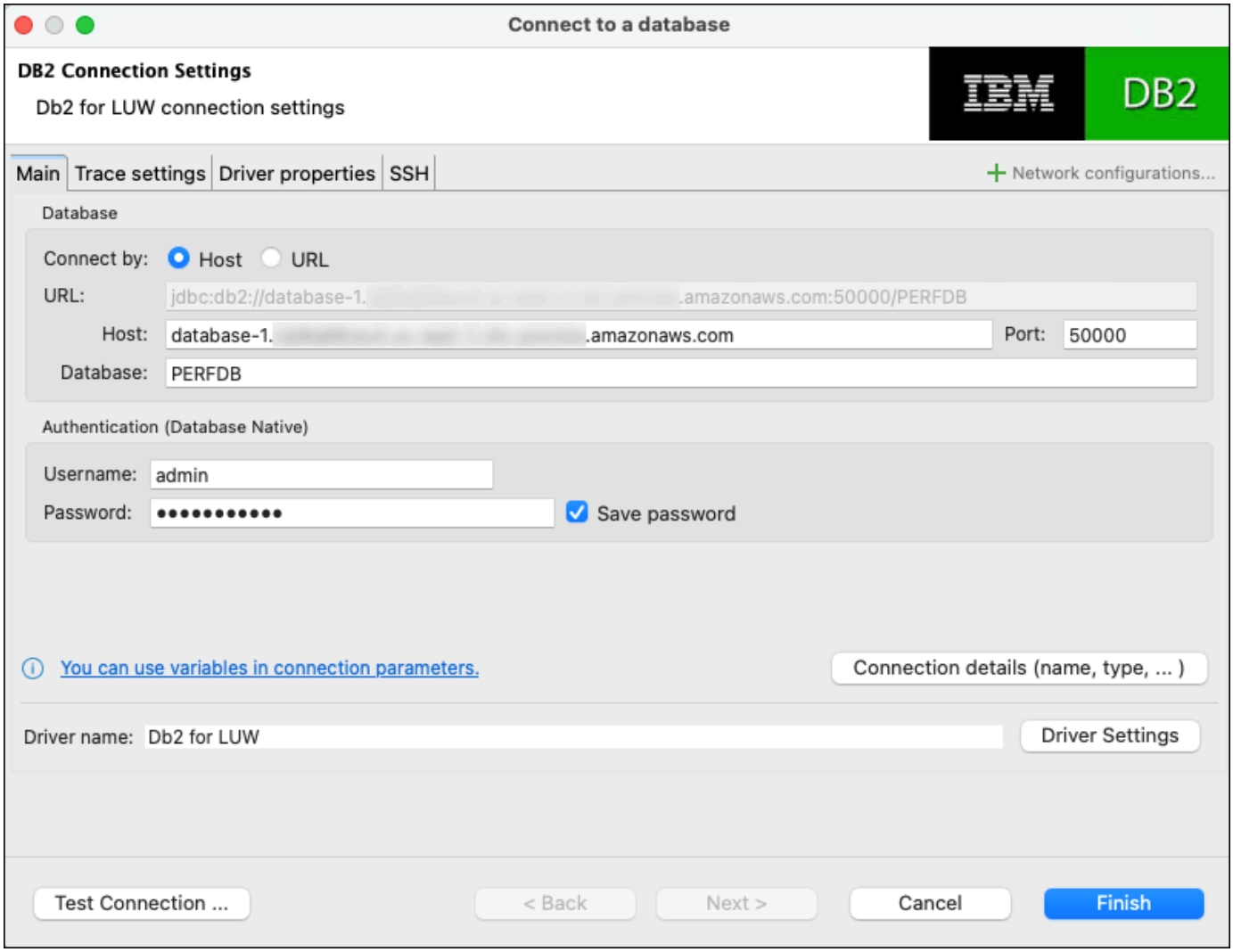
3. 데이터베이스에 연결 창에서 RDS for Db2 DB 인스턴스 정보를 입력합니다.

   a. 다음 정보를 입력합니다.

      - 호스트에 DB 인스턴스의 DNS 이름을 입력합니다.
      - 포트에 DB 인스턴스의 포트 번호를 입력합니다.
      - 데이터베이스에 데이터베이스 이름을 입력합니다.
      - 사용자 이름에 DB 인스턴스의 데이터베이스 관리자 이름을 입력합니다.
      - 암호에 DB 인스턴스의 데이터베이스 관리자 암호를 입력합니다.

   b. 암호 저장을 선택합니다.

   c. 드라이버 설정을 선택합니다.



4. 드라이버 편집 창에서 추가 보안 속성을 지정합니다.

   a. 드라이버 속성 탭을 선택합니다.

   b. 2개의 사용자 속성을 추가합니다.

      i. 컨텍스트 메뉴를 열고(마우스 오른쪽 버튼 클릭) 새 속성 추가를 선택합니다.

      ii. 속성 이름에 encryptionAlgorithm을 추가한 다음 확인을 선택합니다.

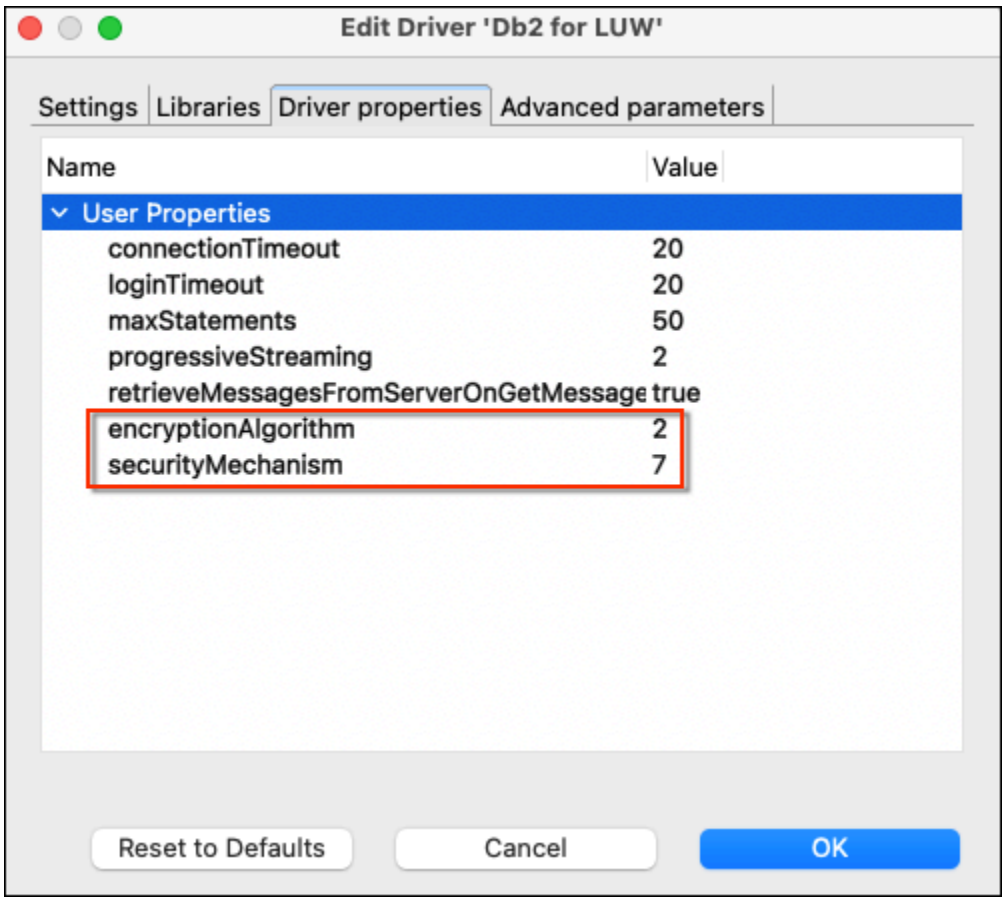
      iii. encryptionAlgorithm 행을 선택한 상태에서 값 열을 선택하고 2를 추가합니다.

      iv. 컨텍스트 메뉴를 열고(마우스 오른쪽 버튼 클릭) 새 속성 추가를 선택합니다.

      v. 속성 이름에 securityMechanism을 추가한 다음 확인을 선택합니다.

      vi. securityMechanism 행을 선택한 상태에서 값 열을 선택하고 7을 추가합니다.

   c. 확인을 선택합니다.

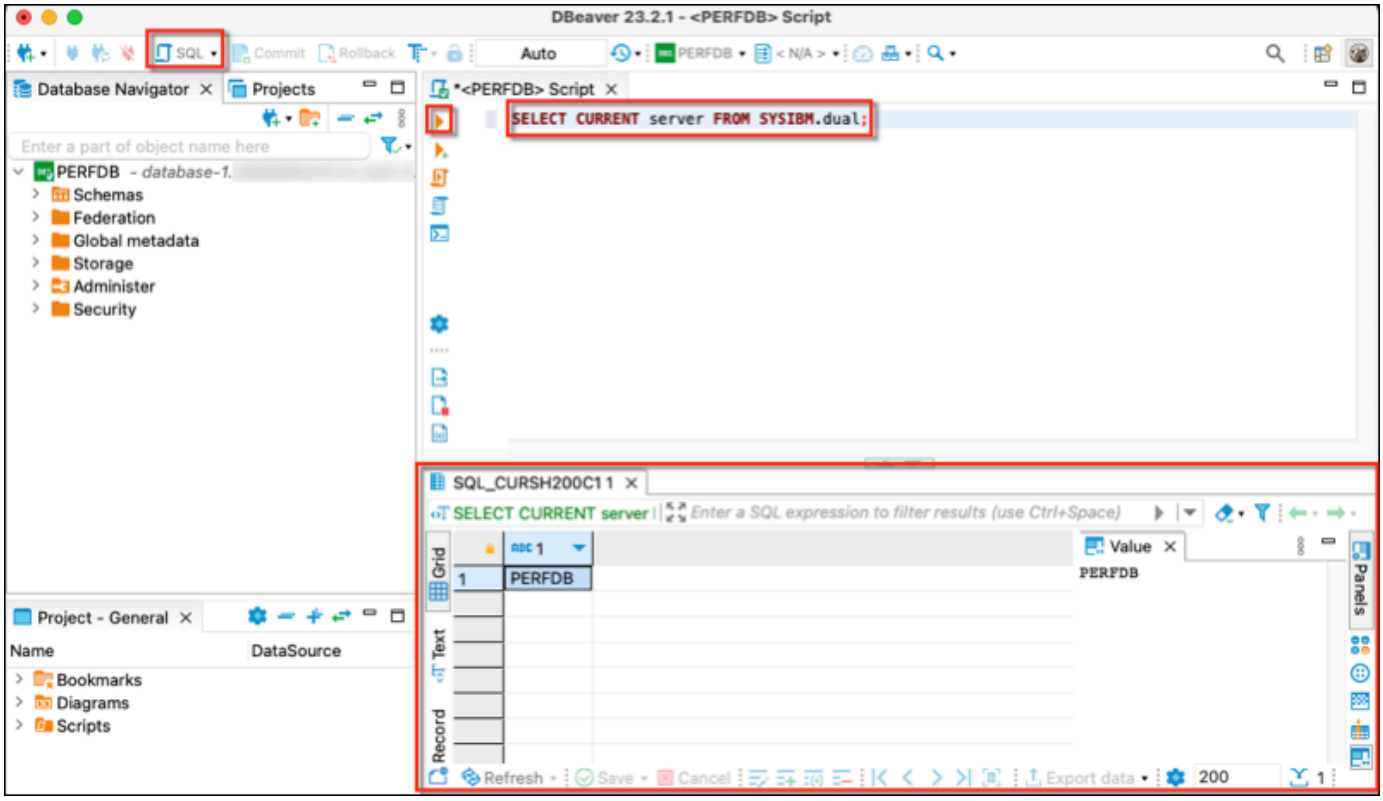


5. 데이터베이스에 연결 창에서 연결 테스트를 선택합니다. 컴퓨터에 DB2 JBDC 드라이버가 설치되어 있지 않으면 드라이버가 자동으로 다운로드됩니다.
6. 확인을 선택합니다.
7. 마침을 클릭합니다.
8. 데이터베이스 탐색 탭에서 데이터베이스 이름을 선택합니다. 이제 객체를 탐색할 수 있습니다.

이제 SQL 명령을 실행할 준비가 되었습니다.

SQL 명령을 실행하고 결과를 확인하려면

1. 상단 메뉴에서 SQL을 선택합니다. 그러면 SQL 스크립트 패널이 열립니다.
2. 스크립트 패널에서 SQL 명령을 입력합니다.
3. 명령을 실행하려면 SQL 쿼리 실행 버튼을 선택합니다.
4. SQL 결과 패널에서 SQL 쿼리 결과를 확인합니다.



IBM Db2 Data Management Console을 사용하여 Amazon RDS for Db2 DB 인스턴스에 연결

IBM Db2 Data Management Console을 사용하여 Amazon RDS for Db2 DB 인스턴스에 연결할 수 있습니다. IBM Db2 Data Management Console에서는 여러 RDS for Db2 DB 인스턴스를 관리하고 모니터링할 수 있습니다.

> **Note**
>
> RDS for Db2 DB 인스턴스와 동일한 VPC 및 보안 그룹에 있는 Amazon EC2 Linux 또는 Windows 시스템이 있어야 합니다. VPC 및 보안 그룹은 내부 네트워크를 통해 DB 인스턴스에 대한 연결을 제어합니다.

IBM Db2 Data Management Console에서는 메타데이터 및 성능 지표를 저장하기 위한 리포지토리 Db2 데이터베이스가 필요하지만, RDS for Db2용 리포지토리 데이터베이스를 자동으로 생성할 수는 없습니다. 먼저 하나 이상의 RDS for Db2 DB 인스턴스를 모니터링하려면 리포지토리 데이터베이스를 만들어야 합니다. 그런 다음 IBM Db2 Data Management Console을 사용하여 IBM Db2 Data Management Console을 설치하고 RDS for Db2 DB 인스턴스에 연결합니다.

주제

- 1단계: DB 인스턴스를 모니터링하기 위한 리포지토리 데이터베이스 생성
- 2단계: IBM Db2 Data Management Console 설치 및 설정
- 3단계: 리포지토리 데이터베이스 구성 및 RDS for Db2 DB 인스턴스에 연결
- IBM Db2 Data Management Console 사용하기

1단계: DB 인스턴스를 모니터링하기 위한 리포지토리 데이터베이스 생성

적절한 크기의 기존 RDS for Db2 DB 인스턴스를 다른 RDS for Db2 DB 인스턴스를 모니터링하기 위한 IBM Db2 Data Management Console용 리포지토리로 사용할 수 있습니다. 하지만 관리자에게는 버퍼 풀과 테이블스페이스를 생성할 SYSCTRL 권한이 없으므로, IBM Db2 Data Management Console 리포지토리 생성을 사용하여 리포지토리 데이터베이스를 생성하는 것은 실패합니다. 대신 리포지토리 데이터베이스를 생성해야 합니다. 이 리포지토리 데이터베이스는 RDS for Db2 DB 인스턴스를 모니터링합니다.

두 방법으로 리포지토리 데이터베이스를 만들 수 있습니다. RDS for Db2 데이터베이스를 생성한 다음 버퍼 풀, 사용자 테이블스페이스 및 시스템 임시 테이블스페이스를 수동으로 생성할 수 있습니다. 또는 IBM Db2 Data Management Console 리포지토리 데이터베이스를 호스팅할 별도의 Amazon EC2 인스턴스를 생성할 수 있습니다.

주제

- 버퍼 풀, 사용자 테이블스페이스 및 시스템 임시 테이블스페이스 수동 생성
- IBM Db2 Data Management Console 리포지토리를 호스팅하기 위한 Amazon EC2 인스턴스 생성

버퍼 풀, 사용자 테이블스페이스 및 시스템 임시 테이블스페이스 수동 생성

버퍼 풀, 사용자 테이블스페이스 및 시스템 임시 테이블스페이스를 생성하려면

1. rdsadmin 데이터베이스에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 자체 정보로 대체합니다.

   ```
   db2 connect to rdsadmin user master_username using master_password
   ```

2. IBM Db2 Data Management Console의 버퍼 풀을 생성합니다. 다음 예제에서 *database_name*을 IBM Db2 Data Management Console에서 RDS for Db2 DB 인스턴스를 모니터링하기 위해 만든 리포지토리 이름으로 바꿉니다.

```
db2 "call rdsadmin.create_bufferpool('database_name',
     'BP4CONSOLE', 1000, 'Y', 'Y', 32768)"
```

3. IBM Db2 Data Management Console의 테이블스페이스 생성 다음 예제에서 *database_name*을 IBM Db2 Data Management Console에서 RDS for Db2 DB 인스턴스를 모니터링하기 위해 만든 리포지토리 이름으로 바꿉니다.

```
db2 "call rdsadmin.create_tablespace('database_name',
     'TS4CONSOLE', 'BP4CONSOLE', 32768)"
```

4. IBM Db2 Data Management Console의 임시 테이블스페이스를 생성합니다. 다음 예제에서 *database_name*을 IBM Db2 Data Management Console에서 RDS for Db2 DB 인스턴스를 모니터링하기 위해 만든 리포지토리 이름으로 바꿉니다.

```
db2 "call rdsadmin.create_tablespace('database_name',
    'TS4CONSOLE_TEMP', 'BP4CONSOLE', 32768, 0, 0, 'S')"
```

이제 IBM Db2 Data Management Console을 설치할 준비가 되었습니다. 설치 및 설정에 대한 자세한 내용은 2단계: IBM Db2 Data Management Console 설치 및 설정 섹션을 참조하세요.

IBM Db2 Data Management Console 리포지토리를 호스팅하기 위한 Amazon EC2 인스턴스 생성

별도의 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 생성하여 IBM Db2 Data Management Console 리포지토리를 호스팅할 수 있습니다. Amazon EC2 인스턴스 생성에 대한 자세한 내용은 Amazon EC2 사용 설명서에서 자습서: Amazon EC2 Linux 인스턴스 시작하기를 참조하세요.

2단계: IBM Db2 Data Management Console 설치 및 설정

버퍼 풀, 사용자 테이블스페이스 및 시스템 임시 테이블스페이스를 생성했으면 IBM Db2 Data Management Console을 설치하고 설정할 준비가 되었습니다.

> **Important**
>
> RDS for Db2 DB 인스턴스와 동일한 VPC 및 보안 그룹에 있는 Amazon EC2 Linux 또는 Windows 시스템이 있어야 합니다. VPC 및 보안 그룹은 내부 네트워크를 통해 DB 인스턴스에 대한 연결을 제어합니다. 또한 IBM Db2 Data Management Console에 대한 리포지토리 데이터베이스를 이미 생성했어야 합니다.

IBM Db2 Data Management Console을 설치 및 설정하려면

1. IBM 지원 웹 사이트의 IBM Db2 Data Management Console 버전 3.1x 릴리스에서 IBM Db2 Data Management Console을 다운로드합니다.
2. IBM Db2 Data Management Console을 설치합니다.
3. IBM Db2 Data Management Console을 열고 Amazon EC2 시스템의 IP 주소와 Amazon EC2 인스턴스에 대한 HTTP 또는 HTTPS 연결에 사용하는 포트 번호를 사용합니다. 예는 `http://xx.xx.xx.xx:11080` 또는 `https://xx.xx.xx.xx.11081` 섹션을 참조하세요. *xx.xx.xx.xx*를 Amazon EC2 시스템의 IP 주소로 바꿉니다. `11080` 및 `11081`은 HTTP 및 HTTPS 연결을 위한 기본 포트입니다.
4. (선택 사항) Amazon EC2 인스턴스에서 포트 80 또는 443을 사용하려면 Apache httpd 또는 Nginx HTTP 서버를 사용하여 IBM Db2 Data Management Console 포트를 포트 80 또는 443으로 프록시할 수 있습니다. 자세한 내용은 Apache HTTP Server Project 및 nginx 웹 사이트를 참조하세요.

   IBM Db2 Data Management Console에 대한 연결을 허용하려면 보안 그룹의 인바운드 규칙을 편집해야 합니다. 프록시를 사용하는 경우 TCP/IP 포트 80 또는 443을 변경하여 IBM Db2 Data Management Console 포트로 리디렉션합니다. 프록시를 사용하지 않는 경우 TCP/IP 포트 80 또는 443을 기본 포트 11080(HTTP) 또는 11081(HTTPS)로 변경합니다.

이제 IBM Db2 Data Management Console에 로그인하여 리포지토리 데이터베이스를 구성하고 RDS for Db2 DB 인스턴스에 연결할 준비가 되었습니다. 자세한 내용은 리포지토리 데이터베이스 구성 및 DB 인스턴스에 연결 단원을 참조하십시오.

3단계: 리포지토리 데이터베이스 구성 및 RDS for Db2 DB 인스턴스에 연결

리포지토리 데이터베이스에 처음 연결하면 IBM Db2 Data Management Console이 리포지토리를 자동으로 구성합니다. 리포지토리 데이터베이스를 구성한 후 IBM Db2 Data Management Console에 데이터베이스 연결을 추가할 수 있습니다.

RDS for Db2 DB 인스턴스에 연결하려면 인스턴스의 DNS 이름과 포트 번호가 필요합니다. 찾는 방법에 대한 자세한 내용은 엔드포인트 찾기 섹션을 참조하세요. 또한 RDS for Db2 DB 인스턴스를 만들 때 정의한 데이터베이스 이름, 마스터 사용자 이름, 마스터 암호도 알아야 합니다. 찾는 방법에 대한 자세한 내용은 DB 인스턴스 생성 섹션을 참조하세요. 인터넷을 통해 연결하는 경우 데이터베이스 포트로의 트래픽을 허용합니다. 자세한 내용은 DB 인스턴스 생성 단원을 참조하십시오.

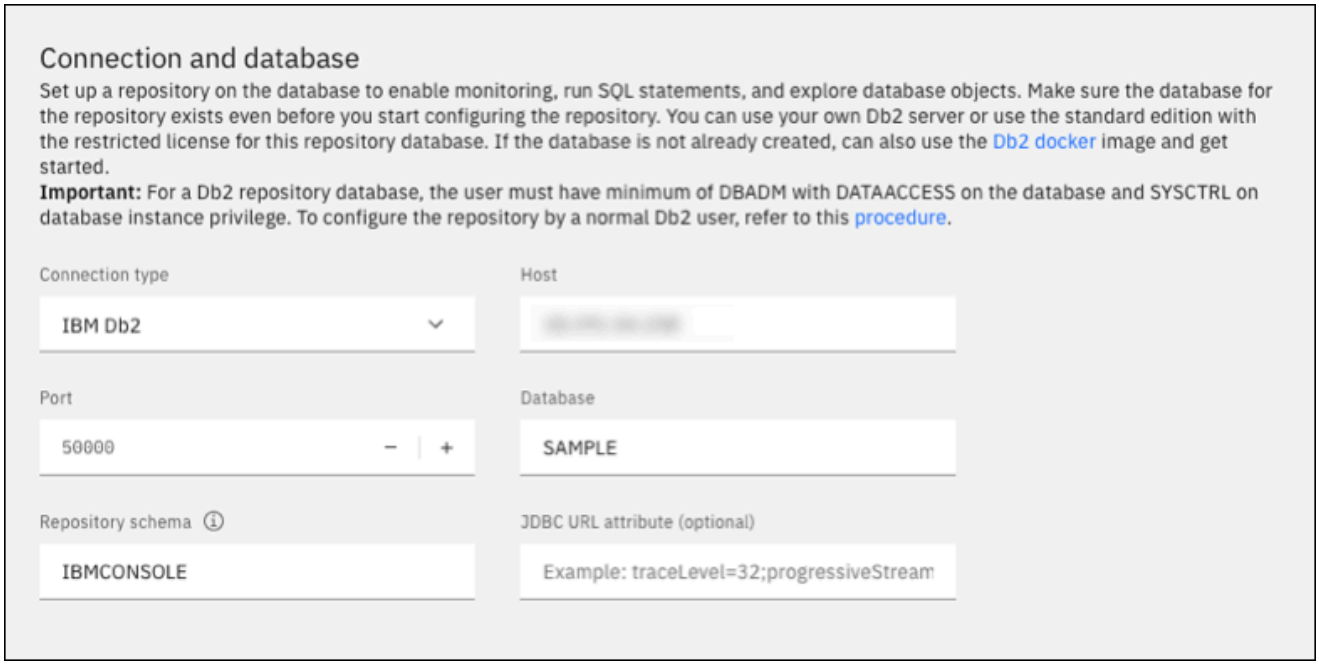
IBM Db2 Data Management Console을 사용하여 RDS for Db2 DB 인스턴스에 연결하려면

1. 설치 중에 설정한 자격 증명으로 IBM Db2 Data Management Console에 로그인합니다.

2. 리포지토리를 구성합니다.

   a. 연결 및 데이터베이스 섹션에서 RDS for Db2 DB 인스턴스에 대한 다음 정보를 입력합니다.

   - 호스트에 DB 인스턴스의 DNS 이름을 입력합니다.
   - 포트에 DB 인스턴스의 포트 번호를 입력합니다.
   - 데이터베이스에 데이터베이스 이름을 입력합니다.



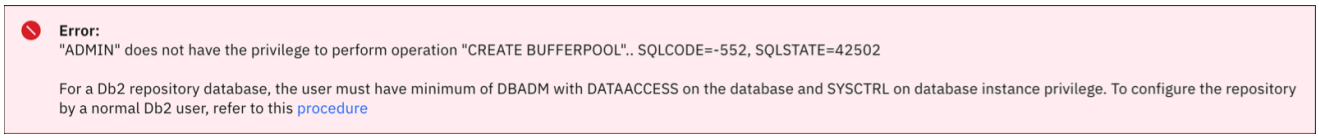
   b. 보안 및 보안 인증 정보 섹션에서 RDS for Db2 DB 인스턴스에 대한 다음 정보를 입력합니다.

   - 보안 유형에서 암호화된 사용자 및 암호를 선택합니다.
   - 사용자 이름에 DB 인스턴스의 데이터베이스 관리자 이름을 입력합니다.
   - 암호에 DB 인스턴스의 데이터베이스 관리자 암호를 입력합니다.

   c. 연결 테스트를 선택합니다.

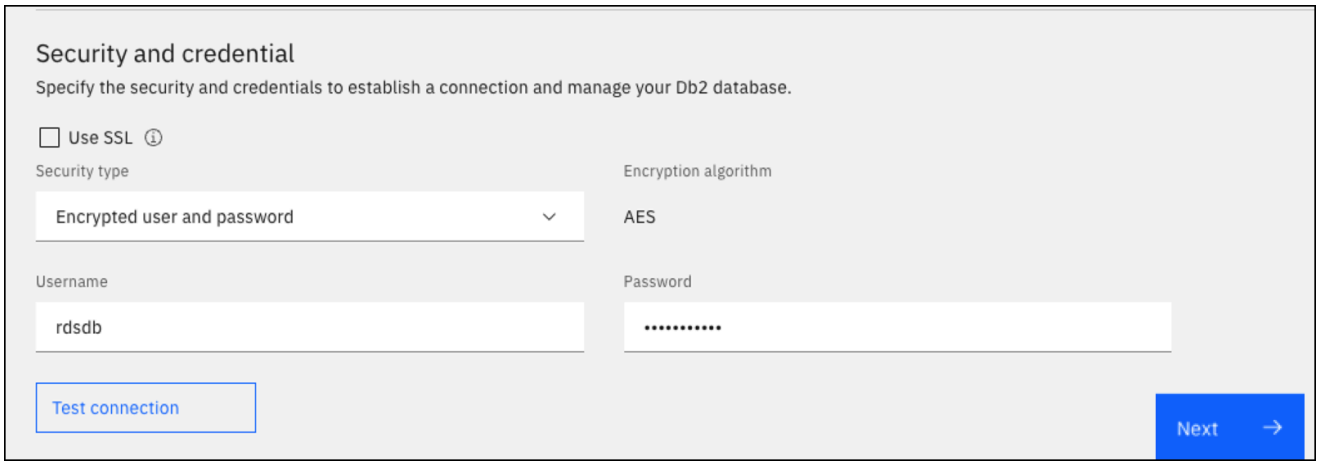
> **Note**
>
> 연결에 실패할 경우 보안 그룹의 인바운드 규칙을 통해 데이터베이스 포트가 열려 있는지 확인하세요. 자세한 내용은 Amazon RDS for Db2를 사용하는 보안 그룹에 대한 고려 사항 단원을 참조하십시오.

RDS for Db2에서 버퍼 풀, 사용자 테이블스페이스 및 시스템 임시 테이블스페이스를 수동으로 생성하지 않은 경우 다음 오류 메시지가 표시될 수 있습니다.



RDS for Db2 DB 인스턴스를 모니터링하려면 IBM Db2 Data Management Console 리포지토리의 버퍼 표, 테이블스페이스 및 객체를 생성했는지 확인합니다. 또는 Amazon EC2 Db2 DB 인스턴스를 사용하여 RDS for Db2 DB 인스턴스를 모니터링하기 위한 IBM Db2 Data Management Console 리포지토리를 호스팅할 수 있습니다. 자세한 내용은 1단계: DB 인스턴스를 모니터링하기 위한 리포지토리 데이터베이스 생성 단원을 참조하십시오.

d. 연결을 성공적으로 테스트한 후 다음을 선택합니다.

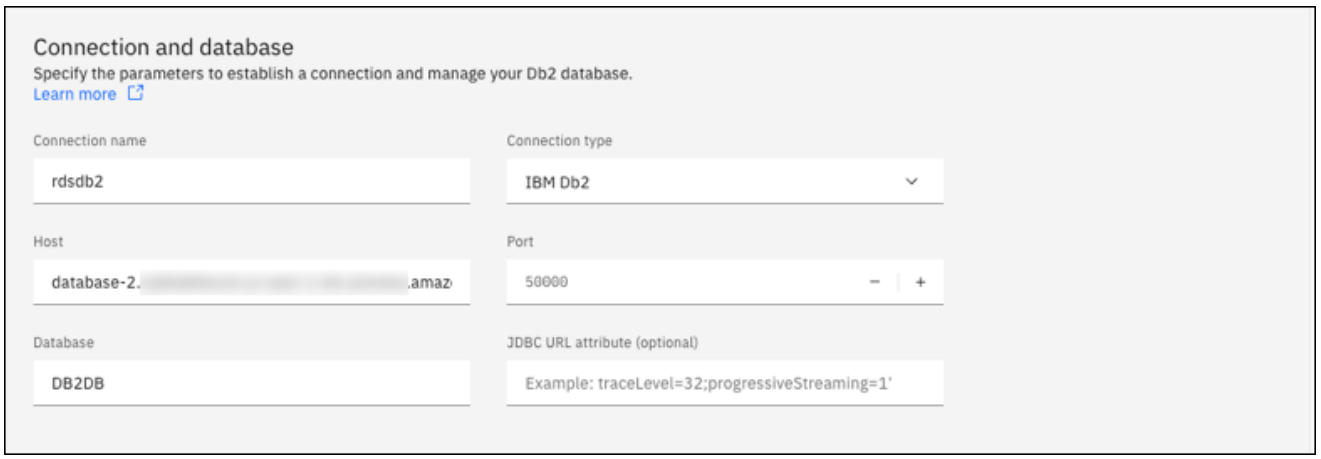


IBM Db2 Data Management Console이 RDS for Db2 DB 인스턴스에서 버퍼 풀, 사용자 테이블스페이스 및 시스템 임시 테이블스페이스를 찾은 경우 IBM Db2 Data Management Console은 리포지토리 데이터베이스를 자동으로 구성합니다. Amazon EC2 인스턴스에서 Db2 인스턴스를 리포지토리 데이터베이스로 사용하는 경우 IBM Db2 Data Management Console은 버퍼 풀 및 기타 객체를 자동으로 생성합니다.

3. 통계 이벤트 모니터 옵트인 설정 창에서 다음을 선택합니다.
4. (선택 사항) 새 연결을 추가합니다. 관리 및 모니터링을 위해 다른 RDS for Db2 DB 인스턴스를 사용하려면 리포지토리가 아닌 RDS for Db2 DB 인스턴스에 연결을 추가합니다.

   a. 연결 및 데이터베이스 섹션에서 관리 및 모니터링에 사용할 RDS for Db2 DB 인스턴스에 대한 다음 정보를 입력합니다.

- 연결 이름에 Db2 데이터베이스 식별자를 입력합니다.
- 호스트에 DB 인스턴스의 DNS 이름을 입력합니다.
- 포트에 DB 인스턴스의 포트 번호를 입력합니다.
- 데이터베이스에 데이터베이스 이름을 입력합니다.



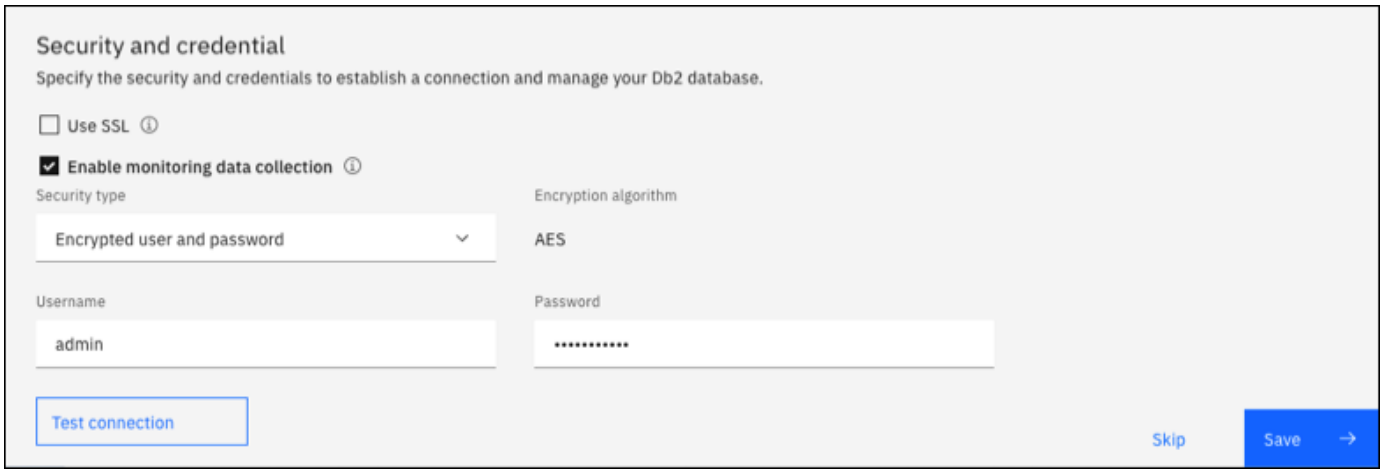
b. 보안 및 보안 인증 정보 섹션에서 모니터링 데이터 수집 활성화를 선택합니다.

c. RDS for Db2 DB 인스턴스에 대한 다음 정보를 입력합니다.

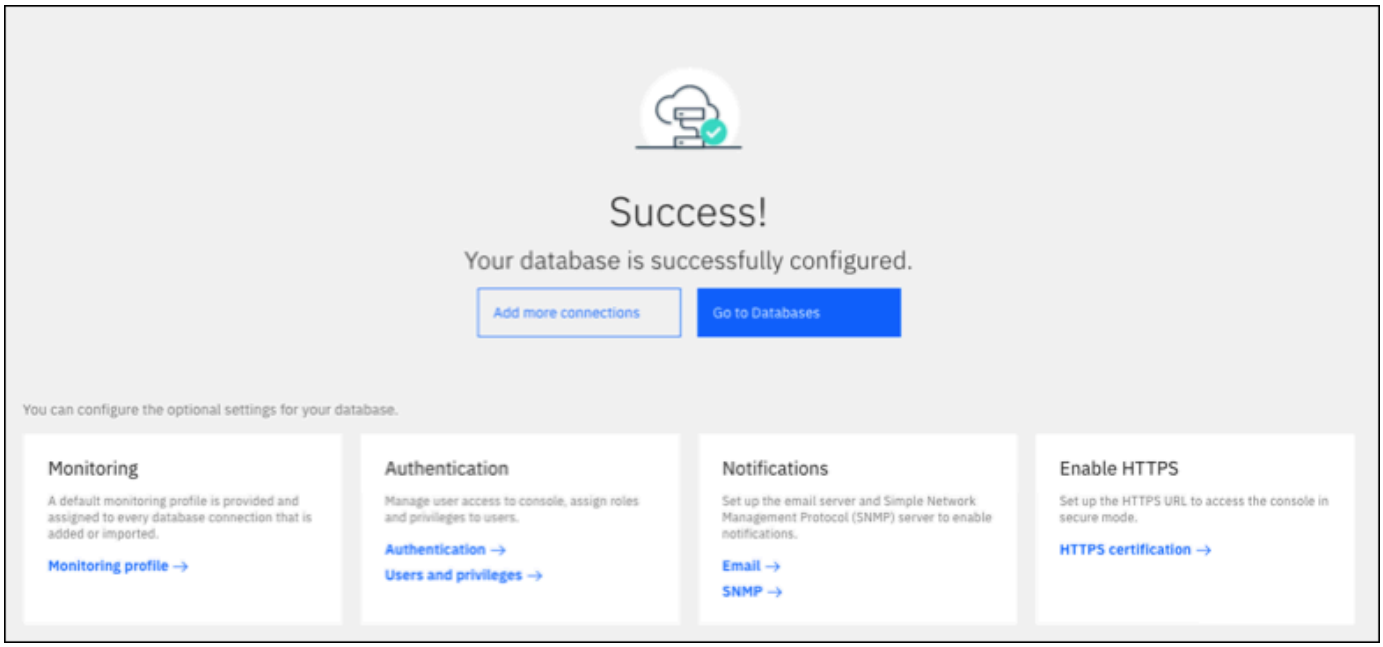
- 사용자 이름에 DB 인스턴스의 데이터베이스 관리자 이름을 입력합니다.
- 암호에 DB 인스턴스의 데이터베이스 관리자 암호를 입력합니다.

d. 연결 테스트를 선택합니다.

e. 연결을 성공적으로 테스트한 후 저장을 선택합니다.



연결이 추가된 후 다음과 유사한 창이 표시됩니다. 이 창은 데이터베이스가 성공적으로 구성되었음을 나타냅니다.



5. 데이터베이스로 이동을 선택합니다. 다음과 유사한 데이터베이스 창이 표시됩니다. 이 창은 지표, 상태 및 연결을 보여주는 대시보드입니다.



이제 IBM Db2 Data Management Console을 사용할 수 있습니다.

IBM Db2 Data Management Console 사용하기

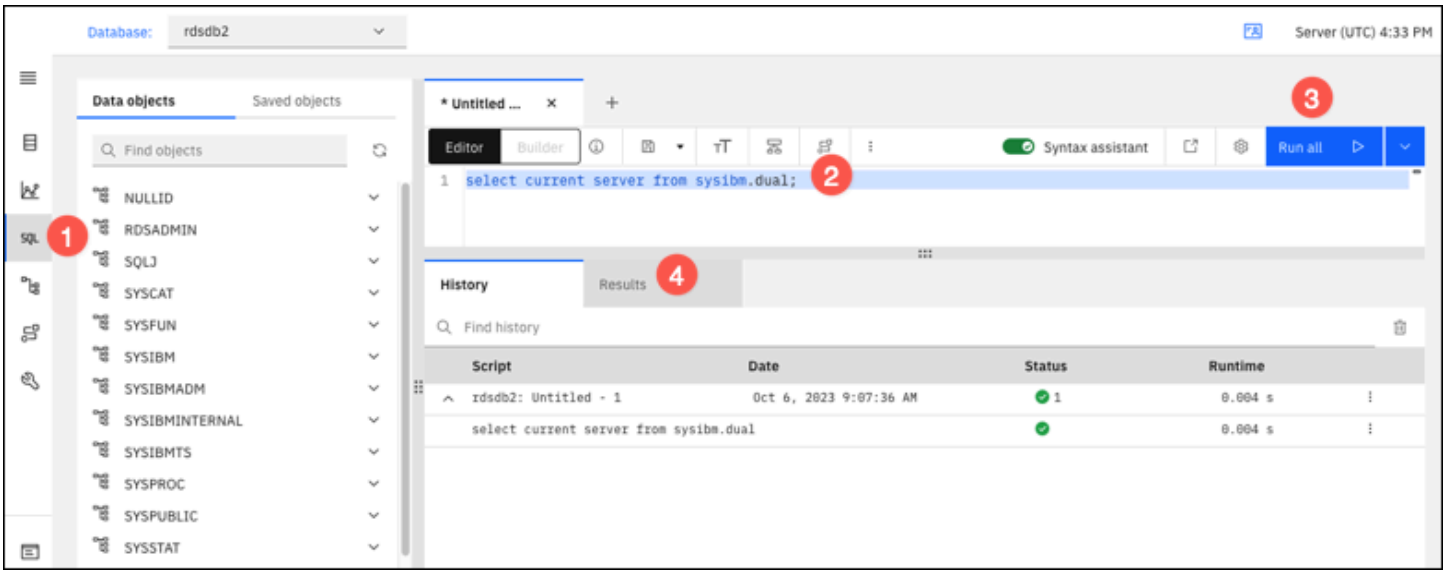
IBM Db2 Data Management Console을 사용하여 다음 유형의 작업을 수행할 수 있습니다.

- 여러 RDS for Db2 DB 인스턴스를 관리합니다.
- SQL 명령을 실행합니다.

- 데이터 및 데이터베이스 객체를 탐색, 생성 또는 변경합니다.
- SQL에서 `EXPLAIN PLAN` 문을 생성합니다.
- 쿼리를 조정합니다.

SQL 명령을 실행하고 결과를 확인하려면

1. 왼쪽 탐색 모음에서 SQL을 선택합니다.
2. SQL 명령을 입력합니다.
3. 모두 실행을 선택합니다.
4. 결과 탭을 선택하여 결과를 봅니다.



Amazon RDS for Db2를 사용하는 보안 그룹에 대한 고려 사항

Amazon RDS for Db2 DB 인스턴스에 연결하려면 필요한 IP 주소와 네트워크 구성이 포함되어 있는 보안 그룹과 연결되어야 합니다. RDS for Db2 DB 인스턴스는 기본 보안 그룹을 사용할 수 있습니다. RDS for Db2 DB 인스턴스를 생성할 때 구성할 필요가 없는 기본 보안 그룹을 할당한 경우에는 방화벽이 인터넷 연결을 차단합니다. 새 보안 그룹 생성에 대한 자세한 내용은 보안 그룹을 통한 액세스 제어 단원을 참조하십시오.

새 보안 그룹을 생성하였으면 보안 그룹과 연동되도록 DB 인스턴스 설정을 변경합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

SSL을 사용하여 DB 인스턴스 연결을 암호화함으로써 보안을 강화할 수 있습니다. 자세한 내용은 Amazon RDS for Db2 DB 인스턴스에 SSL/TLS 사용 단원을 참조하십시오.

Amazon RDS for Db2 DB 인스턴스 연결 보안

Amazon RDS for Db2는 RDS for Db2 DB 인스턴스의 보안을 개선하는 방법을 지원합니다.

주제

- Amazon RDS for Db2 DB 인스턴스에 SSL/TLS 사용
- Amazon RDS for Db2에 대한 Kerberos 인증 사용

Amazon RDS for Db2 DB 인스턴스에 SSL/TLS 사용

SSL은 클라이언트와 서버 간의 네트워크 연결을 보호하는 데 사용되는 업계 표준 프로토콜입니다. SSL 버전 3.0 이후에 이름이 TLS로 변경되었지만, 여전히 이 프로토콜을 SSL로 지칭하는 경우가 많습니다. Amazon RDS는 Amazon RDS for Db2 DB 인스턴스에 SSL 암호화를 지원합니다. SSL/TLS를 사용하여 애플리케이션 클라이언트와 RDS for Db2 DB 인스턴스 간의 연결을 암호화할 수 있습니다. SSL/TLS 지원 기능은 RDS for Db2에 대한 모든 AWS 리전에서 사용할 수 있습니다.

RDS for Db2 DB 인스턴스에 대해 SSL/TLS 암호화를 활성화하려면 DB 인스턴스와 연결된 파라미터 그룹에 Db2 SSL 옵션을 추가합니다. Amazon RDS는 Db2에서 요구하는 대로 SSL/TLS 연결을 위해 두 번째 포트를 사용합니다. 이로 인해 DB 인스턴스와 Db2 클라이언트 간에 클리어 텍스트 통신과 SSL로 암호화된 통신이 동시에 발생할 수 있습니다. 예를 들어 이 포트를 클리어 텍스트 통신에 사용하여 VPC 내의 다른 리소스와 통신하면서 동일한 포트를 SSL로 암호화된 통신에 사용하여 VPC 외부의 리소스와 통신할 수 있습니다.

주제

- SSL/TLS 연결 생성하기
- Db2 데이터베이스 서버에 연결

SSL/TLS 연결 생성하기

SSL/TLS 연결을 만들려면 인증 기관(CA)을 선택하고 모든 AWS 리전에 대한 인증서 번들을 다운로드한 다음 사용자 지정 파라미터 그룹에 파라미터를 추가합니다.

1단계: CA 선택 및 인증서 다운로드

인증 기관(CA)을 선택하고 모든 AWS 리전에 대한 인증서 번들을 다운로드합니다. 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 단원을 참조하십시오.

2단계: 사용자 지정 파라미터 그룹의 파라미터 업데이트

> **Important**
>
> RDS for Db2용 기존 보유 라이선스 사용(BYOL) 모델을 사용하는 경우 IBM Customer ID 및 IBM Site ID용으로 만든 사용자 지정 파라미터 그룹을 수정합니다. RDS for Db2에 대해 다른 라이선스 모델을 사용하는 경우 절차에 따라 사용자 지정 파라미터 그룹에 파라미터를 추가합니다. 자세한 내용은 Amazon RDS for Db2 라이선스 옵션 단원을 참조하십시오.

RDS for Db2 DB 인스턴스의 기본 파라미터 그룹은 수정할 수 없습니다. 따라서 사용자 지정 파라미터 그룹을 생성하여 수정한 후 RDS for Db2 DB 인스턴스에 연결해야 합니다. 파라미터 그룹에 대한 자세한 내용은 Amazon RDS DB 인스턴스용 DB 파라미터 그룹 단원을 참조하세요.

다음 표에 있는 파라미터 설정을 사용합니다.

파라미터	값
DB2COMM	TCPIP,SSL 또는 SSL
SSL_SVCENAME	<any port number except the number used for the non-SSL port>

사용자 지정 파라미터 그룹의 파라미터를 업데이트하려면

1. create-db-parameter-group 명령을 실행하여 사용자 지정 파라미터 그룹을 생성합니다.

   다음 필수 옵션을 포함합니다.

   - `--db-parameter-group-name` – 생성하려는 파라미터 그룹의 이름입니다.
   - `--db-parameter-group-family` – Db2 엔진 에디션 및 메이저 버전입니다. 유효한 값: `db2-se-11-5`, `db2-ae-11.5`.
   - `--description` – 이 파라미터 그룹에 대한 설명입니다.

   DB 파라미터 그룹을 생성하는 것에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹 생성 단원을 참조하십시오.

2. modify-db-parameter-group 명령을 실행하여 만든 사용자 지정 파라미터 그룹의 파라미터를 수정합니다.

   다음 필수 옵션을 포함합니다.

   - `--db-parameter-group-name` – 생성한 파라미터 그룹의 이름입니다.
   - `--parameters` – 파라미터 업데이트를 위한 파라미터 이름, 값, 응용 방법으로 구성된 배열입니다.

   파라미터 그룹 수정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요.
3. 파라미터 그룹을 RDS for Db2 DB 인스턴스에 연결합니다. 자세한 내용은 Amazon RDS의 DB 인스턴스에 DB 파라미터 그룹 연결 단원을 참조하십시오.

Db2 데이터베이스 서버에 연결

Db2 데이터베이스 서버 연결 지침은 언어별로 다릅니다.

Java

Java를 사용하여 Db2 데이터베이스 서버에 연결하려면

1. JDBC 드라이버를 다운로드합니다. 자세한 내용은 IBM 지원 설명서의 DB2 JDBC 드라이버 버전 및 다운로드를 참조하세요.
2. 다음 콘텐츠로 쉘 스크립트 파일을 만듭니다. 이 스크립트는 번들의 모든 인증서를 Java KeyStore에 추가합니다.

   > **Important**
   >
   > 스크립트의 경로에 `keytool`이 존재하는지 확인하여 스크립트가 찾을 수 있도록 하세요. Db2 클라이언트를 사용하는 경우 `~sqlib/java/jdk64/jre/bin` 아래에서 `keytool`을 찾을 수 있습니다.

```
#!/bin/bash
PEM_FILE=$1
PASSWORD=$2
KEYSTORE=$3
```

```
# number of certs in the PEM file
CERTS=$(grep 'END CERTIFICATE' $PEM_FILE| wc -l)
for N in $(seq 0 $(($CERTS - 1))); do
    ALIAS="${PEM_FILE%.*}-$N"
    cat $PEM_FILE |
    awk "n==$N { print }; /END CERTIFICATE/ { n++ }" |
    keytool -noprompt -import -trustcacerts -alias $ALIAS -keystore $KEYSTORE -
storepass $PASSWORD
done
```

3. 쉘 스크립트를 실행하고 인증서 번들이 포함된 PEM 파일을 Java KeyStore로 가져오려면 다음 명령을 실행합니다. *shell_file_name.sh*를 쉘 스크립트 파일의 이름으로, *##*를 Java KeyStore의 암호로 변경합니다.

```
./shell_file_name.sh global-bundle.pem password truststore.jks
```

4. Db2 서버에 연결하려면 다음 명령을 실행합니다. 예제의 다음 자리 표시자를 RDS for Db2 DB 인스턴스 정보로 변경합니다.

   - *ip_address* – DB 인스턴스 엔드포인트의 IP 주소입니다.
   - *port* – SSL 연결의 포트 번호입니다. SSL이 아닌 포트에 사용되는 번호를 제외한 모든 포트 번호가 될 수 있습니다.
   - *database_name* – DB 인스턴스에 있는 데이터베이스의 이름입니다.
   - *master_username* – DB 인스턴스의 마스터 사용자 이름입니다.
   - *master_password* – DB 인스턴스의 마스터 암호입니다.

```
export trustStorePassword=MyPassword
java -cp ~/dsdriver/jdbc_sqlj_driver/linuxamd64/db2jcc4.jar \
com.ibm.db2.jcc.DB2Jcc -url \
"jdbc:db2://ip_address:port/database_name:\
sslConnection=true;sslTrustStoreLocation=\
~/truststore.jks;\
sslTrustStorePassword=${trustStorePassword};\
sslVersion=TLSv1.2;\
encryptionAlgorithm=2;\
securityMechanism=7;" \
-user master_username -password master_password
```

Node.js

Node.js를 사용하여 Db2 데이터베이스 서버에 연결하려면

1. node-ibm_db 드라이버를 설치합니다. 자세한 내용은 IBM Db2 설명서의 Linux 및 UNIX 시스템에 node-ibm_db 드라이버 설치를 참조하세요.
2. 다음 콘텐츠를 바탕으로 JavaScript 파일을 생성합니다. 예제의 다음 자리 표시자를 RDS for Db2 DB 인스턴스 정보로 변경합니다.

   - *ip_address* – DB 인스턴스 엔드포인트의 IP 주소입니다.
   - *master_username* – DB 인스턴스의 마스터 사용자 이름입니다.
   - *master_password* – DB 인스턴스의 마스터 암호입니다.
   - *database_name* – DB 인스턴스에 있는 데이터베이스의 이름입니다.
   - *port* – SSL 연결의 포트 번호입니다. SSL이 아닌 포트에 사용되는 번호를 제외한 모든 포트 번호가 될 수 있습니다.

```
var ibmdb = require("ibm_db");
const hostname = "ip_address";
const username = "master_username";
const password = "master_password";
const database = "database_name";
const port = "port";
const certPath = "/root/qa-bundle.pem";
ibmdb.open("DRIVER={DB2};DATABASE=" + database + ";HOSTNAME=" +
 hostname + ";UID=" + username + ";PWD=" + password + ";PORT=" + port +
 ";PROTOCOL=TCPIP;SECURITY=SSL;SSLServerCertificate=" + certPath + ";", function
 (err, conn){
 if (err) return console.log(err);
 conn.close(function () {
 console.log('done');
 });
});
```

3. JavaScript 파일을 실행하려면 다음 명령을 실행합니다.

```
node ssl-test.js
```

Python

Python를 사용하여 Db2 데이터베이스 서버에 연결하려면

1. 다음 콘텐츠가 포함된 Python 파일을 생성합니다. 예제의 다음 자리 표시자를 RDS for Db2 DB 인스턴스 정보로 변경합니다.

   - *port* – SSL 연결의 포트 번호입니다. SSL이 아닌 포트에 사용되는 번호를 제외한 모든 포트 번호가 될 수 있습니다.
   - *master_username* – DB 인스턴스의 마스터 사용자 이름입니다.
   - *master_password* – DB 인스턴스의 마스터 암호입니다.
   - *database_name* – DB 인스턴스에 있는 데이터베이스의 이름입니다.
   - *ip_address* – DB 인스턴스 엔드포인트의 IP 주소입니다.

```
import click
import ibm_db
import sys

port = port;
master_user_id = "master_username" # Master id used to create your DB instance
master_password = "master_password" # Master password used to create your DB
 instance
db_name = "database_name" # If not given "db-name'
vpc_customer_private_ip = "ip_address" # Hosts end points - Customer private IP
 Addressicert_path = "/root/ssl/global-bundle.pem" # cert path

@click.command()
@click.option("--path", help="certificate path")
def db2_connect(path):

    try:
        conn =
 ibm_db.connect(f"DATABASE={db_name};HOSTNAME={vpc_customer_private_ip};PORT={port};

 PROTOCOL=TCPIP;UID={master_user_id};PWD={master_password};SECURITY=ssl;SSLServerCertifi
 "", "")
        try:
            ibm_db.exec_immediate(conn, 'create table tablename (a int);')
            print("Query executed successfully")
        except Exception as e:
```

```
            print(e)
        finally:
            ibm_db.close(conn)
            sys.exit(1)
    except Exception as ex:
        print("Trying to connect...")

if __name__ == "__main__":
    db2_connect()
```

2. 생성한 Python 파일을 실행하는 다음 쉘 스크립트를 생성합니다. *python_file_name.py*를 Python 스크립트 파일의 이름으로 변경합니다.

```
#!/bin/bash
PEM_FILE=$1
# number of certs in the PEM file
CERTS=$(grep 'END CERTIFICATE' $PEM_FILE| wc -l)

for N in $(seq 0 $(($CERTS - 1))); do
    ALIAS="${PEM_FILE%.*}-$N"
    cert=`cat $PEM_FILE | awk "n==$N { print }; /END CERTIFICATE/ { n++ }"`
    cat $PEM_FILE | awk "n==$N { print }; /END CERTIFICATE/ { n++ }" >
 $ALIAS.pem
    python3 <python_file_name.py> --path $ALIAS.pem
    output=`echo $?`
    if [ $output == 1 ]; then
        break
    fi
done
```

3. 인증서 번들과 함께 PEM 파일을 가져오고 쉘 스크립트를 실행하려면 다음 명령을 실행합니다. *shell_file_name.sh*를 쉘 스크립트 파일 이름으로 변경합니다.

```
./shell_file_name.sh global-bundle.pem
```

Amazon RDS for Db2에 대한 Kerberos 인증 사용

Amazon RDS for Db2 DB 인스턴스에 연결할 때 Kerberos 인증을 통해 사용자를 인증할 수 있습니다. DB 인스턴스는 AWS Directory Service for Microsoft Active Directory(AWS Managed Microsoft AD)와 함께 작동하여 Kerberos 인증을 활성화합니다. 사용자가 신뢰하는 도메인에 조인한 RDS for Db2 DB

인스턴스를 사용하여 인증할 경우 AWS Directory Service를 사용하여 만든 디렉터리에 인증 요청이 전달됩니다. 자세한 내용은 AWS Directory Service 관리 가이드의 AWS Directory Service란 무엇입니까?를 참조하세요.

먼저 사용자 보안 인증 정보를 저장할 AWS Managed Microsoft AD 디렉터리를 만듭니다. 그런 다음 AWS Managed Microsoft AD 디렉터리의 도메인 및 기타 정보를 RDS for Db2 DB 인스턴스에 추가합니다. RDS for Db2 DB 인스턴스에 대해 사용자가 인증될 때 AWS Managed Microsoft AD 디렉터리에 인증 요청이 전달됩니다.

모든 자격 증명을 동일한 디렉터리에 보관하면 시간과 노력을 절약할 수 있습니다. 이 접근 방식의 경우, 여러 DB 인스턴스에 대한 자격 증명을 보관하고 관리할 수 있는 중앙 집중식 공간이 있습니다. 디렉터리를 사용하면 전체 보안 프로필을 향상할 수도 있습니다.

Kerberos 인증에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- Amazon RDS for Db2 DB 인스턴스에 대한 Kerberos 인증 설정
- Kerberos 인증을 사용하여 Amazon RDS for Db2에 연결

리전 및 버전 사용 가능 여부

기능 가용성 및 해당 지원은 각 데이터베이스 엔진의 특정 버전 및 AWS 리전 리전에 따라 다릅니다. Kerberos 인증을 사용하는 RDS for Db2의 버전 및 리전 가용성에 대한 자세한 내용은 Amazon RDS에서 Kerberos 인증을 지원하는 리전 및 DB 엔진 섹션을 참조하세요.

> **Note**
>
> RDS for Db2 DB 인스턴스에서 사용 중단된 DB 인스턴스 클래스에는 Kerberos 인증이 지원되지 않습니다. 자세한 내용은 Amazon RDS for Db2 인스턴스 클래스 섹션을 참조하세요.

RDS for Db2 DB 인스턴스에 대한 Kerberos 인증 개요

RDS for Db2 DB 인스턴스에 대해 Kerberos 인증을 설정하려면 다음과 같은 일반적인 단계(나중에 자세히 설명함)를 완료하세요.

1. AWS Managed Microsoft AD를 사용하여 AWS Managed Microsoft AD 디렉터리를 생성합니다. AWS Management Console, AWS Command Line Interface(AWS CLI) 또는 AWS Directory

Service를 사용하여 디렉터리를 생성할 수 있습니다. 자세한 내용은 AWS Directory Service 관리 가이드의 AWS Managed Microsoft AD 디렉터리 생성을 참조하세요.

2. 관리형 IAM 정책 `AmazonRDSDirectoryServiceAccess`를 사용하는 AWS Identity and Access Management(IAM) 역할을 생성합니다. 이 IAM 역할을 사용하여 Amazon RDS에서 디렉터리를 호출할 수 있습니다.

   IAM 역할이 액세스를 허용하려면 AWS 계정의 올바른 AWS 리전에서 AWS Security Token Service(AWS STS) 엔드포인트를 활성화해야 합니다. AWS STS 엔드포인트는 모든 AWS 리전에서 기본적으로 활성화되어 있으며 추가 작업 없이 사용할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 리전에서 AWS STS 활성화 및 비활성화를 참조하세요.

3. AWS Management Console, AWS CLI 또는 RDS API에서 다음 방법 중 하나를 사용하여 RDS for Db2 DB 인스턴스를 생성하거나 수정합니다.
   - 콘솔, create-db-instance 명령 또는 CreateDBInstance API 작업을 사용하여 새 RDS for Db2 DB 인스턴스를 생성합니다. 지침은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
   - 콘솔, modify-db-instance 명령 또는 ModifyDBInstance API 작업을 사용하여 기존 RDS for Db2 DB 인스턴스를 수정합니다. 지침은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.
   - 콘솔, restore-db-instance-from-db-snapshot 명령 또는 RestoreDBInstanceFromDBSnapshot API 작업을 사용하여 DB 스냅샷에서 RDS for Db2 DB 인스턴스를 복원합니다. 지침은 DB 인스턴스 복원 섹션을 참조하세요.
   - 콘솔, restore-db-instance-to-point-in-time 명령 또는 RestoreDBInstanceToPointInTime API 작업을 사용하여 RDS for Db2 DB 인스턴스를 특정 시점으로 복원합니다. 지침은 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 섹션을 참조하세요.

   디렉터리와 동일한 Amazon Virtual Private Cloud(VPC) 또는 다른 AWS 계정이나 VPC에 DB 인스턴스를 배치할 수 있습니다. RDS for Db2 DB 인스턴스를 생성하거나 수정할 때 다음 작업을 수행합니다.
   - 디렉터리를 만들 때 생성된 도메인 식별자(`d-*` 식별자)를 제공합니다.
   - 생성한 IAM 역할의 이름을 제공합니다.
   - DB 인스턴스의 보안 그룹이 디렉터리의 보안 그룹에서 인바운드 트래픽을 수신할 수 있는지 확인합니다.

4. Db2 클라이언트를 구성하고 다음 포트에 대해 클라이언트 호스트와 AWS Directory Service 간에 트래픽이 흐를 수 있는지 확인합니다.
   - TCP/UDP 포트 53 – DNS
   - TCP 88 – Kerberos 인증
   - TCP 389 – LDAP

- TCP 464 – Kerberos 인증

도메인에서 DB 인스턴스 관리

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB 인스턴스 및 DB 인스턴스와 Microsoft Active Directory과의 관계를 관리할 수 있습니다. 예를 들어, Active Directory를 연결하여 Kerberos 인증을 활성화할 수 있습니다. 또한 Active Directory 연결을 제거하여 Kerberos 인증을 비활성화할 수 있습니다. 또한 외부에서 인증할 DB 인스턴스를 한 Microsoft Active Directory에서 다른 디렉터리로 이동할 수 있습니다.

예를 들어, modify-db-instance CLI 명령을 실행하여 다음 작업을 수행할 수 있습니다.

- `--domain` 옵션에 현재 멤버십의 디렉터리 ID를 지정하여 실패한 멤버십에 대한 Kerberos 인증 활성화를 다시 시도합니다.
- `--domain` 옵션에 대해 `none`을 지정하여 DB 인스턴스에서 Kerberos 인증을 비활성화합니다.
- `--domain` 옵션에 대한 새 도메인의 도메인 식별자를 지정하여 한 도메인에서 다른 도메인으로 DB 인스턴스를 이동합니다.

도메인 멤버십 이해

DB 인스턴스를 생성하거나 수정하고 나면 해당 인스턴스는 도메인의 멤버가 됩니다. 콘솔에서 또는 describe-db-instances CLI 명령을 실행하여 도메인 멤버십의 상태를 확인할 수 있습니다. DB 인스턴스의 상태는 다음 중 한 가지가 될 수 있습니다.

- `kerberos-enabled` – DB 인스턴스에 Kerberos 인증이 활성화되어 있습니다.
- `enabling-kerberos` - AWS에서 이 DB 인스턴스에 대한 Kerberos 인증 활성화를 진행 중입니다.
- `pending-enable-kerberos` – 이 DB 인스턴스에 대한 Kerberos 인증 활성화가 보류 중입니다.
- `pending-maintenance-enable-kerberos` - AWS에서 예약된 다음 유지 관리 기간에 DB 인스턴스에 대한 Kerberos 인증을 활성화하려 합니다.
- `pending-disable-kerberos` – 이 DB 인스턴스에 대한 Kerberos 인증 비활성화가 보류 중입니다.
- `pending-maintenance-disable-kerberos` - AWS에서 예약된 다음 유지 관리 기간에 DB 인스턴스에 대한 Kerberos 인증을 비활성화하려 합니다.
- `enable-kerberos-failed` - 구성 문제로 인해 AWS가 DB 인스턴스에 대해 Kerberos 인증을 활성화하지 못했습니다. DB 인스턴스 수정 명령을 다시 실행하기 전에 구성 문제를 해결하세요.

- `disabling-kerberos` - AWS에서 이 DB 인스턴스에 대한 Kerberos 인증 비활성화를 진행 중입니다.

네트워크 연결 문제 또는 잘못된 IAM 역할로 인해 Kerberos 인증 활성화 요청이 실패할 수 있습니다. 경우에 따라 DB 인스턴스를 만들거나 수정할 때 Kerberos 인증을 사용하려고 하면 실패할 수 있습니다. 이런 경우 올바른 IAM 역할을 사용하고 있는지 확인한 다음 DB 인스턴스를 수정하여 도메인에 조인합니다.

Amazon RDS for Db2 DB 인스턴스에 대한 Kerberos 인증 설정

AWS Directory Service for Microsoft Active Directory(AWS Managed Microsoft AD)를 사용하여 RDS for Db2 DB 인스턴스에 대한 Kerberos 인증을 설정합니다. Kerberos 인증을 설정하려면 다음 단계를 따릅니다.

주제

- 1단계: AWS Managed Microsoft AD를 사용하여 디렉터리 생성
- 2단계: Amazon RDS에서 AWS Directory Service에 액세스할 수 있도록 IAM 역할 생성
- 3단계: 사용자 생성 및 구성
- 4단계: AWS Managed Microsoft AD에서 RDS for Db2 관리자 그룹 생성
- 5단계: RDS for Db2 DB 인스턴스 생성 또는 수정
- 6단계: Db2 클라이언트 구성

1단계: AWS Managed Microsoft AD를 사용하여 디렉터리 생성

AWS Directory Service는 AWS 클라우드에서 완전관리형 Active Directory를 생성합니다. AWS Managed Microsoft AD 디렉터리를 생성할 때 AWS Directory Service에서 두 개의 도메인 컨트롤러와 DNS 서버가 자동으로 생성됩니다. 디렉터리 서버는 VPC 내 다른 서브넷에서 생성됩니다. 이러한 중복은 장애가 발생해도 디렉터리에 액세스할 수 있도록 보장하는 데 도움이 됩니다.

AWS Managed Microsoft AD 디렉터리를 생성하는 경우 AWS Directory Service에서 다음 작업이 자동으로 수행됩니다.

- VPC 내에서 Active Directory를 설정합니다.
- 사용자 이름 `Admin`과 지정된 암호를 사용하여 디렉터리 관리자 계정을 생성합니다. 이 계정을 사용하여 디렉터리를 관리할 수 있습니다.

> **Important**
>
> 반드시 이 암호를 저장해야 합니다. AWS Directory Service에서는 이 암호를 저장하지 않으므로 암호를 검색하거나 다시 설정할 수 없습니다.

- 디렉터리 컨트롤러에 대한 보안 그룹을 만듭니다. 보안 그룹이 RDS for Db2 DB 인스턴스와의 통신을 허용해야 합니다.

AWS Directory Service for Microsoft Active Directory를 시작하면 AWS는 디렉터리의 모든 객체를 포함하는 조직 단위(OU)를 생성합니다. 디렉터리를 만들 때 입력한 NetBIOS 이름이 있는 이 OU는 도메인 루트에 있습니다. 도메인 루트는 AWS에서 소유하고 관리합니다.

`Admin` 디렉터리를 사용하여 생성한 AWS Managed Microsoft AD 계정은 OU의 가장 일반적인 관리 활동에 대한 권한이 있습니다.

- 사용자를 생성, 업데이트 또는 삭제합니다.
- 파일 또는 인쇄 서버와 같은 도메인에 리소스를 추가한 다음 해당 리소스에 대한 권한을 OU의 사용자에게 할당합니다.
- 추가 OU 및 컨테이너 생성.
- 권한 위임.
- 삭제된 객체를 Active Directory 휴지통에서 복원합니다.
- AWS Directory Service에서 Windows PowerShell에 대한 Active Directory 및 도메인 이름 서비스(DNS) 모듈을 실행합니다.

또한 `Admin` 계정은 다음 도메인 차원 활동을 수행할 권한이 있습니다.

- DNS 구성 관리(레코드, 영역 및 전달자 추가, 제거 또는 업데이트)
- DNS 이벤트 로그 보기
- 보안 이벤트 로그 보기

AWS Managed Microsoft AD으로 디렉터리를 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/directoryservicev2/에서 AWS Directory Service 콘솔을 엽니다.
2. 디렉터리 설정을 선택합니다.

3. AWS Managed Microsoft AD를 선택합니다. AWS Managed Microsoft AD는 현재 Amazon RDS에서 사용하도록 지원되는 유일한 옵션입니다.
4. 다음을 선택합니다.
5. 디렉터리 정보 입력 페이지에서 다음 정보를 제공합니다.

   - 에디션 – 요구 사항에 맞는 에디션을 선택합니다.
   - 디렉터리 DNS 이름 – 디렉터리를 위한 정규화된 이름(예: `corp.example.com`)입니다.
   - 디렉터리 NetBIOS 이름 – 디렉터리의 선택적 짧은 이름(예: `CORP`)입니다.
   - 디렉터리 설명 – 디렉터리에 대한 선택적 설명입니다.
   - 관리자 암호 – 디렉터리 관리자의 암호입니다. 디렉터리 생성 프로세스에서는 사용자 이름 `Admin`과 이 암호를 사용하여 관리자 계정을 생성합니다.

     디렉터리 관리자 암호는 "admin"이라는 단어를 포함할 수 없습니다. 암호는 대소문자를 구분하며 길이가 8~64자 사이여야 합니다. 또한 다음 네 범주 중 세 개에 해당하는 문자를 1자 이상 포함해야 합니다.

     - 소문자(a–z)
     - 대문자(A–Z)
     - 숫자(0–9)
     - 영숫자 외의 특수 문자(~!@#$%^&*_-+=`|\(){}[]:;"'<>,.?/)
     - 암호 확인 – 관리자 암호를 다시 입력합니다.

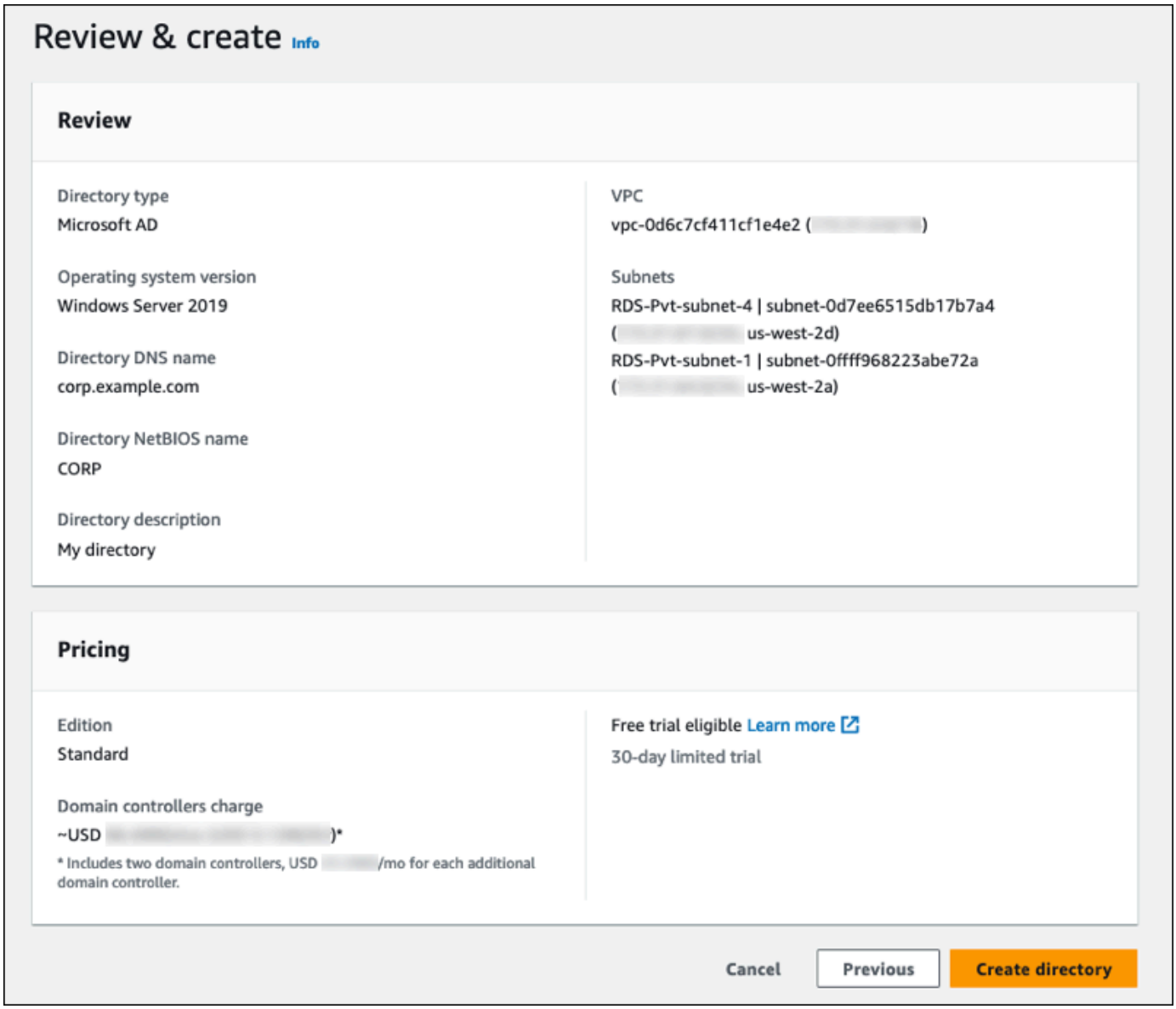
     > **Important**
     >
     > 반드시 이 암호를 저장해야 합니다. AWS Directory Service에서는 이 암호를 저장하지 않으며 암호를 검색하거나 재설정할 수 없습니다.

6. Next(다음)를 선택합니다.
7. Choose VPC and subnets(VPC 및 서브넷 선택) 페이지에 다음 정보를 입력합니다.

   - VPC – 디렉터리에 대한 VPC를 선택합니다. RDS for Db2 DB 인스턴스는 이와 동일한 VPC 또는 다른 VPC에서 생성할 수 있습니다.
   - 서브넷 – 디렉터리 서버에 대한 서브넷을 선택합니다. 두 서브넷이 서로 다른 가용 영역에 있어야 합니다.

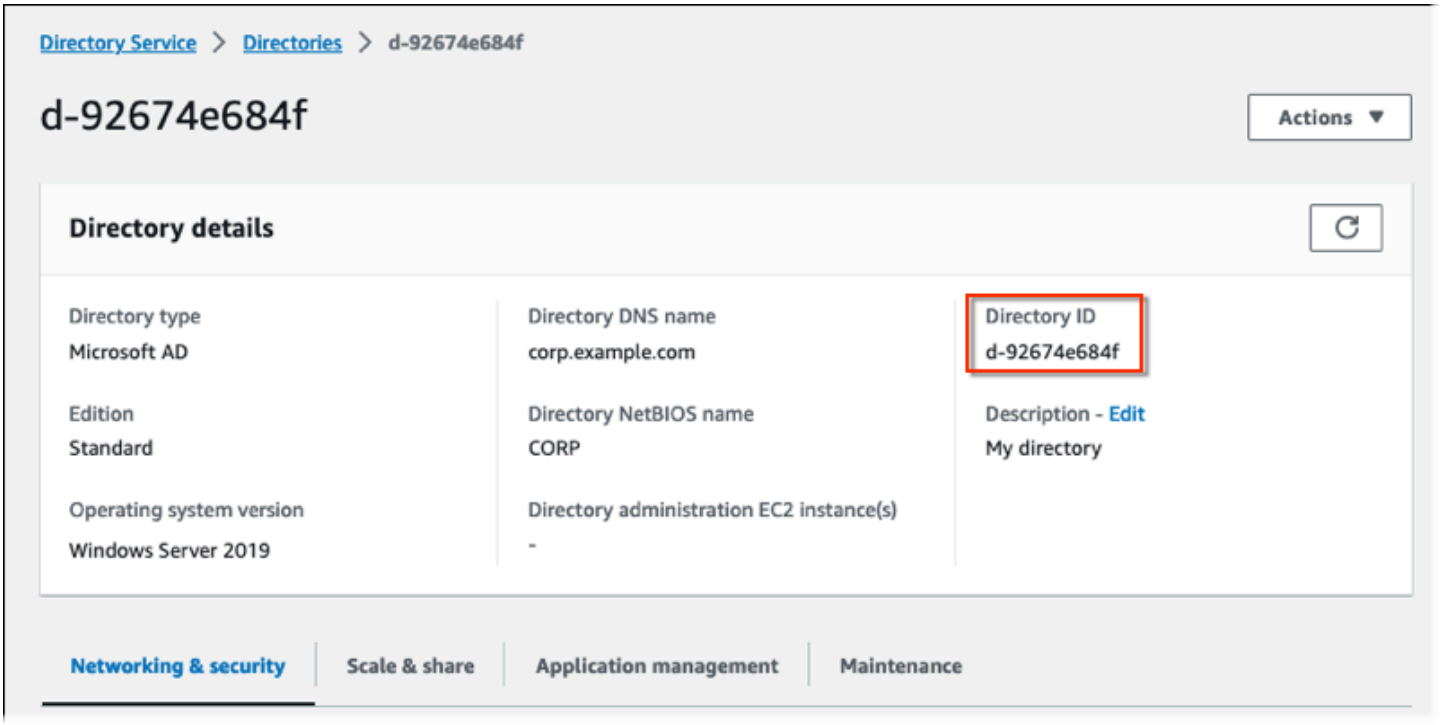
8. Next(다음)를 선택합니다.

9. 디렉터리 정보를 검토합니다. 변경이 필요하면 이전을 선택하여 변경합니다. 정보가 올바르면 Create directory(디렉터리 생성)을 선택합니다.



디렉터리를 생성하는 데 몇 분 정도 걸립니다. 디렉터리가 성공적으로 생성되면 상태 값이 활성으로 변경됩니다.

디렉터리에 대한 정보를 보려면 디렉터리 ID에서 해당 디렉터리 ID를 선택합니다. 디렉터리 ID 값을 적어 두십시오. RDS for Db2 DB 인스턴스를 생성하거나 수정할 때 이 값이 필요합니다.



2단계: Amazon RDS에서 AWS Directory Service에 액세스할 수 있도록 IAM 역할 생성

사용자를 대신해서 Amazon RDS가 AWS Directory Service를 호출하도록 하려면 AWS 계정에서 관리형 IAM 정책 `AmazonRDSDirectoryServiceAccess`를 사용하는 IAM 역할이 필요합니다. 이 역할을 사용하여 Amazon RDS에서 AWS Directory Service를 호출할 수 있습니다.

AWS Management Console을 사용하여 DB 인스턴스를 생성할 때 콘솔 사용자에게 `iam:CreateRole` 권한이 있으면 콘솔에서 필요한 IAM 역할을 자동으로 생성합니다. 이 경우 역할 이름은 `rds-directoryservice-kerberos-access-role`입니다. 그렇지 않으면 IAM 역할을 수동으로 생성해야 합니다. 이 IAM 역할을 생성할 때 `Directory Service`를 선택하고 여기에 AWS 관리형 정책인 `AmazonRDSDirectoryServiceAccess`를 연결합니다.

서비스에 대한 IAM 역할 생성에 대한 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.

> **Note**
>
> RDS for Microsoft SQL Server에 대한 Windows 인증에 사용되는 IAM 역할은 RDS for Db2에 사용할 수 없습니다.

AmazonRDSDirectoryServiceAccess 관리형 정책인을 사용하는 대신 필요한 권한이 포함된 정책을 생성할 수 있습니다. 이 경우 IAM 역할에 다음과 같은 IAM 신뢰 정책이 있어야 합니다.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.amazonaws.com",
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

또한 역할에는 다음과 같은 IAM 역할 정책도 있어야 합니다.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ds:DescribeDirectories",
        "ds:AuthorizeApplication",
        "ds:UnauthorizeApplication",
        "ds:GetAuthorizedApplicationDetails"
      ],
    "Effect": "Allow",
    "Resource": "*"
    }
  ]
```

```
}
```

3단계: 사용자 생성 및 구성

Active Directory Users and Computers 도구를 사용하여 사용자를 만들 수 있습니다. Active Directory Domain Services 및 Active Directory Lightweight Directory Services 도구 중 하나입니다. 자세한 내용은 Microsoft 설명서의 Active Directory 도메인에 사용자 및 컴퓨터 추가를 참조하세요. 이 경우 사용자는 도메인에 속하며 디렉터리에서 ID가 유지되는 개인 또는 기타 엔터티(예: 사용자의 컴퓨터)입니다.

AWS Directory Service 디렉터리에서 사용자를 생성하려면 AWS Directory Service 디렉터리의 멤버인 Windows 기반 Amazon EC2 인스턴스에 연결되어 있어야 합니다. 이와 동시에 사용자를 생성할 권한이 있는 사용자로 로그인한 상태이어야 합니다. 자세한 내용은 AWS Directory Service관리 안내서의 사용자 생성을 참조하십시오.

4단계: AWS Managed Microsoft AD에서 RDS for Db2 관리자 그룹 생성

RDS for Db2는 마스터 사용자 또는 2명의 Amazon RDS 예약 사용자 `rdsdb` 및 `rdsadmin`에 대한 Kerberos 인증을 지원하지 않습니다. 대신 AWS Managed Microsoft AD에서 `masterdba`라는 새로운 그룹을 생성해야 합니다. 자세한 내용은 Microsoft 설명서의 Active Directory에서 그룹 계정 생성을 참조하세요. 이 그룹에 추가하는 모든 사용자는 마스터 사용자 권한을 갖게 됩니다.

Kerberos 인증을 활성화하면 마스터 사용자는 `masterdba` 역할을 잃습니다. 따라서 Kerberos 인증을 비활성화하지 않으면 마스터 사용자는 인스턴스 로컬 사용자 그룹 멤버십에 액세스할 수 없습니다. 암호 로그인을 통해 마스터 사용자를 계속 사용하려면 AWS Managed Microsoft AD에서 마스터 사용자와 동일한 이름을 가진 사용자를 생성하세요. 그런 다음 사용자를 `masterdba` 그룹에 추가합니다.

5단계: RDS for Db2 DB 인스턴스 생성 또는 수정

디렉터리에서 사용할 RDS for Db2 DB 인스턴스를 생성하거나 수정합니다. AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB 인스턴스를 디렉터리에 연결할 수 있습니다. 이 작업을 다음 중 한 가지 방법으로 수행할 수 있습니다.

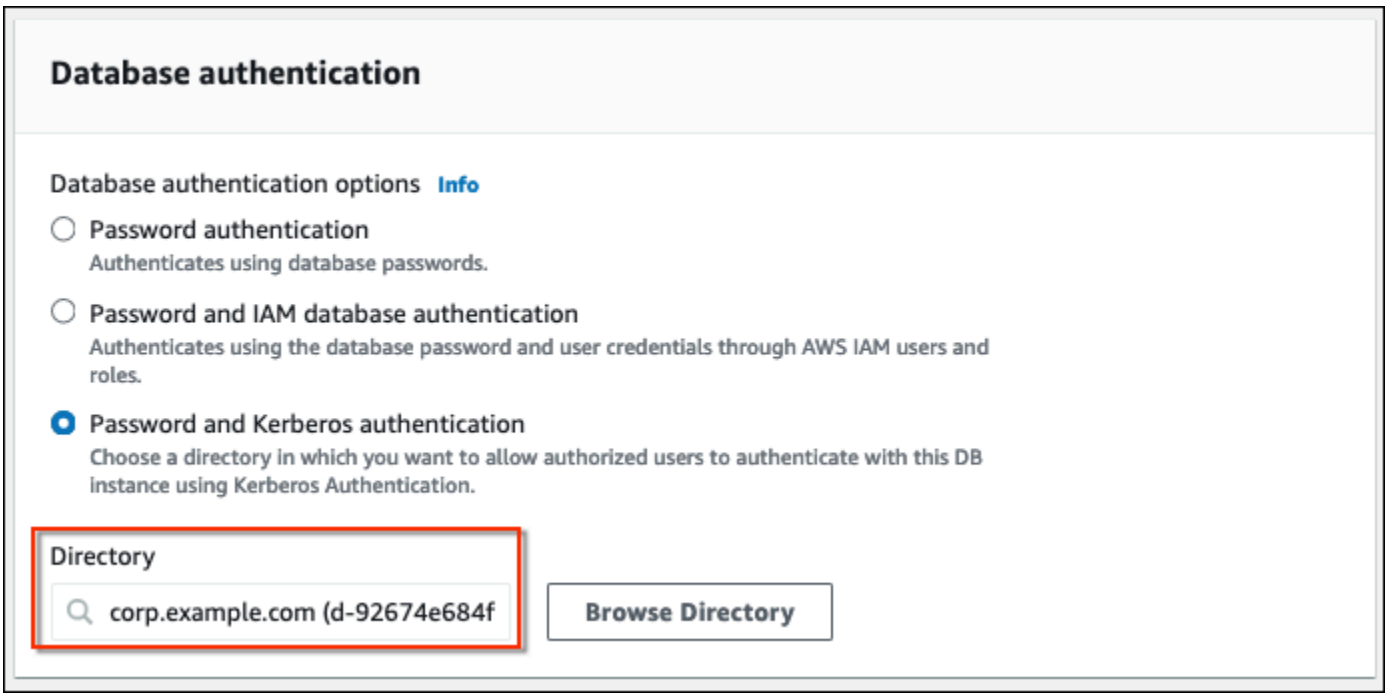
- 콘솔, create-db-instance 명령 또는 CreateDBInstance API 작업을 사용하여 새 RDS for Db2 DB 인스턴스를 생성합니다. 지침은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
- 콘솔, modify-db-instance 명령 또는 ModifyDBInstance API 작업을 사용하여 기존 RDS for Db2 DB 인스턴스를 수정합니다. 지침은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.
- 콘솔, restore-db-instance-from-db-snapshot 명령 또는 RestoreDBInstanceFromDBSnapshot API 작업을 사용하여 DB 스냅샷에서 RDS for Db2 DB 인스턴스를 복원합니다. 지침은 DB 인스턴스 복원 섹션을 참조하세요.

- 콘솔, restore-db-instance-to-point-in-time 명령 또는 RestoreDBInstanceToPointInTime API 작업을 사용하여 RDS for Db2 DB 인스턴스를 특정 시점으로 복원합니다. 지침은 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 섹션을 참조하세요.

Kerberos 인증은 VPC의 RDS for Db2 DB 인스턴스에 대해서만 지원됩니다. DB 인스턴스는 디렉터리와 동일한 VPC 또는 다른 VPC에 있을 수 있습니다. DB 인스턴스가 디렉터리와 통신할 수 있도록 DB 인스턴스는 디렉터리의 VPC 내 송수신을 허용하는 보안 그룹을 사용해야 합니다.

콘솔

콘솔을 사용하여 DB 인스턴스를 생성, 수정 또는 복원할 때 데이터베이스 인증 섹션에서 암호 및 Kerberos 인증을 선택합니다. 그런 다음 [디렉터리 찾아보기(Browse Directory)]를 선택합니다. 디렉터리를 선택하거나 디렉터리 생성을 선택하여 Directory Service를 사용합니다.



AWS CLI

AWS CLI를 사용하는 경우 생성한 도메인 디렉터리를 DB 인스턴스에서 사용하려면 다음과 같은 파라미터가 필요합니다.

- `--domain` 파라미터의 경우 디렉터리를 만들 때 생성된 도메인 식별자('d-*' 식별자)를 사용하세요.
- `--domain-iam-role-name` 파라미터의 경우 귀하가 생성한, 관리형 IAM 정책 `AmazonRDSDirectoryServiceAccess`를 사용하는 역할을 사용하십시오.

다음 예제에서는 디렉터리를 사용하도록 DB 인스턴스를 수정합니다. 예제에서 다음 자리 표시자를 사용자의 값으로 바꿉니다.

- *db_instance_name* – RDS for Db2 DB 인스턴스의 이름입니다.
- *directory_id* – 생성한 AWS Directory Service for Microsoft Active Directory 디렉터리의 ID입니다.
- *role_name* – 생성한 IAM 역할의 이름입니다.

```
aws rds modify-db-instance --db-instance-identifier db_instance_name --domain
 d-directory_id --domain-iam-role-name role_name
```

> **Important**
>
> DB 인스턴스를 수정하여 Kerberos 인증을 활성화하는 경우에는 변경 후 DB 인스턴스를 재부팅하세요.

6단계: Db2 클라이언트 구성

Db2 클라이언트를 구성하려면

1. 도메인을 가리키도록 /etc/krb5.conf 파일(또는 동등한 파일)을 생성합니다.

   > **Note**
   >
   > Windows 운영 체제의 경우 C:\windows\krb5.ini 파일을 생성합니다.

2. 클라이언트 호스트와 AWS Directory Service 간에 트래픽이 흐를 수 있는지 확인합니다. Netcat과 같은 네트워크 유틸리티를 사용하여 다음 작업을 수행하세요.

   a. 포트 53의 DNS를 통한 트래픽을 확인합니다.

   b. 포트 53 및 AWS Directory Service용 포트 88 및 464를 포함하는 Kerberos의 TCP/UDP를 통한 트래픽을 확인합니다.

3. 데이터베이스 포트를 통해 클라이언트 호스트와 DB 인스턴스 간에 트래픽이 흐를 수 있는지 확인합니다. db2 명령을 사용하여 데이터베이스에 연결하고 액세스할 수 있습니다.

다음 예제는 AWS Managed Microsoft AD에 대한 /etc/krb5.conf 파일 콘텐츠입니다.

```
[libdefaults]
default_realm = EXAMPLE.COM
[realms]
EXAMPLE.COM = {
kdc = example.com
admin_server = example.com
}
[domain_realm]
.example.com = EXAMPLE.COM
example.com = EXAMPLE.COM
```

Kerberos 인증을 사용하여 Amazon RDS for Db2에 연결

다음 절차에 따라 Kerberos 인증을 통해 Amazon RDS for Db2 DB 인스턴스에 연결합니다.

Kerberos 인증을 사용하여 RDS for Db2에 연결하려면

1. 명령 프롬프트에서 다음 명령을 실행합니다. 다음 예제에서 *username*을 Microsoft Active Directory 사용자 이름으로 바꿉니다.

   ```
   kinit username
   ```

2. RDS for Db2 DB 인스턴스가 공개 액세스 가능한 VPC를 사용하는 경우 Amazon EC2 클라이언트의 `/etc/hosts` 파일에 DB 인스턴스 엔드포인트의 IP 주소를 추가합니다. 다음 예제에서는 IP 주소를 얻은 다음 `/etc/hosts` 파일에 추가합니다.

   ```
   % dig +short Db2-endpoint.AWS-Region.rds.amazonaws.com
   ;; Truncated, retrying in TCP mode.
   ec2-34-210-197-118.AWS-Region.compute.amazonaws.com.
   34.210.197.118

   % echo "34.210.197.118  Db2-endpoint.AWS-Region.rds.amazonaws.com" >> /etc/hosts
   ```

3. 다음 명령을 사용하여 Active Directory와 연결된 RDS for Db2 DB 인스턴스에 로그인합니다. *database_name*을 RDS for Db2 데이터베이스의 이름으로 바꿉니다.

   ```
   db2 connect to database_name
   ```

Amazon RDS for Db2 DB 인스턴스 관리

이 주제에서는 Amazon RDS for Db2 DB 인스턴스와 관련하여 수행하는 일반적인 관리 작업을 안내합니다. 일부 작업은 모든 Amazon RDS DB 인스턴스에서 동일합니다. RDS for Db2에만 해당되는 작업도 있습니다.

다음 작업은 모든 RDS 데이터베이스에 공통적으로 적용됩니다. 표준 SQL 클라이언트를 사용하여 RDS for Db2 데이터베이스에 연결하는 것과 같이 RDS for Db2 전용 작업도 있습니다.

작업 영역	관련 설명서
인스턴스 클래스, 스토리지 및 PIOPS 프로덕션 인스턴스를 생성하는 경우 Amazon RDS에서 인스턴스 클래스, 스토리지 유형 및 프로비저닝된 IOPS가 작동하는 방식을 알아봅니다.	DB 인스턴스 클래스 Amazon RDS 스토리지 유형
다중 AZ 배포 프로덕션 DB 인스턴스에서는 다중 AZ 배포를 사용해야 합니다. 다중 AZ 배포는 DB 인스턴스를 위해 향상된 가용성, 데이터 내구성 및 내결함성을 제공합니다.	Amazon RDS에 대한 다중 AZ 배포 구성 및 관리
Amazon VPC AWS 계정에 기본 Virtual Private Cloud(VPC)가 있는 경우에는 DB 인스턴스가 기본 VPC 내부에 자동으로 생성됩니다. 계정에 기본 VPC가 없는데 VPC 안에 DB 인스턴스를 만들려면 VPC와 서브넷 그룹을 만든 후 DB 인스턴스를 만듭니다.	VPC에서 DB 인스턴스를 사용한 작업
보안 그룹 기본적으로 DB 인스턴스는 액세스를 막는 방화벽을 사용합니다. 따라서 DB 인스턴스에 액세스하기 위한 알맞은 IP 주소와 네트워크 구성으로 보안 그룹을 생성해야 합니다.	보안 그룹을 통한 액세스 제어
파라미터 그룹 수(Parameter groups) RDS for Db2 DB 인스턴스에 `rds.ibm_customer_id` 및 `rds.ibm_site_id` 파라미터를 추가해야 하므로, DB 인스턴스	RDS for Db2 DB 인스턴스의 파라미터 그룹에 IBM ID 추가 Amazon RDS의 파라미터 그룹

작업 영역	관련 설명서
를 생성하기 전에 파라미터 그룹을 생성해야 합니다. DB 인스턴스에 다른 특정 데이터베이스 파라미터가 필요한 경우 DB 인스턴스를 생성하기 전에 이 파라미터 그룹에 추가합니다.	
옵션 그룹 수 DB 인스턴스에 특정 데이터베이스 옵션이 필요할 경우, 옵션 그룹을 생성한 후 DB 인스턴스를 생성합니다.	Amazon RDS for Db2 DB 인스턴스 옵션
DB 인스턴스에 연결 보안 그룹을 만들고 이를 DB 인스턴스에 연결한 후 IBM Db2 CLP와 같은 표준 SQL 클라이언트 애플리케이션을 사용하여 DB 인스턴스에 연결할 수 있습니다.	DB 인스턴스에 연결
백업 및 복원 DB 인스턴스를 구성하여 자동 스토리지 백업을 생성하거나 수동 스토리지 스냅샷을 생성한 다음 백업 또는 스냅샷에서 인스턴스를 복원할 수 있습니다.	데이터 백업, 복원 및 내보내기
모니터링(Monitoring) IBM Db2 Data Management Console을 사용하여 RDS for Db2 DB 인스턴스를 모니터링할 수 있습니다. CloudWatch Amazon RDS 지표, 이벤트 및 향상된 모니터링 기능을 통해 RDS for Db2 DB 인스턴스를 모니터링할 수도 있습니다.	IBM Db2 Data Management Console을 사용하여 Amazon RDS for Db2 DB 인스턴스에 연결 Amazon RDS 콘솔에서 지표 보기 Amazon RDS 이벤트 보기 Enhanced Monitoring을 사용하여 OS 지표 모니터링
로그 파일 RDS for Db2 DB 인스턴스의 로그 파일에 액세스할 수 있습니다.	Amazon RDS 로그 파일 모니터링

주제

- Amazon RDS for Db2 DB 인스턴스에 대한 공통 시스템 작업 수행
- Amazon RDS for Db2 DB 인스턴스에 대한 공통 데이터베이스 작업 수행

Amazon RDS for Db2 DB 인스턴스에 대한 공통 시스템 작업 수행

Db2를 실행하는 Amazon RDS DB 인스턴스에서 시스템과 관련된 일반적인 데이터베이스 관리 작업을 수행할 수 있습니다. 관리형 서비스 환경을 제공하기 위해 Amazon RDS는 DB 인스턴스에 대해 shell 액세스를 제공하지 않으며, 고급 권한이 필요한 특정 시스템 절차와 테이블에 대한 액세스를 제한합니다.

RDS for Db2에서 권한을 부여 및 취소하고 원격 데이터베이스에 연결하는 방법에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- RDS for Db2에 대한 권한 부여 및 취소
- 원격 RDS for Db2 DB 인스턴스에 연결

사용자 지정 데이터베이스 엔드포인트 생성

Amazon RDS for Db2로 마이그레이션할 때 사용자 지정 데이터베이스 엔드포인트 URL을 사용하여 애플리케이션 변경을 최소화할 수 있습니다. 예를 들어, `db2.example.com`을 현재 DNS 레코드로 사용하는 경우 Amazon Route 53에 추가할 수 있습니다. Route 53에서는 프라이빗 호스팅 영역을 사용하여 현재 DNS 데이터베이스 엔드포인트를 RDS for Db2 데이터베이스 엔드포인트에 매핑할 수 있습니다. Amazon RDS 데이터베이스 엔드포인트에 대해 사용자 지정 `A` 또는 `CNAME` 레코드를 추가하려면 Amazon Route 53 개발자 안내서의 Amazon Route 53을 사용한 도메인 등록 및 관리를 참조하세요.

> **Note**
>
> 도메인을 Route 53으로 이전할 수 없는 경우 DNS 공급자를 사용하여 RDS for Db2 데이터베이스 엔드포인트 URL에 대한 `CNAME` 레코드를 생성할 수 있습니다. DNS 공급자 설명서를 참조하세요.

RDS for Db2에 대한 권한 부여 및 취소

사용자는 데이터베이스에 연결된 그룹의 멤버십을 통해 데이터베이스에 액세스할 수 있습니다.

다음 프로시저를 사용하여 데이터베이스에 대한 액세스를 제어할 수 있는 권한을 부여하거나 취소할 수 있습니다.

이 프로시저는 로컬 머신에서 IBM Db2 CLP를 실행하여 RDS for Db2 DB 인스턴스에 연결합니다. 로컬 머신에서 실행되는 RDS for Db2 DB 인스턴스에 연결하려면 TCPIP 노드와 데이터베이스를 카탈로그화해야 합니다. 자세한 내용은 IBM Db2 CLP을 사용하여 Amazon RDS for Db2 DB 인스턴스에 연결 단원을 참조하십시오.

주제

- 사용자에게 데이터베이스 액세스 권한 부여
- 사용자의 암호 변경
- 사용자에게 그룹 추가
- 사용자로부터 그룹 제거
- 사용자 제거
- 사용자 표시
- 역할 생성
- 역할 부여
- 역할 취소
- 역할 삭제
- 데이터베이스 권한 부여
- 데이터베이스 권한 취소

사용자에게 데이터베이스 액세스 권한 부여

사용자에게 데이터베이스 액세스 권한을 부여하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 `rdsadmin` 데이터베이스에 연결합니다. 다음 예제에서 *`master_username`* 및 *`master_password`*를 사용자 자체 정보로 대체합니다.

   ```
   db2 connect to rdsadmin user master_username using master_password
   ```

   다음 예제와 비슷한 출력이 생성됩니다.

   ```
   Database Connection Information
   ```

```
Database server        = DB2/LINUXX8664 11.5.8.0
SQL authorization ID   = ADMIN
Local database alias   = RDSADMIN
```

2. `rdsadmin.add_user`를 호출하여 권한 부여 목록에 사용자를 추가합니다. 자세한 내용은 rdsadmin.add_user 단원을 참조하십시오.

```
db2 "call rdsadmin.add_user(
    'username',
    'password',
    'group_name,group_name')"
```

3. (선택 사항) `rdsadmin.add_groups`를 호출하여 사용자에게 그룹을 더 추가합니다. 자세한 내용은 rdsadmin.add_groups 단원을 참조하십시오.

```
db2 "call rdsadmin.add_groups(
    'username',
    'group_name,group_name')"
```

4. 사용자가 사용할 수 있는 권한을 확인합니다. 다음 예제에서 *rds_database_alias*, *master_user*, *master_password*를 사용자 자체 정보로 바꿉니다. *username*도 사용자의 사용자 이름으로 대체합니다.

```
db2 terminate
db2 connect to rds_database_alias user master_user using master_password
db2 "SELECT SUBSTR(AUTHORITY,1,20) AUTHORITY, D_USER, D_GROUP, D_PUBLIC
       FROM TABLE (SYSPROC.AUTH_LIST_AUTHORITIES_FOR_AUTHID ('username', 'U') ) AS
 T
       ORDER BY AUTHORITY"
```

다음 예제와 비슷한 출력이 생성됩니다.

```
AUTHORITY            D_USER D_GROUP D_PUBLIC
-------------------- ------ ------- --------
ACCESSCTRL           N      N       N
BINDADD              N      N       N
CONNECT              N      N       N
CREATETAB            N      N       N
CREATE_EXTERNAL_ROUT N      N       N
CREATE_NOT_FENCED_RO N      N       N
CREATE_SECURE_OBJECT N      N       N
DATAACCESS           N      N       N
```

```
DBADM                N       N       N
EXPLAIN              N       N       N
IMPLICIT_SCHEMA      N       N       N
LOAD                 N       N       N
QUIESCE_CONNECT      N       N       N
SECADM               N       N       N
SQLADM               N       N       N
SYSADM               *       N       *
SYSCTRL              *       N       *
SYSMAINT             *       N       *
SYSMON               *       N       *
WLMADM               N       N       N
```

5. 사용자를 추가한 그룹에 RDS for Db2 역할 ROLE_NULLID_PACKAGES, ROLE_TABLESPACES, ROLE_PROCEDURES를 부여합니다. 자세한 내용은 Amazon RDS for Db2 기본 역할 단원을 참조하십시오.

> **Note**
>
> RESTRICTIVE 모드에서 RDS for Db2 DB 인스턴스를 생성합니다. 따라서 RDS for Db2 역할 ROLE_NULLID_PACKAGES, ROLE_TABLESPACES, ROLE_PROCEDURES는 IBM Db2 CLP 및 Dynamic SQL에 대해 NULLID 패키지에서 실행 권한을 부여합니다. 또한 이러한 역할은 테이블스페이스에 대한 사용자 권한을 부여합니다.

a. Db2 데이터베이스에 연결합니다. 다음 예제에서 *database_name*, *master_user*, *master_password*를 사용자 자체 정보로 바꿉니다.

```
db2 connect to database_name user master_user using master_password
```

b. 그룹에 ROLE_NULLED_PACKAGES 역할을 부여합니다. 다음 예제에서 *group_name*을 역할을 추가하려는 그룹의 이름으로 바꿉니다.

```
db2 "grant role ROLE_NULLID_PACKAGES to group group_name"
```

c. 동일한 그룹에 ROLE_TABLESPACES 역할을 부여합니다. 다음 예제에서 *group_name*을 역할을 추가하려는 그룹의 이름으로 바꿉니다.

```
db2 "grant role ROLE_TABLESPACES to group group_name"
```

d. 동일한 그룹에 ROLE_PROCEDURES 역할을 부여합니다. 다음 예제에서 *group_name*을 역할을 추가하려는 그룹의 이름으로 바꿉니다.

```
db2 "grant role ROLE_PROCEDURES to group group_name"
```

6. 사용자를 추가한 그룹에 connect, bindadd, createtab, IMPLICIT_SCHEMA 권한을 부여합니다. 다음 예제에서 *group_name*을 사용자를 추가한 두 번째 그룹의 이름으로 대체합니다.

```
db2 "grant usage on workload SYSDEFAULTUSERWORKLOAD to public"
db2 "grant connect, bindadd, createtab, implicit_schema on database to
 group group_name"
```

7. 사용자를 추가한 각 추가 그룹에 대해 4~6단계를 반복합니다.

8. 사용자로 연결하고, 표를 만들고, 표에 값을 넣고, 표에서 데이터를 반환하여 사용자의 액세스 권한을 테스트합니다. 다음 예제에서 *rds_database_alias*, *username*, *password*를 데이터베이스 이름과 사용자의 사용자 이름 및 암호로 바꿉니다.

```
db2 connect to rds_database_alias user username using password
db2 "create table t1(c1 int not null)"
db2 "insert into t1 values (1),(2),(3),(4)"
db2 "select * from t1"
```

사용자의 암호 변경

사용자의 암호를 변경하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 rdsadmin 데이터베이스에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 자체 정보로 대체합니다.

```
db2 connect to rdsadmin user master_username using master_password
```

2. rdsadmin.change_password를 호출하여 암호를 변경합니다. 자세한 내용은 rdsadmin.change_password 단원을 참조하십시오.

```
db2 "call rdsadmin.change_password(
    'username',
    'new_password')"
```

사용자에게 그룹 추가

사용자에게 그룹을 추가하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 `rdsadmin` 데이터베이스에 연결합니다. 다음 예제에서 *`master_username`* 및 *`master_password`*를 사용자 자체 정보로 대체합니다.

   ```
   db2 connect to rdsadmin user master_username using master_password
   ```

2. `rdsadmin.add_groups`를 호출하여 사용자에게 그룹을 추가합니다. 자세한 내용은 rdsadmin.add_groups 단원을 참조하십시오.

   ```
   db2 "call rdsadmin.add_groups(
       'username',
       'group_name,group_name')"
   ```

사용자로부터 그룹 제거

사용자로부터 그룹을 제거하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 `rdsadmin` 데이터베이스에 연결합니다. 다음 예제에서 *`master_username`* 및 *`master_password`*를 사용자 자체 정보로 대체합니다.

   ```
   db2 connect to rdsadmin user master_username using master_password
   ```

2. `rdsadmin.remove_groups`를 호출하여 그룹을 제거합니다. 자세한 내용은 rdsadmin.remove_groups 단원을 참조하십시오.

   ```
   db2 "call rdsadmin.remove_groups(
       'username',
       'group_name,group_name')"
   ```

사용자 제거

권한 부여 목록에서 사용자를 제거하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 `rdsadmin` 데이터베이스에 연결합니다. 다음 예제에서 *`master_username`* 및 *`master_password`*를 사용자 자체 정보로 대체합니다.

   ```
   db2 connect to rdsadmin user master_username using master_password
   ```

2. `rdsadmin.remove_user`를 호출하여 권한 부여 목록에서 사용자를 제거합니다. 자세한 내용은 rdsadmin.remove_user 단원을 참조하십시오.

   ```
   db2 "call rdsadmin.remove_user('username')"
   ```

사용자 표시

권한 부여 목록에 사용자를 표시하려면 `rdsadmin.list_users` 저장 프로시저를 호출하세요. 자세한 내용은 rdsadmin.list_users 단원을 참조하십시오.

```
db2 "call rdsadmin.list_users()"
```

역할 생성

rdsadmin.create_role 저장 프로시저를 사용하여 역할을 생성할 수 있습니다.

역할을 생성하려면

1. `rdsadmin` 데이터베이스에 연결합니다. 다음 예제에서 *`master_username`* 및 *`master_password`*를 사용자 정보로 대체합니다.

   ```
   db2 connect to rdsadmin user master_username using master_password
   ```

2. 콘텐츠를 출력하도록 Db2를 설정합니다.

   ```
   db2 set serveroutput on
   ```

3. 역할을 생성합니다. 자세한 내용은 the section called “rdsadmin.create_role” 단원을 참조하십시오.

```
db2 "call rdsadmin.create_role(
    'database_name',
    'role_name')"
```

4. 콘텐츠를 출력하지 않도록 Db2를 설정합니다.

```
db2 set serveroutput off
```

역할 부여

rdsadmin.grant_role 저장 프로시저를 사용하여 역할, 사용자 또는 그룹에 역할을 할당할 수 있습니다.

역할을 할당하려면

1. rdsadmin 데이터베이스에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 정보로 대체합니다.

```
db2 connect to rdsadmin user master_username using master_password
```

2. 콘텐츠를 출력하도록 Db2를 설정합니다.

```
db2 set serveroutput on
```

3. 역할을 할당합니다. 자세한 내용은 the section called “rdsadmin.grant_role” 단원을 참조하십시오.

```
db2 "call rdsadmin.grant_role(
    'database_name',
    'role_name',
    'grantee',
    'admin_option')"
```

4. 콘텐츠를 출력하지 않도록 Db2를 설정합니다.

```
db2 set serveroutput off
```

역할 취소

rdsadmin.revoke_role 저장 프로시저를 사용하여 역할, 사용자 또는 그룹에서 역할을 취소할 수 있습니다.

역할을 취소하려면 다음과 같이 하세요.

1. rdsadmin 데이터베이스에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 정보로 대체합니다.

```
db2 connect to rdsadmin user master_username using master_password
```

2. 역할을 취소합니다. 자세한 내용은 the section called “rdsadmin.revoke_role” 단원을 참조하십시오.

```
db2 "call rdsadmin.revoke_role(
    ?,
    'database_name',
    'role_name',
    'grantee')"
```

역할 삭제

??? 저장 프로시저를 사용하여 역할을 삭제할 수 있습니다.

역할을 삭제하는 방법

1. rdsadmin 데이터베이스에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 정보로 대체합니다.

```
db2 connect to rdsadmin user master_username using master_password
```

2. 역할을 삭제합니다. 자세한 내용은 ??? 단원을 참조하십시오.

```
db2 "call rdsadmin.drop_role(
    ?,
    'database_name',
    'role_name')"
```

데이터베이스 권한 부여

DBADM 권한을 가진 마스터 사용자는 역할, 사용자 또는 그룹에 DBADM, ACCESSCTRL 또는 DATAACCESS 권한을 부여할 수 있습니다.

데이터베이스에 권한을 부여하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 rdsadmin 데이터베이스에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 자체 정보로 대체합니다.

```
db2 connect to rdsadmin user master_username using master_password
```

2. rdsadmin.dbadm_grant를 호출하여 사용자에게 액세스 권한을 부여합니다. 자세한 내용은 rdsadmin.dbadm_grant 단원을 참조하십시오.

```
db2 "call rdsadmin.dbadm_grant(
    ?,
    'database_name,
    'authorization',
    'grantee')"
```

사용 사례

다음 절차는 역할 생성, 역할에 DBADM 권한 부여, 사용자에게 역할 할당 및 그룹에 역할 할당 과정을 안내합니다.

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 rdsadmin 데이터베이스에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 자체 정보로 대체합니다.

```
db2 connect to rdsadmin user master_username using master_password
```

2. TESTDB 데이터베이스에 대해 PROD_ROLE 역할을 생성합니다. 자세한 내용은 rdsadmin.create_role 단원을 참조하십시오.

```
db2 "call rdsadmin.create_role(
    'TESTDB',
    'PROD_ROLE')"
```

3. PROD_USER 사용자에게 역할을 할당합니다. PROD_USER에 역할을 할당할 수 있는 관리자 권한이 부여됩니다. 자세한 내용은 rdsadmin.grant_role 단원을 참조하십시오.

```
db2 "call rdsadmin.grant_role(
```

```
    ?,
    'TESTDB',
    'PROD_ROLE',
    'USER PROD_USER',
    'Y')"
```

4. (선택 사항) 추가 권한을 제공합니다. 다음 예제에서는 FUNDPROD 데이터베이스에 대해 PROD_ROLE 역할에 DBADM 권한을 부여합니다. 자세한 내용은 rdsadmin.dbadm_grant 단원을 참조하십시오.

```
db2 "call rdsadmin.dbadm_grant(
    ?,
    'FUNDPROD',
    'DBADM',
    'ROLE PROD_ROLE')"
```

5. 세션을 종료합니다.

```
db2 terminate
```

6. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 TESTDB 데이터베이스에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 자체 정보로 대체합니다.

```
db2 connect to TESTDB user master_username using master_password
```

7. 역할에 더 많은 권한을 추가합니다.

```
db2 "grant connect, implicit_schema on database to role PROD_ROLE"
```

8. 그룹에 PROD_ROLE 역할을 부여합니다.

```
db2 "grant role PROD_ROLE to group PRODGRP"
```

PRODGRP 그룹에 속한 사용자는 이제 TESTDB 데이터베이스 연결, 테이블 생성 또는 스키마 생성과 같은 작업을 수행할 수 있습니다.

데이터베이스 권한 취소

DBADM 권한을 가진 마스터 사용자는 역할, 사용자 또는 그룹의 DBADM, ACCESSCTRL 또는 DATAACCESS 권한을 취소할 수 있습니다.

데이터베이스 권한을 취소하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 rdsadmin 데이터베이스에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 자체 정보로 대체합니다.

```
db2 connect to rdsadmin user master_username using master_password
```

2. rdsadmin.dbadm_revoke를 호출하여 사용자 액세스 권한을 취소합니다. 자세한 내용은 rdsadmin.dbadm_revoke 단원을 참조하십시오.

```
db2 "call rdsadmin.dbadm_revoke(
    ?,
    'database_name',
    'authorization',
    'grantee')"
```

원격 RDS for Db2 DB 인스턴스에 연결

다음 단계에 따라 원격 RDS for Db2 DB 인스턴스에 연결하고 get snapshot 작업을 실행합니다.

원격 RDS for Db2 DB 인스턴스에 연결하려면

1. 클라이언트 측 IBM Db2 CLP 세션을 실행합니다. RDS for Db2 DB 인스턴스 및 데이터베이스 카탈로그화에 대한 자세한 내용은 IBM Db2 CLP을 사용하여 Amazon RDS for Db2 DB 인스턴스에 연결 섹션을 참조하세요. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 기록해 둡니다.
2. RDS for Db2 DB 인스턴스에 연결합니다. 다음 예제에서 *node_name*, *master_username*, *master_password*를 카탈로그화한 TCPIP 노드 이름과 RDS for Db2 DB 인스턴스의 마스터 사용자 이름 및 마스터 암호로 대체합니다.

```
db2 attach to node_name user master_username using master_password
```

원격 RDS for Db2 DB 인스턴스에 연결한 후 다음 명령 및 기타 get snapshot 명령을 실행할 수 있습니다. 자세한 내용은 IBM Db2 설명서의 GET SNAPSHOT 명령을 참조하세요.

```
db2 list applications
db2 get snapshot for all databases
```

```
db2 get snapshot for database manager
db2 get snapshot for all applications
```

Amazon RDS for Db2 DB 인스턴스에 대한 공통 데이터베이스 작업 수행

Amazon RDS for Db2 DB 인스턴스에서 데이터베이스와 관련된 일반적인 DBA 작업을 수행할 수 있습니다. 관리형 서비스 환경을 제공하기 위해 Amazon RDS는 DB 인스턴스에 대해 shell 액세스를 제공하지 않습니다. 또한 마스터 사용자는 SYSADM, SYSMAINT 또는 SYSCTRL 권한을 요구하는 명령이나 유틸리티를 실행할 수 없습니다.

버퍼 풀, 데이터베이스 및 테이블스페이스의 일반적인 작업에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- 버퍼 풀의 일반적인 작업
- 공통 데이터베이스 작업
- 테이블스페이스의 일반적인 작업

버퍼 풀의 일반적인 작업

RDS for Db2 데이터베이스의 버퍼 풀을 생성, 변경 또는 삭제할 수 있습니다. 버퍼 풀을 생성, 변경 또는 삭제하려면 마스터 사용자가 사용할 수 없는 더 높은 수준의 SYSADM 또는 SYSCTRL 권한이 필요합니다. 대신 Amazon RDS 저장 프로시저를 사용하세요.

버퍼 풀을 플러시할 수도 있습니다.

주제

- 버퍼 풀 생성
- 버퍼 풀 변경
- 버퍼 풀 삭제
- 버퍼 풀 플러시

버퍼 풀 생성

RDS for Db2 데이터베이스용 버퍼 풀을 만들려면 `rdsadmin.create_bufferpool` 저장 프로시저를 호출합니다. 자세한 내용은 IBM Db2 설명서의 CREATE BUFFERPOOL 문을 참조하세요.

버퍼 풀을 생성하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 rdsadmin 데이터베이스에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 자체 정보로 대체합니다.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. rdsadmin.create_bufferpool을 호출하여 버퍼 풀을 생성합니다. 자세한 내용은 rdsadmin.create_bufferpool 섹션을 참조하세요.

```
db2 "call rdsadmin.create_bufferpool(
    'database_name',
    'buffer_pool_name',
    buffer_pool_size,
    'immediate',
    'automatic',
    page_size,
    number_block_pages,
    block_size)"
```

버퍼 풀 변경

RDS for Db2 데이터베이스용 버퍼 풀을 변경하려면 rdsadmin.alter_bufferpool 저장 프로시저를 호출합니다. 자세한 내용은 IBM Db2 설명서의 ALTER BUFFERPOOL 문을 참조하세요.

버퍼 풀을 변경하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 rdsadmin 데이터베이스에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 자체 정보로 대체합니다.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. rdsadmin.alter_bufferpool을 호출하여 버퍼 풀을 변경합니다. 자세한 내용은 rdsadmin.alter_bufferpool 섹션을 참조하세요.

```
db2 "call rdsadmin.alter_bufferpool(
    'database_name',
    'buffer_pool_name',
```

```
    buffer_pool_size,
    'immediate',
    'automatic',
    change_number_blocks,
    number_block_pages,
    block_size)"
```

버퍼 풀 삭제

RDS for Db2 데이터베이스용 버퍼 풀을 삭제하려면 `rdsadmin.drop_bufferpool` 저장 프로시저를 호출합니다. 자세한 내용은 IBM Db2 설명서의 버퍼 풀 삭제를 참조하세요.

> **Important**
>
> 삭제하려는 버퍼 풀에 테이블스페이스가 할당되어 있지 않은지 확인합니다.

버퍼 풀을 삭제하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 `rdsadmin` 데이터베이스에 연결합니다. 다음 예제에서 *`master_username`* 및 *`master_password`*를 사용자 자체 정보로 대체합니다.

   ```
   db2 "connect to rdsadmin user master_username using master_password"
   ```

2. `rdsadmin.drop_bufferpool`을 호출하여 버퍼 풀을 삭제합니다. 자세한 내용은 rdsadmin.drop_bufferpool 섹션을 참조하세요.

   ```
   db2 "call rdsadmin.drop_bufferpool(
       'database_name',
       'buffer_pool_name')"
   ```

버퍼 풀 플러시

RDS for Db2가 메모리의 페이지를 스토리지에 기록하도록 버퍼 풀을 플러시하여 체크포인트를 강제 적용할 수 있습니다.

> **Note**
>
> 버퍼 풀을 플러시할 필요는 없습니다. Db2는 트랜잭션을 커밋하기 전에 로그를 동기적으로 기록합니다. 더티 페이지가 여전히 버퍼 풀에 있을 수 있지만, Db2는 이러한 페이지를 스토리지에 비동기적으로 작성합니다. 시스템이 예기치 않게 종료되더라도 데이터베이스를 다시 시작하면 Db2가 자동으로 충돌 복구를 수행합니다. 충돌 복구 중에 Db2는 커밋된 변경 내용을 데이터베이스에 쓰거나 커밋되지 않은 트랜잭션의 변경 내용을 롤백합니다.

버퍼 풀을 플러시하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 Db2 데이터베이스에 연결합니다. 다음 예제에서 *rds_database_alias*, *master_username*, *master_password*를 사용자 자체 정보로 바꿉니다.

   ```
   db2 connect to rds_database_alias user master_username using master_password
   ```

2. 버퍼 풀을 플러시합니다.

   ```
   db2 flush bufferpools all
   ```

공통 데이터베이스 작업

RDS for Db2 DB 인스턴스에서 데이터베이스를 생성, 삭제 또는 복원할 수 있습니다. 데이터베이스를 생성, 삭제 또는 복원하려면 더 높은 수준의 SYSADM 권한이 필요하며, 이 권한은 마스터 사용자가 사용할 수 없습니다. 대신 Amazon RDS 저장 프로시저를 사용하세요.

또한 모니터링, 유지 관리, 데이터베이스 정보 수집과 같은 일반적인 관리 작업을 수행할 수 있습니다.

주제

- 데이터베이스 생성
- 데이터베이스 설정 구성
- 데이터베이스 파라미터 수정
- 로그 보존 구성
- 로그 정보 나열
- 데이터베이스 비활성화

- 데이터베이스 활성화
- 데이터베이스 재활성화
- 데이터베이스 삭제
- 데이터베이스 복원
- 데이터베이스 나열
- 데이터베이스에 대한 정보 수집
- 데이터베이스의 애플리케이션 강제 종료
- 성능 보고서 생성

데이터베이스 생성

RDS for Db2 DB 인스턴스에서 데이터베이스를 만들려면 `rdsadmin.create_database` 저장 프로시저를 직접 호출합니다. 자세한 내용은 IBM Db2 설명서의 CREATE DATABASE 명령을 참조하세요.

> **Note**
>
> `db2_compatibility_vector` 파라미터를 수정하려는 경우 데이터베이스를 만들기 전에 파라미터를 수정합니다. 자세한 내용은 db2_compatibility_vector 파라미터 설정 섹션을 참조하세요.

데이터베이스를 생성하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 `rdsadmin` 데이터베이스에 연결합니다. 다음 예제에서 *`master_username`* 및 *`master_password`*를 사용자 자체 정보로 대체합니다.

   ```
   db2 "connect to rdsadmin user master_username using master_password"
   ```

2. `rdsadmin.create_database`를 직접 호출하여 데이터베이스를 생성합니다. 자세한 내용은 rdsadmin.create_database 섹션을 참조하세요.

   ```
   db2 "call rdsadmin.create_database('database_name')"
   ```

3. (선택 사항) 만들려는 각 데이터베이스에 대해 `rdsadmin.create_database`를 직접 호출하여 추가 데이터베이스를 만듭니다. 각 Db2 DB 인스턴스에는 최대 50개의 데이터베이스를 포함할 수 있습니다. 자세한 내용은 rdsadmin.create_database 섹션을 참조하세요.

```
db2 "call rdsadmin.create_database('database_name')"
```

4. (선택 사항) 다음 방법 중 하나를 사용하여 데이터베이스가 만들어졌는지 확인합니다.

   - `rdsadmin.list_databases`을 호출합니다. 자세한 내용은 rdsadmin.list_databases 섹션을 참조하세요.
   - 다음 SQL 명령을 실행합니다.

```
db2 "select varchar(r.task_type,25) as task_type, r.database_name,
    varchar(r.lifecycle,15) as lifecycle, r.created_at, r.database_name,
    varchar(bson_to_json(task_input_params),256) as input_params,
    varchar(r.task_output,1024) as task_output
    from table(rdsadmin.get_task_status(null,null,'create_database'))
    as r order by created_at desc"
```

데이터베이스 설정 구성

RDS for Db2 DB 인스턴스에서 데이터베이스용 설정을 구성하려면 `rdsadmin.set_configuration` 저장 프로시저를 직접 호출합니다. 예를 들어, 복원 작업 중에 생성할 버퍼나 버퍼 조작기 개수를 구성할 수 있습니다.

데이터베이스 설정을 구성하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 `rdsadmin` 데이터베이스에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 자체 정보로 대체합니다.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. (선택 사항) `rdsadmin.show_configuration`을 직접 호출하여 현재 구성 설정을 확인합니다. 자세한 내용은 the section called “rdsadmin.show_configuration” 섹션을 참조하세요.

```
db2 "call rdsadmin.show_configuration('name')"
```

3. `rdsadmin.set_configuration`을 직접 호출하여 데이터베이스 설정을 구성합니다. 자세한 내용은 the section called “rdsadmin.set_configuration” 섹션을 참조하세요.

```
db2 "call rdsadmin.set_configuration(
    'name',
```

```
    'value')"
```

데이터베이스 파라미터 수정

Amazon RDS for Db2는 데이터베이스 관리자 구성 파라미터, 레지스트리 변수, 데이터베이스 구성 파라미터 등 세 가지 유형의 파라미터를 사용합니다. 처음 두 유형은 파라미터 그룹을 통해 업데이트하고 마지막 유형은 rdsadmin.update_db_param 저장 프로시저를 통해 업데이트할 수 있습니다.

> **Note**
>
> 기존 파라미터 값만 수정할 수 있습니다. RDS for Db2는 지원하지 않는 파라미터 새로 추가할 수 없습니다.

이러한 파라미터 및 파라미터 값 수정 방법에 대한 자세한 내용은 the section called “Db2 파라미터” 섹션을 참조하세요.

로그 보존 구성

Amazon RDS에서 RDS for Db2 데이터베이스용 로그 파일을 보관하는 기간을 구성하려면 `rdsadmin.set_archive_log_retention` 저장 프로시저를 직접 호출합니다.

데이터베이스의 로그 보존을 구성하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 `rdsadmin` 데이터베이스에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 자체 정보로 대체합니다.

   ```
   db2 "connect to rdsadmin user master_username using master_password"
   ```

2. (선택 사항) `rdsadmin.show_archive_log_retention`을 직접 호출하여 현재 로그 보존 구성을 확인합니다. 자세한 내용은 the section called “rdsadmin.show_archive_log_retention” 섹션을 참조하세요.

   ```
   db2 "call rdsadmin.show_archive_log_retention(
       ?,
       'database_name')"
   ```

3. `rdsadmin.set_archive_log_retention`을 직접 호출하여 데이터베이스의 로그 보존을 구성합니다. 자세한 내용은 the section called “rdsadmin.set_archive_log_retention” 섹션을 참조하세요.

```
db2 "call rdsadmin.set_archive_log_retention(
    ?,
    'database_name',
    'archive_log_retention_hours')"
```

로그 정보 나열

사용된 총 스토리지 크기와 같은 세부 정보를 포함하여 아카이브 로그 파일에 대한 세부 정보를 나열하려면 `rdsadmin.list_archive_log_information` 저장 프로시저를 호출합니다.

데이터베이스에 대한 로그 정보를 나열하려면 다음과 같이 합니다.

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 `rdsadmin` 데이터베이스에 연결합니다. 다음 예제에서 *`master_username`* 및 *`master_password`*를 사용자 자체 정보로 대체합니다.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. `rdsadmin.list_archive_log_information`을 호출하여 로그 파일 정보 목록을 반환합니다. 자세한 내용은 ??? 섹션을 참조하세요.

```
db2 "call rdsadmin.list_archive_log_information(
    ?,
    'database_name')"
```

데이터베이스 비활성화

RDS for Db2 DB 인스턴스에서 데이터베이스를 비활성화하려면 `rdsadmin.deactivate_database` 저장 프로시저를 직접적으로 호출합니다.

기본적으로 Amazon RDS는 RDS for Db2 DB 인스턴스에서 데이터베이스를 만들 때 데이터베이스를 활성화합니다. 자주 사용하지 않는 데이터베이스를 비활성화하여 메모리 리소스를 절약할 수 있습니다.

데이터베이스를 비활성화하는 방법

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 rdsadmin 데이터베이스에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 자체 정보로 대체합니다.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. rdsadmin.deactivate_database를 직접 호출하여 데이터베이스를 비활성화합니다. 자세한 내용은 ??? 섹션을 참조하세요.

```
db2 "call rdsadmin.deactivate_database(
    ?,
    'database_name')"
```

데이터베이스 활성화

독립형 RDS for Db2 DB 인스턴스에서 데이터베이스를 활성화하려면 rdsadmin.activate_database 저장 프로시저를 직접적으로 호출합니다.

기본적으로 Amazon RDS는 RDS for Db2 DB 인스턴스에서 데이터베이스를 만들 때 데이터베이스를 활성화합니다. 자주 사용하지 않는 데이터베이스를 비활성화하여 메모리 리소스를 보존한 다음 나중에 비활성화된 데이터베이스를 활성화할 수 있습니다.

데이터베이스를 활성화하는 방법

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 rdsadmin 데이터베이스에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 자체 정보로 대체합니다.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. rdsadmin.activate_database를 직접 호출하여 데이터베이스를 활성화합니다. 자세한 내용은 ??? 섹션을 참조하세요.

```
db2 "call rdsadmin.activate_database(
    ?,
    'database_name')"
```

데이터베이스 재활성화

데이터베이스를 재활성화하려면 `rdsadmin.reactivate_database` 저장 프로시저를 직접적으로 호출하세요. 데이터베이스 구성을 변경한 후 RDS for Db2 DB 인스턴스에서 데이터베이스를 재활성화해야 할 수 있습니다. 데이터베이스를 재활성화해야 하는지 확인하려면 데이터베이스에 연결하고 `db2 get db cfg show detail`을 실행합니다.

데이터베이스 구성을 변경한 후 이 저장 프로시저를 직접적으로 호출하여 독립형 RDS for Db2 DB 인스턴스에서 데이터베이스를 재활성화할 수도 있습니다. 또는 먼저 `rdsadmin.deactivate_database` 저장 프로시저를 직접적으로 호출한 다음 `rdsadmin.activate_database` 저장 프로시저를 직접적으로 호출하여 독립형 RDS for Db2 DB 인스턴스에서 데이터베이스를 재활성화할 수 있습니다. 자세한 내용은 데이터베이스 비활성화 및 데이터베이스 활성화(을)를 참조하세요.

데이터베이스를 재활성화하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 `rdsadmin` 데이터베이스에 연결합니다. 다음 예제에서 *`master_username`* 및 *`master_password`*를 사용자 자체 정보로 대체합니다.

   ```
   db2 "connect to rdsadmin user master_username using master_password"
   ```

2. `rdsadmin.reactivate_database`를 직접적으로 호출하여 데이터베이스를 재활성화합니다. 자세한 내용은 ??? 섹션을 참조하세요.

   ```
   db2 "call rdsadmin.reactivate_database(
       ?,
       'database_name')"
   ```

데이터베이스 삭제

RDS for Db2 DB 인스턴스에서 데이터베이스를 삭제하려면 `rdsadmin.drop_database` 저장 프로시저를 직접 호출합니다. 자세한 내용은 IBM Db2 설명서의 데이터베이스 삭제를 참조하세요.

> **Note**
>
> 특정 조건이 충족되는 경우에만 저장 프로시저를 직접 호출하여 데이터베이스를 삭제할 수 있습니다. 자세한 내용은 the section called “사용 노트”rdsadmin.drop_database용 를 참조하십시오.

데이터베이스를 삭제하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 rdsadmin 데이터베이스에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 자체 정보로 대체합니다.

   ```
   db2 "connect to rdsadmin user master_username using master_password"
   ```

2. rdsadmin.drop_database를 직접 호출하여 데이터베이스를 삭제합니다. 자세한 내용은 rdsadmin.drop_database 섹션을 참조하세요.

   ```
   db2 "call rdsadmin.drop_database('database_name')"
   ```

데이터베이스 복원

Amazon S3 버킷에서 RDS for Db2 DB 인스턴스로 데이터베이스를 이동하려면 rdsadmin.restore_database 저장 프로시저를 직접 호출합니다. 자세한 내용은 IBM Db2 설명서의 RESTORE DATABASE 명령을 참조하세요.

데이터베이스를 복원하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 rdsadmin 데이터베이스에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 자체 정보로 대체합니다.

   ```
   db2 "connect to rdsadmin user master_username using master_password"
   ```

2. (선택 사항) rdsadmin.show_configuration을 직접 호출하고 현재 구성 설정을 확인하여 복원 작업을 최적화합니다. 자세한 내용은 the section called “rdsadmin.show_configuration” 섹션을 참조하세요.

```
db2 "call rdsadmin.show_configuration('name')"
```

3. rdsadmin.set_configuration을 직접 호출하여 복원 작업을 최적화하도록 설정을 구성합니다. 이러한 값을 명시적으로 설정하면 대량 데이터가 있는 데이터베이스를 복원할 때 성능이 향상될 수 있습니다. 자세한 내용은 the section called “rdsadmin.set_configuration” 섹션을 참조하세요.

```
db2 "call rdsadmin.set_configuration(
    'name',
    'value')"
```

4. rdsadmin.restore_database를 호출하여 데이터베이스를 복원합니다. 자세한 내용은 the section called “rdsadmin.restore_database” 섹션을 참조하세요.

```
db2 "call rdsadmin.restore_database(
    ?,
    'database_name',
    's3_bucket_name',
    's3_prefix',
    restore_timestamp,
    'backup_type')"
```

5. (선택 사항) rdsadmin.list_databases를 직접 호출하고 복원된 데이터베이스가 나열되었는지 확인하여 데이터베이스가 복원되었는지 확인합니다. 자세한 내용은 rdsadmin.list_databases 섹션을 참조하세요.

6. rdsadmin.rollforward_database를 직접 호출하여 데이터베이스를 다시 온라인 상태로 만들고 추가 트랜잭션 로그를 적용합니다. 자세한 내용은 the section called “rdsadmin.rollforward_database” 섹션을 참조하세요.

```
db2 "call rdsadmin.rollforward_database(
    ?,
    'database_name',
    's3_bucket_name',
    s3_prefix,
    'rollforward_to_option',
    'complete_rollforward')"
```

7. 이전 단계에서 complete_rollforward를 FALSE로 설정한 경우에는 rdsadmin.complete_rollforward를 직접 호출하여 데이터베이스를 다시 온

라인 상태로 전환하는 작업을 완료해야 합니다. 자세한 내용은 the section called "rdsadmin.complete_rollforward" 섹션을 참조하세요.

```
db2 "call rdsadmin.complete_rollforward(
    ?,
    'database_name')"
```

데이터베이스 나열

rdsadmin.list_databases 사용자 정의 함수를 직접 호출하여 Amazon RDS for Db2에서 실행되는 모든 데이터베이스를 나열할 수 있습니다.

데이터베이스를 나열하는 방법

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 rdsadmin 데이터베이스에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 자체 정보로 대체합니다.

   ```
   db2 "connect to rdsadmin user master_username using master_password"
   ```

2. rdsadmin.list_databases를 직접 호출하여 데이터베이스를 나열합니다. 자세한 내용은 rdsadmin.list_databases 섹션을 참조하세요.

   ```
   db2 "select * from table(rdsadmin.list_databases())"
   ```

데이터베이스에 대한 정보 수집

RDS for Db2 DB 인스턴스에서 데이터베이스에 대한 정보를 수집하려면 rdsadmin.db2pd_command 저장 프로시저를 직접적으로 호출합니다. 이 정보를 사용하면 데이터베이스를 모니터링하거나 문제를 해결할 수 있습니다.

데이터베이스에 대한 정보를 수집하려면 다음과 같이 하세요.

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 rdsadmin 데이터베이스에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 자체 정보로 대체합니다.

   ```
   db2 "connect to rdsadmin user master_username using master_password"
   ```

2. `rdsadmin.db2pd_command`를 직접 호출하여 데이터베이스에 대한 정보를 수집합니다. 자세한 내용은 rdsadmin.db2pd_command 섹션을 참조하세요.

```
db2 "call rdsadmin.db2pd_command('db2pd_cmd')"
```

데이터베이스의 애플리케이션 강제 종료

RDS for Db2 인스턴스의 데이터베이스에서 애플리케이션을 강제 종료하려면 `rdsadmin.force_application` 저장 프로시저를 직접적으로 호출합니다. 데이터베이스 유지 관리를 수행하기 전에 데이터베이스에서 애플리케이션을 강제로 제거해야 합니다.

데이터베이스의 애플리케이션을 강제 종료하려면 다음과 같이 하세요.

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 `rdsadmin` 데이터베이스에 연결합니다. 다음 예제에서 *`master_username`* 및 *`master_password`*를 사용자 자체 정보로 대체합니다.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. `rdsadmin.force_application`을 직접적으로 호출하여 데이터베이스의 애플리케이션을 강제 종료합니다. 자세한 내용은 rdsadmin.force_application 섹션을 참조하세요.

```
db2 "call rdsadmin.force_application(
    ?,
    'applications')"
```

성능 보고서 생성

프로시저 또는 스크립트를 사용하여 성능 보고서를 생성할 수 있습니다. 프로시저 사용에 대한 자세한 내용은 IBM Db2 설명서의 DBSUMMARY 프로시저 - 시스템 및 애플리케이션 성능 지표에 대한 요약 보고서 생성을 참조하세요.

Db2는 `~sqllib/sample/perf` 디렉터리에 `db2mon.sh` 파일을 포함합니다. 스크립트를 실행하면 합리적인 비용으로 광범위한 SQL 지표 보고서를 생성할 수 있습니다. `db2mon.sh` 파일 및 관련 스크립트 파일을 다운로드하려면 IBM db2-samples GitHub 리포지토리의 perf 디렉터리를 참조하세요.

스크립트를 사용하여 성능 보고서를 생성하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 Db2 데이터베이스에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 자체 정보로 대체합니다.

```
db2 connect to rdsadmin user master_username using master_password
```

2. `rdsadmin.create_bufferpool`을 호출하여 페이지 크기가 4096인 버퍼 풀 `db2monbp`를 생성합니다. 자세한 내용은 rdsadmin.create_bufferpool 섹션을 참조하세요.

```
db2 "call rdsadmin.create_bufferpool('database_name','db2monbp',4096)"
```

3. `rdsadmin.create_tablespace`을 호출하여 `db2monbp` 버퍼 풀을 사용하는 임시 테이블스페이스 `db2montmptbsp`를 생성합니다. 자세한 내용은 rdsadmin.create_tablespace 섹션을 참조하세요.

```
db2 "call rdsadmin.create_tablespace('database_name',\
    'db2montmptbsp','db2monbp',4096,1000,100,'T')"
```

4. `db2mon.sh` 스크립트를 열고 데이터베이스 연결에 대한 줄을 수정합니다.

   a. 다음과 같은 줄을 제거합니다.

```
db2 -v connect to $dbName
```

   b. 이전 단계의 줄을 다음 줄로 바꿉니다. 다음 예제에서 *master_username* 및 *master_password*를 RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호로 대체합니다.

```
db2 -v connect to $dbName user master_username using master_password
```

   c. 다음과 같은 줄을 제거합니다.

```
db2 -v create bufferpool db2monbp

db2 -v create user temporary tablespace db2montmptbsp bufferpool db2monbp

db2 -v drop tablespace db2montmptbsp
```

```
db2 -v drop bufferpool db2monbp
```

5. db2mon.sh 스크립트를 실행하여 지정된 간격으로 보고서를 출력합니다. 다음 예제에서 *absolute_path*는 스크립트 파일의 전체 경로로, *rds_database_alias*는 데이터베이스 이름으로, *seconds*는 보고서 생성 사이의 시간(0~3,600초)으로 바꿉니다.

```
absolute_path/db2mon.sh rds_database_alias seconds | tee -a db2mon.out
```

예시

다음 예제는 스크립트 파일이 home 디렉터리 아래의 perf 디렉터리에 있음을 보여줍니다.

```
/home/db2inst1/sqllib/samples/perf/db2mon.sh rds_database_alias seconds | tee -a
 db2mon.out
```

6. 버퍼 풀과 db2mon.sh 파일용으로 생성된 테이블스페이스를 삭제합니다. 다음 예제에서 *master_username* 및 *master_password*를 RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호로 대체합니다. *database_name*을 데이터베이스의 이름으로 바꿉니다. 자세한 내용은 rdsadmin.drop_tablespace 및 rdsadmin.drop_bufferpool(을)를 참조하세요.

```
db2 connect to rdsadmin user master_username using master_password

db2 "call rdsadmin.drop_tablespace('database_name','db2montmptbsp')"

db2 "call rdsadmin.drop_bufferpool('database_name','db2monbp')"
```

스토리지 관리

Db2는 자동 스토리지를 사용하여 표, 인덱스, 임시 파일과 같은 데이터베이스 객체의 물리적 스토리지를 관리합니다. 스토리지 공간을 수동으로 할당하고 사용 중인 스토리지 경로를 추적하는 대신, 자동 스토리지를 통해 Db2 시스템은 필요에 따라 스토리지 경로를 생성하고 관리할 수 있습니다. 이를 통해 Db2 데이터베이스 관리를 단순화하고 사람의 실수로 인한 오류 가능성을 줄일 수 있습니다. 자세한 내용은 IBM Db2 설명서의 자동 스토리지를 참조하세요.

RDS for Db2를 사용하면 논리적 볼륨과 파일 시스템을 자동으로 확장하여 스토리지 크기를 동적으로 늘릴 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 작업 단원을 참조하십시오.

테이블스페이스의 일반적인 작업

RDS for Db2 데이터베이스의 테이블스페이스를 생성, 변경, 삭제 또는 이름을 변경할 수 있습니다. 테이블스페이스를 생성, 변경, 삭제 또는 이름을 변경하려면 마스터 사용자가 사용할 수 없는 더 높은 수준의 SYSADM 권한이 필요합니다. 대신 Amazon RDS 저장 프로시저를 사용하세요.

주제

- 테이블스페이스 생성
- 테이블스페이스 변경
- 테이블스페이스 이름 변경
- 테이블스페이스 삭제
- 테이블스페이스의 상태 확인
- 테이블스페이스에 관한 세부 정보 반환
- 테이블스페이스의 상태 및 스토리지 그룹 나열
- 표의 테이블스페이스 나열
- 테이블스페이스 컨테이너 나열

테이블스페이스 생성

RDS for Db2 데이터베이스용 테이블스페이스를 만들려면 `rdsadmin.create_tablespace` 저장 프로시저를 호출합니다. 자세한 내용은 IBM Db2 설명서의 CREATE TABLESPACE 문을 참조하세요.

> **Important**
>
> 테이블스페이스를 만들려면 테이블스페이스와 연결할 페이지 크기가 같은 버퍼 풀이 있어야 합니다. 자세한 내용은 버퍼 풀의 일반적인 작업 단원을 참조하십시오.

테이블스페이스를 생성하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 `rdsadmin` 데이터베이스에 연결합니다. 다음 예제에서 *`master_username`* 및 *`master_password`*를 사용자 자체 정보로 대체합니다.

   ```
   db2 "connect to rdsadmin user master_username using master_password"
   ```

2. `rdsadmin.create_tablespace`를 호출하여 테이블스페이스를 생성합니다. 자세한 내용은 rdsadmin.create_tablespace 단원을 참조하십시오.

```
db2 "call rdsadmin.create_tablespace(
    'database_name',
    'tablespace_name',
    'buffer_pool_name',
    tablespace_initial_size,
    tablespace_increase_size,
    'tablespace_type')"
```

테이블스페이스 변경

RDS for Db2 데이터베이스용 테이블스페이스를 변경하려면 `rdsadmin.alter_tablespace` 저장 프로시저를 호출합니다. 이 저장 프로시저를 사용하여 테이블스페이스의 버퍼 풀을 변경하거나, 하이 워터 마크를 낮추거나, 테이블스페이스를 온라인으로 전환할 수 있습니다. 자세한 내용은 IBM Db2 설명서의 ALTER TABLESPACE 문을 참조하세요.

테이블스페이스를 변경하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 `rdsadmin` 데이터베이스에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 자체 정보로 대체합니다.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. `rdsadmin.alter_tablespace`를 호출하여 테이블스페이스를 변경합니다. 자세한 내용은 rdsadmin.alter_tablespace 단원을 참조하십시오.

```
db2 "call rdsadmin.alter_tablespace(
    'database_name',
    'tablespace_name',
    'buffer_pool_name',
    buffer_pool_size,
    tablespace_increase_size,
    'max_size', 'reduce_max',
    'reduce_stop',
    'reduce_value',
    'lower_high_water',
    'lower_high_water_stop',
```

```
    'switch_online')"
```

테이블스페이스 이름 변경

RDS for Db2 데이터베이스용 테이블스페이스의 이름을 변경하려면 `rdsadmin.rename_tablespace` 저장 프로시저를 호출합니다. 자세한 내용은 IBM Db2 설명서의 RENAME TABLESPACE 문을 참조하세요.

테이블스페이스 이름을 변경하는 방법

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 `rdsadmin` 데이터베이스에 연결합니다. 다음 예제에서 *`master_username`* 및 *`master_password`*를 사용자 자체 정보로 대체합니다.

   ```
   db2 "connect to rdsadmin user master_username using master_password"
   ```

2. `rdsadmin.rename_tablespace`를 직접 호출하여 테이블스페이스의 이름을 변경합니다. 테이블스페이스의 이름으로 지정할 수 있는 것에 대한 제한 사항을 비롯한 자세한 내용은 rdsadmin.rename_tablespace 섹션을 참조하세요.

   ```
   db2 "call rdsadmin.rename_tablespace(
       'database_name',
       'source_tablespace_name',
       'target_tablespace_name')"
   ```

테이블스페이스 삭제

RDS for Db2 데이터베이스용 테이블스페이스를 삭제하려면 `rdsadmin.drop_tablespace` 저장 프로시저를 호출합니다. 테이블스페이스를 삭제하기 전에 먼저 표, 인덱스 또는 대형 객체(LOB)와 같은 테이블스페이스의 모든 객체를 삭제해야 합니다. 자세한 내용은 IBM Db2 설명서의 테이블스페이스 삭제를 참조하세요.

테이블스페이스를 삭제하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 `rdsadmin` 데이터베이스에 연결합니다. 다음 예제에서 *`master_username`* 및 *`master_password`*를 사용자 자체 정보로 대체합니다.

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. rdsadmin.drop_tablespace를 호출하여 테이블스페이스를 삭제합니다. 자세한 내용은 rdsadmin.drop_tablespace 단원을 참조하십시오.

```
db2 "call rdsadmin.drop_tablespace(
    'database_name',
    'tablespace_name')"
```

테이블스페이스의 상태 확인

cast 함수를 사용하여 테이블스페이스의 상태를 확인할 수 있습니다.

테이블스페이스의 상태를 확인하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 Db2 데이터베이스에 연결합니다. 다음 예제에서 *rds_database_alias*, *master_username*, *master_password*를 사용자 자체 정보로 바꿉니다.

```
db2 connect to rds_database_alias user master_username using master_password
```

2. 요약 출력을 반환합니다.

   요약 출력은 다음과 같습니다.

```
db2 "select cast(tbsp_id as smallint) as tbsp_id,
cast(tbsp_name as varchar(35)) as tbsp_name,
cast(tbsp_type as varchar(3)) as tbsp_type,
cast(tbsp_state as varchar(10)) as state,
cast(tbsp_content_type as varchar(8)) as contents from
 table(mon_get_tablespace(null,-1)) order by tbsp_id"
```

테이블스페이스에 관한 세부 정보 반환

cast 함수를 사용하여 한 구성원 또는 모든 구성원의 테이블스페이스 정보를 반환할 수 있습니다.

테이블스페이스에 관한 세부 정보를 반환하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 Db2 데이터베이스에 연결합니다. 다음 예제에서 *rds_database_alias*, *master_username*, *master_password*를 사용자 자체 정보로 바꿉니다.

   ```
   db2 connect to rds_database_alias user master_username using master_password
   ```

2. 한 멤버 또는 모든 멤버의 데이터베이스에 있는 모든 테이블스페이스에 대한 세부 정보를 반환합니다.

   한 멤버의 경우에는 다음과 같습니다.

   ```
   db2 "select cast(member as smallint) as member,
   cast(tbsp_id as smallint) as tbsp_id,
   cast(tbsp_name as varchar(35)) as tbsp_name,
   cast(tbsp_type as varchar(3)) as tbsp_type,
   cast(tbsp_state as varchar(10)) as state,
   cast(tbsp_content_type as varchar(8)) as contents,
   cast(tbsp_total_pages as integer) as total_pages,
   cast(tbsp_used_pages as integer) as used_pages,
   cast(tbsp_free_pages as integer) as free_pages,
   cast(tbsp_page_top as integer) as page_hwm,
   cast(tbsp_page_size as integer) as page_sz,
   cast(tbsp_extent_size as smallint) as extent_sz,
   cast(tbsp_prefetch_size as smallint) as prefetch_sz,
   cast(tbsp_initial_size as integer) as initial_size,
   cast(tbsp_increase_size_percent as smallint) as increase_pct,
   cast(storage_group_name as varchar(12)) as stogroup from
    table(mon_get_tablespace(null,-1)) order by member, tbsp_id "
   ```

   모든 멤버의 경우에는 다음과 같습니다.

   ```
   db2 "select cast(member as smallint) as member
   cast(tbsp_id as smallint) as tbsp_id,
   cast(tbsp_name as varchar(35)) as tbsp_name,
   cast(tbsp_type as varchar(3)) as tbsp_type,
   cast(tbsp_state as varchar(10)) as state,
   cast(tbsp_content_type as varchar(8)) as contents,
   cast(tbsp_total_pages as integer) as total_pages,
   cast(tbsp_used_pages as integer) as used_pages,
   cast(tbsp_free_pages as integer) as free_pages,
   ```

```
cast(tbsp_page_top as integer) as page_hwm,
cast(tbsp_page_size as integer) as page_sz,
cast(tbsp_extent_size as smallint) as extent_sz,
cast(tbsp_prefetch_size as smallint) as prefetch_sz,
cast(tbsp_initial_size as integer) as initial_size,
cast(tbsp_increase_size_percent as smallint) as increase_pct,
cast(storage_group_name as varchar(12)) as stogroup from
 table(mon_get_tablespace(null,-2)) order by member, tbsp_id "
```

테이블스페이스의 상태 및 스토리지 그룹 나열

SQL 문을 실행하여 테이블스페이스의 상태 및 스토리지 그룹을 나열할 수 있습니다.

테이블스페이스의 상태 및 스토리지 그룹을 나열하려면 다음 SQL 문을 실행합니다.

```
db2 "SELECT varchar(tbsp_name, 30) as tbsp_name,
                  varchar(TBSP_STATE, 30) state,
                  tbsp_type,
                  varchar(storage_group_name,30) storage_group
FROM TABLE(MON_GET_TABLESPACE('',-2)) AS t"
```

표의 테이블스페이스 나열

SQL 문을 실행하여 표의 테이블스페이스를 나열할 수 있습니다.

표의 테이블스페이스를 나열하려면 다음 SQL 문을 실행합니다. 다음 예제에서 *SCHEMA_NAME* 및 *TABLE_NAME*을 다음과 같이 스키마 및 테이블 이름으로 바꿉니다.

```
db2 "SELECT
   VARCHAR(SD.TBSPACE,30) AS DATA_SPACE,
   VARCHAR(SL.TBSPACE,30) AS LONG_SPACE,
   VARCHAR(SI.TBSPACE,30) AS INDEX_SPACE
 FROM
   SYSCAT.DATAPARTITIONS P
   JOIN SYSCAT.TABLESPACES SD ON SD.TBSPACEID = P.TBSPACEID
   LEFT JOIN SYSCAT.TABLESPACES SL ON SL.TBSPACEID = P.LONG_TBSPACEID
   LEFT JOIN SYSCAT.TABLESPACES SI ON SI.TBSPACEID = P.INDEX_TBSPACEID
 WHERE
    TABSCHEMA = 'SCHEMA_NAME'
 AND TABNAME   = 'TABLE_NAME'"
```

테이블스페이스 컨테이너 나열

cast 명령을 사용하여 모든 테이블스페이스 컨테이너 또는 특정 테이블스페이스 컨테이너를 나열할 수 있습니다.

테이블스페이스의 테이블스페이스 컨테이너를 나열하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 Db2 데이터베이스에 연결합니다. 다음 예제에서 *rds_database_alias*, *master_username*, *master_password*를 자체 정보로 바꿉니다.

   ```
   db2 connect to rds_database_alias user master_username using master_password
   ```

2. 데이터베이스 또는 특정 테이블스페이스 컨테이너의 모든 테이블스페이스 컨테이너 목록을 반환합니다.

   모든 테이블스페이스 컨테이너의 경우에는 다음과 같습니다.

   ```
   db2 "select cast(member as smallint) as member,
   cast(tbsp_name as varchar(35)) as tbsp_name,
   cast(container_id as smallint) as id,
   cast(container_name as varchar(60)) as container_path, container_type as type from
    table(mon_get_container(null,-2)) order by member,tbsp_id,container_id"
   ```

   특정 테이블스페이스 컨테이너의 경우에는 다음과 같습니다.

   ```
   db2 "select cast(member as smallint) as member,
   cast(tbsp_name as varchar(35)) as tbsp_name,
   cast(container_id as smallint) as id,
   cast(container_name as varchar(60)) as container_path, container_type as type from
    table(mon_get_container('TBSP_1',-2)) order by member, tbsp_id,container_id"
   ```

Amazon RDS for Db2 DB 인스턴스와 Amazon S3 통합

Amazon RDS 저장 프로시저를 사용하는 Amazon Simple Storage Service(S3) 버킷과 Amazon RDS for Db2 DB 인스턴스 간에 파일을 전송할 수 있습니다. 자세한 내용은 Amazon RDS for Db2 저장 프로시저 참조 단원을 참조하십시오.

> **Note**
>
> DB 인스턴스와 Amazon S3 버킷은 같은 AWS 리전에 있어야 합니다.

RDS for Db2를 Amazon S3와 통합하려면 DB 인스턴스가 RDS for Db2가 있는 Amazon S3 버킷에 액세스할 수 있어야 합니다. 현재 S3 버킷이 없는 경우에는 버킷을 생성하세요.

주제

- 1단계: IAM 정책 생성
- 2단계: IAM 역할 생성 및 IAM 정책 연결
- 3단계: RDS for Db2 DB 인스턴스에 IAM 역할 추가

1단계: IAM 정책 생성

이 단계에서는 Amazon S3 버킷에서 RDS DB 인스턴스로 파일을 전송하는 데 필요한 권한을 가진 AWS Identity and Access Management(IAM) 정책을 생성합니다. 이 단계에서는 이미 S3 버킷을 생성한 것으로 가정합니다. 자세한 내용은 Amazon S3 사용자 안내서의 버킷 생성을 참조하세요.

정책을 생성하기 전에 다음 정보를 기록해 둡니다.

- 버킷의 Amazon 리소스 이름(ARN)
- 버킷에서 SSE-KMS 또는 SSE-S3 암호화를 사용하는 경우 AWS Key Management Service(AWS KMS) 키에 대한 ARN.

다음 권한을 포함하는 IAM 정책을 생성합니다.

```
"kms:GenerateDataKey",
"kms:Decrypt",
"s3:PutObject",
"s3:GetObject",
"s3:AbortMultipartUpload",
```

```
"s3:ListBucket",
"s3:DeleteObject",
"s3:GetObjectVersion",
"s3:ListMultipartUploadParts"
```

AWS Management Console 또는 AWS Command Line Interface(AWS CLI)를 사용하여 IAM 정책을 생성할 수 있습니다.

콘솔

Amazon S3 버킷에 Amazon RDS 액세스를 허용하는 IAM 정책을 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 정책을 선택합니다.
3. 정책 생성을 선택한 다음 JSON을 선택합니다.
4. 서비스별 작업을 추가합니다. Amazon S3 버킷에서 Amazon RDS로 파일을 전송하려면 버킷 권한 및 객체 권한을 선택해야 합니다.
5. 리소스(Resources)를 확장합니다. 버킷과 객체 리소스를 지정해야 합니다.
6. Next(다음)를 선택합니다.
7. 정책 이름에 이 정책의 이름을 입력합니다.
8. (선택 사항) 설명에 이 정책에 대한 설명을 입력합니다.
9. 정책 생성을 선택합니다.

AWS CLI

Amazon S3 버킷에 Amazon RDS 액세스를 허용하는 IAM 정책을 생성하려면

1. create-policy 명령을 실행합니다. 다음 예제에서 *iam_policy_name* 및 *s3_bucket_name*을 IAM 정책 이름과 RDS for Db2 데이터베이스가 있는 Amazon S3 버킷 이름으로 대체합니다.

   대상 LinuxmacOS, 또는Unix:

   ```
   aws iam create-policy \
       --policy-name iam_policy_name \
       --policy-document '{
         "Version": "2012-10-17",
         "Statement": [
   ```

```
        {
          "Effect": "Allow",
          "Action": [
            "kms:GenerateDataKey",
            "kms:Decrypt",
            "s3:PutObject",
            "s3:GetObject",
            "s3:AbortMultipartUpload",
            "s3:ListBucket",
            "s3:DeleteObject",
            "s3:GetObjectVersion",
            "s3:ListMultipartUploadParts"
          ],
          "Resource": [
            "arn:aws:s3:::s3_bucket_name/*",
            "arn:aws:s3:::s3_bucket_name"
          ]
        }
      ]
    }'
```

Windows의 경우:

```
aws iam create-policy ^
    --policy-name iam_policy_name ^
    --policy-document '{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
            "Action": [
              "s3:PutObject",
              "s3:GetObject",
              "s3:AbortMultipartUpload",
              "s3:ListBucket",
              "s3:DeleteObject",
              "s3:GetObjectVersion",
              "s3:ListMultipartUploadParts"
            ],
            "Resource": [
              "arn:aws:s3:::s3_bucket_name/*",
              "arn:aws:s3:::s3_bucket_name"
            ]
```

```
        }
      ]
    }'
```

2. 정책이 생성되면 정책의 ARN을 기록해 둡니다. 2단계: IAM 역할 생성 및 IAM 정책 연결에 대한 ARN이 필요합니다.

IAM 정책 생성에 대한 자세한 내용은 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

2단계: IAM 역할 생성 및 IAM 정책 연결

이 단계에서는 1단계: IAM 정책 생성에서 IAM 정책을 생성했다고 가정합니다. 이 단계에서는 RDS for Db2 DB 인스턴스에 대한 IAM 역할을 만든 다음 역할에 IAM 정책을 연결합니다.

AWS Management Console 또는 AWS CLI를 사용하여 DB 인스턴스에 대한 IAM 역할을 만들 수 있습니다.

콘솔

IAM 역할을 생성하여 여기에 IAM 정책을 연결하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할을 선택합니다.
3. 역할 생성을 선택합니다.
4. 신뢰할 수 있는 엔터티 유형에 AWS 서비스를 선택합니다.
5. 서비스 또는 사용 사례의 경우 RDS를 선택한 다음 RDS – 데이터베이스에 역할 추가를 선택합니다.
6. Next(다음)를 선택합니다.
7. 권한 정책의 경우 생성한 IAM 정책의 이름을 검색하여 선택합니다.
8. Next(다음)를 선택합니다.
9. 역할 이름(Role name)에 역할 이름을 입력합니다.
10. (선택 사항)설명에 새 역할에 대한 설명을 입력합니다.
11. 역할 생성을 선택합니다.

AWS CLI

IAM 역할을 생성하여 여기에 IAM 정책을 연결하려면

1. create-role 명령을 실행합니다. 다음 예제에서 *iam_role_name*을 IAM 역할 이름으로 바꿉니다.

   대상 LinuxmacOS, 또는Unix:

```
aws iam create-role \
    --role-name iam_role_name \
    --assume-role-policy-document '{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "rds.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }'
```

   Windows의 경우:

```
aws iam create-role ^
    --role-name iam_role_name ^
    --assume-role-policy-document '{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "rds.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }'
```

2. 역할이 생성되면 역할의 ARN을 기록하십시오. 3단계: RDS for Db2 DB 인스턴스에 IAM 역할 추가에 대한 ARN이 필요합니다.

3. attach-role-policy 명령을 실행합니다. 다음 예제에서 *iam_policy_arn*을 1단계: IAM 정책 생성에서 생성한 IAM 정책의 ARN으로 대체합니다. *iam_role_name*을 방금 생성한 IAM 역할 이름으로 바꿉니다.

   대상 LinuxmacOS, 또는Unix:

```
aws iam attach-role-policy \
   --policy-arn iam_policy_arn \
   --role-name iam_role_name
```

   Windows의 경우:

```
aws iam attach-role-policy ^
   --policy-arn iam_policy_arn ^
   --role-name iam_role_name
```

자세한 내용은 IAM 사용 설명서의 IAM 사용자에게 권한을 위임하기 위한 역할 생성을 참조하십시오.

3단계: RDS for Db2 DB 인스턴스에 IAM 역할 추가

이 단계에서는 IAM 역할을 RDS for Db2 DB 인스턴스에 추가합니다. 다음과 같은 요구 사항을 확인합니다.

- 필수 Amazon S3 권한 정책이 연결된 IAM 역할에 대한 액세스 권한이 있어야 합니다.
- 한 번에 하나의 IAM 역할만 RDS for Db2 DB 인스턴스에 연결할 수 있습니다.
- RDS for Db2 DB 인스턴스는 사용 가능 상태여야 합니다.

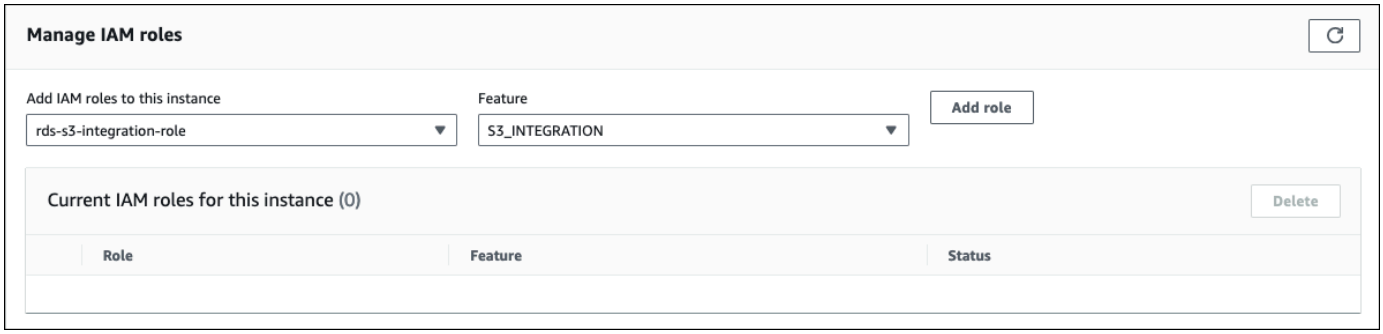
AWS Management Console 또는 AWS CLI를 사용하여 DB 인스턴스에 IAM 역할을 추가할 수 있습니다.

콘솔

IAM 역할을 RDS for Db2 DB 인스턴스에 추가하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
3. RDS for Db2 DB 인스턴스 이름을 선택합니다.

4. 연결성 및 보안(Connectivity & security) 탭에서 페이지 하단의 IAM 역할 관리(Manage IAM roles) 섹션이 나올 때까지 아래로 스크롤합니다.
5. IAM 역할을 이 인스턴스에 추가에는 2단계: IAM 역할 생성 및 IAM 정책 연결에서 생성한 역할을 선택합니다.
6. 기능에서 S3_INTEGRATION을 선택하십시오.
7. [Add role]을 선택합니다.



AWS CLI

RDS for Db2 DB 인스턴스에 IAM 역할을 추가하려면 add-role-to-db-instance 명령을 실행합니다. 다음 예제에서 *db_instance_name* 및 *iam_role_arn*을 DB 인스턴스의 이름과 2단계: IAM 역할 생성 및 IAM 정책 연결에서 생성한 IAM 역할의 ARN으로 대체합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds add-role-to-db-instance \
    --db-instance-identifier db_instance_name \
    --feature-name S3_INTEGRATION \
    --role-arn iam_role_arn \
```

Windows의 경우:

```
aws rds add-role-to-db-instance ^
    --db-instance-identifier db_instance_name ^
    --feature-name S3_INTEGRATION ^
    --role-arn iam_role_arn ^
```

RDS for Db2 DB 인스턴스에 역할이 성공적으로 추가되었는지 확인하려면 describe-db-instances 명령을 실행합니다. 다음 예제에서 *db_instance_name*을 DB 인스턴스 이름으로 바꿉니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-instances \
    --filters "Name=db-instance-id,Values=db_instance_name" \
    --query 'DBInstances[].AssociatedRoles'
```

Windows의 경우:

```
aws rds describe-db-instances ^
    --filters "Name=db-instance-id,Values=db_instance_name" ^
    --query 'DBInstances[].AssociatedRoles'
```

다음 예제와 비슷한 출력이 생성됩니다.

```
[
    [
        {
            "RoleArn": "arn:aws:iam::0123456789012:role/rds-db2-s3-role",
            "FeatureName": "S3_INTEGRATION",
            "Status": "ACTIVE"
        }
    ]
]
```

Amazon RDS for Db2로 데이터 마이그레이션

AWS 또는 기본 Db2 도구를 사용하여 자체 관리형 Db2 데이터베이스를 Amazon RDS for Db2로 마이그레이션할 수 있습니다.

AWS 서비스를 사용하여 Db2 데이터베이스에서 Amazon RDS for Db2로 마이그레이션하는 방법에 대한 자세한 내용은 AWS 서비스를 사용하여 Db2에서 Amazon RDS for Db2로 데이터 마이그레이션 섹션을 참조하세요.

기본 Db2 도구를 사용하여 Db2 데이터베이스에서 Amazon RDS for Db2로 마이그레이션하는 방법에 대한 자세한 내용은 기본 Db2 도구를 사용하여 Db2에서 Amazon RDS for Db2로 데이터 마이그레이션 섹션을 참조하세요.

AWS 서비스를 사용하여 Db2에서 Amazon RDS for Db2로 데이터 마이그레이션

Amazon RDS에서 Db2 데이터베이스의 데이터를 Amazon RDS for Db2로 마이그레이션하는 방법에는 몇 가지가 있습니다. Linux, AIX 또는 Windows 환경에서 Amazon RDS for Db2로 Db2 데이터베이스를 일회성 마이그레이션할 수 있습니다. 가동 중지를 최소화하기 위해 가동 중지 시간이 거의 발생하지 않는 마이그레이션을 수행할 수 있습니다. 데이터를 Amazon S3에 저장하고 한 번에 한 테이블씩 Db2 데이터베이스에 로드하여 데이터를 마이그레이션할 수 있습니다. 복제 또는 AWS Database Migration Service 사용을 통해 동기 마이그레이션을 수행할 수도 있습니다.

Linux 기반 Db2 데이터베이스의 일회성 마이그레이션의 경우 Amazon RDS는 오프라인 및 온라인 백업만 지원합니다. Amazon RDS에서는 증분 백업과 Delta 백업을 지원하지 않습니다. Linux 기반 Db2 데이터베이스의 가동 중지 시간이 거의 없는 마이그레이션의 경우 Amazon RDS에는 온라인 백업이 필요합니다. 가동 중지 시간이 거의 없는 마이그레이션에는 온라인 백업을 사용하고, 가동 중지를 처리할 수 있는 마이그레이션에는 오프라인 백업을 사용하는 것이 좋습니다.

주제

- Linux에서 Amazon RDS for Db2의 Linux로 마이그레이션
- Linux에서 Amazon RDS for Db2의 가동 중지 시간이 거의 없는 Linux로 마이그레이션
- Amazon RDS for Db2에 대해 Linux에서 Linux로 동기식 마이그레이션
- AIX 또는 Windows에서 Amazon RDS for Db2의 Linux로 마이그레이션
- Amazon S3를 통해 Amazon RDS for Db2로 Db2 데이터 마이그레이션
- AWS Database Migration Service(AWS DMS)를 사용하여 Amazon RDS for Db2로 마이그레이션

Linux에서 Amazon RDS for Db2의 Linux로 마이그레이션

이 마이그레이션 접근 방식을 사용하면 자체 관리형 Db2 데이터베이스를 Amazon S3 버킷으로 백업합니다. 그런 다음 Amazon RDS 저장 프로시저를 사용하여 Db2 데이터베이스를 Amazon RDS for Db2 DB 인스턴스로 복원합니다. Amazon S3 사용에 대한 자세한 내용은 Amazon RDS for Db2 DB 인스턴스와 Amazon S3 통합 섹션을 참조하세요.

RDS for Db2의 백업 및 복원은 IBM Db2가 지원되는 업그레이드 경로 및 제한 사항을 따릅니다. 자세한 내용은 IBM Db2 설명서의 Db2 서버의 지원되는 업그레이드 경로 및 Db2 서버의 업그레이드 제한을 참조하세요.

주제

- 기본 복원 사용에 대한 제한 및 권장 사항
- Amazon S3로 데이터베이스 백업
- 기본 자동 스토리지 그룹 생성
- Db2 데이터베이스 복원

기본 복원 사용에 대한 제한 및 권장 사항

기본 복원 사용 시 다음과 같은 제한 및 권장 사항이 적용됩니다.

- Amazon RDS는 지원되는 RDS for Db2 버전과 일치하는 Db2의 온프레미스 버전 마이그레이션만 지원합니다. 지원되는 버전에 대한 자세한 내용은 Amazon RDS에서 지원되는 Db2 마이너 버전 섹션을 참조하세요.
- Amazon RDS는 네이티브 복원을 위한 오프라인 및 온라인 백업만 지원합니다. Amazon RDS에서는 증분 백업이나 Delta 백업을 지원하지 않습니다.
- RDS for Db2 DB 인스턴스가 위치한 리전과 다른 AWS 리전에서는 Amazon S3 버킷으로 복원할 수 없습니다.
- Amazon S3는 Amazon S3 버킷에 업로드되는 파일 크기를 5TB로 제한합니다. 데이터베이스 백업 파일이 5TB를 초과하는 경우 백업 파일을 더 작은 파일로 분할하세요.
- Amazon RDS는 올타리가 없는 외부 루틴, 증분 복원 또는 Delta 복원을 지원하지 않습니다.
- 암호화된 원본 데이터베이스에서 복원할 수 없지만 암호화된 Amazon RDS DB 인스턴스로 복원할 수 있습니다.

데이터베이스를 복원하면 백업이 복사된 다음 RDS for Db2 DB 인스턴스에 추출됩니다. RDS for Db2 DB 인스턴스의 스토리지 공간을 백업 크기와 디스크의 원래 데이터베이스 크기의 합 이상으로 프로비저닝하는 것이 좋습니다.

복원된 데이터베이스의 최대 크기는 지원되는 최대 데이터베이스 크기에서 백업 크기를 뺀 값입니다. 예를 들어, 지원되는 최대 데이터베이스 크기가 64TiB이고 백업 크기가 30TiB인 경우 복원된 데이터베이스의 최대 크기는 34TiB입니다.

```
64 TiB - 30 TiB = 34 TiB
```

Amazon S3로 데이터베이스 백업

Amazon S3에 데이터베이스를 백업하려면 다음 AWS 구성 요소가 필요합니다.

- 백업 파일을 저장하는 Amazon S3 버킷: Amazon RDS로 마이그레이션하려는 모든 백업 파일을 업로드합니다. 가동 중지를 처리할 수 있는 마이그레이션에는 오프라인 백업을 사용하는 것이 좋습니다. S3 버킷이 이미 있다면 해당 버킷을 사용할 수 있습니다. S3 버킷이 없는 경우 Amazon S3 사용 설명서의 버킷 생성을 참조하세요.

  > **Note**
  >
  > 데이터베이스가 크고 S3 버킷으로 전송하는 데 시간이 오래 걸리는 경우 AWS Snow Family 디바이스를 주문하고 AWS에 백업을 수행하도록 요청할 수 있습니다. 파일을 디바이스에 복사하여 Snow Family 팀에 반환하면 팀은 백업된 이미지를 S3 버킷으로 전송합니다. 자세한 내용은 AWS Snow Family 설명서를 참조하십시오.

- S3 버킷에 액세스하기 위한 IAM 역할: 이미 IAM 역할이 있다면 해당 역할을 사용할 수 있습니다. 역할이 없는 경우 2단계: IAM 역할 생성 및 IAM 정책 연결 섹션을 참조하세요.
- IAM 역할에 연결된 신뢰 관계 및 권한이 포함된 IAM 정책: 자세한 내용은 1단계: IAM 정책 생성 섹션을 참조하세요.
- RDS for Db2 DB 인스턴스에 추가된 IAM 역할: 자세한 내용은 3단계: RDS for Db2 DB 인스턴스에 IAM 역할 추가 섹션을 참조하세요.

기본 자동 스토리지 그룹 생성

소스 데이터베이스에 기본 자동 스토리지 그룹이 있어야 합니다. 데이터베이스에 기본 자동 스토리지 그룹이 없는 경우 새 자동 스토리지 그룹을 만들어야 합니다.

기본 자동 스토리지 그룹을 만들려면

1. 소스 데이터베이스에 연결합니다. 다음 예제에서 *source_database*를 데이터베이스 이름으로 바꿉니다.

```
db2 connect to source_database
```

2. 자동 스토리지 그룹을 생성하고 이를 기본값으로 설정합니다. 다음 예제에서 *storage_path*를 스토리지 그룹이 있는 절대 경로로 바꿉니다.

```
db2 "create stogroup IBMSTOGROUP ON storage_path set as default"
```

3. 백엔드 프로세스를 종료합니다.

```
db2 terminate
```

4. 데이터베이스를 비활성화하고 모든 데이터베이스 서비스를 중지합니다. 다음 예제에서 *source_database*를 스토리지 그룹을 만든 대상 데이터베이스의 이름으로 바꿉니다.

```
db2 deactivate db source_database
```

5. 데이터베이스를 백업합니다. 다음 예제에서 *source_database*를 스토리지 그룹을 만든 대상 데이터베이스의 이름으로 바꿉니다. *file_system_path*를 데이터베이스를 백업하려는 위치의 절대 경로로 바꿉니다.

```
db2 backup database source_database to file_system_path
```

Db2 데이터베이스 복원

Amazon S3에 데이터베이스를 백업하고 자동 스토리지 그룹을 만들고 나면 Db2 데이터베이스를 RDS for Db2 DB 인스턴스로 복원할 수 있습니다.

Db2 데이터베이스를 RDS for Db2 DB 인스턴스로 복원하려면

1. RDS for Db2 DB 인스턴스에 연결합니다. 자세한 내용은 DB 인스턴스에 연결 단원을 참조하십시오.
2. (선택 사항) 데이터베이스가 복원 작업을 위한 최적의 설정으로 구성되었는지 확인하기 위해 the section called “rdsadmin.show_configuration”을 직접적으로 호출하여 RESTORE_DATABASE_PARALLELISM 및 RESTORE_DATABASE_NUM_BUFFERS에 대

한 값을 확인할 수 있습니다. 필요에 따라 이러한 값을 변경하려면 the section called “rdsadmin.set_configuration”을 직접적으로 호출합니다. 이러한 값을 명시적으로 설정하면 대량 데이터가 있는 데이터베이스를 복원할 때 성능이 향상될 수 있습니다.

3. `rdsadmin.restore_database`를 호출하여 데이터베이스를 복원합니다. 자세한 내용은 rdsadmin.restore_database 단원을 참조하십시오.

Linux에서 Amazon RDS for Db2의 가동 중지 시간이 거의 없는 Linux로 마이그레이션

이 마이그레이션 접근 방식을 사용하면 자체 관리형 Db2 데이터베이스(소스)에서 Amazon RDS for Db2로 Linux 기반 Db2 데이터베이스를 마이그레이션합니다. 이 접근 방식을 사용하면 애플리케이션 또는 사용자의 중단이나 가동 중지 시간이 최소화되거나 전혀 발생하지 않습니다. 이 접근 방식은 데이터베이스를 백업하고 로그 재생을 통해 복원하므로, 진행 중인 작업의 중단을 방지하고 데이터베이스의 고가용성을 제공합니다.

가동 중지 시간이 거의 없는 마이그레이션을 달성하기 위해 RDS for Db2는 로그 재생을 통한 복원을 구현합니다. 이 접근 방식은 자체 관리형 Linux 기반 Db2 데이터베이스의 백업을 가져와 RDS for Db2 서버에 복원합니다. Amazon RDS 저장 프로시저를 사용하면 후속 트랜잭션 로그를 적용하여 데이터베이스를 최신 상태로 유지합니다.

주제

- 가동 중지 시간이 거의 없는 마이그레이션에 대한 제한 사항 및 권장 사항
- Amazon S3로 데이터베이스 백업
- 기본 자동 스토리지 그룹 생성
- Db2 데이터베이스 마이그레이션

가동 중지 시간이 거의 없는 마이그레이션에 대한 제한 사항 및 권장 사항

가동 중지 시간이 거의 없는 마이그레이션을 사용하는 경우 다음과 같은 제한 사항과 권장 사항이 적용됩니다.

- Amazon RDS는 가동 중지 시간이 거의 없는 마이그레이션을 위해 온라인 백업이 필요합니다. 아카이브된 트랜잭션 로그를 업로드할 때 Amazon RDS가 데이터베이스를 롤포워드 보류 상태로 유지하기 때문입니다. 자세한 내용은 the section called “Db2 데이터베이스 마이그레이션” 단원을 참조하십시오.
- RDS for Db2 DB 인스턴스가 위치한 리전과 다른 AWS 리전에서는 Amazon S3 버킷으로 복원할 수 없습니다.

- Amazon S3는 S3 버킷에 업로드되는 파일 크기를 5TB로 제한합니다. 데이터베이스 백업 파일이 5TB를 초과하는 경우 백업 파일을 더 작은 파일로 분할하세요.
- Amazon RDS는 올타리가 없는 외부 루틴, 증분 복원 또는 Delta 복원을 지원하지 않습니다.
- 암호화된 원본 데이터베이스에서 복원할 수 없지만 암호화된 Amazon RDS DB 인스턴스로 복원할 수 있습니다.

데이터베이스를 복원하면 Amazon RDS가 백업을 복사한 다음 RDS for Db2 DB 인스턴스에서 추출합니다. RDS for Db2 DB 인스턴스의 스토리지 공간을 백업 크기와 디스크의 원래 데이터베이스 크기의 합 이상으로 프로비저닝하는 것이 좋습니다.

복원된 데이터베이스의 최대 크기는 지원되는 최대 데이터베이스 크기에서 백업 크기를 뺀 값입니다. 예를 들어, 지원되는 최대 데이터베이스 크기가 64TiB이고 백업 크기가 30TiB인 경우 복원된 데이터베이스의 최대 크기는 34TiB입니다.

```
64 TiB - 30 TiB = 34 TiB
```

Amazon S3로 데이터베이스 백업

Amazon S3에 데이터베이스를 백업하려면 다음 AWS 구성 요소가 필요합니다.

- 백업 파일을 저장하는 Amazon S3 버킷: Amazon RDS로 마이그레이션하려는 모든 백업 파일을 업로드합니다. Amazon RDS는 가동 중지 시간이 거의 없는 마이그레이션을 위해 온라인 백업이 필요합니다. S3 버킷이 이미 있다면 해당 버킷을 사용할 수 있습니다. S3 버킷이 없는 경우 Amazon S3 사용 설명서의 버킷 생성을 참조하세요.

  > **Note**
  >
  > 데이터베이스가 크고 S3 버킷으로 전송하는 데 시간이 오래 걸리는 경우 AWS Snow Family 디바이스를 주문하고 AWS에 백업을 수행하도록 요청할 수 있습니다. 파일을 디바이스에 복사하여 Snow Family 팀에 반환하면 팀은 백업된 이미지를 S3 버킷으로 전송합니다. 자세한 내용은 AWS Snow Family 설명서를 참조하십시오.

- S3 버킷에 액세스하기 위한 IAM 역할: 이미 AWS Identity and Access Management(IAM) 역할이 있다면 해당 역할을 사용할 수 있습니다. 역할이 없는 경우 2단계: IAM 역할 생성 및 IAM 정책 연결 섹션을 참조하세요.
- IAM 역할에 연결된 신뢰 관계 및 권한이 포함된 IAM 정책: 자세한 내용은 1단계: IAM 정책 생성 섹션을 참조하세요.

- RDS for Db2 DB 인스턴스에 추가된 IAM 역할: 자세한 내용은 3단계: RDS for Db2 DB 인스턴스에 IAM 역할 추가 섹션을 참조하세요.

기본 자동 스토리지 그룹 생성

소스 데이터베이스에 기본 자동 스토리지 그룹이 있어야 합니다. 데이터베이스에 기본 자동 스토리지 그룹이 없는 경우 새 자동 스토리지 그룹을 만들어야 합니다.

기본 자동 스토리지 그룹을 만들려면

1. 소스 데이터베이스에 연결합니다. 다음 예제에서 *source_database*를 데이터베이스 이름으로 바꿉니다.

```
db2 connect to source_database
```

2. 자동 스토리지 그룹을 생성하고 이를 기본값으로 설정합니다. 다음 예제에서 *storage_path*를 스토리지 그룹이 있는 절대 경로로 바꿉니다.

```
db2 "create stogroup IBMSTOGROUP ON storage_path set as default"
```

3. 백엔드 프로세스를 종료합니다.

```
db2 terminate
```

Db2 데이터베이스 마이그레이션

Amazon S3에 데이터베이스를 백업하고 자동 스토리지 그룹을 만들고 나면 Db2 데이터베이스를 RDS for Db2 DB 인스턴스로 마이그레이션할 수 있습니다.

가동 중지 시간이 거의 없는 마이그레이션을 수행하려면

1. 소스 데이터베이스를 온라인으로 백업합니다. 자세한 내용은 IBM Db2 설명서의 BACKUP DATABASE 명령을 참조하세요.
2. Amazon S3 버킷에 데이터베이스 백업을 복사합니다. Amazon S3 사용에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서를 참조하세요.
3. RDS for Db2 DB 인스턴스의 *master_username* 및 *master_password*를 사용하여 `rdsadmin` 서버에 연결합니다.

```
db2 connect to rdsadmin user master_username using master_password
```

4. (선택 사항) 데이터베이스가 복원 작업을 위한 최적의 설정으로 구성되었는지 확인하기 위해 the section called “rdsadmin.show_configuration”을 직접적으로 호출하여 RESTORE_DATABASE_PARALLELISM 및 RESTORE_DATABASE_NUM_BUFFERS에 대한 값을 확인할 수 있습니다. 필요에 따라 이러한 값을 변경하려면 the section called “rdsadmin.set_configuration”을 직접적으로 호출합니다. 이러한 값을 명시적으로 설정하면 대량 데이터가 있는 데이터베이스를 복원할 때 성능이 향상될 수 있습니다.
5. rdsadmin.restore_database를 호출하여 RDS for Db2 서버의 백업을 복원합니다. backup_type를 ONLINE으로 설정합니다. 자세한 내용은 rdsadmin.restore_database 단원을 참조하십시오.
6. 소스 서버에서 S3 버킷으로 아카이브 로그를 복사합니다. 자세한 내용은 IBM Db2 설명서의 아카이브 로깅을 참조하세요.
7. rdsadmin.rollforward_database를 호출하여 필요한 횟수만큼 아카이브 로그를 적용합니다. 데이터베이스를 ROLL-FORWARD PENDING 상태로 유지하려면 complete_rollforward를 FALSE로 설정합니다. 자세한 내용은 rdsadmin.rollforward_database 단원을 참조하십시오.
8. 모든 아카이브 로그를 적용한 후 rdsadmin.complete_rollforward를 호출하여 데이터베이스를 온라인 상태로 전환합니다. 자세한 내용은 rdsadmin.complete_rollforward 단원을 참조하십시오.
9. 데이터베이스의 애플리케이션 엔드포인트를 업데이트하거나 트래픽을 RDS for Db2 서버로 리디렉션하도록 DNS 엔드포인트를 업데이트하여 애플리케이션 연결을 RDS for Db2 서버로 전환합니다. RDS for Db2 데이터베이스 엔드포인트와 함께 자체 관리형 Db2 데이터베이스에서 Db2 자동 클라이언트 경로 변경 기능을 사용할 수도 있습니다. 자세한 내용은 IBM Db2 설명서의 자동 클라이언트 경로 변경 설명 및 설정을 참조하세요.
10. (선택 사항) 소스 데이터베이스를 종료합니다.

Amazon RDS for Db2에 대해 Linux에서 Linux로 동기식 마이그레이션

이 마이그레이션 접근 방식을 사용하면 자체 관리형 Db2 데이터베이스와 Amazon RDS for Db2 DB 인스턴스 간에 복제를 설정합니다. 자체 관리형 데이터베이스의 변경 내용은 거의 실시간으로 RDS for Db2 DB 인스턴스에 복제됩니다. 이 접근 방식을 사용하면 마이그레이션 프로세스 중에 지속적인 가용성을 제공하고 가동 중지를 최소화할 수 있습니다.

AIX 또는 Windows에서 Amazon RDS for Db2의 Linux로 마이그레이션

이 마이그레이션 접근 방식에서는 기본 Db2 도구를 사용하여 자체 관리형 Db2 데이터베이스를 Amazon S3 버킷에 백업합니다. 기본 Db2 도구에는 `export` 유틸리티, `db2move` 시스템 명령 또는 `db2look` 시스템 명령이 포함됩니다. Db2 데이터베이스는 자체 관리형으로 또는 Amazon Elastic Compute Cloud(Amazon EC2)에서 사용할 수 있습니다. AIX 또는 Windows 시스템에서 Amazon S3 버킷으로 데이터를 이동할 수 있습니다. 그런 다음 Db2 클라이언트를 사용하여 S3 버킷에서 Amazon RDS for Db2 데이터베이스로 직접 데이터를 로드합니다. 가동 중지 시간은 데이터베이스 크기에 따라 다릅니다. Amazon S3 사용에 대한 자세한 내용은 Amazon RDS for Db2 DB 인스턴스와 Amazon S3 통합 섹션을 참조하세요.

Db2 데이터베이스를 RDS for Db2로 마이그레이션하려면

1. 데이터베이스 백업을 준비합니다. 자체 관리형 Db2 시스템에 백업을 보관할 수 있는 충분한 양의 스토리지를 구성합니다.
2. 데이터베이스를 백업합니다.
   a. db2look 시스템 명령을 실행하여 모든 객체의 데이터 정의 언어(DDL) 파일을 추출합니다.
   b. Db2 내보내기 유틸리티, db2move 시스템 명령 또는 CREATE EXTERNAL TABLE 문을 실행하여 Db2 표 데이터를 Db2 시스템의 스토리지로 언로드합니다.
3. 백업을 Amazon S3 버킷으로 이동합니다. 자세한 내용은 Amazon RDS for Db2 DB 인스턴스와 Amazon S3 통합 단원을 참조하십시오.

   > **Note**
   >
   > 데이터베이스가 크고 S3 버킷으로 전송하는 데 시간이 오래 걸리는 경우 AWS Snow Family 디바이스를 주문하고 AWS에 백업을 수행하도록 요청할 수 있습니다. 파일을 디바이스에 복사하여 Snow Family 팀에 반환하면 팀은 백업된 이미지를 S3 버킷으로 전송합니다. 자세한 내용은 AWS Snow Family 설명서를 참조하십시오.

4. Db2 클라이언트를 사용하여 S3 버킷에서 RDS for Db2 데이터베이스로 직접 데이터를 로드합니다. 자세한 내용은 Amazon S3로 마이그레이션 단원을 참조하십시오.

Amazon S3를 통해 Amazon RDS for Db2로 Db2 데이터 마이그레이션

이 마이그레이션 접근 방식을 사용하면 먼저 단일 테이블의 데이터를 데이터 파일에 저장하고 Amazon S3 버킷에 저장합니다. 그런 다음 LOAD 명령을 사용하여 해당 데이터 파일의 데이터를 Amazon RDS

for Db2 데이터베이스의 테이블로 로드합니다. Amazon S3 사용에 대한 자세한 내용은 Amazon RDS for Db2 DB 인스턴스와 Amazon S3 통합 섹션을 참조하세요.

주제

- Amazon S3에 데이터 저장
- RDS for Db2 테이블에 데이터 로드

Amazon S3에 데이터 저장

단일 테이블의 데이터를 Amazon S3에 저장하려면 데이터베이스 유틸리티를 사용하여 데이터베이스 관리 시스템(DBMS)에서 CSV 파일로 데이터를 추출하세요. 그런 다음 각 파일을 Amazon S3에 업로드합니다.

Amazon S3에 데이터 파일을 저장하려면 다음 AWS 구성 요소가 필요합니다.

- 백업 파일을 저장할 Amazon S3 버킷: S3 버킷이 이미 있다면 해당 버킷을 사용할 수 있습니다. S3 버킷이 없는 경우 Amazon S3 사용 설명서의 버킷 생성을 참조하세요.
- S3 버킷에 액세스하기 위한 IAM 역할: 이미 IAM 역할이 있다면 해당 역할을 사용할 수 있습니다. 역할이 없는 경우 2단계: IAM 역할 생성 및 IAM 정책 연결 섹션을 참조하세요.
- IAM 역할에 연결된 신뢰 관계 및 권한이 포함된 IAM 정책: 자세한 내용은 1단계: IAM 정책 생성 섹션을 참조하세요.
- RDS for Db2 DB 인스턴스에 추가된 IAM 역할: 자세한 내용은 3단계: RDS for Db2 DB 인스턴스에 IAM 역할 추가 섹션을 참조하세요.

RDS for Db2 테이블에 데이터 로드

Amazon S3에 데이터 파일을 저장한 후 이러한 파일의 데이터를 RDS for Db2 DB 인스턴스의 개별 테이블로 로드할 수 있습니다.

Db2 테이블 데이터를 RDS for Db2 DB 데이터베이스 테이블에 로드하려면

1. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 `rdsadmin` 데이터베이스에 연결합니다. 다음 예제에서 *`master_username`* 및 *`master_password`*를 사용자 자체 정보로 대체합니다.

```
db2 connect to rdsadmin user master_username using master_password
```

2. 저장된 파일이 보관되는 Amazon S3 버킷으로 연결되는 스토리지 액세스 별칭을 카탈로그화합니다. 다음 단계에서 사용할 수 있도록 이 별칭의 이름을 적어 둡니다. 동일한 Amazon S3 버킷에 저장된 데이터 파일에서 여러 테이블을 로드하려는 경우 이 단계를 한 번만 수행하면 됩니다.

   다음 예제는 *jorge_souza*라는 사용자에게 *amzn-s3-demo-bucket*이라는 이름의 버킷에 대한 액세스 권한을 부여하는 *my_s3_alias*라는 별칭을 카탈로그화합니다.

   ```
   db2 "call rdsadmin.catalog_storage_access(?, 'my_s3_alias', 'amzn-s3-demo-bucket',
    'USER', 'jorge_souza')"
   ```

   이 저장 절차에 대한 자세한 내용은 the section called “rdsadmin.catalog_storage_access” 단원을 참조하세요.

3. Amazon S3 버킷으로 연결되는 스토리지 액세스 별칭을 사용하여 LOAD 명령을 실행합니다.

   > **Note**
   >
   > LOAD 명령에서 오류가 반환되는 경우 Amazon S3용 VPC 게이트웨이 엔드포인트를 생성하고 보안 그룹에 아웃바운드 규칙을 추가해야 할 수 있습니다. 자세한 내용은 the section called “파일 I/O 오류” 단원을 참조하십시오.

   다음 예제에서는 *my_s3_datafile.csv*라는 데이터 파일의 데이터를 *my_db2_table*이라는 테이블로 로드합니다. 이 예제에서는 *my_s3_alias*라는 별칭이 연결된 Amazon S3 버킷에 데이터 파일이 있다고 가정합니다.

   ```
   db2 "load from db2remote://my_s3_alias//my_s3_datafile.csv of DEL insert
    into my_db2_table";
   ```

   다음 예제에서는 *my_table1_export.ixf*라는 데이터 파일의 LOB를 *my_db2_table*이라는 테이블로 로드합니다. 이 예제에서는 *my_s3_alias*라는 별칭이 연결된 Amazon S3 버킷에 데이터 파일이 있다고 가정합니다.

   ```
   db2 "call sysproc.admin_cmd('load from
    "db2remote://my_s3_alias//my_table1_export.ixf" of ixf
           lobs from "db2remote://my_s3_alias//" xml from "db2remote://my_s3_alias//"
           modified by lobsinfile implicitlyhiddeninclude identityoverride
    generatedoverride periodoverride transactionidoverride
           messages on server
   ```

```
        replace into "my_schema"."my_db2_table"
                                nonrecoverable
        indexing mode incremental allow no access')"
```

RDS for Db2 DB 인스턴스에서 테이블에 로드하려는 Amazon S3 버킷의 각 데이터 파일에 대해 이 단계를 반복합니다.

LOAD 명령에 대한 자세한 내용은 LOAD 명령을 참조하세요.

AWS Database Migration Service(AWS DMS)를 사용하여 Amazon RDS for Db2로 마이그레이션

일회성 마이그레이션에 AWS DMS를 사용하고 Linux, Unix(예: AIX) 및 Windows의 Db2에서 Amazon RDS for Db2로 동기화할 수 있습니다. 자세한 내용은 AWS Database Migration Service란 무엇인가요?를 참조하세요.

기본 Db2 도구를 사용하여 Db2에서 Amazon RDS for Db2로 데이터 마이그레이션

몇 가지 네이티브 Db2 도구, 유틸리티 및 명령을 사용하여 Db2 데이터베이스의 데이터를 Amazon RDS for Db2 데이터베이스로 직접 이동할 수 있습니다. 이러한 기본 Db2 도구를 사용하려면 클라이언트 머신을 RDS for Db2 DB 인스턴스에 연결할 수 있어야 합니다. 자세한 내용은 Amazon RDS for Db2 DB 인스턴스에 클라이언트 머신 연결 단원을 참조하십시오.

> **Note**
>
> 데이터를 이동하는 또 다른 방법은 먼저 Amazon S3 버킷에 저장한 다음 LOAD 명령을 사용하여 RDS for Db2 데이터베이스의 테이블로 데이터를 전송하는 것입니다. 이 방법은 RDS for Db2와 S3 간의 네트워크 연결이 양호하기 때문에 대량의 데이터를 마이그레이션할 때 최상의 성능을 제공합니다. 자세한 내용은 the section called “Amazon S3로 마이그레이션” 단원을 참조하십시오.

도구 이름	사용 사례	제한 사항
db2look	자체 관리형 Db2 데이터베이스의 메타데이터를 RDS for Db2 데이터베이스로 복사합니다.	• 버퍼 풀 생성, 테이블스페이스 생성, 역할 생성을 위한 구문을 RDS for Db2 저장 프로

도구 이름	사용 사례	제한 사항
		시저에서 사용하는 구문과 일치하도록 수정해야 합니다.
IMPORT 명령	소형 표와 대형 객체(LOB)가 있는 표를 클라이언트 머신에서 RDS for Db2 DB 인스턴스로 마이그레이션합니다.	• INSERT 및 DELETE 로깅 작업으로 인해 LOAD 유틸리티보다 느립니다. • 네트워크 대역폭이 제한되어 성능이 떨어집니다.
INGEST 유틸리티	클라이언트 머신에 대형 객체(LOB)가 없는 파일 및 파이프의 데이터를 RDS for Db2 DB 인스턴스로 지속적으로 스트리밍합니다. INSERT 및 MERGE 작업을 지원합니다.	• LOB가 포함된 데이터 파일은 스트리밍할 수 없습니다. 대신 IMPORT 명령을 사용하세요. • 자체 관리형 Db2 데이터베이스와 RDS for Db2 데이터베이스 간에는 연결이 필요합니다.
INSERT 명령	자체 관리형 Db2 데이터베이스에서 RDS for Db2 데이터베이스로 소형 표의 데이터를 복사합니다.	• 자체 관리형 Db2 데이터베이스와 RDS for Db2 데이터베이스 간에는 연결이 필요합니다. • 네트워크 대역폭이 제한되어 성능이 떨어집니다.
LOAD CLIENT 명령	대형 객체(LOB)가 없는 소형 표를 클라이언트 머신에서 RDS for Db2 DB 인스턴스로 마이그레이션합니다.	• LOB가 포함된 데이터 파일은 마이그레이션할 수 없습니다. 대신 IMPORT 명령을 사용하세요. • 네트워크 대역폭이 제한되어 성능이 떨어집니다.

Amazon RDS for Db2 DB 인스턴스에 클라이언트 머신 연결

기본 Db2 도구를 사용하여 Db2 데이터베이스의 데이터를 Amazon RDS for Db2 데이터베이스로 이동하려면 먼저 클라이언트 머신을 RDS for Db2 DB 인스턴스에 연결해야 합니다.

클라이언트 머신은 다음 중 하나일 수 있습니다.

- Linux, Windows 또는 macOS의 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스. 이 인스턴스는 RDS for Db2 DB 인스턴스, AWS Cloud9 또는 AWS CloudShell과 동일한 Virtual Private Cloud(VPC)에 있어야 합니다.
- Amazon EC2 인스턴스의 자체 관리형 Db2 인스턴스. 인스턴스는 동일한 VPC에 있어야 합니다.
- Amazon EC2 인스턴스의 자체 관리형 Db2 인스턴스. VPC 피어링을 활성화한 경우 인스턴스가 서로 다른 VPC에 있을 수 있습니다. 자세한 내용은 Amazon Virtual Private Cloud VPC 피어링 가이드의 VPC 피어링 연결 생성을 참조하세요.
- 자체 관리 환경에서 Linux, Windows 또는 macOS를 실행하는 로컬 시스템. RDS for Db2에 공개적으로 연결할 수 있거나 자체 관리형 Db2 인스턴스 및 AWS 간에 VPN 연결을 활성화해야 합니다.

클라이언트 머신을 RDS for Db2 DB 인스턴스에 연결하려면 IBM Db2 Data Management Console을 사용하여 클라이언트 머신에 로그인하세요. 자세한 내용은 Amazon RDS DB 인스턴스 생성 및 IBM Db2 Data Management Console 단원을 참조하세요.

AWS Database Migration Service(AWS DMS)를 사용하여 데이터베이스에 대해 쿼리를 실행하고, SQL 실행 계획을 실행하고, 데이터베이스를 모니터링할 수 있습니다. 자세한 내용은 AWS Database Migration Service 사용 설명서의 AWS Database Migration Service란 무엇인가요?를 참조하세요.

클라이언트 머신을 RDS for Db2 DB 인스턴스에 성공적으로 연결했으면 기본 Db2 도구를 사용하여 데이터를 복사할 수 있습니다. 자세한 내용은 기본 Db2 도구를 사용하여 Db2에서 Amazon RDS for Db2로 데이터 마이그레이션 단원을 참조하십시오.

db2look을 사용하여 Db2에서 Amazon RDS for Db2로 데이터베이스 메타데이터 복사

`db2look`은 데이터 정의 언어(DDL) 파일, 객체, 권한, 구성, WLM 및 데이터베이스 레이아웃을 추출하는 기본 Db2 도구입니다. `db2look`을 사용하여 자체 관리형 Db2 데이터베이스의 데이터베이스 메타데이터를 Amazon RDS for Db2 데이터베이스로 복사할 수 있습니다. 자세한 내용은 IBM Db2 설명서의 db2look을 사용한 데이터베이스 모방을 참조하세요.

데이터베이스 메타데이터를 복사하려면

1. 자체 관리형 Db2 시스템에서 db2look 도구를 실행하여 DDL 파일을 추출합니다. 다음 예제에서 *database_name*을 Db2 데이터베이스 이름으로 바꿉니다.

```
db2look -d database_name -e -l -a -f -wlm -cor -createdb -printdbcfg -o db2look.sql
```

2. 클라이언트 머신이 소스(자체 관리형 Db2) 데이터베이스 및 RDS for Db2 DB 인스턴스에 액세스할 수 있는 경우 원격 인스턴스에 직접 연결하여 클라이언트 머신에 db2look.sql 파일을 생성할 수 있습니다. 그런 다음 원격 자체 관리형 Db2 인스턴스를 카탈로그화합니다.

   a. 노드를 카탈로그화합니다. 다음 예제에서 *dns_ip_address* 및 *port*를 자체 관리형 Db2 데이터베이스의 DNS 이름 또는 IP 주소 및 포트 번호로 대체합니다.

```
db2 catalog tcpip node srcnode REMOTE dns_ip_address server port
```

   b. 데이터베이스를 카탈로그화합니다. 다음 예제에서 *source_database_name* 및 *source_database_alias*를 자체 관리형 Db2 데이터베이스의 이름과 이 데이터베이스에 사용할 별칭으로 바꿉니다.

```
db2 catalog database source_database_name as source_database_alias at node
 srcnode \
    authentication server_encrypt
```

   c. 소스 데이터베이스에 연결합니다. 다음 예제에서 *source_database_alias*, *user_id*, *user_password*를 이전 단계에서 만든 별칭과 자체 관리형 Db2 데이터베이스의 사용자 ID 및 암호로 대체합니다.

```
db2look -d source_database_alias -i user_id -w user_password -e -l -a -f -wlm \
    -cor -createdb -printdbcfg -o db2look.sql
```

3. 클라이언트 머신에서 원격 자체 관리형 Db2 데이터베이스에 액세스할 수 없는 경우 db2look.sql 파일을 클라이언트 머신에 복사합니다. 그런 다음 RDS for Db2 DB 인스턴스를 카탈로그화합니다.

   a. 노드를 카탈로그화합니다. 다음 예제에서 *dns_ip_address* 및 *port*를 RDS for Db2 DB 인스턴스의 DNS 이름 또는 IP 주소 및 포트 번호로 대체합니다.

```
db2 catalog tcpip node remnode REMOTE dns_ip_address server port
```

b. 데이터베이스를 카탈로그화합니다. 다음 예제에서 *rds_database_name* 및 *rds_database_alias*를 RDS for Db2 데이터베이스의 이름과 이 데이터베이스에 사용할 별칭으로 바꿉니다.

```
db2 catalog database rds_database_name as rds_database_alias at node remnode \
    authentication server_encrypt
```

c. RDS for Db2를 관리하는 관리 데이터베이스를 카탈로그화합니다. 데이터를 저장하는 데는 이 데이터베이스를 사용할 수 없습니다.

```
db2 catalog database rdsadmin as rdsadmin at node remnode authentication
 server_encrypt
```

4. 버퍼 풀과 테이블스페이스를 생성합니다. 관리자는 버퍼 풀이나 테이블스페이스를 생성할 권한이 없습니다. 하지만 Amazon RDS 저장 프로시저를 사용하여 생성할 수는 있습니다.

a. `db2look.sql` 파일에서 버퍼 풀과 테이블스페이스의 이름과 정의를 찾습니다.

b. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 Amazon RDS에 연결합니다. 다음 예제에서 *master_username* 및 *master_password*를 사용자 자체 정보로 대체합니다.

```
db2 connect to rdsadmin user master_username using master_password
```

c. `rdsadmin.create_bufferpool`을 호출하여 버퍼 풀을 생성합니다. 자세한 내용은 rdsadmin.create_bufferpool 단원을 참조하십시오.

```
db2 "call rdsadmin.create_bufferpool(
    'database_name',
    'buffer_pool_name',
    buffer_pool_size,
    'immediate',
    'automatic',
    page_size,
    number_block_pages,
    block_size)"
```

d. `rdsadmin.create_tablespace`를 호출하여 테이블스페이스를 생성합니다. 자세한 내용은 rdsadmin.create_tablespace 단원을 참조하십시오.

```
db2 "call rdsadmin.create_tablespace(
```

```
    'database_name',
    'tablespace_name',
    'buffer_pool_name',
    tablespace_initial_size,
    tablespace_increase_size,
    'tablespace_type')"
```

e. 추가하려는 각 추가 버퍼 풀 또는 테이블스페이스에 대해 c 또는 d 단계를 반복합니다.

f. 연결을 종료합니다.

```
db2 terminate
```

5. 표와 객체를 생성합니다.

a. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 RDS for Db2 데이터베이스에 연결합니다. 다음 예제에서 *rds_database_name*, *master_username*, *master_password*를 사용자 자체 정보로 바꿉니다.

```
db2 connect to rds_database_name user master_username using master_password
```

b. `db2look.sql` 파일을 실행합니다.

```
db2 -tvf db2look.sql
```

c. 연결을 종료합니다.

```
db2 terminate
```

IMPORT 명령을 사용하여 클라이언트 시스템에서 Amazon RDS for Db2로 데이터 가져오기

클라이언트 머신에서 `IMPORT` 명령을 사용하여 Amazon RDS for Db2 서버로 데이터를 가져올 수 있습니다.

> **Important**
>
> `IMPORT` 명령 메서드는 소형 표와 대형 객체(LOB)가 포함된 표를 마이그레이션하는 데 유용합니다. `INSERT` 및 `DELETE` 로깅 작업 때문에 `IMPORT` 명령이 `LOAD` 유틸리티보다 느립니다. 클라이언트 머신과 RDS for Db2 사이의 네트워크 대역폭이 제한적인 경우 다른 마이그레이션

방법을 사용하는 것이 좋습니다. 자세한 내용은 기본 Db2 도구를 사용하여 Db2에서 Amazon RDS for Db2로 데이터 마이그레이션 단원을 참조하십시오.

RDS for Db2 서버로 데이터를 가져오려면

1. IBM Db2 Data Management Console을 사용하여 클라이언트 머신에 로그인합니다. 자세한 내용은 IBM Db2 Data Management Console을 사용하여 Amazon RDS for Db2 DB 인스턴스에 연결 단원을 참조하십시오.
2. 클라이언트 머신에 있는 RDS for Db2 데이터베이스를 카탈로그화합니다.

   a. 노드를 카탈로그화합니다. 다음 예제에서 *dns_ip_address* 및 *port*를 자체 관리형 Db2 데이터베이스의 DNS 이름 또는 IP 주소 및 포트 번호로 대체합니다.

   ```
   db2 catalog tcpip node srcnode REMOTE dns_ip_address server port
   ```

   b. 데이터베이스를 카탈로그화합니다. 다음 예제에서 *source_database_name* 및 *source_database_alias*를 자체 관리형 Db2 데이터베이스의 이름과 이 데이터베이스에 사용할 별칭으로 바꿉니다.

   ```
   db2 catalog database source_database_name as source_database_alias at node
    srcnode \
       authentication server_encrypt
   ```

3. 소스 데이터베이스에 연결합니다. 다음 예제에서 *source_database_alias*, *user_id*, *user_password*를 이전 단계에서 만든 별칭과 자체 관리형 Db2 데이터베이스의 사용자 ID 및 암호로 대체합니다.

   ```
   db2look -d source_database_alias -i user_id -w user_password -e -l -a -f -wlm \
       -cor -createdb -printdbcfg -o db2look.sql
   ```

4. 자체 관리형 Db2 시스템에서 EXPORT 명령을 사용하여 데이터 파일을 생성합니다. 다음 예제에서 *directory*를 데이터 파일이 있는 클라이언트 머신의 디렉터리로 바꿉니다. *file_name* 및 *table_name*을 데이터 파일 이름 및 표 이름으로 바꿉니다.

   ```
   db2 "export to /directory/file_name.txt of del lobs to /directory/lobs/ \
       modified by coldel\| select * from table_name"
   ```

5. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 RDS for Db2 데이터베이스에 연결합니다. 다음 예제에서 *rds_database_alias*, *master_username*, *master_password*를 사용자 자체 정보로 바꿉니다.

```
db2 connect to rds_database_alias user master_username using master_password
```

6. IMPORT 명령을 사용하여 클라이언트 머신의 파일에서 원격 RDS for Db2 데이터베이스로 데이터를 가져올 수 있습니다. 자세한 내용은 IBM Db2 설명서의 IMPORT 명령을 참조하세요. 다음 예제에서 *directory* 및 *file_name*을 데이터 파일이 있는 클라이언트 머신의 디렉터리와 데이터 파일 이름으로 바꿉니다. *SCHEMA_NAME* 및 *TABLE_NAME*을 스키마 및 표 이름으로 바꿉니다.

```
db2 "IMPORT from /directory/file_name.tbl OF DEL LOBS FROM /directory/lobs/ \
    modified by coldel\| replace into SCHEMA_NAME.TABLE_NAME"
```

7. 연결을 종료합니다.

```
db2 terminate
```

LOAD 명령을 사용하여 클라이언트 시스템에서 Amazon RDS for Db2로 데이터 가져오기

LOAD CLIENT 명령을 사용하여 클라이언트 머신의 파일 데이터를 RDS for Db2 서버로 로드할 수 있습니다. RDS for Db2 서버에는 SSH 연결이 없으므로, 자체 관리형 Db2 서버 또는 Db2 클라이언트 머신에서 LOAD CLIENT 명령을 사용할 수 있습니다.

> **Important**
>
> LOAD CLIENT 명령 메서드는 소형 표를 마이그레이션하는 데 유용합니다. 클라이언트와 RDS for Db2 사이의 네트워크 대역폭이 제한적인 경우 다른 마이그레이션 방법을 사용하는 것이 좋습니다. 자세한 내용은 기본 Db2 도구를 사용하여 Db2에서 Amazon RDS for Db2로 데이터 마이그레이션을 참조하세요.
>
> 데이터 파일에 대형 객체 파일 이름에 대한 참조가 포함된 경우 Db2 서버에 대형 객체(LOB)가 있어야 하므로, LOAD 명령이 작동하지 않습니다. 클라이언트 머신에서 RDS for Db2 서버로 LOB를 로드하려고 하면 SQL3025N 오류가 발생합니다. 대신 IMPORT 명령을 사용하세요.

RDS for Db2 서버로 데이터를 로드하려면

1. IBM Db2 Data Management Console을 사용하여 클라이언트 머신에 로그인합니다. 자세한 내용은 IBM Db2 Data Management Console을 사용하여 Amazon RDS for Db2 DB 인스턴스에 연결 단원을 참조하십시오.
2. 클라이언트 머신에 있는 RDS for Db2 데이터베이스를 카탈로그화합니다.

   a. 노드를 카탈로그화합니다. 다음 예제에서 *dns_ip_address* 및 *port*를 자체 관리형 Db2 데이터베이스의 DNS 이름 또는 IP 주소 및 포트 번호로 대체합니다.

   ```
   db2 catalog tcpip node srcnode REMOTE dns_ip_address server port
   ```

   b. 데이터베이스를 카탈로그화합니다. 다음 예제에서 *source_database_name* 및 *source_database_alias*를 자체 관리형 Db2 데이터베이스의 이름과 이 데이터베이스에 사용할 별칭으로 바꿉니다.

   ```
   db2 catalog database source_database_name as source_database_alias at node
    srcnode \
       authentication server_encrypt
   ```

3. 소스 데이터베이스에 연결합니다. 다음 예제에서 *source_database_alias*, *user_id*, *user_password*를 이전 단계에서 만든 별칭과 자체 관리형 Db2 데이터베이스의 사용자 ID 및 암호로 대체합니다.

   ```
   db2look -d source_database_alias -i user_id -w user_password -e -l -a -f -wlm \
       -cor -createdb -printdbcfg -o db2look.sql
   ```

4. 자체 관리형 Db2 시스템에서 EXPORT 명령을 사용하여 데이터 파일을 생성합니다. 다음 예제에서 *directory*를 데이터 파일이 있는 클라이언트 머신의 디렉터리로 바꿉니다. *file_name* 및 *TABLE_NAME*을 데이터 파일 이름 및 표 이름으로 바꿉니다.

   ```
   db2 "export to /directory/file_name.txt of del modified by coldel\| \
       select * from TPCH.TABLE_NAME"
   ```

5. RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 RDS for Db2 데이터베이스에 연결합니다. 다음 예제에서 *rds_database_alias*, *master_username*, *master_password*를 사용자 자체 정보로 바꿉니다.

   ```
   db2 connect to rds_database_alias user master_username using master_password
   ```

6. LOAD 명령을 사용하여 클라이언트 머신의 파일에서 원격 RDS for Db2 데이터베이스로 데이터를 로드합니다. 자세한 내용은 IBM Db2 설명서의 LOAD 명령을 참조하세요. 다음 예제에서 *directory*를 데이터 파일이 있는 클라이언트 머신의 디렉터리로 바꿉니다. *file_name* 및 *TABLE_NAME*을 데이터 파일 이름 및 표 이름으로 바꿉니다.

```
db2 "LOAD CLIENT from /directory/file_name.txt \
    modified by coldel\| replace into TPCH.TABLE_NAME \
    nonrecoverable without prompting"
```

7. 연결을 종료합니다.

```
db2 terminate
```

INSERT 명령을 사용하여 Db2에서 Amazon RDS for Db2로 데이터 가져오기

자체 관리형 Db2 서버에서 INSERT 명령을 사용하여 Amazon RDS for Db2 데이터베이스에 데이터를 삽입할 수 있습니다. 이 마이그레이션 방식에서는 원격 RDS for Db2 DB 인스턴스의 별명을 사용합니다. 자체 관리형 Db2 데이터베이스(소스)는 RDS for Db2 데이터베이스(대상)에 연결할 수 있어야 합니다.

> **Important**
>
> INSERT 명령 메서드는 소형 표를 마이그레이션하는 데 유용합니다. 자체 관리형 Db2 데이터베이스와 RDS for Db2 데이터베이스 간의 네트워크 대역폭이 제한적인 경우 다른 마이그레이션 방법을 사용하는 것이 좋습니다. 자세한 내용은 기본 Db2 도구를 사용하여 Db2에서 Amazon RDS for Db2로 데이터 마이그레이션 단원을 참조하십시오.

자체 관리형 Db2 데이터베이스의 데이터를 RDS for Db2 데이터베이스로 복사하려면

1. 자체 관리형 Db2 인스턴스에서 RDS for Db2 DB 인스턴스를 카탈로그화합니다.

   a. 노드를 카탈로그화합니다. 다음 예제에서 *dns_ip_address* 및 *port*를 자체 관리형 Db2 데이터베이스의 DNS 이름 또는 IP 주소 및 포트 번호로 대체합니다.

   ```
   db2 catalog tcpip node remnode REMOTE dns_ip_address SERVER port
   ```

   b. 데이터베이스를 카탈로그화합니다. 다음 예제에서 *rds_database_name*을 RDS for Db2 DB 인스턴스의 데이터베이스 이름으로 바꿉니다.

```
db2 catalog database rds_database_name as remdb at node remnode \
    authentication server_encrypt
```

2. 자체 관리형 Db2 인스턴스에서 페더레이션을 활성화합니다. 다음 예제에서 *source_database_name*을 자체 관리형 Db2 인스턴스의 데이터베이스 이름으로 바꿉니다.

```
db2 update dbm cfg using FEDERATED YES source_database_name
```

3. RDS for Db2 DB 인스턴스에 표를 생성합니다.

   a. 노드를 카탈로그화합니다. 다음 예제에서 *dns_ip_address* 및 *port*를 자체 관리형 Db2 데이터베이스의 DNS 이름 또는 IP 주소 및 포트 번호로 대체합니다.

```
db2 catalog tcpip node srcnode REMOTE dns_ip_address server port
```

   b. 데이터베이스를 카탈로그화합니다. 다음 예제에서 *source_database_name* 및 *source_database_alias*를 자체 관리형 Db2 데이터베이스의 이름과 이 데이터베이스에 사용할 별칭으로 바꿉니다.

```
db2 catalog database source_database_name as source_database_alias at node
 srcnode \
    authentication server_encrypt
```

4. 소스 데이터베이스에 연결합니다. 다음 예제에서 *source_database_alias*, *user_id*, *user_password*를 이전 단계에서 만든 별칭과 자체 관리형 Db2 데이터베이스의 사용자 ID 및 암호로 대체합니다.

```
db2look -d source_database_alias -i user_id -w user_password -e -l -a -f -wlm \
    -cor -createdb -printdbcfg -o db2look.sql
```

5. 페더레이션을 설정하고 자체 관리형 Db2 인스턴스에서 RDS for Db2 데이터베이스 표의 별명을 생성합니다.

   a. 로컬 데이터베이스에 연결합니다. 다음 예제에서 *source_database_name*을 자체 관리형 Db2 인스턴스의 데이터베이스 이름으로 바꿉니다.

```
db2 connect to source_database_name
```

   b. Db2 데이터 소스에 액세스하기 위한 래퍼를 생성합니다.

```
db2 create wrapper drda
```

c. 페더레이션된 데이터베이스에서 데이터 소스를 정의합니다. 다음 예제에서 *admin* 및 *admin_password*를 자체 관리형 Db2 인스턴스의 보안 인증 정보로 대체합니다. *rds_database_name*을 RDS for Db2 DB 인스턴스의 데이터베이스 이름으로 바꿉니다.

```
db2 "create server rdsdb2 type DB2/LUW version '11.5.9.0' \
    wrapper drda authorization "admin" password "admin_password" \
    options( dbname 'rds_database_name', node 'remnode')"
```

d. 두 데이터베이스의 사용자를 매핑합니다. 다음 예제에서 *master_username* 및 *master_password*를 RDS for Db2 DB 인스턴스의 보안 인증 정보로 대체합니다.

```
db2 "create user mapping for user server rdsdb2 \
    options (REMOTE_AUTHID 'master_username', REMOTE_PASSWORD
 'master_password')"
```

e. RDS for Db2 서버와의 연결을 확인합니다.

```
db2 set passthru rdsdb2
```

f. 원격 RDS for Db2 데이터베이스에서 표의 별명을 생성합니다. 다음 예제에서 *NICKNAME* 및 *TABLE_NAME*을 표의 별명과 표 이름으로 바꿉니다.

```
db2 create nickname REMOTE.NICKNAME for RDSDB2.TABLE_NAME.NICKNAME
```

6. 원격 RDS for Db2 데이터베이스 표에 데이터를 삽입합니다. 자체 관리형 Db2 인스턴스의 로컬 표에 있는 `select` 문에 별명을 사용합니다. 다음 예제에서 *NICKNAME* 및 *TABLE_NAME*을 표의 별명과 표 이름으로 바꿉니다.

```
db2 "INSERT into REMOTE.NICKNAME select * from RDS2DB2.TABLE_NAME.NICKNAME"
```

INGEST 유틸리티를 사용하여 Db2에서 Amazon RDS for Db2로 데이터 가져오기

`INGEST` 유틸리티를 사용하여 클라이언트 머신의 파일 및 파이프에서 대상 Amazon RDS for Db2 DB 인스턴스로 데이터를 지속적으로 스트리밍할 수 있습니다. `INGEST` 유틸리티는 `INSERT` 및 `MERGE` 작업을 지원합니다. 자세한 내용은 IBM Db2 설명서의 수집 유틸리티를 참조하세요.

이 INGEST 유틸리티는 별명을 지원하므로, 이를 사용하여 자체 관리형 Db2 데이터베이스의 데이터를 RDS for Db2 데이터베이스로 전송할 수 있습니다. 이 접근 방식은 두 데이터베이스 간에 네트워크 연결이 있는 한 사용할 수 있습니다.

> **Important**
>
> INGEST 유틸리티는 대형 객체(LOB)를 지원하지 않습니다. 대신 IMPORT 명령을 사용하세요.

INGEST 유틸리티의 RESTARTABLE 기능을 사용하려면 RDS for Db2 데이터베이스에서 다음 명령을 실행합니다.

```
db2 "call sysproc.sysinstallobjects('INGEST','C',NULL,NULL)"
```

Amazon RDS for Db2 페더레이션

Amazon RDS for Db2 데이터베이스를 페더레이션 데이터베이스로 사용할 수 있습니다. RDS for Db2에 대한 페더레이션을 설정한 후에는 RDS for Db2 데이터베이스에서 여러 데이터베이스의 데이터에 액세스하고 이를 쿼리할 수 있습니다. 페더레이션을 사용하면 데이터를 RDS for Db2 데이터베이스로 마이그레이션하거나 데이터를 단일 데이터베이스로 통합하지 않아도 됩니다.

RDS for Db2 데이터베이스를 페더레이션 데이터베이스로 사용하면 데이터를 서로 다른 데이터베이스에 보관하면서 모든 RDS for Db2 기능에 계속 액세스하고 다양한 AWS 서비스를 활용할 수 있습니다. 동일한 유형의 서로 다른 데이터베이스를 연결하는 동종 페더레이션과 유형이 서로 다른 여러 데이터베이스를 연결하는 이기종 페더레이션을 모두 설정할 수 있습니다.

먼저 RDS for Db2의 Db2 데이터베이스를 원격 데이터베이스에 연결합니다. 그런 다음 연결된 모든 데이터베이스에 대해 쿼리를 실행할 수 있습니다. 예를 들어 RDS for Db2 데이터베이스의 테이블을 z/OS 기반 원격 Db2 데이터베이스의 테이블과 조인하는 SQL JOIN 문을 실행할 수 있습니다.

주제

- 동종 페더레이션
- 이기종 페더레이션

동종 페더레이션

RDS for Db2 데이터베이스와 다음 Db2 제품군 간에 동종 페더레이션을 설정할 수 있습니다.

- Linux, UNIX 및 Windows(LUW)용 Db2
- Db2 iSeries
- Db2 for z/OS

RDS for Db2 동종 페더레이션은 다음 동작을 지원하지 않습니다.

- RDS for Db2 호스트 데이터베이스에서 노드 디렉터리와 원격 데이터베이스를 설정하는 `CATALOG` 명령 실행
- z/OS 기반 Db2에 대해 페더레이션할 때 워크로드 밸런싱(WLB) 설정
- IBM 데이터 서버 드라이버 구성 파일(`db2dsdriver.cfg`) 구성

RDS for Db2 동종 페더레이션의 요구 사항은 다음과 같습니다.

- UNFENCED 모드에서 DRDA 래퍼를 생성해야 합니다. 그러지 않으면 RDS for Db2에서 페더레이션이 작동하지 않습니다.
- RDS for Db2 호스트 데이터베이스에서 원격 호스트 데이터베이스로 들어오는 트래픽과 나가는 트래픽을 허용해야 합니다. 자세한 내용은 보안 그룹을 생성하여 VPC 내부의 DB 인스턴스에 대한 액세스를 제공 단원을 참조하십시오.

주제

- 1단계: DRDA 래퍼 및 페더레이션 서버 생성
- 2단계: 사용자 매핑 만들기
- 3단계: 연결 확인

1단계: DRDA 래퍼 및 페더레이션 서버 생성

동종 페더레이션을 위해 DRDA 래퍼와 페더레이션 서버를 생성합니다. 원격 호스트에 연결할 때는 HOST, PORT 및 DBNAME을 사용합니다.

원격 Db2 데이터베이스 유형에 따라 다음 방법 중 하나를 선택합니다.

- Linux, UNIX 및 Windows(LUX)용 Db2 데이터베이스 - 다음 SQL 명령을 실행합니다. 다음 예제에서는 *server_name*을 페더레이션에 사용할 서버의 이름으로 바꿉니다. *db2_version*을 원격 Db2 데이터베이스 버전으로 바꿉니다. *username*과 *password*를 연결하려는 원격 Db2 데이터베이스의 자격 증명으로 바꿉니다. *db_name*, *dns_name* 및 *port*를 연결하려는 원격 Db2 데이터베이스의 적절한 값으로 바꿉니다.

  ```
  create wrapper drda options(DB2_FENCED 'N');
  create server server_name type DB2/LUW wrapper drda version 'db2_version'
   authorization "master_username" password "master_password" options (add DBNAME
   'db_name',add HOST 'dns_name',add PORT 'port');
  ```

  예

  ```
  create wrapper drda options(DB2_FENCED 'N');
  create server SERVER1 type DB2/LUW wrapper drda version '11.5' authorization
   "sysuser" password "******" options (add DBNAME 'TESTDB2',add HOST
   'ip-123-45-67-899.us-west-1.compute.internal',add PORT '25010');
  ```

- Db2 iSeries – 다음 SQL 명령을 실행합니다. 다음 예제에서는 *wrapper_name* 및 *library_name*을 DRDA 래퍼 및 래퍼 라이브러리 파일의 이름으로 바꿉니다. *server_name*을 페

더레이션에 사용할 서버의 이름으로 바꿉니다. *db2_version*을 원격 Db2 데이터베이스 버전으로 바꿉니다. *username* 및 *password*를 연결하려는 원격 Db2 데이터베이스의 자격 증명으로 바꿉니다. *dns_name*, *port* 및 *db_name*을 연결하려는 원격 Db2 데이터베이스의 적절한 값으로 바꿉니다.

```
create wrapper wrapper_name library 'library name' options(DB2_FENCED 'N');
create server server_name type db2/mvs version db2_version wrapper wrapper_name
 authorization "sername" password "password" options (HOST 'dns_name', PORT 'port',
 DBNAME 'db_name');
```

예

```
create wrapper WRAPPER1 library 'libdb2drda.so' options(DB2_FENCED 'N');
create server SERVER1 type db2/mvs version 11 wrapper WRAPPER1 authorization
 "sysuser" password "******" options (HOST 'test1.123.com', PORT '446', DBNAME
 'STLEC1');
```

- z/OS용 Db2 – 다음 SQL 명령을 실행합니다. 다음 예제에서는 *wrapper_name* 및 *library_name*을 DRDA 래퍼 및 래퍼 라이브러리 파일의 이름으로 바꿉니다. *server_name*을 페더레이션에 사용할 서버의 이름으로 바꿉니다. *db2_version*을 원격 Db2 데이터베이스 버전으로 바꿉니다. *username* 및 *password*를 연결하려는 원격 Db2 데이터베이스의 자격 증명으로 바꿉니다. *dns_name*, *port* 및 *db_name*을 연결하려는 원격 Db2 데이터베이스의 적절한 값으로 바꿉니다.

```
create wrapper wrapper_name library 'library_name' options(DB2_FENCED 'N');
create server server_name type db2/mvs version db2_version wrapper wrapper_name
 authorization "username" password "password" options (HOST 'dns_name', PORT 'port',
 DBNAME 'db_name');
```

예

```
create wrapper WRAPPER1 library 'libdb2drda.so' OPTIONS(DB2_FENCED 'N');
create server SERVER1 type db2/mvs version 11 wrapper WRAPPER1 authorization
 "sysuser" password "******" options (HOST 'test1.123.com', PORT '446', DBNAME
 'STLEC1');
```

2단계: 사용자 매핑 만들기

다음 SQL 명령어를 실행하여 사용자 매핑을 만들고 페더레이션 서버를 데이터 소스 서버와 연결합니다. 다음 예제에서는 *server_name*을 작업을 수행하려는 원격 서버의 이름으로 바꿉니다. 이 서버는 1단계에서 만든 서버입니다. *username* 및 *password*를 이 원격 서버의 자격 증명으로 바꿉니다.

```
create user mapping for user server server_name options (REMOTE_AUTHID 'username',
 REMOTE_PASSWORD 'password');
```

자세한 내용은 IBM Db2 설명서의 사용자 매핑을 참조하세요.

3단계: 연결 확인

연결을 확인하여 페더레이션 설정이 성공했는지 확인합니다. SET PASSTHRU 명령으로 세션을 열어 네이티브 SQL 명령을 원격 데이터 소스로 보낸 다음 원격 데이터 서버에 테이블을 생성합니다.

1. 세션을 열고 닫아 데이터 소스에 SQL을 제출합니다. 다음 예제에서는 *server_name*을 1단계에서 페더레이션용으로 만든 서버의 이름으로 바꿉니다.

   ```
   set passthru server_name;
   ```

2. 새 테이블을 만듭니다. 다음 예제에서는 *column_name*, *data_type* 및 *value*를 테이블에 적합한 항목으로 바꿉니다.

   ```
   create table table_name
    ( column_name data_type(value), column_name data_type(value);
   ```

   자세한 내용은 IBM Db2 설명서의 CREATE TABLE 문을 참조하세요.

3. 인덱스를 만들고 테이블에 행 값을 삽입한 다음 연결을 재설정합니다. 연결을 재설정하면 연결이 끊어지지만 백엔드 프로세스는 그대로 유지됩니다. 다음 예제에서 *index_name*, *table_name*, *column_name* 및 *columnx_value*를 해당 정보로 대체합니다.

   ```
   create index index_name on table_name(column_name);
   insert into table_name values(column1_value,column2_value,column3_value);
   insert into table_name values(column1_value,column2_value,column3_value);
   set passthru reset;

   connect reset;
   ```

4. 원격 Db2 데이터베이스에 연결하고, 원격 서버의 닉네임을 만들고, 작업을 수행합니다. 원격 Db2 데이터베이스의 데이터에 모두 액세스했으면 연결을 재설정한 다음 종료하세요. 다음 예제에서 *database_name*을 원격 Db2 데이터베이스 이름으로 바꿉니다. *nickname*을 이름으로 바꿉니다. *server_name* 및 *table_name*을 작업을 수행하려는 원격 서버 및 해당 서버의 테이블 이름으로 바꿉니다. *username*을 원격 서버의 정보로 바꿉니다. *sql_command*를 원격 서버에서 수행할 작업으로 바꿉니다.

```
connect to database_name;
create nickname nickname for server_name."username"."table_name";
select sql_command from nickname;
connect reset;
terminate;
```

예

다음 예제에서는 페더레이션 서버 testdb10에서 작업을 수행할 수 있는 패스스루 세션을 만듭니다.

그런 다음 데이터 유형이 서로 다른 세 개의 열로 구성된 테이블 t1이 생성됩니다.

그런 다음 이 예제에서는 테이블 t1의 세 열에 인덱스 i1_t1을 만듭니다. 이후 해당 세 열에 값이 포함된 두 행이 삽입된 다음 연결이 해제됩니다.

마지막으로, 예제에서는 원격 Db2 데이터베이스 testdb2에 연결하고 페더레이션 서버 testdb10의 테이블 t1에 대한 닉네임을 만듭니다. 해당 데이터 소스의 사용자 이름 TESTUSER로 닉네임이 생성됩니다. SQL 명령은 테이블 t1의 모든 데이터를 출력합니다. 이 예제는 연결을 해제하고 세션을 종료합니다.

```
set passthru testdbl0;

create table t1 ( c1 decimal(13,0), c2 char(200), c3 int);

create index i1_t1 on t1(c3);
insert into t1 values(1,'Test',1);
insert into t1 values(2,'Test 2',2);
connect reset;

connect to testdb2;
create nickname remote_t1 for testdbl0."TESTUSER"."T1";
select * from remote_t1;
connect reset;
```

```
terminate;
```

이기종 페더레이션

RDS for Db2 데이터베이스와 Oracle 및 Microsoft SQL Server 등의 다른 데이터 소스 간에 이기종 페더레이션을 설정할 수 있습니다. Db2 LUW가 지원하는 데이터 소스의 전체 목록은 IBM 지원 사이트의 Db2 LUW V11.5에 번들로 제공되는 페더레이션의 데이터 소스 지원 매트릭스를 참조하세요.

RDS for Db2 이기종 페더레이션은 다음 항목을 지원하지 않습니다.

- 다른 데이터 소스의 네이티브 래퍼
- 다른 데이터 소스의 JDBC 래퍼
- Sybase, Informix 및 Teradata 데이터 소스와의 페더레이션(이러한 데이터 소스는 RDS for Db2에 클라이언트 소프트웨어를 설치해야 하기 때문)

RDS for Db2 이기종 페더레이션에는 다음과 같은 요구 사항이 있습니다.

- RDS for Db2는 ODBC 래퍼 메서드만 지원합니다.
- 래퍼의 명시적 정의를 생성하는 경우 `DB2_FENCED` 옵션을 `'N'`으로 설정해야 합니다. ODBC에 유효한 래퍼 옵션 목록은 IBM Db2 설명서의 ODBC 옵션을 참조하세요.
- RDS for Db2 호스트 데이터베이스에서 원격 호스트 데이터베이스로 들어오고 나가는 트래픽을 허용해야 합니다. 자세한 내용은 보안 그룹을 생성하여 VPC 내부의 DB 인스턴스에 대한 액세스를 제공 단원을 참조하십시오.

Oracle과의 페더레이션에 대한 자세한 내용은 IBM Support 사이트의 Db2 페더레이션 및 ODBC 드라이버를 사용하여 Oracle을 쿼리하는 방법을 참조하세요.

페더레이션을 지원하는 데이터 소스에 대한 자세한 내용은 IBM Support 사이트의 Db2 LUW V11.5에 번들로 제공되는 페더레이션의 데이터 소스 지원 매트릭스를 참조하세요.

주제

- 1단계: ODBC 래퍼 만들기
- 2단계: 페더레이션 서버 만들기
- 3단계: 사용자 매핑 만들기
- 4단계: 연결 확인

1단계: ODBC 래퍼 만들기

다음 명령을 실행하여 래퍼를 생성합니다.

```
db2 "create wrapper odbc options( module '/home/rdsdb/sqllib/federation/odbc/lib/
libodbc.so')"
```

2단계: 페더레이션 서버 만들기

다음 명령을 실행하여 페더레이션 서버를 생성합니다. 다음 예제에서는 *server_name*을 페더레이션에 사용할 서버의 이름으로 바꿉니다. *wrapper_type*을 적절한 래퍼로 바꿉니다. *db_version*을 원격 데이터베이스 버전으로 바꿉니다. *dns_name*, *port* 및 *service_name*을 연결하려는 원격 데이터베이스의 적절한 값으로 바꿉니다.

```
db2 "create server server_name type wrapper_type version db_version options (HOST
 'dns_name', PORT 'port', SERVICE_NAME 'service_name')“
```

래퍼 유형에 대한 자세한 내용은 IBM Support 사이트의 Db2 LUW V11.5에 번들로 제공되는 페더레이션의 데이터 소스 지원 매트릭스를 참조하세요.

예

다음 예제에서는 원격 Oracle 데이터베이스를 위한 페더레이션 서버를 만듭니다.

```
db2 "create server server1 type oracle_odbc version 12.1 options (HOST
 'test1.amazon.com', PORT '1521', SERVICE_NAME 'pdborcl.amazon.com')“
```

3단계: 사용자 매핑 만들기

다음 SQL 명령어를 실행하여 사용자 매핑을 만들고 페더레이션 서버를 데이터 소스 서버와 연결합니다. 다음 예제에서는 *server_name*을 작업을 수행하려는 원격 서버의 이름으로 바꿉니다. 이 서버는 2단계에서 만든 서버입니다. *username* 및 *password*를 이 원격 서버의 자격 증명으로 바꿉니다.

```
create user mapping for user server server_name options (REMOTE_AUTHID 'username',
 REMOTE_PASSWORD 'password');
```

자세한 내용은 IBM Db2 설명서의 사용자 매핑을 참조하세요.

4단계: 연결 확인

연결을 확인하여 페더레이션 설정이 성공했는지 확인합니다. SET PASSTHRU 명령으로 세션을 열어 네이티브 SQL 명령을 원격 데이터 소스로 보낸 다음 원격 데이터 서버에 테이블을 생성합니다.

1. 세션을 열고 닫아 데이터 소스에 SQL을 제출합니다. 다음 예제에서는 *server_name*을 2단계에서 페더레이션용으로 만든 서버의 이름으로 바꿉니다.

```
set passthru server_name;
```

2. 새 테이블을 만듭니다. 다음 예제에서는 *column_name*, *data_type* 및 *value*를 테이블에 적합한 항목으로 바꿉니다.

```
create table table_name
 ( column_name data_type(value), column_name data_type(value);
```

   자세한 내용은 IBM Db2 설명서의 CREATE TABLE 문을 참조하세요.

3. 인덱스를 만들고 테이블에 행 값을 삽입한 다음 연결을 재설정합니다. 연결을 재설정하면 연결이 끊어지지만 백엔드 프로세스는 그대로 유지됩니다. 다음 예제에서 *index_name*, *table_name*, *column_name* 및 *columnx_value*를 해당 정보로 대체합니다.

```
create index index_name on table_name(column_name);
insert into table_name values(column1_value,column2_value,column3_value);
insert into table_name values(column1_value,column2_value,column3_value);
set passthru reset;

connect reset;
```

4. 원격 Db2 데이터베이스에 연결하고, 원격 서버의 닉네임을 만들고, 작업을 수행합니다. 원격 Db2 데이터베이스의 데이터에 모두 액세스했으면 연결을 재설정한 다음 종료하세요. 다음 예제에서 *database_name*을 원격 Db2 데이터베이스 이름으로 바꿉니다. *nickname*을 이름으로 바꿉니다. *server_name* 및 *table_name*을 작업을 수행하려는 원격 서버 및 해당 서버의 테이블 이름으로 바꿉니다. *username*을 원격 서버의 정보로 바꿉니다. *sql_command*를 원격 서버에서 수행할 작업으로 바꿉니다.

```
connect to database_name;
create nickname nickname for server_name."username"."table_name";
select sql_command from nickname;
connect reset;
```

```
terminate;
```

예

다음 예제에서는 페더레이션 서버 testdb10에서 작업을 수행할 수 있는 패스스루 세션을 만듭니다.

그런 다음 데이터 유형이 서로 다른 세 개의 열로 구성된 테이블 t1이 생성됩니다.

그런 다음 이 예제에서는 테이블 t1의 세 열에 인덱스 i1_t1을 만듭니다. 이후 해당 세 열에 값이 포함된 두 행이 삽입된 다음 연결이 해제됩니다.

마지막으로, 예제에서는 원격 Db2 데이터베이스 testdb2에 연결하고 페더레이션 서버 testdb10의 테이블 t1에 대한 닉네임을 만듭니다. 해당 데이터 소스의 사용자 이름 TESTUSER로 닉네임이 생성됩니다. SQL 명령은 테이블 t1의 모든 데이터를 출력합니다. 이 예제는 연결을 해제하고 세션을 종료합니다.

```
set passthru testdbl0;

create table t1 ( c1 decimal(13,0), c2 char(200), c3 int);

create index i1_t1 on t1(c3);
insert into t1 values(1,'Test',1);
insert into t1 values(2,'Test 2',2);
connect reset;

connect to testdb2;
create nickname remote_t1 for testdbl0."TESTUSER"."T1";
select * from remote_t1;
connect reset;
terminate;
```

Amazon RDS for Db2의 복제본 작업

RDS for Db2 DB 인스턴스 간 복제를 구성하기 위해 복제본 데이터베이스를 생성할 수 있습니다. RDS for Db2 복제본과 다른 DB 엔진 간의 차이점에 대한 요약은 DB 엔진별 읽기 전용 복제본의 차이점 섹션을 참조하세요.

Db2 복제본 데이터베이스는 기본 데이터베이스의 물리적 복사본입니다. 대기 모드의 Db2 복제본을 대기 복제본이라고 합니다. Db2는 복제본에서의 읽기 또는 쓰기를 허용하지 않지만, 복제본을 승격하여 쓰기 가능한 상태로 만들 수 있습니다. 승격된 복제본에는 승격 요청이 이루어진 시점까지 복제된 데이터가 있습니다. RDS for Db2는 복제에 Db2 고가용성 재해 복구(HADR)를 사용합니다. 자세한 내용은 IBM Db2 설명서의 High availability disaster recovery (HADR)를 참조하세요.

대기 복제본

Db2 복제본을 생성하거나 수정할 때 대기 모드로 배치할 수 있습니다. 대기 복제본의 경우 복제본 데이터베이스가 사용자 연결을 수락하지 않도록 Db2 환경 변수 `DB2_HADR_ROS`가 `OFF`로 설정됩니다. 대기 복제본의 주된 용도는 리전 간 재해 복구입니다.

대기 복제본은 읽기 전용 워크로드를 처리할 수 없습니다. 대기 복제본에는 아카이브 로그가 없습니다.

소스 DB 인스턴스 하나에서 최대 세 개까지 복제본을 생성할 수 있습니다.

데이터베이스 활성화

Db2 HADR은 데이터베이스 수준에서 구성됩니다. 복제본을 생성하면 `rdsadmin`을 포함해 RDS가 완전히 관리하는 모든 Db2 데이터베이스에 HADR이 설정됩니다. Db2 복제본을 생성하기 전에 모든 데이터베이스를 명시적으로 활성화해야 합니다. 활성화하지 않으면 복제본 생성이 실패하고 Amazon RDS가 이벤트를 내보냅니다. DB 인스턴스에 하나 이상의 복제본이 생긴 후에는 `rdsadmin.activate_database` 또는 `rdsadmin.deactivate_database` 저장 프로시저를 사용하여 DB 인스턴스에서 데이터베이스를 활성화하거나 비활성화할 수 없습니다. 그러나 `rdsadmin.reactivate_database` 저장 프로시저를 직접 호출하여 데이터베이스를 다시 활성화할 수 있습니다. 자세한 내용은 데이터베이스 관리 섹션을 참조하세요.

HADR 구성

데이터베이스에 연결한 다음, `db2 get db cfg`를 실행하여 데이터베이스에 대한 모든 HADR 구성을 볼 수 있습니다.

아카이브 로그 보존

Amazon RDS는 다음 조건이 충족되면 기본 DB 인스턴스에서 로그를 제거합니다.

- 로그가 최소 2시간 이상 경과됨
- 아카이브 로그 보존 시간에 대한 설정이 초과됨
- 아카이브 로그가 모든 복제본 DB 인스턴스에 성공적으로 복제됨 이 조건은 동일한 AWS 리전의 DB 인스턴스와 교차 리전 DB 인스턴스에 적용됩니다.

아카이브 로그 보존 시간 설정에 대한 자세한 내용은 rdsadmin.set_archive_log_retention 단원을 참조하세요.

Amazon RDS는 각 데이터베이스를 개별적으로 확인하고 정리합니다. 데이터베이스의 HADR 연결이 끊어지거나 연결에 대한 정보를 사용할 수 없는 경우 Amazon RDS는 해당 데이터베이스를 건너뛰고 아카이브 로그를 제거하지 않습니다.

Db2 복제 중 중단

복제본을 생성하면 Amazon RDS가 소스 DB 인스턴스의 DB 스냅샷을 캡처하고 복제를 시작합니다. DB 스냅샷 작업이 시작될 때 소스 DB 인스턴스에서 매우 짧은 I/O 보류가 발생합니다. 이러한 I/O 중단은 일반적으로 1초 정도 지속됩니다. 그러나 소스 DB 인스턴스가 다중 AZ 배포인 경우 소스 DB 인스턴스에 I/O 보류가 발생하지 않습니다. 이는 다중 AZ 배포의 경우 보조 DB 인스턴스에서 스냅샷이 생성되기 때문입니다.

DB 스냅샷은 Db2 복제본이 됩니다. Amazon RDS는 서비스 중단 없이 소스 데이터베이스 및 복제본에 필요한 파라미터와 권한을 설정합니다. 마찬가지로 복제본을 삭제해도 중단이 발생하지 않습니다.

RDS for Db2 복제본에 대한 요구 사항 및 고려 사항

Db2 복제본을 생성하기 전에 다음 요구 사항 및 고려 사항을 숙지하세요.

Db2 복제본에 대한 버전 및 라이선스 요구 사항

RDS for Db2 복제본을 생성하기 전에 다음 정보를 검토하세요.

- Db2 Advanced Edition(AE)과 Standard Edition(SE) 모두 Bring Your Own License(BYOL) 모델과 AWS Marketplace 모델을 통한 Db2 라이선스 모두에 대해 대기 상태의 복제본을 생성할 수 있습니다.

- 읽기 전용 복제본에 대한 사용자 지정 파라미터 그룹을 지정해야 합니다. BYOL 모델을 사용하는 복제본의 경우 이 사용자 지정 파라미터 그룹에 IBM Site ID 및 IBM Customer ID가 포함되어야 합니다. AWS Management Console, AWS CLI 또는 RDS API를 사용하여 읽기 전용 복제본에 대해 이 사용자 지정 파라미터 그룹을 지정할 수 있습니다. 자세한 내용은 Db2에 기존 보유 라이선스 사용을 위한 IBM ID 섹션을 참조하세요.
- 대기 복제본의 경우 DB 인스턴스 크기와 관계없이 vCPU 수는 2입니다. 이어지는 항목에서는 RDS for Db2 라이선스 유형 간의 차이점을 설명합니다.
  - BYOL 모델의 경우 AWS License Manager 구성은 RDS for Db2 DB 인스턴스가 두 개의 vCPU를 사용함을 보여줍니다.
  - AWS Marketplace 모델을 통한 Db2 라이선스의 경우 청구서에 두 개의 vCPU에 대한 라이선스 비용이 반영됩니다.
- 모든 Db2 11.5 버전은 복제본 DB 인스턴스를 지원합니다. Db2 복제본은 마이너 버전 업그레이드를 지원합니다.

RDS for Db2 복제본에 대한 백업 및 복구 고려 사항

RDS for Db2 복제본을 생성하기 전에 다음 사항을 고려하세요.

- RDS for Db2 복제본의 스냅샷을 생성하거나 자동 백업을 설정하려면 백업 보존 기간을 수동으로 설정해야 합니다. 자동 백업은 기본적으로 켜져 있지 않습니다.
- 복제본 백업을 복원할 때는 백업이 완료된 시간이 아니라 데이터베이스 시간으로 복원됩니다. 이 데이터베이스 시간이란 백업에 있는 데이터의 가장 최근에 적용된 트랜잭션 시간입니다. 복제본이 기본 데이터베이스보다 몇 분 또는 몇 시간 정도 지연될 수 있기 때문에 이러한 차이는 상당합니다. 여러 데이터베이스가 있는 경우 RDS for Db2는 가장 이른 데이터베이스 시간을 사용합니다.

  차이점을 찾으려면 AWS CLI describe-db-snapshots 명령을 실행하거나 RDS API DescribeDBSnapshots 작업을 직접 호출합니다. `SnapshotDatabaseTime` 값을 `OriginalSnapshotCreateTime` 값과 비교합니다. `SnapshotDatabaseTime` 값은 복제본 백업의 데이터베이스 시간입니다. `OriginalSnapshotCreateTime` 값은 기본 데이터베이스에 적용된 최신 트랜잭션입니다.

RDS for Db2 복제본에 대한 복제 고려 사항

RDS for Db2 복제본을 생성하기 전에 다음 사항을 고려하세요.

- 복제는 RDS for Db2 DB 인스턴스의 모든 데이터베이스에 Db2 HADR을 사용합니다.

- 복제는 LOAD 명령을 지원하지 않습니다. 소스 DB 인스턴스에서 LOAD 명령을 실행하면 일관되지 않은 데이터가 수신됩니다.
- RDS for Db2는 다음 항목을 복제하지 않습니다.
  - 스토리지 액세스. 스토리지 액세스에 의존하는 외부 테이블과 같은 데이터에 유의하세요.
  - 인라인이 아닌 LOB
  - 외부 저장 프로시저의 바이너리(C 또는 Java)
- 대기 복제본의 경우 RDS for Db2는 다음 항목을 복제합니다.
  - 마스터 사용자를 제외한 로컬 사용자
  - 데이터베이스 구성 파라미터

RDS for Db2 복제본에 대한 기타 고려 사항

RDS for Db2 복제본을 생성하기 전에 다음 사항을 고려하세요.

- RDS for Db2는 데이터베이스 구성을 복제본에 복제합니다. RDS for Db2는 복제본을 승격할 때 각 데이터베이스를 비활성화하고 활성화합니다.
- RDS for Db2는 마스터 사용자가 아닌 로컬 사용자를 복제본에 복제합니다. 복제본에서 마스터 사용자를 수정할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.
- 모든 데이터베이스는 활성 상태여야 합니다. 데이터베이스 활성화에 대한 자세한 내용은 데이터베이스 관리 섹션을 참조하세요.
- 복제본을 생성하기 전에 데이터베이스 생성, 삭제, 복원 또는 롤링 포워드를 위한 모든 저장 프로시저를 완료해야 합니다. 이러한 저장 프로시저 사용에 대한 자세한 내용은 데이터베이스 관리 섹션을 참조하세요.
- 복제본이 생성되면 Amazon RDS는 소스 DB 인스턴스의 모든 데이터베이스에 대한 데이터베이스 수준 파라미터 `blocknonlogged`를 `YES`로 설정합니다. 소스 복제본이 다시 단독 인스턴스가 되면 Amazon RDS는 값을 다시 `NO`로 설정합니다. 자세한 내용은 IBM Db2 설명서의 blocknonlogged - Block creation of tables that allow non-logged activity configuration parameter를 참조하세요.
- 복제본이 생성되면 Amazon RDS는 소스 DB 인스턴스의 모든 데이터베이스에 대한 데이터베이스 수준 파라미터 `logindexbuild`를 `YES`로 설정합니다. 소스 복제본이 다시 단독 인스턴스가 되면 Amazon RDS는 값을 다시 `NO`로 설정합니다. 자세한 내용은 IBM Db2 설명서의 logindexbuild - Log index pages created configuration parameter를 참조하세요.

RDS for Db2 복제본 생성 준비

복제본 사용을 시작하려면 먼저 다음 작업을 수행해야 합니다.

업무

- 자동 백업 활성화
- 컴퓨팅 및 스토리지 리소스 계획
- 데이터베이스 준비

자동 백업 활성화

임의의 DB 인스턴스를 원본 DB 인스턴스로 사용하려면 원본 DB 인스턴스의 자동 백업을 활성화해야 합니다. 이 절차를 수행하는 방법에 대한 자세한 내용은 자동 백업 활성화 단원을 참조하세요.

컴퓨팅 및 스토리지 리소스 계획

원본 DB 인스턴스와 해당 복제본이 운영 로드에 맞게 컴퓨팅 및 스토리지 면에서 제대로 크기가 조정되었는지 확인합니다. 복제본이 컴퓨팅, 네트워크 또는 스토리지 리소스 용량에 도달하면 복제본은 해당 소스에서 변경 사항을 수신하거나 적용하는 것을 중지합니다. RDS for Db2는 개입을 통해 소스 DB 인스턴스와 해당 복제본 간의 긴 복제본 지연 시간을 완화하지 않습니다. 복제본의 스토리지 및 CPU 리소스를 해당 원본 및 다른 복제본과 별도로 수정할 수 있습니다.

데이터베이스 준비

복제본을 생성하기 전에 DB 인스턴스에 존재해야 하는 모든 데이터베이스가 DB 인스턴스에 포함되어 있는지 확인합니다. 복제본을 생성한 후에는 DB 인스턴스에서 데이터베이스를 생성, 삭제 또는 기본 복원할 수 없습니다. rdsadmin.create_database, rdsadmin.drop_database 또는 rdsadmin.restore_database 저장 프로시저에 대한 직접 호출은 모두 실패합니다.

또한 DB 인스턴스의 모든 데이터베이스가 활성 상태인지 확인합니다. 활성 상태가 아니면 복제본 생성이 실패합니다. 데이터베이스 활성화에 대한 자세한 내용은 데이터베이스 관리 섹션을 참조하세요.

대기 모드에서 RDS for Db2 복제본 생성

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 복제본을 대기 모드로 생성할 수 있습니다.

콘솔

소스 RDS for Db2 DB 인스턴스에서 대기 복제본을 생성하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
3. 대기 복제본의 소스로 사용할 RDS for Db2 DB 인스턴스를 선택합니다.
4. 작업에서 읽기 전용 복제본 만들기를 선택합니다.
5. 복제본 모드에서 대기를 선택합니다.
6. 사용하려는 설정을 선택합니다. DB 인스턴스 식별자에 읽기 전용 복제본의 이름을 입력합니다. 필요에 따라 다른 설정을 조정합니다.
7. 리전에서 대기 복제본을 시작할 AWS 리전을 선택합니다.
8. 인스턴스 크기 및 스토리지 유형을 선택합니다. 읽기 전용 복제본의 원본 DB 인스턴스와 동일한 DB 인스턴스 클래스와 스토리지 유형을 사용하는 것이 좋습니다.
9. 대기 복제본에 대한 장애 조치 지원을 위해 다른 가용 영역에 대기 복제본을 생성하려면 다중 AZ 배포에서 대기 인스턴스 생성을 선택합니다.
10. 사용하고자 하는 기타 설정을 선택합니다.
11. 복제본 생성을 선택합니다.

데이터베이스 페이지에서 대기 복제본에는 복제본 역할이 있습니다.

AWS CLI

Db2 복제본을 대기 모드로 생성하려면 AWS CLI 명령 create-db-instance-read-replica에서 `--replica-mode`를 `mounted`로 설정합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance-read-replica \
    --db-instance-identifier my_read_replica \
    --source-db-instance-identifier my_db_instance \
    --replica-mode mounted
```

Windows의 경우:

```
aws rds create-db-instance-read-replica ^
    --db-instance-identifier my_read_replica ^
    --source-db-instance-identifier my_db_instance ^
    --replica-mode mounted
```

RDS API

Db2 복제본을 대기 모드로 생성하려면 RDS API 작업 CreateDBInstanceReadReplica에서 `ReplicaMode=mounted`를 지정합니다.

RDS for Db2 복제본 백업 작업

RDS for Db2 복제본의 백업을 생성하고 복원할 수 있습니다. RDS for Db2는 자동 백업과 수동 스냅샷을 모두 지원합니다. 자세한 내용은 데이터 백업, 복원 및 내보내기 섹션을 참조하세요. 이어지는 섹션에서는 기본 데이터베이스와 RDS for Db2 복제본의 백업 관리 간의 주요 차이점을 설명합니다.

주제

- RDS for Db2 복제본 백업 켜기
- RDS for Db2 복제본 백업 복원

RDS for Db2 복제본 백업 켜기

RDS for Db2 복제본에서는 기본적으로 자동 백업이 켜져 있지 않습니다. 백업 보존 기간을 0이 아닌 양수 값으로 설정하여 자동 백업을 켭니다.

콘솔

자동 백업을 즉시 켜는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 다음 변경하려는 DB 인스턴스를 선택합니다.
3. 수정을 선택합니다.
4. 백업 보존 기간으로 0이 아닌 양수 값(예: 3일)을 선택합니다.
5. [Continue]를 선택합니다.
6. 즉시 적용을 선택합니다.
7. DB 인스턴스 수정을 선택하여 변경 내용을 저장하고 자동 백업을 활성화합니다.

AWS CLI

자동 백업을 켜려면 AWS CLI modify-db-instance 명령을 사용합니다.

다음 파라미터를 포함합니다.

- `--db-instance-identifier`
- `--backup-retention-period`
- `--apply-immediately` 또는 `--no-apply-immediately`

다음 예에서는 백업 보존 기간을 3일로 설정하여 자동 백업을 켭니다. 변경이 바로 적용됩니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier my_db_instance  \
    --backup-retention-period 3 \
    --apply-immediately
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier my_db_instance  ^
    --backup-retention-period 3 ^
    --apply-immediately
```

RDS API

자동 백업을 켜려면 RDS API ModifyDBInstance 작업을 다음 필수 파라미터와 함께 사용합니다.

- `DBInstanceIdentifier`
- `BackupRetentionPeriod`

RDS for Db2 복제본 백업 복원

기본 데이터베이스의 백업을 복원하는 것과 동일한 방식으로 RDS for Db2 복제본 백업을 복원할 수 있습니다. 자세한 정보는 다음의 주제를 참조하세요.

- DB 인스턴스 복원
- Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원

복제본 백업을 복원할 때 주요 고려 사항은 복원할 시점을 결정하는 것입니다. 이 데이터베이스 시간이란 백업에 있는 데이터의 가장 최근에 적용된 트랜잭션 시간입니다. 복제본 백업을 복원할 때는 백업이 완료된 시간이 아니라 데이터베이스 시간으로 복원됩니다. 복제본이 기본 데이터베이스보다 몇 분 또는 몇 시간 정도 지연될 수 있기 때문에 이러한 차이는 상당합니다. 따라서 복제본 백업의 데이터베이스 시간 및 복원하려는 시점이 백업 생성 시간보다 훨씬 이를 수 있습니다.

데이터베이스 시간과 생성 시간의 차이를 찾으려면 AWS CLI describe-db-snapshots 명령을 실행하거나 RDS API DescribeDBSnapshots 작업을 직접 호출합니다. `SnapshotDatabaseTime` 값과 `OriginalSnapshotCreateTime` 값을 비교합니다. `SnapshotDatabaseTime` 값은 복제본 백업의 모든 데이터베이스 중에서 가장 이른 데이터베이스 시간입니다. `OriginalSnapshotCreateTime` 값은 기본 데이터베이스에 적용된 최신 트랜잭션입니다. 복제 지연은 여러 데이터베이스에서 다를 수 있으며 데이터베이스 시간은 이 두 번 사이에 있을 수 있습니다.

다음 AWS CLI 예는 두 시간의 차이를 보여줍니다.

```
aws rds describe-db-snapshots \
    --db-instance-identifier my_db2_replica
    --db-snapshot-identifier my_replica_snapshot

{
    "DBSnapshots": [
        {
            "DBSnapshotIdentifier": "my_replica_snapshot",
            "DBInstanceIdentifier": "my_db2_replica",
            "SnapshotDatabaseTime": "2022-07-26T17:49:44Z",
            ...
            "OriginalSnapshotCreateTime": "2021-07-26T19:49:44Z"
        }
    ]
}
```

RDS for Db2 복제 문제 해결

이 콘텐츠에서는 가능한 RDS for Db2 복제 문제 및 해결 방법을 설명합니다.

주제

- Db2 복제 지연 모니터링

- Db2 복제 오류

Db2 복제 지연 모니터링

Amazon CloudWatch에서 복제 지연 시간을 모니터링하려면 Amazon RDS `ReplicaLag` 지표를 확인합니다. 복제본 지연 시간에 대한 자세한 내용은 읽기 전용 복제본 모니터링 및 Amazon RDS에 대한 Amazon CloudWatch 지표 단원을 참조하세요.

대기 복제본의 경우 지연 시간이 너무 길면 `MON_GET_HADR` 테이블에 소스 DB 인스턴스의 상태를 쿼리합니다. 복제본 DB 인스턴스는 사용자 연결을 허용하지 않으므로 복제본 DB 인스턴스를 쿼리하지 마세요.

Db2 복제 오류

Db2 복제는 여러 가지 이유로 오류 상태일 수 있습니다. 이벤트와 DB 인스턴스 상태를 모니터링하여 DB 인스턴스가 복제 중인지 확인합니다. DB 인스턴스가 복제 중인지 확인하려면 Amazon RDS 콘솔에서 복제본에 대한 진단 로그를 확인합니다. 로그에서 HADR 메시지의 오류를 찾습니다. 로그 시퀀스 번호와 기본 시퀀스 번호를 비교합니다. 자세한 내용은 Amazon RDS for Db2 데이터베이스 로그 파일 섹션을 참조하세요.

Amazon RDS for Db2 DB 인스턴스 옵션

다음은 Db2 DB 엔진을 실행하는 Amazon RDS 인스턴스에 사용할 수 있는 옵션 또는 추가 기능을 보여줍니다. 이러한 옵션들을 활성화하려면 먼저 사용자 정의 옵션 그룹에 추가한 다음 옵션 그룹과 DB 인스턴스를 연동시켜야 합니다. 옵션 그룹 작업에 대한 자세한 내용은 옵션 그룹 작업 단원을 참조하십시오.

Amazon RDS는 Db2에 대해 다음 옵션을 지원합니다.

옵션	옵션 ID
Db2 감사 로깅	DB2_AUDIT

Db2 감사 로깅

Db2 감사 로깅을 사용하면 Amazon RDS는 사용자의 데이터베이스 로그온, 데이터베이스에 대해 실행되는 쿼리 등의 데이터베이스 활동을 기록합니다. RDS는 사용자가 제공한 AWS Identity and Access Management(IAM) 역할을 사용하여 완료된 감사 로그를 Amazon S3 버킷에 업로드합니다.

주제

- Db2 감사 로깅 설정
- Db2 감사 로깅 관리
- 감사 로그 보기
- Db2 감사 로깅의 문제 해결

Db2 감사 로깅 설정

Amazon RDS for Db2 데이터베이스에 대한 감사 로깅을 활성화하려면 RDS for Db2 DB 인스턴스에서 `DB2_AUDIT` 옵션을 활성화합니다. 그런 다음 특정 데이터베이스에 대해 기능을 활성화하도록 감사 정책을 구성합니다. RDS for Db2 DB 인스턴스에서 옵션을 활성화하려면 `DB2_AUDIT` 옵션에 대한 옵션 설정을 구성합니다. Amazon S3 버킷의 Amazon 리소스 이름(ARN)과 버킷에 액세스할 수 있는 권한이 있는 IAM 역할을 제공하면 됩니다.

RDS for Db2 데이터베이스에 대해 Db2 감사 로깅을 설정하려면 다음 단계를 완료합니다.

주제

- 1단계: Amazon S3 버킷 생성
- 2단계: IAM 정책 생성
- 3단계: IAM 역할 생성 및 IAM 정책 연결
- 4단계: Db2 감사 로깅을 위한 옵션 그룹 구성
- 5단계: 감사 정책 구성
- 6단계: 감사 구성 확인

1단계: Amazon S3 버킷 생성

아직 생성하지 않은 경우 Amazon RDS에서 RDS for Db2 데이터베이스의 감사 로그 파일을 업로드할 Amazon S3 버킷을 생성합니다. 감사 파일의 대상으로 사용하는 S3 버킷에는 다음 제한이 적용됩니다.

- RDS for Db2 DB 인스턴스와 동일한 AWS 리전에 있어야 합니다.

- 대중에게 공개되어서는 안 됩니다.
- 버킷 소유자는 IAM 역할 소유자여야 합니다.

Amazon S3 버킷을 생성하는 방법을 알아보려면 Amazon S3 사용 설명서에서 버킷 생성을 참조하세요.

감사 로깅을 활성화하면 Amazon RDS가 자동으로 DB 인스턴스의 로그를 다음 위치로 전송합니다.

- DB 인스턴스 수준 로그 – *bucket_name*/db2-audit-logs/*dbi_resource_id*/*date_time_utc*/
- 데이터베이스 수준 로그 – *bucket_name*/db2-audit-logs/*dbi_resource_id*/*date_time_utc*/*db_name*/

버킷에 대한 Amazon 리소스 이름(ARN)을 기록합니다. 이 정보는 후속 단계를 완료하는 데 필요합니다.

2단계: IAM 정책 생성

감사 로그 파일을 DB 인스턴스에서 Amazon S3 버킷으로 전송하는 데 필요한 권한을 가진 IAM 정책을 생성합니다. 이 단계에서는 S3 버킷이 있다고 가정합니다.

정책을 생성하기 전에 다음 정보를 수집합니다.

- 버킷의 ARN.
- 버킷이 SSE-KMS 암호화를 사용하는 경우 AWS Key Management Service(AWS KMS) 키의 ARN.

다음 권한을 포함하는 IAM 정책을 생성합니다.

```
"s3:ListBucket",
 "s3:GetBucketAcl",
 "s3:GetBucketLocation",
 "s3:PutObject",
 "s3:ListMultipartUploadParts",
 "s3:AbortMultipartUpload",
 "s3:ListAllMyBuckets"
```

> **Note**
>
> Amazon RDS는 동일한 AWS 계정가 S3 버킷과 RDS for Db2 DB 인스턴스를 모두 소유하고 있는지 확인하기 위해 내부적으로 `s3:ListAllMyBuckets` 작업이 필요합니다.

버킷이 SSE-KMS 암호화를 사용하는 경우 IAM 역할 및 AWS KMS 키에 대해 다음 권한도 포함하세요.

사용자의 IAM 역할에 대한 정책에 다음 권한을 포함합니다.

```
"kms:GenerateDataKey",
 "kms:Decrypt"
```

AWS KMS 키에 대한 키 정책에 다음 권한을 포함합니다. *111122223333*을 계정 번호로*AR0A123456789EXAMPLE*을 IAM 역할 이름으로 변경합니다.

```
{
  "Sid": "Allow RDS role to use the key",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:sts::111122223333:assumed-role/AR0A123456789EXAMPLE/RDS-Db2Audit",
      "arn:aws:iam::111122223333:role/AR0A123456789EXAMPLE"
    ]
  },
  "Action": [
    "kms:GenerateDataKey",
    "kms:Decrypt"
  ],
  "Resource": "*"
}
```

AWS Management Console 또는 AWS Command Line Interface(AWS CLI)를 사용하여 IAM 정책을 생성할 수 있습니다.

콘솔

Amazon S3 버킷에 Amazon RDS 액세스를 허용하는 IAM 정책을 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.

2. 탐색 창에서 정책을 선택합니다.
3. 정책 생성을 선택한 다음 JSON을 선택합니다.
4. 작업 추가에서 S3를 기준으로 필터링합니다. ListBucket, GetBucketAcl, GetBucketLocation 액세스를 추가합니다.
5. 리소스 추가에서 추가를 선택합니다. 리소스 유형에서 버킷을 선택하고 버킷 이름을 입력합니다. 그런 다음 리소스 추가를 선택합니다.
6. 새 문 추가를 선택합니다.
7. 작업 추가에서 S3를 기준으로 필터링합니다. PutObject, ListMultipartUploadParts, AbortMultipartUpload 액세스를 추가합니다.
8. 리소스 추가에서 추가를 선택합니다. 리소스 유형에서 개체를 선택하고 *## ##/**을 입력합니다. 그런 다음 리소스 추가를 선택합니다.
9. 새 문 추가를 선택합니다.
10. 작업 추가에서 S3를 기준으로 필터링합니다. ListAllMyBuckets 액세스를 추가합니다.
11. 리소스 추가에서 추가를 선택합니다. 리소스 유형에서 모든 리소스를 선택합니다. 그런 다음 리소스 추가를 선택합니다.
12. 자체 KMS 키를 사용하여 데이터를 암호화하는 경우 다음 단계를 따릅니다.

    1. 새 문 추가를 선택합니다.
    2. 작업 추가에서 KMS별로 필터링합니다. GenerateDataKey 및 Decrypt 액세스를 추가합니다.
    3. 리소스 추가에서 추가를 선택합니다. 리소스 유형에서 모든 리소스를 선택합니다. 그런 다음 리소스 추가를 선택합니다.
13. 다음을 선택합니다.
14. 정책 이름에 이 정책의 이름을 입력합니다.
15. (선택 사항) 설명에 이 정책에 대한 설명을 입력합니다.
16. 정책 생성을 선택합니다.

AWS CLI

Amazon S3 버킷에 Amazon RDS 액세스를 허용하는 IAM 정책을 생성하려면

1. create-policy 명령을 실행합니다. 다음 예제에서는 *iam_policy_name*과 *amzn-s3-demo-bucket*을 IAM 정책의 이름과 대상 Amazon S3 버킷의 이름으로 바꿉니다.

   대상 LinuxmacOS, 또는Unix:

```
aws iam create-policy \
    --policy-name iam_policy_name \
    --policy-document '{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "Statement1",
                "Effect": "Allow",
                "Action": [
                    "s3:ListBucket",
                    "s3:GetBucketAcl",
                    "s3:GetBucketLocation"
                ],
                "Resource": [
                    "arn:aws:s3:::amzn-s3-demo-bucket"
                ]
            },
            {
                "Sid": "Statement2",
                "Effect": "Allow",
                "Action": [
                    "s3:PutObject",
                    "s3:ListMultipartUploadParts",
                    "s3:AbortMultipartUpload"
                ],
                "Resource": [
                    "arn:aws:s3:::amzn-s3-demo-bucket/*"
                ]
            },
            {
                "Sid": "Statement3",
                "Effect": "Allow",
                "Action": [
                    "s3:ListAllMyBuckets"
                ],
                "Resource": [
                    "*"
                ]
            },
            {
                "Sid": "Statement4",
                "Effect": "Allow",
                "Action": [
```

```
                    "kms:GenerateDataKey",
                    "kms:Decrypt"
                ],
                "Resource": [
                    "*"
                ]
            }
        ]
  }'
```

Windows의 경우:

```
aws iam create-policy ^
    --policy-name iam_policy_name ^
    --policy-document '{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "Statement1",
                "Effect": "Allow",
                "Action": [
                    "s3:ListBucket",
                    "s3:GetBucketAcl",
                    "s3:GetBucketLocation"
                ],
                "Resource": [
                    "arn:aws:s3:::amzn-s3-demo-bucket"
                ]
            },
            {
                "Sid": "Statement2",
                "Effect": "Allow",
                "Action": [
                    "s3:PutObject",
                    "s3:ListMultipartUploadParts",
                    "s3:AbortMultipartUpload"
                ],
                "Resource": [
                    "arn:aws:s3:::amzn-s3-demo-bucket/*"
                ]
            },
            {
                "Sid": "Statement3",
```

```
                "Effect": "Allow",
                "Action": [
                    "s3:ListAllMyBuckets"
                ],
                "Resource": [
                    "*"
                ]
            },
            {
                "Sid": "Statement4",
                "Effect": "Allow",
                "Action": [
                    "kms:GenerateDataKey",
                    "kms:Decrypt"
                ],
                "Resource": [
                    "*"
                ]
            }
        ]
}'
```

2. 정책이 생성되면 정책의 ARN을 기록해 둡니다. 3단계: IAM 역할 생성 및 IAM 정책 연결에 대한 ARN이 필요합니다.

IAM 정책 생성에 대한 자세한 내용은 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

3단계: IAM 역할 생성 및 IAM 정책 연결

이 단계에서는 2단계: IAM 정책 생성에서 IAM 정책을 생성했다고 가정합니다. 이 단계에서는 RDS for Db2 DB 인스턴스에 대한 IAM 역할을 만든 다음 역할에 IAM 정책을 연결합니다.

콘솔 또는 AWS CLI를 사용하여 DB 인스턴스에 대한 IAM 역할을 만들 수 있습니다.

콘솔

IAM 역할을 생성하여 여기에 IAM 정책을 연결하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할을 선택합니다.
3. 역할 생성(Create role)을 선택합니다.

4. 신뢰할 수 있는 엔터티 유형에 AWS 서비스를 선택합니다.
5. 서비스 또는 사용 사례의 경우 RDS를 선택한 다음 RDS – 데이터베이스에 역할 추가를 선택합니다.
6. 다음을 선택합니다.
7. 권한 정책의 경우 생성한 IAM 정책의 이름을 검색하여 선택합니다.
8. 다음을 선택합니다.
9. 역할 이름(Role name)에 역할 이름을 입력합니다.
10. (선택 사항)설명에 새 역할에 대한 설명을 입력합니다.
11. 역할 생성을 선택합니다.

AWS CLI

IAM 역할을 생성하여 여기에 IAM 정책을 연결하려면

1. create-role 명령을 실행합니다. 다음 예제에서 *iam_role_name*을 IAM 역할 이름으로 바꿉니다.

   대상 LinuxmacOS, 또는Unix:

```
aws iam create-role \
    --role-name iam_role_name \
    --assume-role-policy-document '{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "rds.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }'
```

   Windows의 경우:

```
aws iam create-role ^
    --role-name iam_role_name ^
    --assume-role-policy-document '{
      "Version": "2012-10-17",
```

```
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "rds.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }'
```

2. 역할이 생성되면 역할의 ARN을 기록합니다. 다음 4단계: Db2 감사 로깅을 위한 옵션 그룹 구성 단계에서 이 ARN을 사용합니다.
3. attach-role-policy 명령을 실행합니다. 다음 예제에서 *iam_policy_arn*을 2단계: IAM 정책 생성에서 생성한 IAM 정책의 ARN으로 대체합니다. *iam_role_name*을 방금 생성한 IAM 역할 이름으로 바꿉니다.

   대상 LinuxmacOS, 또는Unix:

```
aws iam attach-role-policy \
   --policy-arn iam_policy_arn \
   --role-name iam_role_name
```

   Windows의 경우:

```
aws iam attach-role-policy ^
   --policy-arn iam_policy_arn ^
   --role-name iam_role_name
```

자세한 내용은 IAM 사용 설명서의 IAM 사용자에게 권한을 위임하기 위한 역할 생성을 참조하십시오.

4단계: Db2 감사 로깅을 위한 옵션 그룹 구성

RDS for Db2 DB 인스턴스에 Db2 감사 로깅 옵션을 추가하는 프로세스는 다음과 같습니다.

1. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 복사 또는 수정합니다.
2. 필요한 모든 옵션을 추가하고 구성하십시오.
3. 옵션 그룹을 DB 인스턴스에 연결합니다.

Db2 감사 로깅 옵션을 추가한 후 DB 인스턴스를 재시작할 필요가 없습니다. 옵션 그룹이 활성화되면 S3 버킷에 감사를 생성하고 감사 로그를 저장할 수 있습니다.

DB 인스턴스의 옵션 그룹에 Db2 감사 로깅을 추가 및 구성하려면

1. 다음 중 하나를 선택합니다.
   - 기존 옵션 그룹을 사용합니다.
   - 사용자 지정 DB 옵션 그룹을 생성하고 해당 옵션 그룹을 사용합니다. 자세한 내용은 옵션 그룹 생성 섹션을 참조하세요.
2. 옵션 그룹에 DB2_AUDIT 옵션을 추가하고 옵션 설정을 구성합니다. 옵션 추가에 대한 자세한 내용은 옵션 그룹에 옵션 추가 섹션을 참조하세요.
   - IAM_ROLE_ARN의 경우 the section called “IAM 역할 생성 및 IAM 정책 연결”에서 생성한 IAM 역할의 ARN을 입력합니다.
   - S3_BUCKET_ARN의 경우 Db2 감사 로그에 사용할 S3 버킷의 ARN을 입력합니다. 버킷은 RDS for Db2 DB 인스턴스와 동일한 리전에 있어야 합니다. 입력한 IAM 역할과 관련된 정책은 이 리소스에 필요한 작업을 허용해야 합니다.
3. 옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스에 적용합니다. 다음 중 하나를 선택합니다.
   - 새 DB 인스턴스를 생성하는 경우, 인스턴스를 시작할 때 옵션 그룹을 적용하십시오.
   - 기존 DB 인스턴스의 경우, 해당 인스턴스를 수정한 후 새 옵션 그룹을 연결하여 옵션 그룹을 적용하십시오. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

5단계: 감사 정책 구성

RDS for Db2 데이터베이스에 대한 감사 정책을 구성하려면 RDS for Db2 DB 인스턴스에 대한 마스터 사용자 이름 및 마스터 암호를 사용하여 `rdsadmin` 데이터베이스에 연결합니다. 그런 다음 데이터베이스의 DB 이름과 해당 파라미터 값을 사용하여 `rdsadmin.configure_db_audit` 저장 프로시저를 호출합니다.

다음 예제에서는 데이터베이스에 연결하여 AUDIT, CHECKING, OBJMAINT, SECMAINT, SYSADMIN, VALIDATE 카테고리를 통해 `testdb`에 대한 감사 정책을 구성합니다. `BOTH` 상태 값은 성공과 실패를 모두 로깅하며, `ERROR TYPE`은 기본적으로 `NORMAL`입니다. 이 저장 프로시저를 사용하는 방법에 대한 자세한 내용은 the section called “rdsadmin.configure_db_audit” 섹션을 참조하세요.

```
db2 "connect to rdsadmin user master_user using master_password"
db2 "call rdsadmin.configure_db_audit('testdb', 'ALL', 'BOTH', ?)"
```

6단계: 감사 구성 확인

감사 정책이 올바르게 설정되었는지 확인하려면 감사 구성 상태를 확인합니다.

구성을 확인하려면 RDS for Db2 DB 인스턴스의 마스터 사용자 이름과 마스터 암호를 사용하여 `rdsadmin` 데이터베이스에 연결합니다. 그런 다음 데이터베이스의 DB 이름을 사용하여 다음 SQL 문을 실행합니다. 다음 예제에서 DB 이름은 *testdb*입니다.

```
db2 "select task_id, task_type, database_name, lifecycle,
    varchar(bson_to_json(task_input_params), 500) as task_params,
    cast(task_output as varchar(500)) as task_output
    from table(rdsadmin.get_task_status(null,'testdb','CONFIGURE_DB_AUDIT'))"

Sample Output

TASK_ID              TASK_TYPE            DATABASE_NAME       LIFECYCLE
-------------------- -------------------- --------------- -------------
                   2  CONFIGURE_DB_AUDIT            DB2DB       SUCCESS

... continued ...
TASK_PARAMS
------------------------------------------------------------
{ "AUDIT_CATEGORY" : "ALL", "CATEGORY_SETTING" : "BOTH" }

... continued ...
                                           TASK_OUTPUT
-------------------------------------------------------
2023-12-22T20:27:03.029Z Task execution has started.

2023-12-22T20:27:04.285Z Task execution has completed successfully.
```

Db2 감사 로깅 관리

Db2 감사 로깅을 설정한 후 특정 데이터베이스의 감사 정책을 수정하거나 데이터베이스 수준 또는 전체 DB 인스턴스에서 감사 로깅을 비활성화할 수 있습니다. 로그 파일이 업로드되는 Amazon S3 버킷을 변경할 수도 있습니다.

주제

- Db2 감사 정책 수정
- 로그 파일 위치 수정
- Db2 감사 로깅 비활성화

Db2 감사 정책 수정

특정 RDS for Db2 데이터베이스의 감사 정책을 수정하려면 `rdsadmin.configure_db_audit` 저장 프로시저를 실행합니다. 이 저장 프로시저를 사용하여 감사 정책의 카테고리, 카테고리 설정 및 오류 유형 구성을 변경할 수 있습니다. 자세한 내용은 the section called “rdsadmin.configure_db_audit” 섹션을 참조하세요.

로그 파일 위치 수정

로그 파일이 업로드되는 Amazon S3 버킷을 변경하려면 다음 중 하나를 수행합니다.

- RDS for Db2 DB 인스턴스에 연결된 현재 옵션 그룹 수정 – `DB2_AUDIT` 옵션이 새 버킷을 가리키도록 `S3_BUCKET_ARN` 설정을 업데이트합니다. 또한 연결된 옵션 그룹의 `IAM_ROLE_ARN` 설정으로 지정된 IAM 역할에 연결된 IAM 정책을 업데이트해야 합니다. 이 IAM 정책은 새 버킷에 필요한 액세스 권한을 제공해야 합니다. IAM 정책에 필요한 권한에 대한 자세한 내용은 IAM 정책 생성 섹션을 참조하세요.
- RDS for Db2 DB 인스턴스를 다른 옵션 그룹에 연결 - DB 인스턴스를 수정하여 연결된 옵션 그룹을 변경합니다. 새 옵션 그룹이 올바른 `S3_BUCKET_ARN` 및 `IAM_ROLE_ARN` 설정으로 구성되었는지 확인합니다. `DB2_AUDIT` 옵션에 대해 이러한 설정을 구성하는 방법에 대한 자세한 내용은 옵션 그룹 구성 섹션을 참조하세요.

옵션 그룹을 수정하는 경우 변경 사항을 즉시 적용해야 합니다. 자세한 내용은 the section called “DB 인스턴스 수정” 섹션을 참조하세요.

Db2 감사 로깅 비활성화

Db2 감사 로깅을 비활성화하려면 다음 중 하나를 수행합니다.

- RDS for Db2 DB 인스턴스에 대한 감사 로깅 비활성화 - DB 인스턴스를 수정하고 `DB2_AUDIT` 옵션이 포함된 옵션 그룹을 제거합니다. 자세한 내용은 the section called “DB 인스턴스 수정” 섹션을 참조하세요.
- 특정 데이터베이스에 대한 감사 로깅 비활성화 - 데이터베이스의 DB 이름으로 `rdsadmin.disable_db_audit`를 호출하여 감사 로깅을 중지하고 감사 정책을 제거합니다. 자세한 내용은 the section called “rdsadmin.disable_db_audit” 섹션을 참조하세요.

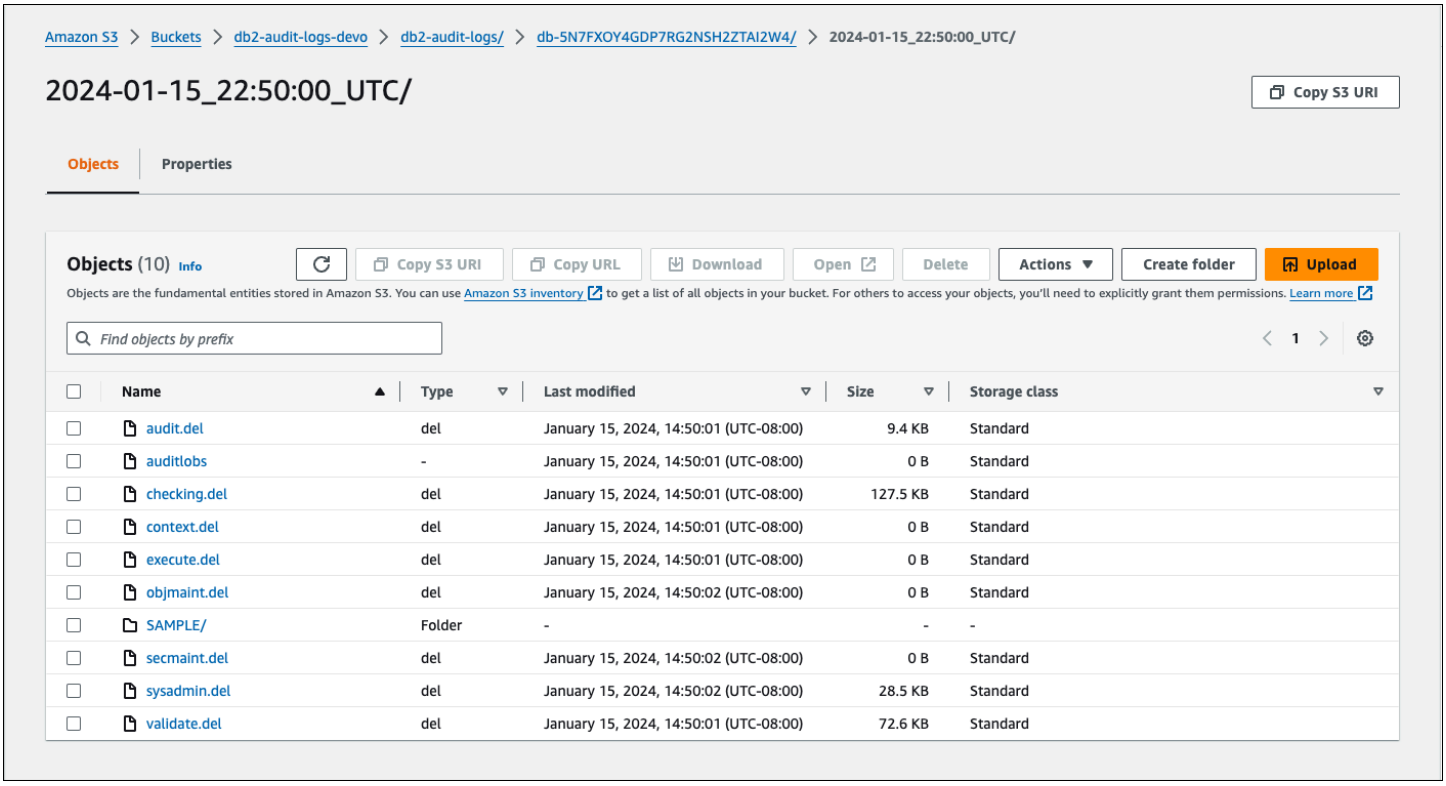
```
db2 "call rdsadmin.disable_db_audit(
    'db_name',
    ?)"
```

감사 로그 보기

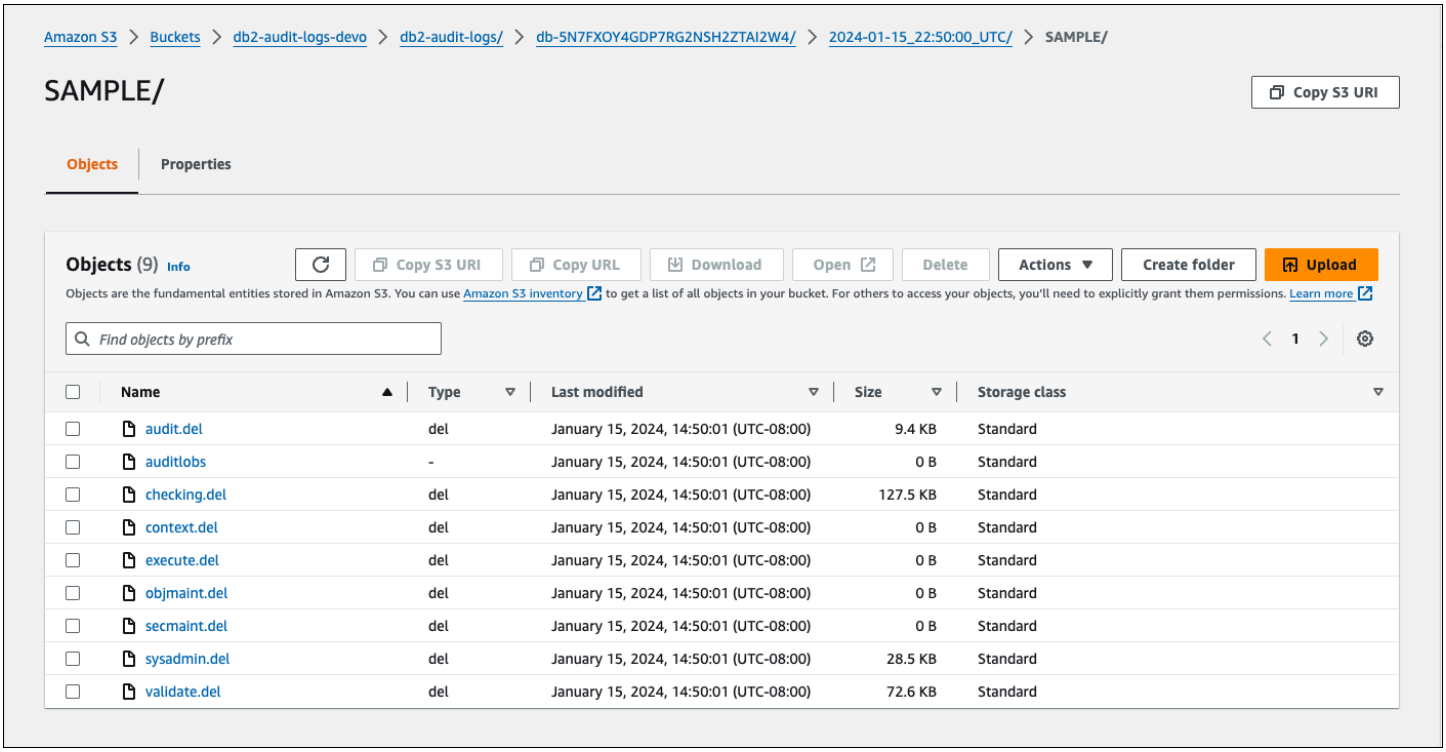
Db2 감사 로깅을 활성화한 후 Amazon S3 버킷의 감사 데이터를 보기 전에 1시간 이상 기다립니다. Amazon RDS는 RDS for Db2 DB 인스턴스의 로그를 다음 위치로 자동 전송합니다.

- DB 인스턴스 수준 로그 – *bucket_name*/db2-audit-logs/*dbi_resource_id*/*date_time_utc*/
- 데이터베이스 수준 로그 – *bucket_name*/db2-audit-logs/*dbi_resource_id*/*date_time_utc*/*db_name*/

Amazon S3 콘솔의 다음 예제 스크린샷은 RDS for Db2 DB 인스턴스 수준 로그 파일의 폴더 목록을 보여줍니다.



Amazon S3 콘솔의 다음 예제 스크린샷은 RDS for Db2 DB 인스턴스의 데이터베이스 수준 로그 파일을 보여줍니다.



Db2 감사 로깅의 문제 해결

다음 정보를 사용하여 Db2 감사 로깅과 관련된 일반적인 문제를 해결합니다.

감사 정책을 구성할 수 없음

저장 프로시저 `rdsadmin.configure_db_audit`을 호출하면 오류가 반환되는 경우 `DB2_AUDIT` 옵션이 있는 옵션 그룹이 RDS for Db2 DB 인스턴스와 연결되어 있지 않을 수 있습니다. DB 인스턴스를 수정하여 옵션 그룹을 추가한 다음 저장 프로시저를 다시 호출하세요. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

Amazon S3 버킷에 데이터 없음

Amazon S3 버킷에서 로깅 데이터가 누락된 경우 다음을 확인합니다.

- Amazon S3 버킷은 RDS for Db2 DB 인스턴스와 동일한 리전에 있습니다.
- `IAM_ROLE_ARN` 옵션 설정에서 지정한 역할은 Amazon S3 버킷에 로그를 업로드하는 데 필요한 권한으로 구성됩니다. 자세한 내용은 IAM 정책 생성 섹션을 참조하세요.
- `IAM_ROLE_ARN` 및 `S3_BUCKET_ARN` 옵션 설정의 ARN은 RDS for Db2 DB 인스턴스와 연결된 옵션 그룹에서 올바르게 표시됩니다. 자세한 내용은 옵션 그룹 구성 섹션을 참조하세요.

데이터베이스에 연결하고 SQL 문을 실행하여 감사 로깅 구성의 작업 상태를 확인할 수 있습니다. 자세한 내용은 감사 구성 확인 섹션을 참조하세요.

또한 이벤트를 확인하여 로그가 누락될 수 있는 이유에 대해 자세히 알아볼 수 있습니다. 이벤트를 보는 방법에 대한 자세한 내용은 the section called “Amazon RDS 콘솔에서 로그, 이벤트 및 스트림 보기” 섹션을 참조하세요.

Amazon RDS for Db2를 위한 외부 저장 프로시저

외부 루틴을 만들고 이를 Amazon RDS for Db2 데이터베이스에 외부 저장 프로시저로 등록할 수 있습니다. 현재 RDS for Db2는 외부 저장 프로시저에 대해 Java 기반 루틴만 지원합니다.

Java 기반 외부 저장 프로시저

Java 기반 외부 저장 프로시저는 RDS for Db2 데이터베이스에 외부 저장 프로시저로 등록하는 외부 Java 루틴입니다.

주제

- Java 기반 외부 저장 프로시저의 제한 사항
- Java 기반 외부 저장 프로시저 구성

Java 기반 외부 저장 프로시저의 제한 사항

외부 루틴을 개발하기 전에 다음 제한 사항 및 제약 조건에 유의하세요.

외부 루틴을 만들려면 Db2에서 제공하는 Java 개발 키트(JDK)를 사용해야 합니다. 자세한 내용은 Java software support for Db2 database products를 참조하세요.

Java 프로그램은 `/tmp` 디렉터리에서만 파일을 생성할 수 있으며, Amazon RDS는 이러한 파일에 대한 실행 파일 또는 Set User ID(SUID) 권한 활성화를 지원하지 않습니다. 또한 Java 프로그램은 소켓 시스템 직접 호출이나 다음 시스템 직접 호출을 사용할 수 없습니다.

- _sysctl
- acct
- afs_syscall
- bpf
- capset
- chown
- chroot
- create_module
- delete_module
- fanotify_init

- fanotify_mark
- finit_module
- fsconfig
- fsopen
- fspick
- get_kernel_syms
- getpmsg
- init_module
- mount
- move_mount
- nfsservctl
- open_by_handle_at
- open_tree
- pivot_root
- putpmsg
- query_module
- quotactl
- reboot
- security
- setdomainname
- setfsuid
- sethostname
- sysfs
- tuxcall
- umount2
- uselib
- ustat
- vhangup
- vserver

Db2의 외부 루틴에 대한 추가 제약 조건은 IBM Db2 설명서의 Restrictions on external routines를 참조하세요.

Java 기반 외부 저장 프로시저 구성

외부 저장 프로시저를 구성하려면 외부 루틴으로 .jar 파일을 만들고 RDS for Db2 데이터베이스에 설치한 다음 외부 저장 프로시저로 등록하세요.

주제

- 1단계: 외부 저장 프로시저 활성화
- 2단계: 외부 루틴을 사용하여 .jar 파일 설치
- 3단계: 외부 저장 프로시저 등록
- 4단계: 외부 저장 프로시저 검증

1단계: 외부 저장 프로시저 활성화

외부 저장 프로시저를 활성화하려면 DB 인스턴스와 연결된 사용자 정의 파라미터 그룹에서 `db2_alternate_authz_behaviour` 파라미터를 다음 값 중 하나로 설정합니다.

- `EXTERNAL_ROUTINE_DBADM` - `DBADM` 권한이 있는 모든 사용자, 그룹 또는 역할에 암시적으로 `CREATE_EXTERNAL_ROUTINE` 권한을 부여합니다.
- `EXTERNAL_ROUTINE_DBAUTH` - `DBADM` 권한이 있는 사용자가 모든 사용자, 그룹 또는 역할에 `CREATE_EXTERNAL_ROUTINE` 권한을 부여하도록 허용합니다. 이 경우 `DBADM` 권한이 있는 사용자를 포함하여 모든 사용자, 그룹 또는 역할에 이 권한이 암시적으로 부여되지 않습니다.

이 설정에 대한 자세한 내용은 IBM Db2 설명서의 GRANT (database authorities) statement를 참조하세요.

AWS Management Console, AWS CLI 또는 Amazon RDS API를 통해 사용자 지정 파라미터 그룹을 생성하고 수정할 수 있습니다.

콘솔

사용자 지정 파라미터 그룹에서 db2_alternate_authz_behaviour 파라미터를 구성하는 방법

1. DB 인스턴스에서 사용하는 것과 다른 사용자 정의 DB 파라미터 그룹을 사용하려면 새로운 DB 파라미터 그룹을 생성합니다. 기존 보유 라이선스 사용(BYOL) 모델을 사용하는 경우 새 사용자 지

정 파라미터 그룹에 IBM ID가 포함되어 있어야 합니다. 이 ID에 대한 자세한 내용은 the section called “Db2에 기존 보유 라이선스 사용을 위한 IBM ID” 섹션을 참조하세요. DB 파라미터 그룹을 생성하는 것에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹 생성 단원을 참조하십시오.

2. 사용자 지정 파라미터 그룹에서 db2_alternate_authz_behaviour 파라미터 값을 설정합니다. 파라미터 그룹 수정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요.

AWS CLI

사용자 지정 파라미터 그룹에서 db2_alternate_authz_behaviour 파라미터를 구성하는 방법

1. DB 인스턴스가 사용하는 것과 다른 사용자 지정 DB 파라미터 그룹을 사용하려면 create-db-parameter-group 명령을 실행하여 사용자 지정 파라미터 그룹을 생성합니다. 기존 보유 라이선스 사용(BYOL) 모델을 사용하는 경우 새 사용자 지정 파라미터 그룹에 IBM ID가 포함되어 있어야 합니다. 이 ID에 대한 자세한 내용은 the section called “Db2에 기존 보유 라이선스 사용을 위한 IBM ID” 섹션을 참조하세요.

   다음 필수 옵션을 포함합니다.

   - `--db-parameter-group-name` – 생성하려는 파라미터 그룹의 이름입니다.
   - `--db-parameter-group-family` – Db2 엔진 에디션 및 메이저 버전입니다. 유효 값은 `db2-se-11.5`및 `db2-ae-11.5`입니다.
   - `--description` – 이 파라미터 그룹에 대한 설명입니다.

   DB 파라미터 그룹을 생성하는 것에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹 생성 단원을 참조하십시오.

   다음 예시는 파라미터 그룹 패밀리 `db2-se-11.5`에 사용자 정의 파라미터 그룹 `MY_EXT_SP_PARAM_GROUP`을 생성하는 방법을 보여줍니다.

   대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-parameter-group \
--region us-east-1 \
--db-parameter-group-name MY_EXT_SP_PARAM_GROUP \
--db-parameter-group-family db2-se-11.5 \
--description "test db2 external routines"
```

Windows의 경우:

```
aws rds create-db-parameter-group ^
--region us-east-1 ^
--db-parameter-group-name MY_EXT_SP_PARAM_GROUP ^
--db-parameter-group-family db2-se-11.5 ^
--description "test db2 external routines"
```

2. modify-db-parameter-group 명령을 실행하여 사용자 지정 파라미터 그룹의 db2_alternate_authz_behaviour 파라미터를 수정합니다.

   다음 필수 옵션을 포함합니다.

   - --db-parameter-group-name – 생성한 파라미터 그룹의 이름입니다.
   - --parameters – 파라미터 업데이트를 위한 파라미터 이름, 값, 응용 방법으로 구성된 배열입니다.

   파라미터 그룹 수정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요.

   다음 예시는 db2_alternate_authz_behaviour의 값을 EXTERNAL_ROUTINE_DBADM으로 설정하여 파라미터 그룹 MY_EXT_SP_PARAM_GROUP을 수정하는 방법을 보여줍니다.

   대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name MY_EXT_SP_PARAM_GROUP \
    --parameters
 "ParameterName='db2_alternate_authz_behaviour',ParameterValue='EXTERNAL_ROUTINE_DBADM',App
```

   Windows의 경우:

```
aws rds modify-db-parameter-group ^
    --db-parameter-group-name MY_EXT_SP_PARAM_GROUP ^
    --parameters
 "ParameterName='db2_alternate_authz_behaviour',ParameterValue='EXTERNAL_ROUTINE_DBADM',App
```

RDS API

사용자 지정 파라미터 그룹에서 db2_alternate_authz_behaviour 파라미터를 구성하는 방법

1. DB 인스턴스에서 사용하는 것과 다른 사용자 정의 DB 파라미터 그룹을 사용하려면 Amazon RDS API CreateDBParameterGroup 작업을 사용하여 새로운 DB 파라미터 그룹을 생성합니다. 기존 보유 라이선스 사용(BYOL) 모델을 사용하는 경우 새 사용자 지정 파라미터 그룹에 IBM Db2 ID가 포함되어 있어야 합니다. 이 ID에 대한 자세한 내용은 the section called “Db2에 기존 보유 라이선스 사용을 위한 IBM ID” 섹션을 참조하세요.

   다음 필수 파라미터를 포함합니다.

   - `DBParameterGroupName`
   - `DBParameterGroupFamily`
   - `Description`

   DB 파라미터 그룹을 생성하는 것에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹 생성 단원을 참조하십시오.
2. RDS API ModifyDBParameterGroup 작업을 사용하여 생성한 사용자 지정 파라미터 그룹의 `db2_alternate_authz_behaviour` 파라미터를 수정합니다.

   다음 필수 파라미터를 포함합니다.

   - `DBParameterGroupName`
   - `Parameters`

   파라미터 그룹 수정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요.

2단계: 외부 루틴을 사용하여 .jar 파일 설치

Java 루틴을 만든 후에는 .jar 파일을 만든 다음 `db2 "call sqlj.install_jar('file:`*`file_path`*`',`*`jar_ID`*`)"`를 실행하여 RDS for Db2 데이터베이스에 설치합니다.

다음 예시는 Java 루틴을 만들어 RDS for Db2 데이터베이스에 설치하는 방법을 보여줍니다. 예시에는 프로세스를 테스트하는 데 사용할 수 있는 간단한 루틴에 대한 샘플 코드가 포함되어 있습니다. 이 예에서는 다음을 가정합니다.

- Java 코드는 Db2가 설치된 서버에서 컴파일됩니다. IBM에서 제공하는 JDK로 컴파일하지 않으면 설명할 수 없는 오류가 발생할 수 있으므로 이 방법이 가장 좋습니다.
- 서버에는 RDS for Db2 데이터베이스가 로컬로 카탈로그되어 있습니다.

다음 샘플 코드를 사용하여 프로세스를 시험해 보려면 코드를 복사한 다음 `MYJAVASP.java`라는 파일에 저장하세요.

```
import java.sql.*;
public class MYJAVASP
{
public static void my_JAVASP (String inparam) throws SQLException, Exception
{
try
{
// Obtain the calling context's connection details.
Connection myConn = DriverManager.getConnection("jdbc:default:connection");
String myQuery = "INSERT INTO TEST.TEST_TABLE VALUES (?, CURRENT DATE)";
PreparedStatement myStmt = myConn.prepareStatement(myQuery);
myStmt.setString(1, inparam);
myStmt.executeUpdate();
}
catch (SQLException sql_ex)
{
throw sql_ex;
}
catch (Exception ex)
{
throw ex;
}
}
```

다음 명령은 Java 루틴을 컴파일합니다.

```
~/sqllib/java/jdk64/bin/javac MYJAVASP.java
```

다음 명령은 .jar 파일을 생성합니다.

```
~/sqllib/java/jdk64/bin/jar cvf MYJAVASP.jar MYJAVASP.class
```

다음 명령은 이름이 `MY_DB2_DATABASE`인 데이터베이스에 연결하고 .jar 파일을 설치합니다.

```
db2 "connect to MY_DB2_DATABASE user master_username using master_password"

db2 "call sqlj.install_jar('file:/tmp/MYJAVASP.jar','MYJAVASP')"
db2 "call sqlj.refresh_classes()"
```

3단계: 외부 저장 프로시저 등록

RDS for Db2 데이터베이스에 .jar 파일을 설치한 후 db2 CREATE PROCEDURE 또는 db2 REPLACE PROCEDURE 명령을 실행하여 저장 프로시저로 등록합니다.

다음 예시는 데이터베이스에 연결하고 이전 단계에서 만든 Java 루틴을 저장 프로시저로 등록하는 방법을 보여줍니다.

```
db2 "connect to MY_DB2_DATABASE user master_username using master_password"

create procedure TESTSP.MYJAVASP (in input char(6))
specific myjavasp
dynamic result sets 0
deterministic
language java
parameter style java
no dbinfo
fenced
threadsafe
modifies sql data
program type sub
external name 'MYJAVASP!my_JAVASP';
```

4단계: 외부 저장 프로시저 검증

다음 단계를 사용하여 이전 단계에서 등록된 샘플 외부 저장 프로시저를 테스트합니다.

외부 저장 프로시저를 검증하는 방법

1. 다음 예시에서처럼 TEST.TEST_TABLE과 같은 테이블을 생성합니다.

   ```
   db2 "create table TEST.TEST_TABLE(C1 char(6), C2 date)"
   ```

2. 새로운 외부 저장 프로시저를 직접 호출합니다. 호출 시 0의 상태가 반환됩니다.

   ```
   db2 "call TESTSP.MYJAVASP('test')"
   ```

```
Return Status = 0
```

3. 1단계에서 만든 테이블을 쿼리하여 저장 프로시저 호출의 결과를 확인합니다.

```
db2 "SELECT * from TEST.TEST_TABLE"
```

이 쿼리는 다음 예시와 유사한 출력을 반환합니다.

```
C1     C2
------ ----------
test   02/05/2024
```

Amazon RDS for Db2에 대해 알려진 문제 및 제한

Amazon RDS for Db2 작업에 대해 알려진 문제 및 제한은 다음과 같습니다.

주제

- 인증 제한
- 울타리가 없는 루틴
- 마이그레이션 중 자동이 아닌 스토리지 테이블스페이스
- db2_compatibility_vector 파라미터 설정
- INVALID 패키지가 포함된 데이터베이스 마이그레이션

인증 제한

Amazon RDS는 기본적으로 db2auth를 JCC_ENFORCE_SECMEC로 설정합니다. 그러나 유선을 통해 사용자 ID 및 암호 암호화를 적용하지 않으려면 파라미터 그룹에서 db2auth 파라미터를 CLEAR_TEXT로 변경하여 이 설정을 재정의할 수 있습니다. 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 단원을 참조하십시오.

울타리가 없는 루틴

RDS for Db2는 울타리가 없는 루틴 생성과 데이터 백업 및 복원을 통한 해당 루틴의 마이그레이션을 지원하지 않습니다. 데이터베이스에 울타리가 없는 루틴이 포함되어 있는지 확인하려면 다음 SQL 명령을 실행합니다.

```
SELECT 'COUNT:' || count(*) FROM SYSCAT.ROUTINES where fenced='N' and routineschema not
 in ('SQLJ','SYSCAT','SYSFUN','SYSIBM','SYSIBMADM','SYSPROC','SYSTOOLS')
```

마이그레이션 중 자동이 아닌 스토리지 테이블스페이스

RDS for Db2는 자동이 아닌 새 스토리지 테이블스페이스 생성을 지원하지 않습니다. 데이터베이스의 일회성 마이그레이션에 기본 복원을 사용하는 경우 RDS for Db2는 자동이 아닌 스토리지 테이블스페이스를 자동 테이블스페이스로 자동 변환한 다음, 데이터베이스를 RDS for Db2로 복원합니다. 일회성 마이그레이션에 대한 자세한 내용은 Linux에서 Amazon RDS for Db2의 Linux로 마이그레이션 및 AIX 또는 Windows에서 Amazon RDS for Db2의 Linux로 마이그레이션 섹션을 참조하세요.

db2_compatibility_vector 파라미터 설정

Amazon RDS를 사용하면 DB 인스턴스를 만들 때 초기 데이터베이스를 만든 다음 연결된 파라미터 그룹에서 파라미터를 수정할 수 있습니다. 그러나 Db2의 경우 파라미터 그룹에서 `db2_compatibility_vector` 파라미터를 설정하려면 먼저 사용자 지정 파라미터 그룹에서 파라미터를 수정하고 데이터베이스 없이 DB 인스턴스를 만든 다음, `rdsadmin.create_database` 저장 프로시저를 사용하여 데이터베이스를 만들어야 합니다.

`db2_compatibility_vector` 파라미터를 설정하는 방법

1. 사용자 지정 DB 파라미터 그룹을 만듭니다. (기본 파라미터 그룹에서는 파라미터를 수정할 수 없습니다.)
2. 파라미터를 수정합니다.
3. DB 인스턴스를 만듭니다.
4. `rdsadmin.create_database` 저장 프로시저를 사용하여 데이터베이스를 만듭니다.
5. 데이터베이스가 포함된 DB 인스턴스와 파라미터 그룹을 연결합니다.

INVALID 패키지가 포함된 데이터베이스 마이그레이션

`RESTORE` 명령을 사용하여 `INVALID` 패키지가 포함된 Db2 데이터베이스를 RDS for Db2로 마이그레이션하면 데이터베이스 사용을 시작할 때 문제가 발생할 수 있습니다. `INVALID` 패키지는 DB 인스턴스 사용자 `rdsdb`의 권한 부여 설정 및 `PUBLIC`에서 권한 부여 제거로 인해 문제를 일으킬 수 있습니다. `INVALID` 패키지로 인해 다음 명령이 실패합니다.

- `db2updv115`
- `db2 "call SYSPROC.ADMIN_REVALIDATE_DB_OBJECTS()"`

`RESTORE` 명령을 사용하여 데이터베이스를 마이그레이션하기 전에 다음 명령을 실행하여 데이터베이스에 `INVALID` 패키지가 포함되어 있지 않은지 확인합니다.

```
db2 "select count(*) from sysibm.sysroutines where valid = 'N'"
```

명령이 0보다 큰 개수를 반환하는 경우 다음 명령을 직접적으로 호출합니다.

```
db2 "call SYSPROC.ADMIN_REVALIDATE_DB_OBJECTS()"
```

그런 다음 이전 명령을 직접적으로 호출하여 데이터베이스에 INVALID 패키지가 더 이상 포함되어 있지 않은지 확인합니다.

```
db2 "select count(*) from sysibm.sysroutines where valid = 'N'"
```

이제 데이터베이스를 백업하여 RDS for Db2 DB 인스턴스로 복원할 준비가 되었습니다.

Amazon RDS for Db2 저장 프로시저 참조

기본 제공 저장 프로시저를 직접 호출하여 Db2 엔진을 실행하는 Amazon RDS for Db2 DB 인스턴스를 관리할 수 있습니다.

저장 프로시저	범주	설명
???	데이터베이스 수	`rdsadmin.activate_database` 저장 프로시저를 사용하여 독립형 RDS for Db2 DB 인스턴스에서 데이터베이스를 활성화합니다.
the section called “rdsadmin.add_groups”	권한 부여 및 취소	`rdsadmin.add_groups` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 대해 사용자에게 하나 이상의 그룹을 추가합니다.
the section called “rdsadmin.add_user”	권한 부여 및 취소	`rdsadmin.add_user` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 대해 권한 부여 목록에 사용자를 추가합니다.
the section called “rdsadmin.alter_bufferpool”	버퍼 풀	`rdsadmin.alter_bufferpool` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 대해 버퍼 풀을 수정합니다.
the section called “rdsadmin.alter_tablespace”	테이블스페이스	`rdsadmin.alter_tablespace` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 대해 테이블스페이스를 수정합니다.
the section called “rdsadmin.catalog_storage_access”	스토리지 액세스	`rdsadmin.catalog_storage_access` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 대해 Db2 데이터 파일이 포함된 Amazon S3 버킷에 액세스하기 위한 스토리지 별칭을 카탈로그화합니다.
the section called “rdsadmin.change_password”	권한 부여 및 취소	`rdsadmin.change_password` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 대해 사용자 암호를 변경합니다.

저장 프로시저	범주	설명
the section called “rdsadmin.complete_rollforward”	데이터베이스 수	`rdsadmin.complete_rollforward` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스를 `ROLL-FORWARD PENDING` 상태에서 온라인 상태로 전환합니다. the section called “rdsadmin.rollforward_database”를 직접 호출했지만 `complete_rollforward` 파라미터를 `FALSE`로 설정하면 `ROLL-FORWARD PENDING` 상태가 발생합니다.
the section called “rdsadmin.configure_db_audit”	감사 정책	`rdsadmin.configure_db_audit` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 대해 감사 정책을 수정합니다. 감사 정책이 없는 경우 이 저장 프로시저를 실행하면 감사 정책이 만들어집니다.
the section called “rdsadmin.create_bufferpool”	버퍼 풀	`rdsadmin.create_bufferpool` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 대해 버퍼 풀을 만듭니다.
the section called “rdsadmin.create_database”	데이터베이스 수	`rdsadmin.create_database` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스를 만듭니다.
the section called “rdsadmin.create_role”	권한 부여 및 취소	`rdsadmin.create_role` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 연결할 역할을 만듭니다.
the section called “rdsadmin.create_tablespace”	테이블스페이스	`rdsadmin.create_tablespace` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 대해 테이블스페이스를 만듭니다.

저장 프로시저	범주	설명
the section called “rdsadmin.db2pd_command”	데이터베이스 수	`rdsadmin.db2pd_command` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 대한 정보를 수집합니다. 이 정보는 RDS for Db2 데이터베이스를 모니터링하고 문제를 해결하는 데 도움이 될 수 있습니다.
the section called “rdsadmin.dbadm_grant”	권한 부여 및 취소	`rdsadmin.dbadm_grant` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 대해 하나 이상의 역할, 사용자 또는 그룹에 하나 이상의 권한 부여 유형(`DBADM`, `ACCESSCTRL` 또는 `DATAACCESS`)을 부여합니다.
the section called “rdsadmin.dbadm_revoke”	권한 부여 및 취소	`rdsadmin.dbadm_revoke` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 대해 하나 이상의 역할, 사용자 또는 그룹에서 하나 이상의 권한 부여 유형(`DBADM`, `ACCESSCTRL` 또는 `DATAACCESS`)을 취소합니다.
???	데이터베이스 수	`rdsadmin.deactivate_database` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스를 비활성화합니다. 데이터베이스를 비활성화하여 메모리 리소스를 절약할 수 있습니다.
the section called “rdsadmin.disable_db_audit”	감사 정책	`rdsadmin.disable_db_audit` 저장 프로시저를 사용하여 감사 로깅을 중지하고 RDS for Db2 DB 인스턴스의 데이터베이스에서 감사 정책을 제거합니다.
the section called “rdsadmin.drop_bufferpool”	버퍼 풀	`rdsadmin.drop_bufferpool` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에서 버퍼 풀을 삭제합니다.

저장 프로시저	범주	설명
the section called “rdsadmin.drop_database”	데이터베이스 수	`rdsadmin.drop_database` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스를 삭제합니다.
???	권한 부여 및 취소	`rdsadmin.drop_role` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에서 역할을 삭제합니다.
the section called “rdsadmin.drop_tablespace”	테이블스페이스	`rdsadmin.drop_tablespace` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에서 테이블스페이스를 삭제합니다.
the section called “rdsadmin.force_application”	데이터베이스 수	유지 관리를 수행하기 위해 `rdsadmin.force_application` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에서 애플리케이션을 강제 종료합니다.
the section called “rdsadmin.grant_role”	권한 부여 및 취소	`rdsadmin.grant_role` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에서 권한 피부여자 역할, 사용자 또는 그룹에 역할을 할당합니다. 또한 이 저장 프로시저를 사용하여 권한 피부여자에게 역할을 할당할 수 있는 DBADM 권한을 부여할 수 있습니다.
???	데이터베이스 수	`rdsadmin.list_archive_log_information` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 대해 아카이브 로그 관련 정보를 반환합니다. 이 정보에는 개별 로그 파일의 크기 및 만든 날짜, 아카이브 로그 파일에 사용되는 총 스토리지와 같은 세부 정보가 포함됩니다.
the section called “rdsadmin.list_users”	권한 부여 및 취소	`rdsadmin.list_users` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 대해 권한 부여 목록에 있는 사용자 목록을 반환합니다.

저장 프로시저	범주	설명
???	데이터베이스 수	데이터베이스 구성을 변경한 후 `rdsadmin.reactivate_database` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스에서 데이터베이스를 재활성화합니다. 독립형 DB 인스턴스에 있는 데이터베이스의 경우 이 저장 프로시저 또는 ??? 저장 프로시저를 사용할 수 있습니다.
the section called “rdsadmin.remove_groups”	권한 부여 및 취소	`rdsadmin.remove_groups` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 대해 사용자로부터 하나 이상의 그룹을 제거합니다.
the section called “rdsadmin.remove_user”	권한 부여 및 취소	`rdsadmin.remove_user` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 대해 권한 부여 목록에서 사용자를 제거합니다.
the section called “rdsadmin.rename_tablespace”	테이블스페이스	`rdsadmin.rename_tablespace` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 대해 테이블스페이스의 이름을 변경합니다.
the section called “rdsadmin.restore_database”	데이터베이스 수	`rdsadmin.restore_database` 저장 프로시저를 사용하여 Amazon S3 버킷에서 RDS for Db2 DB 인스턴스의 데이터베이스를 복원합니다.
the section called “rdsadmin.revoke_role”	권한 부여 및 취소	`rdsadmin.revoke_role` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에서 권한 피부여자 역할, 사용자 또는 그룹의 역할을 취소합니다.

저장 프로시저	범주	설명
the section called “rdsadmin.rollforward_database”	데이터베이스 수	`rdsadmin.rollforward_database` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스를 온라인 상태로 전환하고 the section called “rdsadmin.restore_database”를 직접적으로 호출하여 RDS for Db2 DB 인스턴스를 복원한 후 트랜잭션 로그를 적용합니다.
the section called “rdsadmin.set_archive_log_retention”	데이터베이스 수	`rdsadmin.set_archive_log_retention` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 대해 아카이브 로그 파일을 보존할 기간을 구성합니다. 이 저장 프로시저를 사용하여 아카이브 로그 보존을 비활성화할 수도 있습니다.
the section called “rdsadmin.set_configuration”	데이터베이스 수	`rdsadmin.set_configuration` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 대해 특정 설정을 구성합니다.
the section called “rdsadmin.show_archive_log_retention”	데이터베이스 수	`rdsadmin.show_archive_log_retention` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 대해 현재 아카이브 로그 보존 설정을 반환합니다.
the section called “rdsadmin.show_configuration”	데이터베이스 수	`rdsadmin.show_configuration` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 대해 수정할 수 있는 하나 이상의 설정을 반환합니다.
the section called “rdsadmin.uncatalog_storage_access”	스토리지 액세스	`rdsadmin.uncatalog_storage_access` 저장 프로시저를 사용하여 Db2 데이터 파일이 포함된 Amazon S3 버킷에 액세스하기 위한 스토리지 별칭을 제거합니다.

저장 프로시저	범주	설명
the section called “rdsadmin.update_db_param”	데이터베이스 수	`rdsadmin.update_db_param` 저장 프로시저를 사용하여 RDS for Db2 DB 인스턴스의 데이터베이스에 대해 데이터베이스 파라미터를 업데이트합니다.

주제

- Amazon RDS for Db2 저장 프로시저 고려 사항
- 권한 부여 및 취소
- RDS for Db2 감사 정책에 대해 저장된 프로시저
- RDS for Db2용 버퍼 풀에 대해 저장된 프로시저
- 데이터베이스 관리
- RDS for Db2의 스토리지 액세스에 대해 저장된 프로시저
- RDS for Db2의 테이블스페이스에 대해 저장된 프로시저

Amazon RDS for Db2 저장 프로시저 고려 사항

Db2 엔진을 실행하는 RDS for Db2 DB 인스턴스에 Amazon RDS 시스템 저장 프로시저를 사용하기 전에 다음 정보를 검토하세요.

- 저장 프로시저를 실행하기 전에 RDS for Db2 DB 인스턴스의 마스터 사용자로 `rdsadmin` 데이터베이스에 먼저 연결해야 합니다. 다음 예제에서 *`master_username`* 및 *`master_password`*를 사용자 자체 정보로 대체합니다.

  ```
  db2 "connect to rdsadmin user master_username using master_password"
  ```

- 저장 프로시저는 ERR_MESSAGE 파라미터를 반환하는데, 이 파라미터는 저장 프로시저가 성공적으로 실행되었는지 여부와 성공적으로 실행되지 않은 이유를 나타냅니다.

  예시

  다음 예제는 저장 프로시저가 성공적으로 실행되었음을 나타냅니다.

  ```
  Parameter Name : ERR_MESSAGE
  Parameter Value : -
  Return Status = 0
  ```

  다음 예제는 저장 프로시저에 사용된 Amazon S3 버킷 이름이 유효하지 않아 저장 프로시저가 성공적으로 실행되지 않았음을 나타냅니다.

  ```
  Parameter Name : ERR_MESSAGE
  Parameter Value : Invalid S3 bucket name
  Return Status = -1006
  ```

저장 프로시저를 호출할 때 반환되는 오류 메시지는 the section called “저장된 프로시저 오류” 섹션을 참조하세요.

저장 프로시저의 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

권한 부여 및 취소

다음 저장 프로시저는 Amazon RDS for Db2 데이터베이스에 대한 권한을 부여하고 취소합니다. 이 프로시저를 실행하려면 마스터 사용자가 먼저 `rdsadmin` 데이터베이스에 연결되어야 합니다.

주제

- rdsadmin.create_role
- rdsadmin.grant_role
- rdsadmin.revoke_role
- rdsadmin.add_user
- rdsadmin.change_password
- rdsadmin.list_users
- rdsadmin.remove_user
- rdsadmin.add_groups
- rdsadmin.remove_groups
- rdsadmin.dbadm_grant
- rdsadmin.dbadm_revoke

rdsadmin.create_role

역할을 생성합니다.

구문

```
db2 "call rdsadmin.create_role(
    'database_name',
    'role_name')"
```

파라미터

다음 파라미터는 필수 파라미터입니다.

`database_name`

명령이 실행될 데이터베이스의 이름입니다. 데이터 형식은 `varchar`입니다.

role_name

생성할 역할의 이름입니다. 데이터 형식은 varchar입니다.

사용 노트

역할 생성 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

예시

다음 예제에서는 DB2DB 데이터베이스에 대해 MY_ROLE 역할을 생성합니다.

```
db2 "call rdsadmin.create_role(
    'DB2DB',
    'MY_ROLE')"
```

rdsadmin.grant_role

역할, 사용자 또는 그룹에 역할을 할당합니다.

구문

```
db2 "call rdsadmin.grant_role(
    ?,
    'database_name',
    'role_name',
    'grantee',
    'admin_option')"
```

파라미터

다음 출력 파라미터는 필수입니다.

?

작업의 고유 식별자를 출력하는 파라미터 마커입니다. 이 파라미터는 ?만 허용합니다.

다음 입력 파라미터는 필수입니다.

database_name

명령이 실행될 데이터베이스의 이름입니다. 데이터 형식은 varchar입니다.

role_name

생성할 역할의 이름입니다. 데이터 형식은 varchar입니다.

grantee

권한을 받을 역할, 사용자 또는 그룹입니다. 데이터 형식은 varchar입니다. 유효한 값: ROLE, USER, GROUP, PUBLIC.

형식은 값 뒤에 이름이 와야 합니다. 값과 이름이 여러 개인 경우 쉼표로 구분합니다. 예: 'USER *user1*, *user2*, GROUP *group1*, *group2*'. 이름을 사용자의 정보로 대체합니다.

다음 입력 파라미터는 선택 사항입니다.

admin_option

피부여자 ROLE에 역할을 할당할 DBADM 권한이 있는지 여부를 지정합니다. 데이터 형식은 char입니다. 기본값은 N입니다.

사용 노트

역할 할당 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

예시

다음 예제에서는 데이터베이스 TESTDB의 ROLE_TEST라는 역할을 role1이라는 역할, user1이라는 사용자, group1이라는 그룹에 할당합니다. ROLE_TEST에는 역할을 할당할 수 있는 관리 권한이 부여됩니다.

```
db2 "call rdsadmin.grant_role(
    ?,
    'TESTDB',
    'ROLE_TEST',
    'ROLE role1, USER user1, GROUP group1',
    'Y')"
```

다음 예시에서는 데이터베이스 TESTDB의 ROLE_TEST라는 역할을 PUBLIC에 할당합니다. ROLE_TEST에는 역할을 할당할 수 있는 관리 권한이 없습니다.

```
db2 "call rdsadmin.grant_role(
```

```
    ?,
    'TESTDB',
    'ROLE_TEST',
    'PUBLIC')"
```

rdsadmin.revoke_role

역할, 사용자 또는 그룹에서 역할을 취소합니다.

구문

```
db2 "call rdsadmin.revoke_role(
    ?,
    'database_name',
    'role_name',
    'grantee')"
```

파라미터

다음 출력 파라미터는 필수입니다.

?

작업의 고유 식별자를 출력하는 파라미터 마커입니다. 이 파라미터는 ?만 허용합니다.

다음 입력 파라미터는 필수입니다.

database_name

명령이 실행될 데이터베이스의 이름입니다. 데이터 형식은 varchar입니다.

role_name

취소할 역할의 이름입니다. 데이터 형식은 varchar입니다.

grantee

권한을 잃을 역할, 사용자 또는 그룹입니다. 데이터 형식은 varchar입니다. 유효한 값: ROLE, USER, GROUP, PUBLIC.

형식은 값 뒤에 이름이 와야 합니다. 값과 이름이 여러 개인 경우 쉼표로 구분합니다. 예: 'USER *user1*, *user2*, GROUP *group1*, *group2*'. 이름을 사용자의 정보로 대체합니다.

사용 노트

역할 할당 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

예시

다음 예제는 role1이라는 역할, user1이라는 사용자 및 group1이라는 그룹에서 TESTDB 데이터베이스의 ROLE_TEST라는 역할을 취소합니다.

```
db2 "call rdsadmin.revoke_role(
    ?,
    'TESTDB',
    'ROLE_TEST',
    'ROLE role1, USER user1, GROUP group1')"
```

다음 예제에서는 PUBLIC에서 TESTDB 데이터베이스의 ROLE_TEST라는 역할을 취소합니다.

```
db2 "call rdsadmin.revoke_role(
    ?,
    'TESTDB',
    'ROLE_TEST',
    'PUBLIC')"
```

rdsadmin.add_user

권한 부여 목록에 사용자를 추가합니다.

구문

```
db2 "call rdsadmin.add_user(
    'username',
    'password',
    'group_name,group_name')"
```

파라미터

다음 파라미터는 필수 파라미터입니다.

##

사용자의 사용자 이름입니다. 데이터 형식은 varchar입니다.

password

사용자의 암호입니다. 데이터 형식은 varchar입니다.

다음 파라미터는 선택 사항입니다.

group_name

사용자를 추가할 그룹의 이름입니다. 데이터 형식은 varchar입니다. 기본값은 빈 문자열 또는 null입니다.

사용 노트

그룹 이름을 쉼표로 구분하여 하나 이상의 그룹에 사용자를 추가할 수 있습니다.

새 사용자를 만들거나 기존 사용자에게 그룹을 추가할 때 그룹을 만들 수 있습니다. 혼자서 그룹을 만들 수는 없습니다.

> **Note**
>
> rdsadmin.add_user를 호출하여 추가할 수 있는 최대 사용자 수는 5,000명입니다.

사용자 추가 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

예시

다음 예시에서는 jorge_souza라는 사용자를 만들고 해당 사용자를 sales 및 inside_sales라는 그룹에 할당합니다.

```
db2 "call rdsadmin.add_user(
    'jorge_souza',
    '*******',
    'sales,inside_sales')"
```

rdsadmin.change_password

사용자의 암호를 변경합니다.

구문

```
db2 "call rdsadmin.change_password(
    'username',
    'new_password')"
```

파라미터

다음 파라미터는 필수 파라미터입니다.

##

사용자의 사용자 이름입니다. 데이터 형식은 varchar입니다.

new_password

사용자의 새 암호입니다. 데이터 형식은 varchar입니다.

사용 노트

암호 변경 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

예시

다음 예제에서는 jorge_souza의 암호를 변경합니다.

```
db2 "call rdsadmin.change_password(
    'jorge_souza',
    '*******')"
```

rdsadmin.list_users

권한 부여 목록에 있는 사용자를 나열합니다.

구문

```
db2 "call rdsadmin.list_users()"
```

사용 노트

사용자 나열 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

rdsadmin.remove_user

권한 목록에서 사용자를 제거합니다.

구문

```
db2 "call rdsadmin.remove_user('username')"
```

파라미터

다음 파라미터는 필수입니다.

##

사용자의 사용자 이름입니다. 데이터 형식은 varchar입니다.

사용 노트

사용자 제거 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

예시

다음 예제에서는 RDS for Db2 DB 인스턴스의 데이터베이스에 액세스할 수 없도록 jorge_souza를 제거합니다.

```
db2 "call rdsadmin.remove_user('jorge_souza')"
```

rdsadmin.add_groups

사용자에게 그룹을 추가합니다.

구문

```
db2 "call rdsadmin.add_groups(
    'username',
    'group_name,group_name')"
```

파라미터

다음 파라미터는 필수 파라미터입니다.

##

사용자의 사용자 이름입니다. 데이터 형식은 varchar입니다.

group_name

사용자를 추가할 그룹의 이름입니다. 데이터 형식은 varchar입니다. 기본값은 빈 문자열입니다.

사용 노트

그룹 이름을 쉼표로 구분하여 사용자에게 하나 이상의 그룹을 추가할 수 있습니다. 그룹 추가 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

예시

다음 예제에서는 사용자 jorge_souza에 direct_sales 및 b2b_sales 그룹을 추가합니다.

```
db2 "call rdsadmin.add_groups(
    'jorge_souza',
    'direct_sales,b2b_sales')"
```

rdsadmin.remove_groups

사용자로부터 그룹을 제거합니다.

구문

```
db2 "call rdsadmin.remove_groups(
    'username',
    'group_name,group_name')"
```

파라미터

다음 파라미터는 필수 파라미터입니다.

##

사용자의 사용자 이름입니다. 데이터 형식은 varchar입니다.

group_name

사용자를 제거할 그룹의 이름입니다. 데이터 형식은 varchar입니다.

사용 노트

그룹 이름을 쉼표로 구분하여 사용자로부터 하나 이상의 그룹을 제거할 수 있습니다.

그룹 제거 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

예시

다음 예제에서는 사용자 `jorge_souza`로부터 `direct_sales` 및 `b2b_sales` 그룹을 제거합니다.

```
db2 "call rdsadmin.remove_groups(
    'jorge_souza',
    'direct_sales,b2b_sales')"
```

rdsadmin.dbadm_grant

역할, 사용자 또는 그룹에 DBADM, ACCESSCTRL 또는 DATAACCESS 권한을 부여합니다.

구문

```
db2 "call rdsadmin.dbadm_grant(
    ?,
    'database_name',
    'authorization',
    'grantee')"
```

파라미터

다음 출력 파라미터는 필수입니다.

?

작업의 고유 식별자를 출력하는 파라미터 마커입니다. 이 파라미터는 ?만 허용합니다.

다음 입력 파라미터는 필수입니다.

database_name

명령이 실행될 데이터베이스의 이름입니다. 데이터 형식은 varchar입니다.

##

부여할 권한의 유형입니다. 데이터 형식은 varchar입니다. 유효한 값: DBADM, ACCESSCTRL, DATAACCESS.

유형이 여러 개인 경우 쉼표로 구분합니다.

grantee

권한을 받을 역할, 사용자 또는 그룹입니다. 데이터 형식은 varchar입니다. 유효한 값: ROLE, USER, GROUP.

형식은 값 뒤에 이름이 와야 합니다. 값과 이름이 여러 개인 경우 쉼표로 구분합니다. 예: 'USER *user1*, *user2*, GROUP *group1*, *group2*'. 이름을 사용자의 정보로 대체합니다.

사용 노트

액세스 권한을 받을 역할이 존재해야 합니다.

데이터베이스 관리자 액세스 권한 부여의 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

예시

다음 예제에서는 ROLE_DBA 역할에 대해 TESTDB라는 이름의 데이터베이스에 대한 데이터베이스 관리자 액세스 권한을 부여합니다.

```
db2 "call rdsadmin.dbadm_grant(
    ?,
    'TESTDB',
    'DBADM',
    'ROLE ROLE_DBA')"
```

다음 예제에서는 user1 및 group1에 대해 TESTDB라는 이름의 데이터베이스에 대한 데이터베이스 관리자 액세스 권한을 부여합니다.

```
db2 "call rdsadmin.dbadm_grant(
    ?,
    'TESTDB',
    'DBADM',
    'USER user1, GROUP group1')"
```

다음 예제에서는 user1, user2, group1, group2에 대해 TESTDB라는 이름의 데이터베이스에 대한 데이터베이스 관리자 액세스 권한을 부여합니다.

```
db2 "call rdsadmin.dbadm_grant(
```

```
    ?,
    'TESTDB',
    'DBADM',
    'USER user1, user2, GROUP group1, group2')"
```

rdsadmin.dbadm_revoke

역할, 사용자 또는 그룹에서 DBADM, ACCESSCTRL 또는 DATAACCESS 권한을 취소합니다.

구문

```
db2 "call rdsadmin.dbadm_revoke(
    ?,
    'database_name',
    'authorization',
    'grantee')"
```

파라미터

다음 출력 파라미터는 필수입니다.

?

작업의 고유 별자입니다. 이 파라미터는 ?만 허용합니다.

다음 입력 파라미터는 필수입니다.

database_name

명령이 실행될 데이터베이스의 이름입니다. 데이터 형식은 varchar입니다.

##

취소할 권한의 유형입니다. 데이터 형식은 varchar입니다. 유효한 값: DBADM, ACCESSCTRL, DATAACCESS.

유형이 여러 개인 경우 쉼표로 구분합니다.

grantee

권한을 취소할 역할, 사용자 또는 그룹입니다. 데이터 형식은 varchar입니다. 유효한 값: ROLE, USER, GROUP.

형식은 값 뒤에 이름이 와야 합니다. 값과 이름이 여러 개인 경우 쉼표로 구분합니다. 예: 'USER *user1*, *user2*, GROUP *group1*, *group2*'. 이름을 사용자의 정보로 대체합니다.

사용 노트

데이터베이스 관리자 액세스 권한 취소의 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

예시

다음 예제에서는 ROLE_DBA 역할에 대해 TESTDB라는 이름의 데이터베이스에 대한 데이터베이스 관리자 액세스 권한을 취소합니다.

```
db2 "call rdsadmin.dbadm_revoke(
    ?,
    'TESTDB',
    'DBADM',
    'ROLE ROLE_DBA')"
```

다음 예제에서는 user1 및 group1에 대해 TESTDB라는 이름의 데이터베이스에 대한 데이터베이스 관리자 액세스 권한을 취소합니다.

```
db2 "call rdsadmin.dbadm_revoke(
    ?,
    'TESTDB',
    'DBADM',
    'USER user1, GROUP group1')"
```

다음 예제에서는 user1, user2, group1, group2에 대해 TESTDB라는 이름의 데이터베이스에 대한 데이터베이스 관리자 액세스 권한을 취소합니다.

```
db2 "call rdsadmin.dbadm_revoke(
    ?,
    'TESTDB',
    'DBADM',
    'USER user1, user2, GROUP group1, group2')"
```

RDS for Db2 감사 정책에 대해 저장된 프로시저

이 주제에 설명된 기본 제공 저장 프로시저는 감사 로깅을 사용하는 Amazon RDS for Db2 데이터베이스의 감사 정책을 관리합니다. 자세한 내용은 the section called “Db2 감사 로깅” 단원을 참조하십시오. 이 프로시저를 실행하려면 마스터 사용자가 먼저 `rdsadmin` 데이터베이스에 연결되어야 합니다.

구문, 파라미터, 사용 정보 및 예시에 대한 자세한 내용은 다음 기본 제공 저장 프로시저를 참조하세요.

저장 프로시저

- rdsadmin.configure_db_audit
- rdsadmin.disable_db_audit

rdsadmin.configure_db_audit

*db_name*으로 지정된 RDS for Db2 데이터베이스의 감사 정책을 구성합니다. 구성 중인 정책이 존재하지 않는 경우 이 저장 프로시저를 호출하면 정책이 생성됩니다. 이 정책이 있는 경우 이 저장 프로시저를 호출하면 입력한 파라미터 값으로 해당 저장 프로시저가 수정됩니다.

구문

```
db2 "call rdsadmin.configure_db_audit(
    'db_name',
    'category',
    'category_setting',
    '?')"
```

파라미터

다음 파라미터는 필수 파라미터입니다.

db_name

감사 정책을 구성하는 데 사용할 RDS for Db2 데이터베이스의 DB 이름입니다. 데이터 형식은 `varchar`입니다.

category

감사 정책을 구성할 카테고리의 이름입니다. 데이터 형식은 `varchar`입니다. 이 파라미터에 유효한 값은 다음과 같습니다.

- `ALL` – `ALL`을 사용하면 Amazon RDS에는 `CONTEXT`, `EXECUTE` 또는 `ERROR` 카테고리가 포함되지 않습니다.

- AUDIT
- CHECKING
- CONTEXT
- ERROR
- EXECUTE – 이 카테고리는 데이터를 포함하거나 포함하지 않고 구성할 수 있습니다. 데이터 수단을 사용하면 모든 호스트 변수 및 파라미터 마커에 제공된 입력 데이터 값도 기록할 수 있습니다. 기본값은 데이터 없음입니다. 자세한 내용은 *category_setting* 파라미터에 대한 설명 및 the section called “예시” 섹션을 참조하세요.
- OBJMAINT
- SECMAINT
- SYSADMIN
- VALIDATE

이러한 카테고리에 대한 자세한 내용은 IBM Db2 설명서를 참조하세요.

category_setting

지정된 감사 카테고리의 설정입니다. 데이터 형식은 varchar입니다.

다음 테이블에는 카테고리별로 유효한 카테고리 설정 값이 나와 있습니다.

범주	유효한 카테고리 설정
ALL	BOTH|FAILURE|SUCCESS|NONE
AUDIT	
CHECKING	
CONTEXT	
OBJMAINT	
SECMAINT	
SYSADMIN	
VALIDATE	

범주	유효한 카테고리 설정
ERROR	AUDIT|NORMAL 입니다. 기본값은 NORMAL입니다.
EXECUTE	BOTH,WITH|BOTH,WITHOUT|FAILURE,WITH|FAILURE,WITHOUT|SUCCESS,WITH|SUCCESS,WITHOUT|NONE

사용 노트

rdsadmin.configure_db_audit을 호출하기 전에 감사 정책을 구성하는 대상 데이터베이스가 있는 RDS for Db2 DB 인스턴스가 DB2_AUDIT 옵션이 있는 옵션 그룹과 연결되어 있는지 확인하세요. 자세한 내용은 the section called “Db2 감사 로깅 설정” 단원을 참조하십시오.

감사 정책을 구성한 후 감사 구성 확인의 단계에 따라 데이터베이스의 감사 구성 상태를 확인할 수 있습니다.

category 파라미터에 ALL을 지정한 경우 CONTEXT, EXECUTE 또는 ERROR 카테고리는 포함되지 않습니다. 이러한 카테고리를 감사 정책에 추가하려면 추가하려는 각 카테고리에서 rdsadmin.configure_db_audit을 개별적으로 호출하세요. 자세한 내용은 the section called “예시” 단원을 참조하십시오.

예시

다음 예제는 TESTDB라는 데이터베이스의 감사 정책을 만들거나 수정합니다. 예제 1~5에서 ERROR 카테고리가 이전에 구성되지 않은 경우 이 카테고리는 NORMAL(기본값)로 설정됩니다. 해당 설정을 AUDIT으로 변경하려면 Example 6: Specifying the ERROR category을 따르세요.

예 1: **ALL** 카테고리 지정

```
db2 "call rdsadmin.configure_db_audit('TESTDB', 'ALL', 'BOTH', ?)"
```

이 예제에서 호출은 감사 정책의 AUDIT, CHECKING, OBJMAINT, SECMAINT, SYSADMIN, VALIDATE 카테고리를 구성합니다. BOTH로 지정하면 각 카테고리에 대해 성공한 이벤트와 실패한 이벤트를 모두 감사합니다.

예 2: 데이터로 **EXECUTE** 카테고리 지정

```
db2 "call rdsadmin.configure_db_audit('TESTDB', 'EXECUTE', 'SUCCESS,WITH', ?)"
```

이 예제에서는 호출이 감사 정책의 EXECUTE 카테고리를 구성합니다. SUCCESS,WITH로 지정하면, 이 카테고리의 로그에 성공한 이벤트만 포함되고 호스트 변수 및 파라미터 마커에 제공된 입력 데이터 값이 포함됩니다.

예 3: 데이터 없이 **EXECUTE** 카테고리 지정

```
db2 "call rdsadmin.configure_db_audit('TESTDB', 'EXECUTE', 'FAILURE,WITHOUT', ?)"
```

이 예제에서는 호출이 감사 정책의 EXECUTE 카테고리를 구성합니다. FAILURE,WITHOUT로 지정하면, 이 카테고리의 로그에 실패한 이벤트만 포함되고 호스트 변수 및 파라미터 마커에 제공된 입력 데이터 값이 포함되지 않습니다.

예 4: 상태 이벤트 없이 **EXECUTE** 카테고리 지정

```
db2 "call rdsadmin.configure_db_audit('TESTDB', 'EXECUTE', 'NONE', ?)"
```

이 예제에서는 호출이 감사 정책의 EXECUTE 카테고리를 구성합니다. NONE으로 지정하면 이 카테고리의 어떤 이벤트도 감사되지 않습니다.

예 5: **OBJMAINT** 카테고리 지정

```
db2 "call rdsadmin.configure_db_audit('TESTDB', 'OBJMAINT', 'NONE', ?)"
```

이 예제에서는 호출이 감사 정책의 OBJMAINT 카테고리를 구성합니다. NONE으로 지정하면 이 카테고리의 어떤 이벤트도 감사되지 않습니다.

예 6: **ERROR** 카테고리 지정

```
db2 "call rdsadmin.configure_db_audit('TESTDB', 'ERROR', 'AUDIT', ?)"
```

이 예제에서는 호출이 감사 정책의 ERROR 카테고리를 구성합니다. AUDIT으로 지정하면 감사 로깅 자체에서 발생하는 오류를 포함하여 모든 오류가 로그에 캡처됩니다. 기본 오류 유형은 NORMAL입니다. NORMAL을 사용하면 감사에서 생성된 오류는 무시되고 수행 중인 작업과 관련된 오류에 대한 SQLCODE만 캡처됩니다.

rdsadmin.disable_db_audit

*db_name*으로 지정된 RDS for Db2 데이터베이스에 대한 감사 로깅을 중지하고 해당 데이터베이스에 대해 구성된 감사 정책을 제거합니다.

> **Note**
>
> 이 저장 프로시저는 the section called “rdsadmin.configure_db_audit” 호출을 통해 구성된 감사 정책만 제거합니다.

구문

```
db2 "call rdsadmin.disable_db_audit('db_name', ?)"
```

파라미터

다음 파라미터는 필수 파라미터입니다.

db_name

감사 로깅을 비활성화할 RDS for Db2 데이터베이스의 DB 이름입니다. 데이터 형식은 `varchar`입니다.

사용 노트

`rdsadmin.disable_db_audit`을 호출해도 RDS for Db2 DB 인스턴스에 대한 감사 로깅은 비활성화되지 않습니다. DB 인스턴스 수준에서 감사 로깅을 비활성화하려면 DB 인스턴스에서 옵션 그룹을 제거하세요. 자세한 내용은 Db2 감사 로깅 비활성화 단원을 참조하십시오.

예시

다음 예제는 `TESTDB` 이름이 지정된 데이터베이스에 대한 감사 로깅을 비활성화합니다.

```
db2 "call rdsadmin.disable_db_audit('TESTDB', ?)"
```

RDS for Db2용 버퍼 풀에 대해 저장된 프로시저

이 주제에 설명된 기본 제공 저장 프로시저는 Amazon RDS for Db2 데이터베이스의 버퍼 풀을 관리합니다. 이 프로시저를 실행하려면 마스터 사용자가 먼저 `rdsadmin` 데이터베이스에 연결되어야 합니다.

이러한 저장 프로시저는 다양한 작업에 사용됩니다. 단, 이 목록이 전부는 아닙니다.

- 버퍼 풀의 일반적인 작업
- 성능 보고서 생성
- db2look을 사용하여 데이터베이스 메타데이터 복사
- IBM Db2 Data Management Console을 위한 리포지토리 데이터베이스 만들기

구문, 파라미터, 사용 정보 및 예시에 대한 자세한 내용은 다음 기본 제공 저장 프로시저를 참조하세요.

저장 프로시저

- rdsadmin.create_bufferpool
- rdsadmin.alter_bufferpool
- rdsadmin.drop_bufferpool

rdsadmin.create_bufferpool

버퍼 풀을 생성합니다.

구문

```
db2 "call rdsadmin.create_bufferpool(
    'database_name',
    'buffer_pool_name',
    buffer_pool_size,
    'immediate',
    'automatic',
    page_size,
    number_block_pages,
    block_size)"
```

파라미터

다음 파라미터는 필수 파라미터입니다.

database_name

명령이 실행될 데이터베이스의 이름입니다. 데이터 형식은 varchar입니다.

buffer_pool_name

생성할 버퍼 풀의 이름입니다. 데이터 형식은 varchar입니다.

다음 파라미터는 선택적입니다.

buffer_pool_size

버퍼 풀의 크기입니다(페이지 수). 데이터 형식은 integer입니다. 기본값은 -1입니다.

immediate

명령을 즉시 실행할지 여부를 지정합니다. 데이터 형식은 char입니다. 기본값은 Y입니다.

automatic

버퍼 풀을 자동으로 설정할지를 지정합니다. 데이터 형식은 char입니다. 기본값은 Y입니다.

page_size

버퍼 풀의 페이지 크기입니다. 데이터 형식은 integer입니다. 유효한 값: 4096, 8192, 16384, 32768. 기본값은 8192입니다.

number_block_pages

버퍼 풀의 블록 페이지 수입니다. 데이터 형식은 integer입니다. 기본값은 0입니다.

block_size

블록 페이지의 블록 크기입니다. 데이터 형식은 integer입니다. 유효한 값: 2 ~ 256. 기본값은 32입니다.

사용 노트

버퍼 풀 생성 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

예시

예시 1: 기본 파라미터를 사용하여 버퍼 풀 만들기

다음 예제에서는 기본 파라미터를 사용하는 TESTDB 데이터베이스에 대해 버퍼 풀 BP8을 생성하므로, 버퍼 풀은 8KB 페이지 크기를 사용합니다.

```
db2 "call rdsadmin.create_bufferpool(
    'TESTDB',
    'BP8')"
```

예시 2: 자동 할당으로 즉시 실행할 버퍼 풀 만들기

다음 예시에서는 초기 페이지 수가 1,000인 16KB 페이지 크기를 사용하고 자동으로 설정되는 TESTDB 데이터베이스에 대해 버퍼 풀 BP16을 생성합니다. Db2는 명령을 즉시 실행합니다. 초기 페이지 수를 -1로 사용하는 경우 Db2는 페이지 자동 할당을 사용합니다.

```
db2 "call rdsadmin.create_bufferpool(
    'TESTDB',
    'BP16',
    1000,
    'Y',
    'Y',
    16384)"
```

예시 3: 블록 페이지를 사용하여 즉시 실행할 버퍼 풀 만들기

다음 예시에서는 TESTDB라는 데이터베이스에 대한 BP16이라는 버퍼 풀을 생성합니다. 이 버퍼 풀의 페이지 크기는 16KB이고 초기 페이지 수는 10,000입니다. Db2는 블록 크기가 512인 블록 500페이지를 사용하여 명령을 즉시 실행합니다.

```
db2 "call rdsadmin.create_bufferpool(
    'TESTDB',
    'BP16',
    10000,
    'Y',
    'Y',
    16384,
    500,
    512)"
```

rdsadmin.alter_bufferpool

버퍼 풀을 변경합니다.

구문

```
db2 "call rdsadmin.alter_bufferpool(
    'database_name',
    'buffer_pool_name',
    buffer_pool_size,
    'immediate',
    'automatic',
    change_number_blocks,
    number_block_pages,
    block_size)"
```

파라미터

다음 파라미터는 필수 파라미터입니다.

database_name

명령이 실행될 데이터베이스의 이름입니다. 데이터 형식은 varchar입니다.

buffer_pool_name

변경할 버퍼 풀의 이름입니다. 데이터 형식은 varchar입니다.

buffer_pool_size

버퍼 풀의 크기입니다(페이지 수). 데이터 형식은 integer입니다.

다음 파라미터는 선택적입니다.

immediate

명령을 즉시 실행할지 여부를 지정합니다. 데이터 형식은 char입니다. 기본값은 Y입니다.

automatic

버퍼 풀을 자동으로 설정할지를 지정합니다. 데이터 형식은 char입니다. 기본값은 N입니다.

change_number_blocks

버퍼 풀의 블록 페이지 수가 변경되었는지 여부를 지정합니다. 데이터 형식은 char입니다. 기본값은 N입니다.

number_block_pages

버퍼 풀의 블록 페이지 수입니다. 데이터 형식은 integer입니다. 기본값은 0입니다.

block_size

블록 페이지의 블록 크기입니다. 데이터 형식은 integer입니다. 유효한 값: 2 ~ 256. 기본값은 32입니다.

사용 노트

버퍼 풀 변경 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

예시

다음 예시에서는 TESTDB 데이터베이스의 버퍼 풀 BP16을 비자동으로 변경하고 크기를 10,000페이지로 변경합니다. Db2는 이 명령을 즉시 실행합니다.

```
db2 "call rdsadmin.alter_bufferpool(
    'TESTDB',
    'BP16',
    10000,
    'Y',
    'N')"
```

rdsadmin.drop_bufferpool

버퍼 풀을 삭제합니다.

구문

```
db2 "call rdsadmin.drop_bufferpool(
    'database_name',
    'buffer_pool_name'"
```

파라미터

다음 파라미터는 필수 파라미터입니다.

database_name

버퍼 풀이 속한 데이터베이스의 이름입니다. 데이터 형식은 varchar입니다.

buffer_pool_name

삭제할 버퍼 풀의 이름입니다. 데이터 형식은 varchar입니다.

사용 노트

버퍼 풀 삭제 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

예시

다음 예제에서는 TESTDB라는 데이터베이스에 대해 호출된 BP16 버퍼 풀을 삭제합니다.

```
db2 "call rdsadmin.drop_bufferpool(
    'TESTDB',
    'BP16')"
```

데이터베이스 관리

다음 저장 프로시저는 Amazon RDS for Db2의 데이터베이스를 관리합니다. 이 프로시저를 실행하려면 마스터 사용자가 먼저 `rdsadmin` 데이터베이스에 연결되어야 합니다.

주제

- rdsadmin.create_database
- rdsadmin.drop_database
- rdsadmin.update_db_param
- rdsadmin.set_configuration
- rdsadmin.show_configuration
- rdsadmin.restore_database
- rdsadmin.rollforward_database
- rdsadmin.complete_rollforward
- rdsadmin.db2pd_command
- rdsadmin.force_application
- rdsadmin.set_archive_log_retention
- rdsadmin.show_archive_log_retention

rdsadmin.create_database

데이터베이스를 생성합니다.

구문

```
db2 "call rdsadmin.create_database('database_name')"
```

파라미터

> **Note**
>
> 이 저장 프로시저는 필수 파라미터의 조합을 검증하지 않습니다. rdsadmin.get_task_status를 직접적으로 호출하면 `database_codeset`, `database_territory`, `database_collation`의 유효하지 않은 조합으로 인해 사용자 정의 함수가 오류를 반환할

수 있습니다. 자세한 내용은 IBM Db2 설명서에서 데이터베이스의 코드 페이지, 지역 및 데이터 정렬 선택을 참조하세요.

다음 파라미터는 필수입니다.

database_name

생성할 데이터베이스의 이름입니다. 데이터 형식은 varchar입니다.

다음 파라미터는 선택적입니다.

database_page_size

데이터베이스의 기본 페이지 크기입니다. 유효한 값: 4096, 8192, 16384, 32768. 데이터 형식은 integer입니다. 기본값은 8192입니다.

> **Important**
>
> Amazon RDS는 4KiB, 8KiB 및 16KiB 페이지에 대한 쓰기 원자성을 지원합니다. 반면 32KiB 페이지는 쓰기가 찢기거나 데스크에 데이터가 일부만 기록될 위험이 있습니다. 32KiB 페이지를 사용하는 경우 시점 복구와 자동 백업을 활성화하는 것이 좋습니다. 그렇지 않으면 찢긴 페이지를 복구하지 못할 위험이 있습니다. 자세한 내용은 the section called "백업 소개" 및 the section called "시점 복구" 단원을 참조하세요.

database_code_set

데이터베이스의 코드 집합입니다. 데이터 형식은 varchar입니다. 기본값은 UTF-8입니다.

database_territory

데이터베이스의 두 문자 국가 코드입니다. 데이터 형식은 varchar입니다. 기본값은 US입니다.

database_collation

데이터베이스에 저장된 문자열을 정렬하고 비교하는 방법을 결정하는 데이터 정렬 순서입니다. 데이터 형식은 varchar입니다.

유효한 값:

- COMPATIBILITY - IBM Db2 버전 2 데이터 정렬 시퀀스.

- EBCDIC_819_037 - ISO 라틴 코드 페이지, 데이터 정렬, CCSID 037(EBCDIC 미국 영어).
- EBCDIC_819_500 - ISO 라틴 코드 페이지, 데이터 정렬, CCSID 500(EBCDIC 인터내셔널).
- EBCDIC_850_037 - ASCII 라틴 코드 페이지, 데이터 정렬, CCSID 037(EBCDIC 미국 영어).
- EBCDIC_850_500 - ASCII 라틴 코드 페이지, 데이터 정렬, CCSID 500(EBCDIC 인터내셔널).
- EBCDIC_932_5026 - ASCII 일본어 코드 페이지, 데이터 정렬, CCSID 037(EBCDIC 미국 영어).
- EBCDIC_932_5035 - ASCII 일본어 코드 페이지, 데이터 정렬, CCSID 500(EBCDIC 인터내셔널).
- EBCDIC_1252_037 - Windows 라틴 코드 페이지, 데이터 정렬, CCSID 037(EBCDIC 미국 영어).
- EBCDIC_1252_500 - Windows 라틴 코드 페이지, 데이터 정렬, CCSID 500(EBCDIC 인터내셔널).
- IDENTITY - 기본 데이터 정렬. 문자열은 바이트 단위로 비교됩니다.
- IDENTITY_16BIT - UTF-16 호환성 인코딩 스키마: 8비트(CESU-8) 데이터 정렬 시퀀스. 자세한 내용은 유니코드 컨소시엄 웹 사이트의 유니코드 기술 보고서 #26을 참조하세요.
- NLSCHAR - 태국 코드 페이지(CP874)에서만 사용할 수 있습니다.
- SYSTEM - SYSTEM을 사용하는 경우 데이터베이스는 database_codeset 및 database_territory의 데이터 정렬 시퀀스를 자동으로 사용합니다.

기본값은 IDENTITY입니다.

또한 RDS for Db2는 language-aware-collation 및 locale-sensitive-collation 데이터 정렬 그룹을 지원합니다. 자세한 내용은 IBM Db2 설명서의 유니코드 데이터베이스의 데이터 정렬 선택을 참조하세요.

database_autoconfigure_str

AUTOCONFIGURE 명령 구문입니다(예: 'AUTOCONFIGURE APPLY DB'). 데이터 형식은 varchar입니다. 기본값은 빈 문자열 또는 null입니다.

자세한 내용은 IBM Db2 설명서의 AUTOCONFIGURE 명령을 참조하세요.

사용 노트

Amazon RDS 콘솔 또는 AWS CLI를 사용하여 RDS for Db2 DB 인스턴스를 만들 때 데이터베이스 이름을 지정하지 않은 경우 rdsadmin.create_database를 호출하여 데이터베이스를 만들 수 있습니다 자세한 내용은 DB 인스턴스 생성 단원을 참조하십시오.

특별 고려 사항:

- Db2 인스턴스로 전송된 CREATE DATABASE 명령은 RESTRICTIVE 옵션을 사용합니다.
- RDS for Db2는 AUTOMATIC STORAGE만 사용합니다.
- RDS for Db2는 NUMSEGS 및 DFT_EXTENT_SZ의 기본값을 사용합니다.
- RDS for Db2는 스토리지 암호화를 사용하며 데이터베이스 암호화를 지원하지 않습니다.

이러한 고려 사항에 대한 자세한 내용은 IBM Db2 설명서의 CREATE DATABASE 명령을 참조하세요.

rdsadmin.create_database를 호출하기 전에 rdsadmin 데이터베이스에 연결해야 합니다. 다음 예제에서 *master_username* 및 *master_password*를 RDS for Db2 DB 인스턴스의 정보로 대체합니다.

```
db2 connect to rdsadmin user master_username using master_password
```

데이터베이스 생성 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

예시

다음 예제에서는 일본에 대한 *database_code_set*, *database_territory*, *database_collation* 파라미터의 올바른 조합을 사용하여 TESTJP라는 데이터베이스를 생성합니다.

```
db2 "call rdsadmin.create_database('TESTJP', 4096, 'IBM-437', 'JP', 'SYSTEM')"
```

rdsadmin.drop_database

데이터베이스를 삭제합니다.

구문

```
db2 "call rdsadmin.drop_database('database_name')"
```

파라미터

다음 파라미터는 필수입니다.

database_name

삭제할 데이터베이스의 이름입니다. 데이터 형식은 varchar입니다.

사용 노트

다음 조건이 충족되면 rdsadmin.drop_database만 직접 호출하여 데이터베이스를 삭제할 수 있습니다.

- Amazon RDS 콘솔 또는 AWS CLI를 사용하여 RDS for Db2 DB 인스턴스를 만들 때 데이터베이스 이름을 지정하지 않았습니다. 자세한 내용은 DB 인스턴스 생성 단원을 참조하십시오.
- the section called “rdsadmin.create_database” 저장 프로시저를 호출하여 데이터베이스를 만들었습니다.
- the section called “rdsadmin.restore_database” 저장 프로시저를 호출하여 오프라인 또는 백업 이미지에서 데이터베이스를 복원했습니다.

rdsadmin.drop_database를 호출하기 전에 rdsadmin 데이터베이스에 연결해야 합니다. 다음 예제에서 *master_username* 및 *master_password*를 RDS for Db2 DB 인스턴스의 정보로 대체합니다.

```
db2 connect to rdsadmin user master_username using master_password
```

데이터베이스 삭제 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

예시

다음 예제에서는 TESTDB라는 데이터베이스를 삭제합니다.

```
db2 "call rdsadmin.drop_database('TESTDB')"
```

응답 예제

잘못된 데이터베이스 이름을 전달하면 저장 프로시저가 다음 응답 예제를 반환합니다.

```
SQL0438N Application raised error or warning with diagnostic text: "Cannot
drop database. Database with provided name does not exist". SQLSTATE=99993
```

Amazon RDS 콘솔을 사용하거나 AWS CLI를 사용하여 데이터베이스를 생성한 경우 저장 프로시저는 다음 응답 예제를 반환합니다.

```
Return Status = 0
```

`Return Status = 0` 수신 후 the section called “rdsadmin.get_task_status” 저장 프로시저를 호출합니다. 다음 예제와 비슷한 응답이 상태를 설명합니다.

```
1 ERROR DROP_DATABASE RDSDB 2023-10-10-16.33.03.744122 2023-10-10-16.33.30.143797 -
 2023-10-10-16.33.30.098857 Task execution has started.
2023-10-10-16.33.30.143797 Caught exception during executing task id 1, Aborting task.
Reason Dropping database created via rds CreateDBInstance api is not allowed.
Only database created using rdsadmin.create_database can be dropped
```

rdsadmin.update_db_param

데이터베이스 파라미터를 업데이트합니다.

구문

```
db2 "call rdsadmin.update_db_param(
    'database_name',
    'parameter_to_modify',
    'changed_value')"
```

파라미터

다음 파라미터는 필수 파라미터입니다.

database_name

작업을 실행할 데이터베이스의 이름입니다. 데이터 형식은 varchar입니다.

parameter_to_modify

수정할 파라미터의 이름입니다. 데이터 형식은 varchar입니다. 자세한 내용은 Amazon RDS for Db2 파라미터 단원을 참조하십시오.

changed_value

파라미터 값을 변경할 값입니다. 데이터 형식은 varchar입니다.

사용 노트

데이터베이스 파라미터 업데이트 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

예시

다음 예제에서는 TESTDB라는 데이터베이스의 archretrydelay 파라미터를 100으로 업데이트합니다.

```
db2 "call rdsadmin.update_db_param(
    'TESTDB',
    'archretrydelay',
    '100')"
```

다음 예제에서는 종속성 검사를 피하기 위해 TESTDB라는 데이터베이스에서 생성된 객체의 검증을 연기합니다.

```
db2 "call rdsadmin.update_db_param(
    'TESTDB',
    'auto_reval',
    'deferred_force')"
```

응답 예제

지원되지 않거나 수정할 수 없는 데이터베이스 구성 파라미터를 수정하려고 하면 저장 절차가 다음 응답 예제를 반환합니다.

```
SQL0438N Application raised error or warning with diagnostic text: "Parameter
is either not supported or not modifiable to customers". SQLSTATE=99993
```

rdsadmin.set_configuration

데이터베이스의 특정 설정을 구성합니다.

구문

```
db2 "call rdsadmin.set_configuration(
    'name',
    'value')"
```

파라미터

다음 파라미터는 필수 파라미터입니다.

##

구성 설정의 이름입니다. 데이터 형식은 varchar입니다.

USD ##

구성 설정 값입니다. 데이터 형식은 varchar입니다.

사용 노트

다음 테이블에는 rdsadmin.set_configuration으로 제어할 수 있는 구성 설정이 나와 있습니다.

명칭	설명
RESTORE_DATABASE_NUM_BUFFERS	복원 작업 중에 생성할 버퍼 개수입니다. 이 값은 DB 인스턴스 클래스의 총 메모리 크기보다 작아야 합니다. 이 설정이 구성되지 않은 경우 Db2는 복원 작업 중에 사용할 값을 결정합니다. 자세한 내용은 IBM Db2 설명서를 참조하십시오.
RESTORE_DATABASE_PARALLELISM	복원 작업 중에 생성할 버퍼 조작기 개수입니다. 이 값은 DB 인스턴스의 vCPU 개수보다 두 배 미만이어야 합니다. 이 설정이 구성되지 않은 경우 Db2는 복원 작업 중에 사용할 값을 결정합니다. 자세한 내용은 IBM Db2 설명서를 참조하십시오.

예시

다음 예제에서는 RESTORE_DATABASE_PARALLELISM 구성을 8로 설정합니다.

```
db2 "call rdsadmin.set_configuration(
    'RESTORE_DATABASE_PARALLELISM',
    '8')"
```

다음 예제에서는 RESTORE_DATABASE_NUM_BUFFERS 구성을 150으로 설정합니다.

```
db2 "call rdsadmin.set_configuration(
    'RESTORE_DATABASE_NUM_BUFFERS',
    '150')"
```

rdsadmin.show_configuration

저장 프로시저 rdsadmin.set_configuration을 사용하여 설정할 수 있는 현재 설정을 반환합니다.

구문

```
db2 "call rdsadmin.show_configuration(
    'name')"
```

파라미터

다음 파라미터는 선택 사항입니다.

##

정보를 반환할 구성 설정의 이름입니다. 데이터 형식은 varchar입니다.

유효한 구성 이름은 다음과 같습니다.

- RESTORE_DATABASE_NUM_BUFFERS – 복원 작업 중에 생성할 버퍼의 개수입니다.
- RESTORE_DATABASE_PARALLELISM – 복원 작업 중에 생성할 버퍼 조작기 개수입니다.

사용 노트

구성 설정의 이름을 지정하지 않으면 rdsadmin.show_configuration이 저장 프로시저 rdsadmin.set_configuration을 사용하여 설정할 수 있는 모든 구성 설정에 대한 정보를 반환합니다.

예시

다음 예제는 현재 RESTORE_DATABASE_PARALLELISM 구성에 관한 정보를 반환합니다.

```
db2 "call rdsadmin.show_configuration(
    'RESTORE_DATABASE_PARALLELISM')"
```

rdsadmin.restore_database

데이터베이스를 복원합니다.

구문

```
db2 "call rdsadmin.restore_database(
    ?,
    'database_name',
    's3_bucket_name',
    's3_prefix',
    restore_timestamp,
    'backup_type')"
```

파라미터

다음 출력 파라미터는 필수입니다.

?

오류 메시지를 출력하는 파라미터 표시자입니다. 이 파라미터는 ?만 허용합니다.

다음 입력 파라미터는 필수입니다.

database_name

복원할 데이터베이스의 이름입니다. 이 이름은 백업 이미지의 데이터베이스 이름과 일치해야 합니다. 데이터 형식은 varchar입니다.

s3_bucket_name

백업이 있는 Amazon S3 버킷의 이름입니다. 데이터 형식은 varchar입니다.

s3_prefix

다운로드 중 파일 일치 작업에 사용할 접두사입니다. 데이터 형식은 varchar입니다.

이 파라미터가 비어 있는 경우 Amazon S3 버킷의 모든 파일이 다운로드됩니다. 다음은 접두사의 예제입니다.

```
backupfolder/SAMPLE.0.rdsdb.DBPART000.20230615010101
```

restore_timestamp

데이터베이스 백업 이미지의 타임스탬프입니다. 데이터 형식은 varchar입니다.

타임스탬프는 백업 파일 이름에 포함됩니다. 예를 들어, 20230615010101은 파일 이름 SAMPLE.0.rdsdb.DBPART000.20230615010101.001의 타임스탬프입니다.

backup_type

백업 유형입니다. 데이터 형식은 varchar입니다. 유효한 값: OFFLINE, ONLINE.

가동 중지 시간이 거의 없는 마이그레이션에는 ONLINE을 사용합니다. 자세한 내용은 Linux에서 Amazon RDS for Db2의 가동 중지 시간이 거의 없는 Linux로 마이그레이션 단원을 참조하십시오.

사용 노트

Amazon RDS 콘솔 또는 AWS CLI를 사용하여 RDS for Db2 DB 인스턴스를 만들 때 데이터베이스 이름을 지정하지 않은 경우 rdsadmin.restore_database를 호출하여 데이터베이스를 복원할 수 있습니다 자세한 내용은 DB 인스턴스 생성 단원을 참조하십시오.

데이터베이스를 복원하기 전에 RDS for Db2 DB 인스턴스의 스토리지 공간을 백업 크기와 디스크의 원래 Db2 데이터베이스 크기를 합한 값 이상으로 프로비저닝해야 합니다. 백업을 복원할 때 Amazon RDS는 RDS for Db2 DB 인스턴스에서 백업 파일을 추출합니다.

각 백업 파일은 5TB 이하여야 합니다. 백업 파일이 5TB를 초과하면 해당 백업 파일을 더 작은 크기의 파일들로 나누어야 합니다.

rdsadmin.restore_database 저장 프로시저를 사용해 모든 파일을 복원하도록 하려면 파일 이름의 타임스탬프 뒤에 파일 번호 접미사를 포함하지 않습니다. 예를 들어, *s3_prefix* backupfolder/SAMPLE.0.rdsdb.DBPART000.20230615010101은 다음 파일을 복원합니다.

```
SAMPLE.0.rdsdb.DBPART000.20230615010101.001
SAMPLE.0.rdsdb.DBPART000.20230615010101.002
SAMPLE.0.rdsdb.DBPART000.20230615010101.003
SAMPLE.0.rdsdb.DBPART000.20230615010101.004
SAMPLE.0.rdsdb.DBPART000.20230615010101.005
```

데이터베이스 복원 작업의 성능을 향상시키기 위해 RDS에서 사용할 버퍼 및 버퍼 조작자의 개수를 구성할 수 있습니다. 현재 구성을 확인하려면 the section called “rdsadmin.show_configuration”을 사용합니다. 구성을 변경하려면 the section called “rdsadmin.set_configuration”을 사용합니다.

데이터베이스 복원 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

데이터베이스를 복원한 후 데이터베이스를 온라인 상태로 전환하고 추가 트랜잭션 로그를 적용하려면 rdsadmin.rollforward_database 섹션을 참조하세요.

예시

다음 예제에서는 *s3_prefix* backupfolder/SAMPLE.0.rdsdb.DBPART000.20230615010101이 있는 단일 파일 또는 여러 파일을 사용하여 오프라인 백업을 복원합니다.

```
db2 "call rdsadmin.restore_database(
    ?,
    'SAMPLE',
    'amzn-s3-demo-bucket',
    'backupfolder/SAMPLE.0.rdsdb.DBPART000.20230615010101',
    20230615010101,
    'OFFLINE')"
```

rdsadmin.rollforward_database

rdsadmin.restore_database를 호출하여 데이터베이스를 복원한 후 데이터베이스를 온라인 상태로 만들고 추가 트랜잭션 로그를 적용합니다.

구문

```
db2 "call rdsadmin.rollforward_database(
    ?,
    'database_name',
    's3_bucket_name',
    s3_prefix,
    'rollfoward_to_option',
    'complete_rollforward')"
```

파라미터

다음 출력 파라미터는 필수입니다.

?

오류 메시지를 출력하는 파라미터 표시자입니다. 이 파라미터는 ?만 허용합니다.

다음 입력 파라미터는 필수입니다.

database_name

작업을 수행할 데이터베이스의 이름입니다. 데이터 형식은 varchar입니다.

s3_bucket_name

백업이 있는 Amazon S3 버킷의 이름입니다. 데이터 형식은 varchar입니다.

s3_prefix

다운로드 중 파일 일치 작업에 사용할 접두사입니다. 데이터 형식은 varchar입니다.

이 파라미터가 비어 있는 경우 S3 버킷의 모든 파일이 다운로드됩니다. 다음은 접두사 예제입니다.

```
backupfolder/SAMPLE.0.rdsdb.DBPART000.20230615010101
```

입력 다음 파라미터는 선택적입니다.

rollforward_to_option

롤포워드하려는 지점입니다. 데이터 형식은 varchar입니다. 유효한 값: END_OF_LOGS, END_OF_BACKUP. 기본값은 END OF LOGS입니다.

complete_rollforward

롤포워드 프로세스를 완료할지 여부를 지정합니다. 데이터 형식은 varchar입니다. 기본값은 TRUE입니다.

TRUE인 경우 완료 후 데이터베이스가 온라인 상태이고 액세스할 수 있습니다. FALSE인 경우 데이터베이스는 ROLL-FORWARD PENDING 상태가 유지됩니다.

사용 노트

rdsadmin.restore_database를 호출한 후에는 rollforward_database를 호출하여 S3 버킷의 아카이브 로그를 적용해야 합니다. 이 저장 프로시저를 사용하여 rdsadmin.restore_database 호출 후 추가 트랜잭션 로그를 복원할 수도 있습니다.

complete_rollforward를 FALSE로 설정하면 데이터베이스가 ROLL-FORWARD PENDING 상태이고 오프라인 상태입니다. 데이터베이스를 온라인 상태로 만들려면 rdsadmin.complete_rollforward를 직접 호출해야 합니다.

데이터베이스의 롤포워드 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

예시

다음 예제에서는 트랜잭션 로그가 있는 데이터베이스의 온라인 백업으로 롤포워드한 다음 데이터베이스를 온라인 상태로 만듭니다.

```
db2 "call rdsadmin.rollforward_database(
    ?,
    null,
    null,
    'END_OF_LOGS',
    'TRUE')"
```

다음 예제에서는 트랜잭션 로그가 없는 데이터베이스의 온라인 백업으로 롤포워드한 다음 데이터베이스를 온라인 상태로 만듭니다.

```
db2 "call rdsadmin.rollforward_database(
    ?,
    'TESTDB',
    'amzn-s3-demo-bucket',
    'logsfolder/,
    'END_OF_BACKUP',
    'TRUE')"
```

다음 예제에서는 트랜잭션 로그가 있는 데이터베이스의 온라인 백업으로 롤포워드한 다음 데이터베이스를 온라인 상태로 만들지 않습니다.

```
db2 "call rdsadmin.rollforward_database(
    ?,
    'TESTDB',
    null,
    'onlinebackup/TESTDB',
    'END_OF_LOGS',
    'FALSE')"
```

다음 예제에서는 추가 트랜잭션 로그가 있는 데이터베이스의 온라인 백업으로 롤포워드한 다음 데이터베이스를 온라인 상태로 만들지 않습니다.

```
db2 "call rdsadmin.rollforward_database(
    ?,
    'TESTDB',
    'amzn-s3-demo-bucket',
```

```
    'logsfolder/S0000155.LOG',
    'END_OF_LOGS',
    'FALSE')"
```

rdsadmin.complete_rollforward

데이터베이스를 ROLL-FORWARD PENDING 상태에서 온라인 상태로 만듭니다.

구문

```
db2 "call rdsadmin.complete_rollforward(
    ?,
    'database_name')"
```

파라미터

다음 출력 파라미터는 필수입니다.

?

오류 메시지를 출력하는 파라미터 표시자입니다. 이 파라미터는 ?만 허용합니다.

다음 입력 파라미터는 필수입니다.

database_name

온라인으로 가져올 데이터베이스의 이름입니다. 데이터 형식은 varchar입니다.

사용 노트

complete_rollforward가 FALSE로 설정된 상태로 rdsadmin.rollforward_database를 호출한 경우 데이터베이스는 ROLL-FORWARD PENDING 상태이며 오프라인 상태입니다. 롤포워드 프로세스를 완료하고 데이터베이스를 온라인 상태로 전환하려면 rdsadmin.complete_rollforward를 호출하세요.

롤포워드 프로세스 완료 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

예시

다음 예제에서는 TESTDB 데이터베이스를 온라인 상태로 전환합니다.

```
db2 "call rdsadmin.complete_rollfoward(
    ?,
    'TESTDB')"
```

rdsadmin.db2pd_command

RDS for Db2 데이터베이스 정보를 수집합니다.

구문

```
db2 "call rdsadmin.db2pd_command('db2pd_cmd')"
```

파라미터

다음 입력 파라미터는 필수입니다.

db2pd_cmd

실행하려는 db2pd 명령의 이름입니다. 데이터 형식은 varchar입니다.

파라미터는 하이픈으로 시작되어야 합니다. 파라미터 목록을 알아보려면 IBM 설명서의 db2pd - Db2 데이터베이스 모니터링 및 문제 해결 명령을 참조하세요.

다음 옵션은 JSON에서 지원되지 않습니다.

- -addnode
- -alldatabases
- -alldbp
- -alldbs
- -allmembers
- -alm_in_memory
- -cfinfo
- -cfpool
- -command
- -dbpartitionnum
- -debug
- -dump
- -everything

- `-file | -o`
- `-ha`
- `-interactive`
- `-member`
- `-pages`

  > **Note**
  >
  > `-pages summary`을(를) 지원합니다.

- `-pdcollection`
- `-repeat`
- `-stack`
- `-totalmem`

`file` 하위 옵션은 지원되지 않습니다(예: `db2pd -db testdb -tcbstats file=tcbstat.out`).

`stacks`의 사용은 지원되지 않습니다(예: `db2pd -edus interval=5 top=10 stacks`).

사용 노트

이 저장 프로시저는 RDS for Db2 데이터베이스를 모니터링하고 문제를 해결하는 데 도움이 되는 정보를 수집합니다.

저장 프로시저는 IBM db2pd 유틸리티를 사용하여 다양한 명령을 실행합니다. db2pd 유틸리티를 사용하려면 RDS for Db2 마스터 사용자에게는 없는 SYSADM 인증이 필요합니다. 하지만 Amazon RDS 저장 프로시저를 사용하면 마스터 사용자가 유틸리티를 사용하여 다양한 명령을 실행할 수 있습니다. 유틸리티에 대한 자세한 내용은 IBM 설명서의 db2pd - Db2 데이터베이스 모니터링 및 문제 해결 명령을 참조하세요.

출력은 최대 2GB로 제한됩니다.

데이터베이스의 정보 수집 상태 점검에 대해 알아보려면 rdsadmin.get_task_status 섹션을 참조하세요.

예시

다음 예제는 RDS for Db2 DB 인스턴스의 가동 시간을 반환합니다.

```
db2 "call rdsadmin.db2pd_command('-')"
```

다음 예제는 TESTDB라는 데이터베이스의 가동 시간을 반환합니다.

```
db2 "call rdsadmin.db2pd_command('-db TESTDB -')"
```

다음 예제는 RDS for Db2 DB 인스턴스의 메모리 사용량을 반환합니다.

```
db2 "call rdsadmin.db2pd_command('-dbptnmem')"
```

다음 예제는 RDS for Db2 DB 인스턴스 및 TESTDB라는 데이터베이스의 메모리 집합을 반환합니다.

```
db2 "call rdsadmin.db2pd_command('-inst -db TESTDB -memsets')"
```

rdsadmin.force_application

RDS for Db2 데이터베이스의 애플리케이션을 강제 종료합니다.

구문

```
db2 "call rdsadmin.force_application(
    ?,
    'applications')"
```

파라미터

다음 출력 파라미터는 필수입니다.

?

오류 메시지를 출력하는 파라미터 표시자입니다. 이 파라미터는 ?만 허용합니다.

다음 입력 파라미터는 필수입니다.

######

RDS for Db2 데이터베이스를 강제 종료하려는 애플리케이션입니다. 데이터 형식은 varchar입니다. 유효한 값: ALL 또는 *application_handle*.

여러 애플리케이션의 이름을 쉼표로 구분합니다. 예제: '*application_handle_1*, *application_handle_2*'

사용 노트

이 저장 프로시저는 데이터베이스에서 모든 애플리케이션을 강제로 종료하므로 유지 관리를 수행할 수 있습니다.

저장 프로시저는 IBM FORCE APPLICATION 명령을 사용합니다. FORCE APPLICATION 명령을 사용하려면 RDS for Db2 마스터 사용자에게는 없는 SYSADM, SYSMAINT, SYSCTRL 인증이 필요합니다. 하지만 Amazon RDS 저장 프로시저를 사용하면 마스터 사용자가 명령을 사용할 수 있습니다. 자세한 내용은 IBM 설명서의 FORCE APPLICATION 명령을 참조하세요.

데이터베이스의 애플리케이션 종료 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

예시

다음 예제에서는 RDS for Db2 데이터베이스의 모든 애플리케이션을 강제 종료합니다.

```
db2 "call rdsadmin.force_application(
    ?,
    'ALL')"
```

다음 예제에서는 RDS for Db2 데이터베이스의 9991, 8891, 1192 애플리케이션 핸들을 강제 종료합니다.

```
db2 "call rdsadmin.force_application(
    ?,
    '9991, 8891, 1192')"
```

rdsadmin.set_archive_log_retention

일부 RDS for Db2 데이터베이스의 아카이브 로그 파일을 보관하는 시간(시간 기준)을 구성합니다.

구문

```
db2 "call rdsadmin.set_archive_log_retention(
    ?,
    'database_name',
```

```
    'archive_log_retention_hours')"
```

파라미터

다음 출력 파라미터는 필수입니다.

?

오류 메시지를 출력하는 파라미터 표시자입니다. 이 파라미터는 ?만 허용합니다.

다음 입력 파라미터는 필수입니다.

database_name

아카이브 로그 보존을 구성할 데이터베이스의 이름입니다. 데이터 형식은 varchar입니다.

archive_log_retention_hours

아카이브 로그 파일을 유지하는 데 걸리는 시간입니다. 데이터 형식은 smallint입니다. 기본값은 0이고 최댓값은 168(7일)입니다.

값이 0인 경우 Amazon RDS는 아카이브 로그 파일을 유지하지 않습니다.

사용 노트

기본적으로 RDS for Db2는 로그를 5분 동안 보존합니다. 변경 데이터 캡처(CDC) 또는 IBM Q Replication에 AWS DMS 등의 복제 도구를 사용하는 경우 도구에 로그 보존을 5분 이상으로 설정하는 것이 좋습니다.

the section called “rdsadmin.show_archive_log_retention”을 직접적으로 호출하여 현재 아카이브 로그 보존 설정을 볼 수 있습니다.

rdsadmin 데이터베이스에서는 아카이브 로그 보존 설정을 구성할 수 없습니다.

예시

다음 예제에서는 TESTDB라는 데이터베이스의 아카이브 로그 보존 시간을 24시간으로 설정합니다.

```
db2 "call rdsadmin.set_archive_log_retention(
    ?,
    'TESTDB',
```

```
    '24')"
```

다음 예제에서는 TESTDB라는 데이터베이스의 아카이브 로그 보존을 비활성화합니다.

```
db2 "call rdsadmin.set_archive_log_retention(
    ?,
    'TESTDB',
    '0')"
```

rdsadmin.show_archive_log_retention

일부 데이터베이스의 현재 아카이브 로그 보존 설정을 반환합니다.

구문

```
db2 "call rdsadmin.show_archive_log_retention(
    ?,
    'database_name')"
```

파라미터

다음 출력 파라미터는 필수입니다.

?

오류 메시지를 출력하는 파라미터 표시자입니다. 이 파라미터는 ?만 허용합니다.

다음 입력 파라미터는 필수입니다.

database_name

아카이브 로그 보존 설정을 보여주는 데이터베이스의 이름입니다. 데이터 형식은 varchar입니다.

예시

다음 예제에서는 TESTDB라는 데이터베이스의 아카이브 로그 보존 설정을 보여줍니다.

```
db2 "call rdsadmin.show_archive_log_retention(
    ?
    'TESTDB')"
```

RDS for Db2의 스토리지 액세스에 대해 저장된 프로시저

이 주제에 설명된 기본 제공 저장 프로시저는 Amazon S3를 사용하여 데이터를 마이그레이션하는 RDS for Db2 데이터베이스에 대한 스토리지 액세스를 관리합니다. 자세한 내용은 the section called "Amazon S3로 마이그레이션" 단원을 참조하십시오.

구문, 파라미터, 사용 정보 및 예시에 대한 자세한 내용은 다음 기본 제공 저장 프로시저를 참조하세요.

저장 프로시저

- rdsadmin.catalog_storage_access
- rdsadmin.uncatalog_storage_access

rdsadmin.catalog_storage_access

Db2 데이터 파일이 있는 Amazon S3 버킷에 액세스하기 위한 스토리지 별칭을 카탈로그화합니다.

구문

```
db2 "call rdsadmin.catalog_storage_access(
    ?,
    'alias',
    's3_bucket_name',
    'grantee_type',
    'grantee'
    )"
```

파라미터

다음 출력 파라미터는 필수입니다.

?

오류 메시지를 출력하는 파라미터 표시자입니다. 데이터 유형은 varchar입니다.

다음 입력 파라미터는 필수입니다.

##

Amazon S3 버킷의 원격 스토리지에 액세스하기 위한 별칭 이름입니다. 데이터 유형은 varchar입니다.

s3_bucket_name

데이터가 있는 Amazon S3 버킷의 이름입니다. 데이터 형식은 varchar입니다.

grantee_type

권한을 받을 피부여자 유형입니다. 데이터 형식은 varchar입니다. 유효한 값: USER, GROUP.

grantee

권한을 받을 사용자 또는 그룹입니다. 데이터 형식은 varchar입니다.

사용 노트

Amazon RDS에서는 RDS for Db2 DB 인스턴스에 추가한 IAM 역할에 카탈로그화된 별칭이 포함되어 있습니다. DB 인스턴스에서 IAM 역할을 제거하면 Amazon RDS가 별칭을 삭제합니다. 자세한 내용은 the section called “Amazon S3로 마이그레이션” 단원을 참조하십시오.

별칭 카탈로그화 상태 확인에 대한 자세한 내용은 rdsadmin.get_task_status 단원을 참조하세요.

예시

다음 예제에서는 SAMPLE이라는 별칭이 등록됩니다. 사용자 jorge_souza에게 amzn-s3-demo-bucket이라는 Amazon S3 버킷에 대한 액세스가 부여됩니다.

```
db2 "call rdsadmin.catalog_storage_access(
    ?,
    'SAMPLE',
    'amzn-s3-demo-bucket',
    'USER',
    'jorge_souza')"
```

rdsadmin.uncatalog_storage_access

스토리지 액세스 별칭을 제거합니다.

구문

```
db2 "call rdsadmin.uncatalog_storage_access(
    ?,
    'alias')"
```

파라미터

다음 출력 파라미터는 필수입니다.

?

오류 메시지를 출력하는 파라미터 표시자입니다. 데이터 유형은 varchar입니다.

다음 입력 파라미터는 필수입니다.

##

제거할 스토리지 별칭의 이름입니다. 데이터 유형은 varchar입니다.

사용 노트

별칭 제거 상태 확인에 대한 자세한 내용은 rdsadmin.get_task_status 단원을 참조하세요.

예시

다음 예제에서는 SAMPLE이라는 별칭이 제거됩니다. 이 별칭은 연결된 Amazon S3 버킷에 대한 액세스를 더 이상 제공하지 않습니다.

```
db2 "call rdsadmin.uncatalog_storage_access(
    ?,
    'SAMPLE')"
```

RDS for Db2의 테이블스페이스에 대해 저장된 프로시저

이 주제에서 설명하는 기본 제공 저장 프로시저는 Amazon RDS for Db2 데이터베이스의 테이블스페이스를 관리합니다. 이 프로시저를 실행하려면 마스터 사용자가 먼저 `rdsadmin` 데이터베이스에 연결되어야 합니다.

이러한 저장 프로시저는 다양한 작업에 사용됩니다. 단, 이 목록이 전부는 아닙니다.

- 테이블스페이스의 일반적인 작업
- 성능 보고서 생성
- db2look을 사용하여 데이터베이스 메타데이터 복사
- IBM Db2 Data Management Console을 위한 리포지토리 데이터베이스 만들기

구문, 파라미터, 사용 정보 및 예시에 대한 자세한 내용은 다음 기본 제공 저장 프로시저를 참조하세요.

저장 프로시저

- rdsadmin.create_tablespace
- rdsadmin.alter_tablespace
- rdsadmin.rename_tablespace
- rdsadmin.drop_tablespace

rdsadmin.create_tablespace

테이블스페이스를 생성합니다.

구문

```
db2 "call rdsadmin.create_tablespace(
    'database_name',
    'tablespace_name',
    'buffer_pool_name',
    tablespace_page_size,
    tablespace_initial_size,
    tablespace_increase_size,
    'tablespace_type',
    'tablespace_prefetch_size')"
```

파라미터

다음 파라미터는 필수 파라미터입니다.

database_name

테이블스페이스를 만들 데이터베이스의 이름입니다. 데이터 형식은 varchar입니다.

tablespace_name

생성할 테이블 이름입니다. 데이터 형식은 varchar입니다.

테이블스페이스 이름에는 다음과 같은 제한 사항이 있습니다.

- 이 데이터베이스에 있는 기존 테이블스페이스의 이름과 같으면 안 됩니다.
- _$#@a-zA-Z0-9만 포함할 수 있습니다.
- _ 또는 $ 기호로 시작할 수 없습니다.
- SYS로 시작할 수 없습니다.

다음 파라미터는 선택적입니다.

buffer_pool_name

테이블스페이스를 할당할 버퍼 풀의 이름입니다. 데이터 형식은 varchar입니다. 기본값은 빈 문자열입니다.

> **Important**
>
> 테이블스페이스와 연결하려면 페이지 크기가 같은 버퍼 풀이 이미 있어야 합니다.

tablespace_page_size

테이블스페이스의 페이지 크기(바이트)입니다. 데이터 형식은 integer입니다. 유효한 값: 4096, 8192, 16384, 32768. 기본값은 rdsadmin.create_database를 직접 호출하여 데이터베이스를 만들 때 사용된 페이지 크기입니다.

> **Important**
>
> Amazon RDS는 4KiB, 8KiB 및 16KiB 페이지에 대한 쓰기 원자성을 지원합니다. 반면 32KiB 페이지는 쓰기가 찢기거나 데스크에 데이터가 일부만 기록될 위험이 있습니다.

32KiB 페이지를 사용하는 경우 시점 복구와 자동 백업을 활성화하는 것이 좋습니다. 그렇지 않으면 찢긴 페이지를 복구하지 못할 위험이 있습니다. 자세한 내용은 the section called "백업 소개" 및 the section called "시점 복구"(을)를 참조하세요.

tablespace_initial_size

테이블스페이스의 초기 크기로, 킬로바이트 단위(KB)입니다. 데이터 형식은 integer입니다. 유효한 값: 48 이상. 기본값은 null입니다.

값을 설정하지 않으면 Db2가 적절한 값을 설정합니다.

> **Note**
>
> 임시 테이블스페이스는 시스템에서 관리되므로, 이 파라미터는 임시 테이블스페이스에 적용할 수 없습니다.

tablespace_increase_size

테이블스페이스가 가득 찼을 때 테이블스페이스를 늘릴 비율입니다. 데이터 형식은 integer입니다. 유효한 값은 1~100입니다. 기본값은 null입니다.

값을 설정하지 않으면 Db2가 적절한 값을 설정합니다.

> **Note**
>
> 임시 테이블스페이스는 시스템에서 관리되므로, 이 파라미터는 임시 테이블스페이스에 적용할 수 없습니다.

tablespace_type

테이블스페이스의 유형입니다. 데이터 형식은 char입니다. 유효한 값: U(사용자 데이터용), T(사용자 임시 데이터용) 또는 S(시스템 임시 데이터용) 기본값은 U입니다.

tablespace_prefetch_size

테이블스페이스의 프리페치 페이지 크기입니다. 데이터 형식은 char입니다. 유효한 값: AUTOMATIC(대소문자를 구분하지 않음) 또는 0이 아닌 32,767 이하의 양의 정수

사용 노트

RDS for Db2는 항상 데이터를 위한 대규모 데이터베이스를 만듭니다.

테이블스페이스 생성 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

예시

예시 1: 테이블스페이스 만들기 및 버퍼 풀 할당

다음 예제에서는 SP8이라는 테이블스페이스를 생성하고 TESTDB라는 데이터베이스에 BP8이라는 버퍼 풀을 할당합니다. 테이블스페이스의 초기 테이블스페이스 페이지 크기는 4,096바이트이고 초기 테이블스페이스가 1,000KB이며 테이블 크기 증가가 50%로 설정되어 있습니다.

```
db2 "call rdsadmin.create_tablespace(
    'TESTDB',
    'SP8',
    'BP8',
    4096,
    1000,
    50)"
```

예시 2: 임시 테이블스페이스 만들기 및 버퍼 풀 할당

다음은 SP8이라는 이름의 임시 테이블스페이스를 생성하는 예시입니다. TESTDB라는 데이터베이스에 크기가 8KiB인 BP8 버퍼 풀을 할당합니다.

```
db2 "call rdsadmin.create_tablespace(
    'TESTDB',
    'SP8',
    'BP8',
    8192,
    NULL,
    NULL,
    'T')"
```

예시 3: 테이블스페이스 생성 및 프리페치 페이지 크기 할당

다음 예시에서는 TESTDB라는 데이터베이스에 대해 SP8이라는 테이블스페이스를 생성합니다. 테이블스페이스의 초기 테이블스페이스 증가 크기는 50이고 프리페치 페이지 크기는 800입니다.

```
db2 "call rdsadmin.create_tablespace(
```

```
    'TESTDB',
    'SP8',
    NULL,
    NULL,
    NULL,
    50,
    NULL,
    '800')"
```

rdsadmin.alter_tablespace

테이블스페이스를 변경합니다.

구문

```
db2 "call rdsadmin.alter_tablespace(
    'database_name',
    'tablespace_name',
    'buffer_pool_name',
    tablespace_increase_size,
    'max_size',
    'reduce_max',
    'reduce_stop',
    'reduce_value',
    'lower_high_water',
    'lower_high_water_stop',
    'switch_online',
    'tablespace_prefetch_size')"
```

파라미터

다음 파라미터는 필수 파라미터입니다.

database_name

테이블스페이스를 사용하는 데이터베이스의 이름입니다. 데이터 형식은 varchar입니다.

tablespace_name

수정할 테이블스페이스의 이름입니다. 데이터 형식은 varchar입니다.

다음 파라미터는 선택적입니다.

buffer_pool_name

테이블스페이스를 할당할 버퍼 풀의 이름입니다. 데이터 형식은 varchar입니다. 기본값은 빈 문자열입니다.

> **Important**
> 테이블스페이스와 연결하려면 페이지 크기가 같은 버퍼 풀이 이미 있어야 합니다.

tablespace_increase_size

테이블스페이스가 가득 찼을 때 테이블스페이스를 늘릴 비율입니다. 데이터 형식은 integer입니다. 유효한 값은 1~100입니다. 기본값은 0입니다.

max_size

테이블스페이스의 최대 크기입니다. 데이터 형식은 varchar입니다. 유효한 값은 *##* K | M | G 또는 NONE입니다. 기본값은 NONE입니다.

reduce_max

하이 워터 마크를 최대 한도까지 줄일지 여부를 지정합니다. 데이터 형식은 char입니다. 기본값은 N입니다.

reduce_stop

이전 reduce_max 또는 reduce_value 명령을 중단할지 여부를 지정합니다. 데이터 형식은 char입니다. 기본값은 N입니다.

reduce_value

테이블스페이스 하이 워터 마크를 줄이는 데 사용할 백분율 값입니다. 데이터 형식은 varchar입니다. 유효한 값은 *##* K | M | G 또는 1~100입니다. 기본값은 N입니다.

lower_high_water

ALTER TABLESPACE LOWER HIGH WATER MARK 명령을 실행할지 여부를 지정합니다. 데이터 형식은 char입니다. 기본값은 N입니다.

lower_high_water_stop

ALTER TABLESPACE LOWER HIGH WATER MARK STOP 명령을 실행할지 여부를 지정합니다. 데이터 형식은 char입니다. 기본값은 N입니다.

switch_online

ALTER TABLESPACE SWITCH ONLINE 명령을 실행할지 여부를 지정합니다. 데이터 형식은 char입니다. 기본값은 N입니다.

tablespace_prefetch_size

테이블스페이스의 프리페치 페이지 크기입니다. 데이터 형식은 char입니다. 유효한 값: AUTOMATIC(대소문자를 구분하지 않음) 또는 0이 아닌 32,767 이하의 양의 정수

> **Note**
>
> 이 파라미터는 buffer_pool_name, table_increase_size, max_size, switch_online과 함께만 작동합니다. reduce_max, reduce_stop, reduce_value, lower_high_water, lower_high_water_stop과 함께는 작동하지 않습니다.

사용 노트

저장된 프로시저를 직접적으로 호출하기 전에 다음 고려 사항을 검토하세요.

- rdsadmin.alter_tablespace 저장 프로시저는 사용자 임시 데이터의 tablespace_type이 T로 설정된 테이블스페이스에서는 작동하지 않습니다.
- 선택적 파라미터인 reduce_max, reduce_stop, reduce_value, lower_high_water, lower_high_water_stop, switch_online 파라미터는 함께 사용할 수 없습니다. rdsadmin.alter_tablespace 명령에서 다른 선택적 파라미터(예: buffer_pool_name)와 조합할 수 없습니다. 자세한 내용은 Statement not valid 섹션을 참조하세요.

테이블스페이스 변경 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

저장 프로시저를 호출할 때 반환되는 오류 메시지는 the section called “저장된 프로시저 오류” 섹션을 참조하세요.

예시

예시 1: 높은 워터마크 낮추기

다음 예제에서는 SP8이라는 테이블스페이스를 변경하고 TESTDB라는 데이터베이스에 BP8이라는 버퍼 풀을 할당하여 하이 워터 마크를 낮춥니다.

```
db2 "call rdsadmin.alter_tablespace(
```

```
    'TESTDB',
    'SP8',
    'BP8',
    NULL,
    NULL,
    'Y')"
```

예시 2: 높은 워터마크 줄이기

다음 예시는 TESTDB 데이터베이스의 TBSP_TEST라는 테이블스페이스에서 REDUCE MAX 명령을 실행합니다.

```
db2 "call rdsadmin.alter_tablespace(
    'TESTDB',
    'TBSP_TEST',
    NULL,
    NULL,
    NULL,
    'Y')"
```

예시 3: 높은 워터마크를 줄이기 위한 명령 중단

다음 예시는 TESTDB 데이터베이스의 TBSP_TEST라는 테이블스페이스에서 REDUCE STOP 명령을 실행합니다.

```
db2 "call rdsadmin.alter_tablespace(
    'TESTDB',
    'TBSP_TEST',
    NULL,
    NULL,
    NULL,
    NULL,
    'Y')"
```

예시 4: 기존 프리페치 페이지 크기 변경

다음 예시에서는 TSBP_TEST라는 테이블스페이스에서 ALTER TABLESPACE SWITCH ONLINE 명령을 실행하고 기존 프리페치 페이지 크기를 64로 변경합니다.

```
db2 "call rdsadmin.alter_tablespace(
    'TESTDB',
```

```
    'TBSP_TEST',
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    'Y',
    '64')"
```

rdsadmin.rename_tablespace

테이블스페이스 이름을 변경합니다.

구문

```
db2 "call rdsadmin.rename_tablespace(
    ?,
    'database_name',
    'source_tablespace_name',
    'target_tablespace_name')"
```

파라미터

다음 파라미터는 필수 파라미터입니다.

?

오류 메시지를 출력하는 파라미터 표시자입니다. 이 파라미터는 ?만 허용합니다.

database_name

테이블스페이스가 속한 데이터베이스의 이름입니다. 데이터 형식은 varchar입니다.

source_tablespace_name

이름을 변경할 테이블스페이스의 이름입니다. 데이터 형식은 varchar입니다.

target_tablespace_name

테이블스페이스의 새로운 이름입니다. 데이터 형식은 varchar입니다.

새로운 이름에는 다음과 같은 제한 사항이 있습니다.

- 기존 테이블스페이스의 이름과 같으면 안 됩니다.
- _$#@a-zA-Z0-9만 포함할 수 있습니다.
- _ 또는 $ 기호로 시작할 수 없습니다.
- SYS로 시작할 수 없습니다.

사용 노트

테이블스페이스 이름 변경 상태 확인에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

rdsadmin 데이터베이스에 속하는 테이블스페이스의 이름은 변경할 수 없습니다.

예시

다음 예시에서는 TESTDB라는 데이터베이스에서 SP8이라는 테이블스페이스를 SP9로 변경합니다.

```
db2 "call rdsadmin.rename_tablespace(
    ?,
    'TESTDB',
    'SP8',
    'SP9')"
```

rdsadmin.drop_tablespace

테이블스페이스를 삭제합니다.

구문

```
db2 "call rdsadmin.drop_tablespace(
    'database_name',
    'tablespace_name')"
```

파라미터

다음 파라미터는 필수 파라미터입니다.

database_name

테이블스페이스가 속한 데이터베이스의 이름입니다. 데이터 형식은 varchar입니다.

tablespace_name

삭제할 테이블스페이스의 이름입니다. 데이터 형식은 varchar입니다.

사용 노트

테이블스페이스 삭제 상태 점검에 대한 자세한 내용은 rdsadmin.get_task_status 섹션을 참조하세요.

예시

다음 예시에서는 TESTDB라는 데이터베이스에서 SP8이라는 테이블스페이스를 삭제합니다.

```
db2 "call rdsadmin.drop_tablespace(
    'TESTDB',
    'SP8')"
```

Amazon RDS for Db2 사용자 정의 함수 참조

Db2 엔진을 실행 중인 Amazon RDS DB 인스턴스에 사용할 수 있는 사용자 정의 함수는 다음과 같습니다.

주제

- rdsadmin.get_task_status
- rdsadmin.list_databases

rdsadmin.get_task_status

작업의 상태를 반환합니다.

구문

```
db2 "select task_id, task_type, database_name, lifecycle,
    varchar(bson_to_json(task_input_params), 500) as task_params,
    cast(task_output as varchar(500)) as task_output
    from table(rdsadmin.get_task_status(task_id,'database_name','task_type'))"
```

파라미터

다음 파라미터는 선택 사항입니다. 파라미터를 제공하지 않으면 사용자 정의 함수는 모든 데이터베이스의 모든 작업 상태를 반환합니다. Amazon RDS는 35일 동안 작업 기록을 유지합니다.

task_id

실행 중인 작업의 ID입니다. 이 ID는 작업을 실행할 때 반환됩니다. 기본값: `0`.

database_name

작업이 실행되고 있는 데이터베이스의 이름입니다.

task_type

쿼리할 작업의 유형입니다. 유효한 값은 ADD_GROUPS, ADD_USER, ALTER_BUFFERPOOL, ALTER_TABLESPACE, CHANGE_PASSWORD, COMPLETE_ROLLFORWARD, CREATE_BUFFERPOOL, CREATE_DATABASE, CREATE_ROLE, CREATE_TABLESPACE, DROP_BUFFERPOOL, DROP_DATABASE, DROP_TABLESPACE, LIST_USERS, REMOVE_GROUPS, REMOVE_USER, RESTORE_DB, ROLLFORWARD_DB_LOG, ROLLFORWARD_STATUS, UPDATE_DB_PARAM입니다.

사용 노트

`rdsadmin.get_task_status` 사용자 정의 함수를 사용하여 다음 Amazon RDS for Db2 작업의 상태를 확인할 수 있습니다. 단, 이 목록이 전부는 아닙니다.

- 버퍼 풀 생성, 변경 또는 삭제
- 테이블스페이스 생성, 변경 또는 삭제
- 데이터베이스 생성 또는 삭제
- Amazon S3에서 데이터베이스 백업 복원
- Amazon S3에서 데이터베이스 로그 롤포워딩

예시

다음 예제에서는 `rdsadmin.get_task_status`가 호출될 때 반환된 열을 표시합니다.

```
db2 "describe select * from table(rdsadmin.get_task_status())"
```

다음 예제에서는 모든 작업의 상태를 나열합니다.

```
db2 "select task_id, task_type, database_name, lifecycle,
    varchar(bson_to_json(task_input_params), 500) as task_params,
    cast(task_output as varchar(500)) as task_output
    from table(rdsadmin.get_task_status(null,null,null))"
```

다음 예제에서는 특정 작업의 상태를 나열합니다.

```
db2 "select task_id, task_type, database_name,
    varchar(bson_to_json(task_input_params), 500) as task_params
    from table(rdsadmin.get_task_status(1,null,null))"
```

다음 예제에서는 특정 작업 및 데이터베이스의 상태를 나열합니다.

```
db2 "select task_id, task_type, database_name,
    varchar(bson_to_json(task_input_params), 500) as task_params
    from table(rdsadmin.get_task_status(2,'SAMPLE',null))"
```

다음 예제에서는 모든 `ADD_GROUPS` 작업의 상태를 나열합니다.

```
db2 "select task_id, task_type, database_name,
    varchar(bson_to_json(task_input_params), 500) as task_params
    from table(rdsadmin.get_task_status(null,null,'add_groups'))"
```

다음 예제에서는 특정 데이터베이스의 모든 작업 상태를 나열합니다.

```
db2 "select task_id, task_type, database_name,
    varchar(bson_to_json(task_input_params), 500) as task_params
    from table(rdsadmin.get_task_status(null,'testdb', null))"
```

다음 예제는 JSON 값을 열로 출력합니다.

```
db2 "select varchar(r.task_type,25) as task_type, varchar(r.lifecycle,10) as lifecycle,
 r.created_at, u.* from
    table(rdsadmin.get_task_status(null,null,'restore_db')) as r,
 json_table(r.task_input_params, 'strict $' columns(s3_prefix varchar(500)
    null on empty, s3_bucket_name varchar(500) null on empty) error on error ) as U"
```

응답

`rdsadmin.get_task_status` 사용자 정의 함수는 다음 열을 반환합니다.

`TASK_ID`

작업의 ID입니다.

`TASK_TYPE`

입력 파라미터에 따라 다릅니다.

- `ADD_GROUPS` – 그룹을 추가합니다.
- `ADD_USER` – 사용자를 추가합니다.
- `ALTER_BUFFERPOOL` – 버퍼 풀을 변경합니다.
- `ALTER_TABLESPACE` – 테이블스페이스를 변경합니다.
- `CHANGE_PASSWORD` – 사용자의 암호를 변경합니다.
- `COMPLETE_ROLLFORWARD` – `rdsadmin.rollforward_database` 작업을 완료하고 데이터베이스를 활성화합니다.
- `CREATE_BUFFERPOOL` – 버퍼 풀을 생성합니다.
- `CREATE_DATABASE` – 데이터베이스를 생성합니다.

- CREATE_ROLE – 사용자의 Db2 역할을 생성합니다.
- CREATE_TABLESPACE – 테이블스페이스를 생성합니다.
- DROP_BUFFERPOOL – 버퍼 풀을 삭제합니다.
- DROP_DATABASE – 데이터베이스를 삭제합니다.
- DROP_TABLESPACE – 테이블스페이스를 삭제합니다.
- LIST_USERS – 모든 사용자를 나열합니다.
- REMOVE_GROUPS – 그룹을 제거합니다.
- REMOVE_USER – 사용자를 제거합니다.
- RESTORE_DB – 전체 데이터베이스를 복원합니다.
- ROLLFORWARD_DB_LOG – 데이터베이스 로그에 대한 rdsadmin.rollforward_database 작업을 수행합니다.
- ROLLFORWARD_STATUS – rdsadmin.rollforward_database 작업 상태를 반환합니다.
- UPDATE_DB_PARAM – 데이터 파라미터를 업데이트합니다.

DATABASE_NAME

작업이 연결되어 있는 데이터베이스의 이름입니다.

COMPLETED_WORK_BYTES

작업이 복원한 바이트 수입니다.

DURATION_MINS

작업을 완료하는 데 걸린 시간입니다.

LIFECYCLE

작업의 상태입니다. 가능한 상태:

- CREATED – Amazon RDS에 작업을 제출한 후 Amazon RDS는 상태를 CREATED로 설정합니다.
- IN_PROGRESS – 작업이 시작되면 Amazon RDS는 상태를 IN_PROGRESS로 설정합니다. CREATED에서 IN_PROGRESS로 상태가 변경되려면 최대 5분이 걸릴 수 있습니다.
- SUCCESS – 작업이 완료되면 Amazon RDS는 상태를 SUCCESS로 설정합니다.
- ERROR – 복원 작업이 실패할 경우 Amazon RDS는 상태를 ERROR로 설정합니다. 오류에 대한 자세한 내용은 TASK_OUPUT 섹션을 참조하세요.

CREATED_BY

명령을 생성한 authid입니다.

CREATED_AT

작업을 생성한 날짜와 시간입니다.

LAST_UPDATED_AT

작업이 마지막으로 업데이트된 날짜와 시간입니다.

TASK_INPUT_PARAMS

파라미터는 작업 유형에 따라 다릅니다. 모든 입력 파라미터는 JSON 객체로 표시됩니다. 예를 들어, RESTORE_DB 작업의 JSON 키는 다음과 같습니다.

- DBNAME
- RESTORE_TIMESTAMP
- S3_BUCKET_NAME
- S3_PREFIX

TASK_OUTPUT

작업에 대한 추가 정보입니다. 기본 복원 중에 오류가 발생하면 이 열에 오류에 대한 정보가 포함됩니다.

응답 예제

다음 응답 예제는 TESTJP라는 데이터베이스가 성공적으로 생성되었음을 보여줍니다. 자세한 내용은 the section called “rdsadmin.create_database” 저장 프로시저를 참조하세요.

```
`1 SUCCESS CREATE_DATABASE RDSDB 2023-10-24-18.32.44.962689 2023-10-24-18.34.50.038523
 1 TESTJP { "CODESET" : "IBM-437", "TERRITORY" : "JP", "COLLATION" : "SYSTEM",
 "AUTOCONFIGURE_CMD" : "", "PAGESIZE" : 4096 }
2023-10-24-18.33.30.079048 Task execution has started.

2023-10-24-18.34.50.038523 Task execution has completed successfully`.
```

다음 응답 예제는 데이터베이스 삭제가 실패한 이유를 설명합니다. 자세한 내용은 the section called “rdsadmin.drop_database” 저장 프로시저를 참조하세요.

```
1 ERROR DROP_DATABASE RDSDB 2023-10-10-16.33.03.744122 2023-10-10-16.33.30.143797 -
 2023-10-10-16.33.30.098857 Task execution has started.
```

```
2023-10-10-16.33.30.143797 Caught exception during executing task id 1, Aborting task.
Reason Dropping database created via rds CreateDBInstance api is not allowed.
Only database created using rdsadmin.create_database can be dropped
```

다음 응답 예제는 데이터베이스의 성공적인 복원을 보여줍니다. 자세한 내용은 the section called "rdsadmin.restore_database" 저장 프로시저를 참조하세요.

```
1 RESTORE_DB  SAMPLE  SUCCESS

{ "S3_BUCKET_NAME" : "amzn-s3-demo-bucket", "S3_PREFIX" :
 "SAMPLE.0.rdsdb3.DBPART000.20230413183211.001", "RESTORE_TIMESTAMP" :
 "20230413183211", "BACKUP_TYPE" : "offline" }




2023-11-06-18.31.03.115795 Task execution has started.
2023-11-06-18.31.04.300231 Preparing to download
2023-11-06-18.31.08.368827 Download complete. Starting Restore
2023-11-06-18.33.13.891356 Task Completed Successfully
```

rdsadmin.list_databases

RDS for Db2 DB 인스턴스에서 실행되는 모든 데이터베이스 목록을 반환합니다.

구문

```
db2 "select * from table(rdsadmin.list_databases())"
```

사용 노트

이 사용자 정의 함수는 데이터베이스가 활성화된 상태인지 비활성화된 상태인지 지정하지 않습니다.

목록에 데이터베이스가 표시되지 않으면 the section called "rdsadmin.get_task_status" 사용자 정의 함수를 직접 호출하고 오류 메시지를 찾습니다.

응답

`rdsadmin.list_databases` 사용자 정의 함수는 다음 열을 반환합니다.

DATABASE_NAME

데이터베이스의 이름입니다.

CREATE_TIME

데이터베이스가 만들어진 날짜와 시간입니다.

응답 예제

다음 응답 예시는 데이터베이스 목록과 데이터베이스가 만들어진 시간을 보여줍니다. rdsadmin은 Amazon RDS가 관리하고 출력에 항상 표시되는 데이터베이스입니다.

```
DATABASE_NAME   CREATE_TIME
--------------- --------------------------
rdsadmin        2024-10-22-03.37.48.535671
TEST            2024-10-22-03.39.36.818679
TEST1           2024-10-22-03.57.15.218009
TEST2           2024-10-22-03.59.28.029556
```

Amazon RDS for Db2 문제 해결

다음 콘텐츠는 RDS for Db2에 발생할 수 있는 문제를 해결하는 데 도움이 될 수 있습니다.

일반적인 Amazon RDS 문제 해결에 대한 자세한 내용은 Amazon RDS 문제 해결 단원을 참조하세요.

주제

- 데이터베이스 연결 오류
- 파일 I/O 오류
- 저장된 프로시저 오류

데이터베이스 연결 오류

다음 오류 메시지는 서버에 메모리가 충분하지 않아 데이터베이스가 연결되지 않았음을 나타냅니다.

```
SQL1643C The database manager failed to allocate shared memory because the
database manager instance memory limit has been reached.
```

DB 인스턴스의 메모리를 늘린 다음 데이터베이스에 다시 연결해 보세요. 메모리 사용량 및 데이터베이스 권장 사항에 대한 자세한 내용은 the section called “여러 Db2 데이터베이스” 섹션을 참조하세요. RDS for Db2 데이터베이스의 메모리를 업데이트하는 방법에 대한 자세한 내용은 the section called “rdsadmin.update_db_param” 섹션을 참조하세요.

파일 I/O 오류

LOAD 명령을 사용하거나 rdsadmin.restore_database 저장 프로시저를 직접 호출할 때 등 다양한 이유로 파일 I/O 오류가 발생할 수 있습니다.

이 예에서는 다음 LOAD 명령을 실행할 수 있습니다.

```
db2 "call sysproc.admin_cmd('load from "DB2REMOTE://s3test//public/datapump/t6.del" of
 del lobs from "DB2REMOTE://s3test/public/datapump/" modified by lobsinfile MESSAGES ON
 SERVER insert INTO RDSDB.t6 nonrecoverable ')"
```

LOAD 명령은 다음 메시지를 반환합니다.

```

  Result set 1
  --------------
```

```
 ROWS_READ            ROWS_SKIPPED         ROWS_LOADED          ROWS_REJECTED
ROWS_DELETED         ROWS_COMMITTED       ROWS_PARTITIONED     NUM_AGENTINFO_ENTRIES
MSG_RETRIEVAL




MSG_REMOVAL
 -------------------- -------------------- -------------------- --------------------
-------------------- -------------------- -------------------- --------------------
--------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------
                     -                    -                    -                    -
                    -                    -                    -                    -
SELECT SQLCODE, MSG FROM TABLE(SYSPROC.ADMIN_GET_MSGS('1594987316_285548770')) AS MSG




                                                                                  CALL
SYSPROC.ADMIN_REMOVE_MSGS('1594987316_285548770')

 1 record(s) selected.

 Return Status = 0

SQL20397W  Routine "SYSPROC.ADMIN_CMD" execution has completed, but at least
one error, "SQL1652", was encountered during the execution. More information
is available.  SQLSTATE=01H52
```

오류 메시지를 보려면 이전 응답에서 제안한 대로 SQL 명령을 실행합니다. `SELECT SQLCODE, MSG FROM TABLE(SYSPROC.ADMIN_GET_MSGS('1594987316_285548770')) AS MSG`는 다음 메시지를 반환합니다.

```
SQLCODE   MSG
---------
 --------------------------------------------------------------------------------
SQL2025N  An I/O error occurred. Error code “438”. Media on which this error occurred:
 “DB2REMOTE://s3test//public/datapump/t6.del”

SQL3500W The utility is beginning the LOAD phase at time “07/05/2024 21:21:48.082954”
```

```
SQL1652N File I/O error occurred
```

Db2 진단 로그에는 다음과 유사한 로그 파일이 포함되어 있습니다.

```
2024-07-05-21.20.09.440609+000 I1191321E864          LEVEL: Error
PID     : 2710                 TID : 139619509200640 PROC : db2sysc 0
INSTANCE: rdsdb                NODE : 000            DB   : NTP
APPHDL  : 0-12180              APPID: xxx.xx.x.xxx.xxxxx.xxxxxxxxxxxx
UOWID   : 5                    ACTID: 1
AUTHID  : ADMIN                HOSTNAME: ip-xx-xx-x-xx
EDUID   : 147                  EDUNAME: db2lmr 0
FUNCTION: DB2 UDB, oper system services, sqloS3Client_GetObjectInfo, probe:219
MESSAGE : ZRC=0x870F01B6=-2029059658=SQLO_FAILED
          "An unexpected error is encountered"
DATA #1 : String, 29 bytes
S3:HeadObject request failed.
DATA #2 : signed integer, 4 bytes
99
DATA #3 : String, 0 bytes
Object not dumped: Address: 0x00007EFC08A9AE38 Size: 0 Reason: Zero-length data
DATA #4 : String, 33 bytes
curlCode: 28, Timeout was reached
```

이 파일 I/O 오류는 다양한 시나리오로 인해 발생할 수 있습니다. 예를 들어 RDS for Db2 DB 인스턴스를 만드는 데 사용되는 보안 그룹과 연결된 VPC에 Amazon S3 게이트웨이 엔드포인트가 없을 수 있습니다. 이 엔드포인트는 RDS for Db2가 Amazon S3에 액세스할 수 있도록 하는 데 필수적입니다. RDS for Db2 DB 인스턴스가 프라이빗 서브넷에 있는 경우 Amazon S3 게이트웨이 엔드포인트가 필요합니다. Amazon RDS 서브넷 그룹을 구성하여 DB 인스턴스가 프라이빗 서브넷을 사용하는지 아니면 퍼블릭 서브넷을 사용하는지 지정할 수 있습니다. 자세한 내용은 DB 서브넷 그룹을 사용한 작업 섹션을 참조하세요.

주제

- 1단계: Amazon S3에 대한 VPC 게이트웨이 엔드포인트 생성
- 2단계: Amazon S3용 VPC 게이트웨이 엔드포인트가 존재하는지 확인

1단계: Amazon S3에 대한 VPC 게이트웨이 엔드포인트 생성

RDS for Db2 DB 인스턴스가 Amazon S3와 상호 작용하려면 프라이빗 서브넷에서 사용할 VPC와 Amazon S3 게이트웨이 엔드포인트를 만듭니다.

S3에 대한 VPC 게이트웨이 엔드포인트를 만드는 방법

1. VPC를 생성합니다. 자세한 내용은 Amazon Virtual Private Cloud 사용 설명서의 VPC 생성을 참조하세요.
2. 프라이빗 서브넷에서 사용할 Amazon S3 게이트웨이 엔드포인트를 만듭니다. 자세한 내용은 AWS PrivateLink 안내서의 Gateway endpoints를 참조하세요.

2단계: Amazon S3용 VPC 게이트웨이 엔드포인트가 존재하는지 확인

AWS Management Console 또는 AWS CLI를 사용하여 Amazon S3 게이트웨이 엔드포인트를 성공적으로 만들었는지 확인합니다.

콘솔

Amazon S3 게이트웨이 엔드포인트를 확인하는 방법

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/vpc에서 Amazon VPC 콘솔을 엽니다.
2. 콘솔의 오른쪽 상단에서 VPC의 AWS 리전을 선택합니다.
3. 만든 VPC를 선택합니다.
4. 리소스 맵 탭의 네트워크 연결에서 Amazon S3 게이트웨이 엔드포인트가 나열되었는지 확인합니다.

AWS CLI

Amazon S3 게이트웨이 엔드포인트를 확인하려면 describe-vpc-endpoints 명령을 실행합니다. 다음 예시에서 *vpc_id*를 VPC ID로, *region*을 AWS 리전로, *profile*을 프로필 이름으로 바꿉니다.

대상 LinuxmacOS, 또는Unix:

```
aws ec2 describe-vpc-endpoints \
    --filters "Name=vpc-id,Values=$vpc_id" \
    "Name=service-name,\
    Values=com.amazonaws.${region}.s3" \
    --region $region --profile=$profile \
    --query "VpcEndpoints[*].VpcEndpointId" --output text
```

Windows의 경우:

```
aws ec2 describe-vpc-endpoints ^
    --filters "Name=vpc-id,Values=$vpc_id" ^
    "Name=service-name,^
    Values=com.amazonaws.${region}.s3" ^
    --region $region --profile=$profile ^
    --query "VpcEndpoints[*].VpcEndpointId" --output text
```

이 명령은 Amazon S3 게이트웨이 엔드포인트가 있는 경우 다음 예시와 유사한 출력을 생성합니다.

```
[
    "vpce-0ea810434ff0b97e4"
]
```

이 명령은 Amazon S3 게이트웨이 엔드포인트가 없는 경우 다음 예시와 유사한 출력을 생성합니다.

```
[]
```

Amazon S3 게이트웨이 엔드포인트가 나열되지 않았으면 1단계: Amazon S3에 대한 VPC 게이트웨이 엔드포인트 생성를 실행합니다.

저장된 프로시저 오류

이 섹션에서는 저장 프로시저를 직접적으로 호출할 때 반환되는 다양한 오류와 이를 해결하는 방법을 설명합니다.

범주	저장된 프로시저 오류
데이터베이스 수	rdsadmin.activate_database 오류
데이터베이스 수	rdsadmin.create_database 오류
데이터베이스 수	rdsadmin.deactivate_database 오류
데이터베이스 수	rdsadmin.drop_database 오류
데이터베이스 수	rdsadmin.reactivate_database 오류
데이터베이스 수	rdsadmin.restore_database 오류
데이터베이스 수	rdsadmin.update_db_param 오류

범주	저장된 프로시저 오류
테이블스페이스	rdsadmin.alter_tablespace 오류

rdsadmin.activate_database 오류

??? 저장된 프로시저를 직접적으로 호출할 때 다음과 같은 오류가 발생할 수 있습니다.

오류	오류 메시지
Failed to allocate shared memory	`SQL1643C The database manager failed to allocate shared memory because the database manager instance memory limit has been reached.`
Unable to activate because of running processes	`The database can’t be activated because it's in the process of being created or restored.`

공유 메모리 할당 실패

다음 오류 메시지는 DB 인스턴스에 메모리가 충분하지 않아 저장 프로시저가 데이터베이스를 활성화하지 못했음을 나타냅니다.

```
SQL1643C The database manager failed to allocate shared memory because the database
 manager instance memory limit has been reached.
```

DB 인스턴스의 메모리를 늘린 다음 `rdsadmin.activate_database` 저장 프로시저를 다시 직접 호출합니다. 메모리 사용량 및 데이터베이스 권장 사항에 대한 자세한 내용은 the section called “여러 Db2 데이터베이스” 섹션을 참조하세요.

실행 중인 프로세스로 인해 활성화할 수 없음

다음 오류 메시지는 `rdsadmin.create_database` 또는 `rdsadmin.restore_database` 저장 프로시저가 실행 중이므로 저장 프로시저가 데이터베이스를 활성화할 수 없음을 나타냅니다.

```
The database can’t be activated because it's in the process of being created or
 restored.
```

몇 분 기다린 후 rdsadmin.activate_database 저장 프로시저를 다시 직접적으로 호출합니다.

rdsadmin.alter_tablespace 오류

the section called “rdsadmin.alter_tablespace” 저장된 프로시저를 직접적으로 호출할 때 다음과 같은 오류가 발생할 수 있습니다.

오류	오류 메시지
Statement not valid	DB21034E The command was processed as an SQL statement because it was not a valid Command Line Processor command. During SQL processing it returned: SQL1763N Invalid ALTER TABLESPACE statement for table space "TBSP_TEST" due to reason "12"
tablespace_prefetch_size value not valid	Invalid tablespace_prefetch_size. Set value to AUTOMATIC or to a non-zero positive numerical value.
tablespace_prefetch_size numerical value not valid	Invalid tablespace_prefetch_size. The number of pages can't be greater than 32767.
Parameter can't be used with tablespace_prefetch_size	You can't use tablespace_prefetch_size with {*parameter* }.
Tablespace change failed	The change to tablespace { *tablespace_name}* failed because you can only alter LARGE or REGULAR tablespaces.

문이 유효하지 않음

다음 오류 메시지는 저장 프로시저가 상호 배타적인 선택적 파라미터를 다른 선택적 파라미터와 결합했음을 나타냅니다. rdsadmin.alter_tablespace 저장 프로시저의 선택적 파라미터인 reduce_max, reduce_stop, reduce_value, lower_high_water, lower_high_water_stop, switch_online 파라미터는 함께 사용할 수 없습니다. rdsadmin.alter_tablespace 저장 프로시저에서 다른 선택적 파라미터(예:

buffer_pool_name)와 조합할 수 없습니다. 이를 결합하면 rdsadmin.get_task_status 사용자 정의 함수를 호출할 때 Db2가 이 오류 메시지를 반환합니다.

```
DB21034E The command was processed as an SQL statement because it was not a valid
 Command Line Processor command. During SQL processing it returned:
SQL1763N Invalid ALTER TABLESPACE statement for table space "TBSP_TEST" due to reason
 "12"
```

상호 배타적인 선택적 파라미터를 다른 선택적 파라미터와 결합하지 않고 rdsadmin.alter_tablespace 저장 프로시저를 다시 호출합니다. 그런 다음 rdsadmin.get_task_status 사용자 정의 함수를 호출합니다. 자세한 내용은 rdsadmin.alter_tablespace 및 rdsadmin.get_task_status를 참조하세요.

tablespace_prefetch_size 값이 유효하지 않음

다음 오류 메시지는 tablespace_prefetch_size를 AUTOMATIC 또는 0이 아닌 양수 값으로 설정하지 않았음을 나타냅니다. 예를 들어 testinput으로 설정하려고 했습니다.

```
Invalid tablespace_prefetch_size. Set value to AUTOMATIC or to a non-zero positive
 numerical value.
```

rdsadmin.alter_tablespace 저장 프로시저를 다시 직접적으로 호출하고 tablespace_prefetch_size를 AUTOMATIC 또는 0이 아닌 양수 값으로 설정합니다.

tablespace_prefetch_size 숫자 값이 유효하지 않음

다음 오류 메시지는 tablespace_prefetch_size를 32,767보다 큰 숫자 값으로 설정했음을 나타냅니다.

```
Invalid tablespace_prefetch_size. The number of pages can't be greater than 32767.
```

rdsadmin.alter_tablespace 저장 프로시저를 다시 직접적으로 호출하고 tablespace_prefetch_size를 32,767 이하의 0이 아닌 양수 값으로 설정합니다.

파라미터를 tablespace_prefetch_size와 함께 사용할 수 없음

다음 오류 메시지는 tablespace_prefetch_size에 호환되지 않는 파라미터를 사용하려고 했음을 나타냅니다.

```
You can't use tablespace_prefetch_size with {parameter}.
```

rdsadmin.alter_tablespace 저장 프로시저를 다시 직접적으로 호출하고 tablespace_prefetch_size에 호환되는 파라미터만 사용합니다. tablespace_prefetch_size에 사용할 수 있는 파라미터에 대한 자세한 내용은 rdsadmin.alter_tablespace 섹션을 참조하세요.

테이블스페이스 변경 실패

다음 오류 메시지는 테이블스페이스를 변경하려고 했음을 나타냅니다.

```
The change to tablespace {tablespace_name} failed because you can only alter LARGE or
 REGULAR tablespaces.
```

rdsadmin.create_database 오류

the section called “rdsadmin.create_database” 저장된 프로시저를 직접적으로 호출할 때 다음과 같은 오류가 발생할 수 있습니다.

오류	오류 메시지
Failed to allocate shared memory	SQL1643C The database manager failed to allocate shared memory because the database manager instance memory limit has been reached.

공유 메모리 할당 실패

다음 오류 메시지는 DB 인스턴스에 메모리가 충분하지 않아 저장 프로시저가 데이터베이스를 만들지 못했음을 나타냅니다.

```
SQL1643C The database manager failed to allocate shared memory because the database
 manager instance memory limit has been reached.
```

DB 인스턴스의 메모리를 늘린 다음 rdsadmin.create_database 저장 프로시저를 다시 직접 호출합니다. 메모리 사용량 및 데이터베이스 권장 사항에 대한 자세한 내용은 the section called “여러 Db2 데이터베이스” 섹션을 참조하세요.

데이터베이스가 만들어졌는지 확인하려면 the section called “rdsadmin.list_databases” 사용자 정의 함수를 직접 호출하고 새 데이터베이스가 나열되었는지 확인합니다.

rdsadmin.deactivate_database 오류

??? 저장된 프로시저를 직접적으로 호출할 때 다음과 같은 오류가 발생할 수 있습니다.

오류	오류 메시지
Unable to deactivate because of running processes	`The database can’t be deactivated because it's in the process of being created or restored.`

실행 중인 프로세스로 인해 비활성화할 수 없음

다음 오류 메시지는 `rdsadmin.create_database` 또는 `rdsadmin.restore_database` 저장 프로시저가 실행 중이므로 저장 프로시저가 데이터베이스를 비활성화할 수 없음을 나타냅니다.

```
The database can’t be deactivated because it's in the process of being created or
 restored.
```

몇 분 기다린 후 `rdsadmin.deactivate_database` 저장 프로시저를 다시 직접적으로 호출합니다.

rdsadmin.drop_database 오류

the section called “rdsadmin.drop_database” 저장된 프로시저를 직접적으로 호출할 때 다음과 같은 오류가 발생할 수 있습니다.

오류	오류 메시지
Database name doesn't exist	`SQL0438N Application raised error or warning with diagnostic text: "Cannot drop database. Database with provided name does not exist". SQLSTATE=99993`
Return status = 0	`Return Status = 0`
Dropping database not allowed	`1 ERROR DROP_DATABASE RDSDB 2023-10-10-16.33.03.744122 2023-10-10-16.33.30.143797 - 2023-10-10-16.33.30.098857 Task execution has started. 2023-10-10-16.33.30.143797 Caught exception during executing task id 1, Aborting task. Reason Dropping`

오류	오류 메시지
	`database created via rds CreateDBInstance api is not allowed. Only database created using rdsadmin.create_database can be dropped`

데이터베이스 이름이 존재하지 않음

다음 오류 메시지는 `rdsadmin.drop_database` 저장 프로시저에서 잘못된 데이터베이스 이름을 전달했음을 나타냅니다.

```
SQL0438N Application raised error or warning with diagnostic text: "Cannot
drop database. Database with provided name does not exist". SQLSTATE=99993
```

올바른 데이터베이스 이름으로 `rdsadmin.drop_database` 저장 프로시저를 다시 호출합니다. 데이터베이스가 삭제되었는지 확인하려면 the section called “rdsadmin.list_databases” 사용자 정의 함수를 직접 호출하고 삭제된 데이터베이스가 나열되지 않았는지 확인합니다.

반품 상태 = 0

다음 오류 메시지는 저장된 프로시저를 완료할 수 없음을 나타냅니다.

```
Return Status = 0
```

`Return Status = 0`을 수신한 후 rdsadmin.get_task_status 사용자 정의 함수를 호출합니다.

데이터베이스 삭제가 허용되지 않음

다음 오류 메시지는 Amazon RDS 콘솔 또는 AWS CLI를 사용하여 데이터베이스를 생성했음을 나타냅니다. `rdsadmin.drop_database` 저장 프로시저를 호출하여 데이터베이스를 생성한 경우에만 the section called “rdsadmin.create_database” 저장 프로시저를 사용할 수 있습니다.

```
1 ERROR DROP_DATABASE RDSDB 2023-10-10-16.33.03.744122 2023-10-10-16.33.30.143797 -
 2023-10-10-16.33.30.098857 Task execution has started.
2023-10-10-16.33.30.143797 Caught exception during executing task id 1, Aborting task.
Reason Dropping database created via rds CreateDBInstance api is not allowed.
Only database created using rdsadmin.create_database can be dropped
```

Amazon RDS 콘솔 또는 AWS CLI를 사용하여 생성한 데이터베이스를 삭제하려면 클라이언트를 사용하여 데이터베이스에 연결한 다음 적절한 명령을 실행합니다.

rdsadmin.reactivate_database 오류

??? 저장된 프로시저를 직접적으로 호출할 때 다음과 같은 오류가 발생할 수 있습니다.

오류	오류 메시지
Failed to allocate shared memory	`SQL1643C The database manager failed to allocate shared memory because the database manager instance memory limit has been reached.`
Unable to reactivate because of running processes	`The database can’t be reactivated because it's in the process of being created or restored.`

공유 메모리 할당 실패

다음 오류 메시지는 DB 인스턴스에 메모리가 충분하지 않아 저장 프로시저가 데이터베이스를 활성화하지 못했음을 나타냅니다.

```
SQL1643C The database manager failed to allocate shared memory because the database
 manager instance memory limit has been reached.
```

DB 인스턴스의 메모리를 늘린 다음 `rdsadmin.activate_database` 저장 프로시저를 다시 직접 호출합니다. 메모리 사용량 및 데이터베이스 권장 사항에 대한 자세한 내용은 the section called “여러 Db2 데이터베이스” 섹션을 참조하세요.

실행 중인 프로세스로 인해 재활성화할 수 없음

다음 오류 메시지는 `rdsadmin.create_database` 또는 `rdsadmin.restore_database` 저장 프로시저가 실행 중이므로 저장 프로시저가 데이터베이스를 재활성화할 수 없음을 나타냅니다.

```
The database can’t be reactivated because it's in the process of being created or
 restored.
```

몇 분 기다린 후 `rdsadmin.reactivate_database` 저장 프로시저를 다시 직접적으로 호출합니다.

rdsadmin.restore_database 오류

the section called “rdsadmin.restore_database” 저장된 프로시저를 직접적으로 호출할 때 다음과 같은 오류가 발생할 수 있습니다.

오류	오류 메시지
Insufficient disk space	`Aborting task. Reason Restoring your database failed because of insufficient disk space. Increase the storage for your DB instance and rerun the rdsadmin.restore_database stored procedure.`
Internal error	`Caught exception during executing task id 104, Aborting task. Reason Internal Error`
Non-fenced routines not allowed	`Caught exception during executing task id 2, Aborting task. Reason Non fenced routines are not allowed. Please delete the routines and retry the restore.`
Tablespaces not restored	`Reason SQL0970N The system attempted to write to a read-only file. Reason SQL2563W The Restore process has completed successfully. However one or more table spaces from the backup were not restored.`

디스크 공간 부족

다음 오류 메시지는 DB 인스턴스의 디스크 공간이 부족하여 데이터베이스를 복원할 수 없음을 나타냅니다.

```
Aborting task. Reason Restoring your database failed because of insufficient
 disk space. Increase the storage for your DB instance and rerun the
 rdsadmin.restore_database stored procedure.
```

DB 인스턴스의 여유 공간은 백업 이미지 크기의 두 배 이상이어야 합니다. 백업 이미지가 압축된 경우 DB 인스턴스의 여유 공간이 백업 이미지 크기의 3배 이상이어야 합니다. 자세한 내용은 the section called “DB 인스턴스 스토리지 용량 증가” 섹션을 참조하세요.

디스크 공간을 늘리고 `rdsadmin.restore_database` 저장 프로시저를 다시 직접 호출합니다. 데이터베이스가 복원되었는지 확인하려면 the section called “rdsadmin.list_databases” 사용자 정의 함수를 직접 호출하고 복원된 데이터베이스가 나열되었는지 확인합니다.

내부 오류

다음 오류 메시지는 저장된 프로시저에 내부 오류가 발생했음을 나타냅니다.

```
Caught exception during executing task id 104, Aborting task. Reason Internal Error
```

AWS Support에 문의하세요.

격리되지 않은 루틴은 허용되지 않음

다음 오류 메시지는 데이터베이스에 격리되지 않은 루틴이 포함되어 있음을 나타냅니다.

```
Caught exception during executing task id 2, Aborting task. Reason Non fenced routines
 are not allowed. Please delete the routines and retry the restore.
```

RDS for Db2는 격리되지 않은 루틴 생성을 지원하지 않습니다. 소스 데이터베이스에서 격리되지 않은 루틴을 제거한 다음 `rdsadmin.restore_database`를 다시 직접적으로 호출합니다. 데이터베이스가 복원되었는지 확인하려면 the section called “rdsadmin.list_databases” 사용자 정의 함수를 직접 호출하고 복원된 데이터베이스가 나열되었는지 확인합니다. 자세한 내용은 the section called “울타리가 없는 루틴” 섹션을 참조하세요.

테이블스페이스가 복원되지 않음

다음 오류 메시지는 RDS for Db2가 데이터베이스를 성공적으로 복원했지만 하나 이상의 테이블스페이스를 복원할 수 없음을 나타냅니다.

```
Reason SQL0970N The system attempted to write to a read-only file.
Reason SQL2563W The Restore process has completed successfully. However one or more
 table spaces from the backup were not restored.
```

RDS for Db2는 비자동 스토리지를 지원하지 않습니다. 비자동 스토리지를 자동 스토리지로 변환한 다음 `rdsadmin.restore_database`를 다시 직접적으로 호출합니다. 자세한 내용은 IBM Db2 설명서에서 자동 스토리지를 사용하도록 비자동 스토리지 데이터베이스 변환을 참조하세요.

비자동 SMS 스토리지가 있는 데이터베이스는 수동 복원이 필요합니다. 데이터베이스에 비자동 SMS 스토리지가 있는 경우 AWS 지원 팀에 문의하세요.

비자동 스토리지 및 일회성 마이그레이션에 대한 자세한 내용은 the section called “마이그레이션 중 자동이 아닌 스토리지 테이블스페이스” 섹션을 참조하세요.

rdsadmin.update_db_param 오류

the section called “rdsadmin.update_db_param” 저장된 프로시저를 직접적으로 호출할 때 다음과 같은 오류가 발생할 수 있습니다.

오류	오류 메시지
Parameter not supported or modifiable	`SQL0438N Application raised error or warning with diagnostic text: "Parameter is either not supported or not modifiable to customers". SQLSTATE=99993`

파라미터가 지원되지 않거나 수정할 수 없음

다음 오류 메시지는 지원되지 않거나 수정할 수 없는 데이터베이스 구성 파라미터를 수정하려고 했음을 나타냅니다.

```
SQL0438N Application raised error or warning with diagnostic text: "Parameter
is either not supported or not modifiable to customers". SQLSTATE=99993
```

파라미터 그룹을 보고 수정할 수 있는 파라미터를 확인할 수 있습니다. 자세한 내용은 the section called “DB 파라미터 그룹의 파라미터 값 보기” 섹션을 참조하세요.

Amazon RDS for MariaDB

Amazon RDS는 여러 버전의 MariaDB for DB 인스턴스를 지원합니다. 지원되는 버전에 대한 전체 정보는 Amazon RDS MariaDB 버전 섹션을 참조하세요.

MariaDB DB 인스턴스를 생성하려면 Amazon RDS 관리 도구 또는 인터페이스를 사용합니다. 그런 다음 Amazon RDS 도구를 사용하여 DB 인스턴스를 관리하는 작업을 수행할 수 있습니다. 이러한 작업에는 다음이 포함됩니다.

- DB 인스턴스 재구성 또는 크기 조정
- DB 인스턴스로의 연결 승인
- 백업 또는 스냅샷에서 생성 및 복원
- 보조 다중 AZ 생성
- 읽기 전용 복제본 생성
- DB 인스턴스의 성능 모니터링

DB 인스턴스의 데이터를 저장하고 이에 액세스하려면 표준 MariaDB 유틸리티 및 애플리케이션을 사용하면 됩니다.

MariaDB는 모든 AWS 리전에서 이용할 수 있습니다. AWS 리전에 대한 자세한 정보는 리전, 가용 영역 및 로컬 영역 섹션을 참조하세요.

Amazon RDS for MariaDB 데이터베이스를 사용하여 HIPAA 인증 애플리케이션을 개발할 수 있습니다. 예를 들어 AWS와 체결한 비즈니스 제휴 계약(BAA)에 따라 보호 대상 건강 정보(PHI)를 비롯한 의료 관련 정보를 저장할 수 있습니다. 자세한 내용은 HIPAA 규정 준수를 참조하세요. AWS 범위 내 서비스는 서드 파티 감사 기관의 철저한 평가를 거쳐 인증, 규정 준수 증명 또는 운영 권한(ATO)을 받았습니다. 자세한 내용은 규정 준수 프로그램 제공 범위 내 AWS 서비스를 참조하세요.

DB 인스턴스를 생성하기 전에 Amazon RDS 환경 설정 단계를 완료합니다. DB 인스턴스를 생성하면 RDS 마스터 사용자에 DBA 권한이 부여되나, 몇 가지 제한이 적용됩니다. 추가 데이터베이스 계정 생성과 같은 관리 작업에 이 계정을 사용합니다.

다음을 생성할 수 있습니다.

- DB 인스턴스
- DB 스냅샷

- 특정 시점 복원
- 자동 백업
- 수동 백업

Amazon VPC를 기반으로 Virtual Private Cloud(VPC)에서 MariaDB를 실행하는 DB 인스턴스를 사용할 수 있습니다. 또한 다양한 옵션을 활성화하여 MariaDB DB 인스턴스에 기능을 추가할 수 있습니다. Amazon RDS는 고가용성 장애 조치 솔루션으로 MariaDB용 다중 AZ 배포를 지원합니다.

> **Important**
>
> 관리형 서비스 환경을 제공하기 위해 Amazon RDS는 DB 인스턴스에 대해 shell 액세스를 제공하지 않습니다. 또한 고급 권한이 필요한 특정 시스템 절차와 테이블에 대한 액세스를 제한합니다. mysql 클라이언트와 같은 표준 SQL 클라이언트를 사용하여 데이터베이스에 액세스할 수 있습니다. 그러나 Telnet 또는 SSH(Secure Shell)를 사용하여 호스트에 직접 액세스할 수는 없습니다.

주제

- Amazon RDS에서의 MariaDB 기능 지원
- Amazon RDS MariaDB 버전
- MariaDB DB 인스턴스에 연결
- MariaDB DB 인스턴스 연결 보안
- Amazon RDS Optimized Reads로 RDS for MariaDB 쿼리 성능 개선
- Amazon RDS Optimized Writes for MariaDB를 통한 쓰기 성능 개선
- MariaDB DB 엔진 업그레이드
- MariaDB DB 스냅샷 엔진 버전 업그레이드
- Amazon RDS for MariaDB DB 인스턴스로 데이터 가져오기
- Amazon RDS에서 MariaDB 복제 작업
- MariaDB 데이터베이스 엔진을 위한 옵션
- MariaDB에 대한 파라미터
- MySQL DB 스냅샷에서 MariaDB DB 인스턴스로 데이터 마이그레이션
- Amazon RDS SQL의 MariaDB 참조

- MariaDB DB 인스턴스의 현지 시간대
- RDS for MariaDB에 대해 알려진 문제 및 제한 사항

Amazon RDS에서의 MariaDB 기능 지원

RDS for MariaDB는 MariaDB의 특징과 기능을 대부분 지원합니다. 일부 기능에는 제한된 지원 또는 제한된 권한이 있을 수 있습니다.

데이터베이스 관련 새로운 소식 페이지에서 새로운 Amazon RDS 기능을 필터링할 수 있습니다. [제품(Products)]에서 [Amazon RDS]를 선택합니다. 그런 다음 **`MariaDB 2023`**과 같은 키워드를 사용하여 검색합니다.

> **Note**
>
> 다음 목록이 전부는 아닙니다.

Amazon RDS의 MariaDB 기능 지원에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- Amazon RDS MariaDB에 대해 지원되는 스토리지 엔진
- Amazon RDS의 MariaDB를 위한 캐시 워밍
- Amazon RDS에서 지원하지 않는 MariaDB 기능

Amazon RDS for MariaDB 메이저 버전의 MariaDB 기능 지원

아래 섹션에서 다음 Amazon RDS for MariaDB 메이저 버전의 MariaDB 기능 지원에 대해 알아볼 수 있습니다.

주제

- Amazon RDS에서의 MariaDB 11.4 지원
- Amazon RDS에서의 MariaDB 10.11 지원
- Amazon RDS에서의 MariaDB 10.6 지원
- Amazon RDS에서의 MariaDB 10.5 지원

- Amazon RDS에서의 MariaDB 10.4 지원

지원되는 Amazon RDS for MariaDB 마이너 버전에 대한 자세한 내용은 Amazon RDS MariaDB 버전 섹션을 참조하세요.

Amazon RDS에서의 MariaDB 11.4 지원

Amazon RDS는 MariaDB 버전 11.4 이상을 실행하는 DB 인스턴스에서 다음과 같은 새로운 기능을 지원합니다.

- 암호화 라이브러리 - RDS for MariaDB는 OpenSSL을 FIPS 140-3 인증을 받은 AWS Libcrypto(AWS-LC)로 대체했습니다.
- Simple Password Check 플러그인 - MariaDB Simple Password Check 플러그인을 사용하여 암호에 특정 유형의 특정 문자 수가 포함되어 있는지 확인할 수 있습니다. 자세한 내용은 the section called “암호 확인 플러그인” 단원을 참조하십시오.
- Cracklib Password Check 플러그인 - MariaDB Cracklib Password Check 플러그인을 사용하여 새 암호의 강도를 확인할 수 있습니다. 자세한 내용은 the section called “암호 확인 플러그인” 단원을 참조하십시오.
- InnoDB 개선 사항 – 이러한 개선 사항에는 다음과 같은 항목이 포함됩니다.
  - 변경 버퍼가 제거되었습니다. 자세한 내용은 InnoDB 변경 버퍼링을 참조하세요.
  - InnoDB 조각 모음이 제거되었습니다. 자세한 내용은 InnoDB 조각 모음을 참조하세요.
- 새 권한 - 이제 관리 사용자에게도 `SHOW CREATE ROUTINE` 권한이 부여됩니다. 이 권한을 통해 피부여자는 다른 사용자가 소유한 루틴의 `SHOW CREATE` 정의 문을 볼 수 있습니다. 자세한 내용은 데이터베이스 권한을 참조하세요.
- 복제 개선 - MariaDB 버전 11.4 DB 인스턴스는 binlog 인덱싱을 지원합니다. 각 binlog 파일에 대해 GTID 인덱스를 생성할 수 있습니다. 이러한 인덱스는 GTID를 찾는 데 걸리는 시간을 줄여 복제 성능을 개선합니다. 자세한 내용은 Binlog 인덱싱을 참조하세요.
- 지원 종료 또는 삭제된 파라미터 – 다음 파라미터는 MariaDB 버전 11.4 DB 인스턴스의 지원을 종료하거나 삭제했습니다.
  - `engine_condition_pushdown`이 optimizer_switch에서 제거됨
  - innodb_change_buffer_max_size
  - innodb_defragment
  - `TLSv1.0` 및 `TLSv1.1`이 tls_version에서 제거됨
- 파라미터의 새 기본값 - innodb_undo_tablespaces 파라미터의 기본값이 `0`에서 `3`으로 변경되었습니다.

- 파라미터의 새로운 유횻값 – 다음 파라미터에는 MariaDB 버전 11.4 DB 인스턴스에 대한 새 유효한 값이 포함됩니다.
  - 이제 binlog_row_image 파라미터의 유효한 값에 FULL_NODUP가 포함됩니다.
  - 이제 OLD_MODE 파라미터의 유효한 값에 NO_NULL_COLLATION_IDS가 포함됩니다.
- 새 파라미터 – 다음 파라미터는 MariaDB 버전 11.4 DB 인스턴스에 새롭게 추가되었습니다.
  - transaction_isolation 파라미터는 tx_isolation 파라미터를 대체합니다.
  - transaction_read_only 파라미터는 tx_read_only 파라미터를 대체합니다.
  - block_encryption_mode 파라미터는 AES_ENCRYPT() 및 AES_DECRYPT() 함수의 기본 블록 암호화 모드를 정의합니다.
  - character_set_collations는 문자 세트 기본 데이터 정렬에 대한 재정의를 정의합니다.
  - binlog_gtid_index, binlog_gtid_index_page_size,binlog_gtid_index_span_min은 binlog GTID 인덱스의 속성을 정의합니다. 자세한 내용은 Binlog 인덱싱을 참조하세요.

MariaDB 11.4 전체 기능 목록과 설명서는 MariaDB 웹 사이트에서 MariaDB 11.4의 변경 사항 및 개선 사항과 릴리스 정보 - MariaDB 11.4 시리즈를 참조하세요.

지원되지 않는 기능 목록은 Amazon RDS에서 지원하지 않는 MariaDB 기능 단원을 참조하세요.

Amazon RDS에서의 MariaDB 10.11 지원

Amazon RDS는 MariaDB 버전 10.11 이상을 실행하는 DB 인스턴스에서 다음과 같은 새로운 기능을 지원합니다.

- 암호 재사용 확인 플러그인 – MariaDB 암호 재사용 확인 플러그인을 사용하여 사용자가 암호를 재사용하는 것을 방지하고 암호 보존 기간을 설정할 수 있습니다. 자세한 내용을 알아보려면 Password Reuse Check Plugin을 참조하세요.
- 퍼블릭에 권한 부여 인증 – 서버에 대한 액세스 권한이 있는 모든 사용자에게 권한을 부여할 수 있습니다. 자세한 내용은 GRANT TO PUBLIC을 참조하세요.
- 슈퍼 및 읽기 전용 관리자 권한 분리 – 이전에 슈퍼 권한을 가졌던 사용자를 비롯해 모든 사용자에게 부여된 읽기 전용 관리자 권한을 제거할 수 있습니다.
- 보안 – 이제 MariaDB 클라이언트의 기본값으로 --ssl 옵션을 설정할 수 있습니다. 구성이 잘못된 경우 MariaDB는 더 이상 SSL을 자동으로 비활성화하지 않습니다.
- SQL 명령 및 함수 – 이제 SHOW ANALYZE FORMAT=JSON 명령 및 함수 ROW_NUMBER, SFORMAT, RANDOM_BYTES를 사용할 수 있습니다. SFORMAT를 사용하면 문자열 형식을 지정할 수 있고, 이는 기본적으로 활성화되어 있습니다. 단일 명령으로 파티션을 테이블로, 테이블을 파티션으로 변환할

수 있습니다. JSON_*() 함수 관련 몇 가지 개선 사항도 있습니다. DES_ENCRYPT 및 DES_DECRYPT 함수는 버전 10.10 이상에서 더 이상 사용할 수 없습니다. 자세한 내용은 SFORMAT를 참조하세요.

- InnoDB 개선 사항 – 이러한 개선 사항에는 다음과 같은 항목이 포함됩니다.
  - 쓰기 증폭을 줄이고 동시성을 개선하기 위해 다시 실행 로그의 성능을 개선했습니다.
  - 데이터 디렉터리를 다시 초기화하지 않고도 다시 실행 테이블스페이스를 변경할 수 있습니다. 이 개선 사항을 통해 컨트롤 플레인 오버헤드를 줄여줍니다. 다시 시작해야 하지만 다시 실행 테이블스페이스를 변경한 후 다시 초기화할 필요는 없습니다.
  - CHECK TABLE … EXTENDED 및 내부적으로 내림차순 인덱스를 지원합니다.
  - 벌크 인서트가 개선되었습니다.
- Binlog 변경 – 이러한 변경 사항에는 다음 항목이 포함됩니다.
  - 두 단계를 거쳐 ALTER를 로깅하여 복제 지연 시간을 줄입니다. binlog_alter_two_phase 파라미터는 기본적으로 비활성화되어 있지만 파라미터 그룹을 통해 활성화할 수 있습니다.
  - explicit_defaults_for_timestamp를 로깅합니다.
  - 트랜잭션을 안전하게 롤백할 수 있으면 더 이상 INCIDENT_EVENT를 로깅하지 않습니다.
- 복제 개선 사항 – MariaDB 버전 10.11 DB 인스턴스는 마스터가 지원하는 경우 GTID 복제본을 기본적으로 사용합니다. 또한 Seconds_Behind_Master가 더 정확해졌습니다.
- 클라이언트 – mysqlbinlog 및 mariadb-dump에 새 명령줄 옵션을 사용할 수 있습니다. mariadb-dump를 사용해 과거 데이터를 덤프 및 복원할 수 있습니다.
- 시스템 버전 관리 – 기록을 수정할 수 있습니다. MariaDB는 새 파티션을 자동으로 생성합니다.
- 원자성 DDL – 이제 CREATE OR REPLACE는 원자적입니다. 문이 성공하거나 아니면 완전히 실패합니다.
- 다시 실행 로그 쓰기 – 다시 실행 로그는 비동기식으로 작성됩니다.
- 저장된 함수 – 이제 저장된 함수에서는 저장된 프로시저와 마찬가지로 IN, OUT, INOUT 파라미터를 지원합니다.
- 지원 종료 또는 삭제된 파라미터 – 다음 파라미터는 MariaDB 버전 10.11 DB 인스턴스의 지원을 종료하거나 삭제했습니다.
  - innodb_change_buffering
  - innodb_disallow_writes
  - innodb_log_write_ahead_size
  - innodb_prefix_index_cluster_optimization
  - keep_files_on_create
  - old

- 동적 파라미터 – 이제 다음 파라미터는 MariaDB 버전 10.11 DB 인스턴스에 대한 동적 파라미터입니다.
  - innodb_log_file_size
  - innodb_write_io_threads
  - innodb_read_io_threads
- 파라미터의 새로운 기본값 – 다음 파라미터에는 MariaDB 버전 10.11 DB 인스턴스에 대한 새로운 기본값이 포함됩니다.
  - explicit_defaults_for_timestamp 파라미터의 기본값은 OFF에서 ON으로 변경되었습니다.
  - optimizer_prune_level 파라미터의 기본값은 1에서 2로 변경되었습니다.
- 파라미터의 새로운 유횻값 – 다음 파라미터에는 MariaDB 버전 10.11 DB 인스턴스에 대한 새로운 유횻값이 포함됩니다.
  - old 파라미터의 유횻값은 old_mode 파라미터의 값에 통합되었습니다.
  - 이제 histogram_type 파라미터의 유횻값은 JSON_HB에 포함됩니다.
  - 이제 innodb_log_buffer_size 파라미터의 유횻값 범위는262144~4294967295입니다(256KB~4096MB).
  - 이제 innodb_log_file_size 파라미터의 유횻값 범위는 4194304~512GB입니다(4MB~512GB).
  - 이제 optimizer_prune_level 파라미터의 유횻값은 2에 포함됩니다.
- 새 파라미터 – 다음 파라미터는 MariaDB 버전 10.11 DB 인스턴스에 새롭게 추가되었습니다.
  - binlog_alter_two_phase 파라미터는 복제본 성능을 개선할 수 있습니다.
  - log_slow_min_examined_row_limit 파라미터는 성능을 개선할 수 있습니다.
  - log_slow_query 파라미터와 log_slow_query_file 파라미터는 각각 slow_query_log 및 slow_query_log_file의 별칭입니다.
  - optimizer_extra_pruning_depth
  - system_versioning_insert_history

MariaDB 10.11 전체 기능 목록과 설명서는 MariaDB 웹 사이트에서 MariaDB 10.11의 변경 사항 및 개선 사항과 릴리스 정보 - MariaDB 10.11 시리즈를 참조하세요.

지원되지 않는 기능 목록은 Amazon RDS에서 지원하지 않는 MariaDB 기능 단원을 참조하세요.

Amazon RDS에서의 MariaDB 10.6 지원

Amazon RDS는 MariaDB 버전 10.6 이상을 실행하는 DB 인스턴스에서 다음과 같은 새로운 기능을 지원합니다.

- MyRocks 스토리지 엔진 - RDS for MariaDB와 함께 MyRocks 스토리지 엔진을 사용하여 쓰기 집약적인 고성능 웹 애플리케이션의 스토리지 소비를 최적화할 수 있습니다. 자세한 내용은 Amazon RDS MariaDB에 대해 지원되는 스토리지 엔진 및 MyRocks.를 참조하세요.
- AWS Identity and Access Management(IAM) DB 인증 - IAM DB 인증을 사용하여 MariaDB DB 인스턴스에 대한 연결을 중앙 집중식으로 관리하고 보안을 강화할 수 있습니다. 자세한 내용은 MariaDB, MySQL 및 PostgreSQL IAM 데이터베이스 인증 단원을 참조하십시오.
- 업그레이드 옵션 - 이제 이전의 주요 릴리스(10.3, 10.4, 10.5)에서 RDS for MariaDB 버전 10.6으로 업그레이드할 수 있습니다. 또한 기존 MySQL 5.6 또는 5.7 DB 인스턴스의 스냅샷을 MariaDB 10.6 인스턴스로 복원할 수 있습니다. 자세한 내용은 MariaDB DB 엔진 업그레이드을 참조하세요.
- 지연 복제 - 이제 읽기 전용 복제본이 소스 데이터베이스보다 지연되는 구성 가능한 기간을 설정할 수 있습니다. 표준 MariaDB 복제 구성에서는 소스와 복제본 간의 복제 지연이 최소화됩니다. 지연 복제를 재해 복구를 위한 전략으로 사용할 수 있습니다. 자세한 정보는 MariaDB를 사용한 지연 복제 구성의 내용을 참조하세요.
- Oracle PL/SQL 호환성 - RDS for MariaDB 버전 10.6을 사용하면 레거시 Oracle 애플리케이션을 Amazon RDS로 보다 쉽게 마이그레이션할 수 있습니다. 자세한 내용은 SQL_MODE=ORACLE 섹션을 참조하세요.
- 원자성 DDL - 동적 데이터 언어(DDL) 문은 RDS for MariaDB 버전 10.6을 사용하면 상대적으로 충돌 안정성을 개선할 수 있습니다. `CREATE TABLE`, `ALTER TABLE`, `RENAME TABLE`, `DROP TABLE`, `DROP DATABASE` 및 관련 DDL 문은 이제 원자성입니다. 명령이 성공하거나 완전히 반전됩니다. 자세한 내용은 원자성 DDL을 참조하세요.
- 기타 기능 향상 - 이러한 기능 향상에는 SQL 내에서 JSON 데이터를 관계형 형식으로 변환하고, Innodb를 사용하여 빈 테이블 데이터를 더 빠르게 로드하는 `JSON_TABLE` 함수를 제공합니다. 여기에는 분석 및 문제 해결, 사용되지 않는 인덱스를 무시하기 위한 최적화 도구 개선 및 성능 개선을 위한 새로운 `sys_schema`도 포함됩니다. 자세한 내용은 JSON_TABLE을 참조하세요.
- 파라미터의 새로운 기본값 – 다음 파라미터에는 MariaDB 버전 10.6 DB 인스턴스에 대한 새로운 기본값이 있습니다.
  - 다음 파라미터의 기본값은 `utf8`에서 `utf8mb3`으로 변경되었습니다.
    - character_set_client
    - character_set_connection
    - character_set_results
    - character_set_system

    이러한 파라미터에 대한 기본값이 변경되었지만 기능적 변경은 없습니다. 자세한 내용은 MariaDB 설명서의 지원되는 문자 집합 및 데이터 정렬을 참조하세요.

- collation_connection 파라미터의 기본값이 utf8_general_ci에서 utf8mb3_general_ci로 변경되었습니다. 이러한 파라미터에 대한 기본값이 변경되었지만 기능적 변경은 없습니다.
- old_mode 파라미터의 기본값이 설정 해제에서 UTF8_IS_UTF8MB3으로 변경되었습니다. 이러한 파라미터에 대한 기본값이 변경되었지만 기능적 변경은 없습니다.

MariaDB 10.6 전체 기능 목록과 설명서는 MariaDB 웹 사이트에서 MariaDB 10.6의 변경 사항 및 개선 사항과 릴리스 정보 - MariaDB 10.6시리즈를 참조하세요.

지원되지 않는 기능 목록은 Amazon RDS에서 지원하지 않는 MariaDB 기능 단원을 참조하세요.

Amazon RDS에서의 MariaDB 10.5 지원

Amazon RDS는 MariaDB 버전 10.5 이상을 실행하는 DB 인스턴스에서 다음과 같은 새로운 기능을 지원합니다.

- InnoDB 기능 향상 – MariaDB 버전 10.5에는 향상된 InnoDB 기능이 포함되어 있습니다. 자세한 내용은 MariaDB 설명서의 InnoDB: 성능 향상 등을 참조하세요.
- 성능 스키마 업데이트 – MariaDB 버전 10.5에는 성능 스키마 업데이트가 포함되어 있습니다. 자세한 내용은 MariaDB 설명서의 MySQL 5.7 계측 및 테이블과 일치하도록 성능 스키마 업데이트를 참조하세요.
- InnoDB Redo 로그의 1개 파일 – MariaDB 버전 10.5 이전 버전에서는 innodb_log_files_in_group 파라미터 값이 2로 설정되었습니다. MariaDB 버전 10.5에서 이 파라미터의 값은 1로 설정됩니다.

  이전 버전에서 MariaDB 버전 10.5로 업그레이드하고 이 파라미터를 수정하지 않으면 innodb_log_file_size 파라미터 값이 변경되지 않습니다. 하지만 2개가 아닌 1개의 로그 파일에 적용됩니다. 그 결과 업그레이드된 MariaDB 버전 10.5 DB 인스턴스가 업그레이드 전에 사용했던 Redo 로그 크기의 절반을 사용합니다. 이 변경 사항은 성능에 눈에 띄는 영향을 미칠 수 있습니다. 이 문제를 해결하기 위해 innodb_log_file_size 파라미터 값을 두 배로 늘릴 수 있습니다. 파라미터 수정에 대한 자세한 정보는 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 단원을 참조하세요.
- SHOW SLAVE STATUS 명령이 지원되지 않음 – MariaDB 버전 10.5 이전 버전에서는 SHOW SLAVE STATUS 명령에 REPLICATION SLAVE 권한이 필요했습니다. MariaDB 버전 10.5에서는 이에 상응하는 SHOW REPLICA STATUS 명령에 REPLICATION REPLICA ADMIN 권한이 필요합니다. 이 새 권한은 RDS 마스터 사용자에게 부여되지 않습니다.

SHOW REPLICA STATUS 명령을 사용하는 대신 새 mysql.rds_replica_status 저장 프로시저를 실행하여 유사한 정보를 반환합니다. 자세한 내용은 mysql.rds_replica_status 섹션을 참조하세요.

- SHOW RELAYLOG EVENTS 명령이 지원되지 않음 – MariaDB 버전 10.5 이전 버전에서는 SHOW RELAYLOG EVENTS 명령에 REPLICATION SLAVE 권한이 필요했습니다. MariaDB 버전 10.5에서 이 명령에는 REPLICATION REPLICA ADMIN 권한이 필요합니다. 이 새 권한은 RDS 마스터 사용자에게 부여되지 않습니다.
- 파라미터의 새로운 기본값 – 다음 파라미터에는 MariaDB 버전 10.5 DB 인스턴스에 대한 새로운 기본값이 있습니다.
  - max_connections 파라미터의 기본값이 LEAST({DBInstanceClassMemory/25165760},12000)로 변경되었습니다. LEAST 파라미터 함수에 대한 자세한 내용은 DB 파라미터 함수 섹션을 참조하세요.
  - innodb_adaptive_hash_index 파라미터의 기본값이 OFF(0)로 변경되었습니다.
  - innodb_checksum_algorithm 파라미터의 기본값이 full_crc32로 변경되었습니다.
  - innodb_log_file_size 파라미터의 기본값이 2GB로 변경되었습니다.

MariaDB 10.5 전체 기능 목록과 설명서는 MariaDB 웹 사이트에서 MariaDB 10.4의 변경 사항 및 개선 사항과 릴리스 정보 - MariaDB 10.5 시리즈를 참조하세요.

지원되지 않는 기능 목록은 Amazon RDS에서 지원하지 않는 MariaDB 기능 단원을 참조하세요.

Amazon RDS에서의 MariaDB 10.4 지원

Amazon RDS는 MariaDB 버전 10.4 이상을 실행하는 DB 인스턴스에서 다음과 같은 새로운 기능을 지원합니다.

- 사용자 계정 보안 기능 향상 – 암호 만료 및 계정 잠금 개선
- Optimizer 기능 향상 – Optimizer Trace 기능
- InnoDB 기능 향상 – 즉각적인 DROP COLUMN 지원과 VARCHAR 및 ROW_FORMAT=DYNAMIC에 대한 즉각적인 ROW_FORMAT=COMPACT 확장
- 새 파라미터 – tcp_nodedelay, tls_version 및 gtid_cleanup_batch_size 포함

MariaDB 10.4 전체 기능 목록과 설명서는 MariaDB 웹 사이트에서 MariaDB 10.4의 변경 사항 및 개선 사항과 릴리스 정보 - MariaDB 10.4 시리즈를 참조하세요.

지원되지 않는 기능 목록은 Amazon RDS에서 지원하지 않는 MariaDB 기능 단원을 참조하세요.

Amazon RDS MariaDB에 대해 지원되는 스토리지 엔진

RDS for MariaDB는 다음과 같은 스토리지 엔진을 지원합니다.

주제

- InnoDB 스토리지 엔진
- MyRocks 스토리지 엔진

현재 다른 스토리지 엔진은 RDS for MariaDB에서 지원되지 않습니다.

InnoDB 스토리지 엔진

MariaDB는 다양한 기능을 가진 여러 스토리지 엔진을 지원하지만, 모든 엔진이 복구와 데이터 내구성에 최적화되어 있지는 않습니다. InnoDB는 Amazon RDS에서 MariaDB DB 인스턴스용으로 권장되는 스토리지 엔진입니다. 특정 시점으로 복구 및 스냅샷 복원 같은 Amazon RDS 기능을 사용하려면 복구 가능한 스토리지 엔진이 필요하며, 이러한 기능은 MariaDB 버전에 대한 권장 스토리지 엔진에서만 지원됩니다.

자세한 내용은 InnoDB를 참조하세요.

MyRocks 스토리지 엔진

RDS for MariaDB 버전 10.6 이상에서 MyRocks 스토리지 엔진을 사용할 수 있습니다. 프로덕션 데이터베이스에서 MyRocks 스토리지 엔진을 사용하기 전에 철저한 벤치마킹 및 테스트를 수행하여 사용 사례에 대해 InnoDB에 비한 잠재적 이점을 확인하는 것이 좋습니다.

MariaDB 버전 10.6의 기본 파라미터 그룹에는 MyRocks 파라미터가 포함되어 있습니다. 자세한 정보는 MariaDB에 대한 파라미터 및 Amazon RDS의 파라미터 그룹 단원을 참조하세요.

MyRocks 스토리지 엔진을 사용하는 테이블을 만들려면 `ENGINE=RocksDB` 문에서 `CREATE TABLE`를 지정합니다. 다음 예제에서는 MyRocks 스토리지 엔진을 사용하는 테이블을 생성합니다.

```
CREATE TABLE test (a INT NOT NULL, b CHAR(10)) ENGINE=RocksDB;
```

InnoDB 및 MyRocks 테이블 모두에 걸쳐 있는 트랜잭션은 실행하지 않는 것이 좋습니다. MariaDB는 스토리지 엔진 간 트랜잭션에 대해 ACID(원자성, 일관성, 격리, 내구성)를 보장하지 않습니다. DB 인스턴스에 InnoDB와 MyRocks 테이블을 둘 다 가질 수 있지만, 한 스토리지 엔진에서 다른 스토리지 엔진으로 마이그레이션하는 경우를 제외하고는 이 방법을 사용하지 않는 것이 좋습니다. InnoDB와

MyRocks 테이블이 모두 DB 인스턴스에 존재하는 경우 각 스토리지 엔진마다 고유한 버퍼 풀이 있으므로 성능이 저하될 수 있습니다.

MyRocks는 SERIALIZABLE 격리 또는 갭 잠금을 지원하지 않습니다. 따라서 일반적으로 명령문 기반 복제와 함께 MyRocks를 사용할 수 없습니다. 자세한 내용은 MyRocks 및 복제를 참조하세요.

현재 다음 MyRocks 파라미터만 수정할 수 있습니다.

- rocksdb_block_cache_size
- rocksdb_bulk_load
- rocksdb_bulk_load_size
- rocksdb_deadlock_detect
- rocksdb_deadlock_detect_depth
- rocksdb_max_latest_deadlocks

MyRocks 스토리지 엔진과 InnoDB 스토리지 엔진은 rocksdb_block_cache_size 및 innodb_buffer_pool_size 파라미터에 따라 메모리를 놓고 경쟁할 수 있습니다. MyRocks 스토리지 엔진만 특정 DB 인스턴스에서 사용해야 하는 경우도 있습니다. 이 경우 innodb_buffer_pool_size minimal 파라미터를 최소 값으로 설정하고 rocksdb_block_cache_size를 가능한 높게 설정하는 것이 좋습니다.

DescribeDBLogFiles 및 DownloadDBLogFilePortion 작업을 사용하여 MyRocks 로그 파일에 액세스할 수 있습니다.

MyRocks 대한 자세한 내용은 MariaDB 웹사이트의 MyRocks를 참조하세요.

Amazon RDS의 MariaDB를 위한 캐시 워밍

InnoDB 캐시 워밍은 DB 인스턴스가 종료될 때 버퍼 풀의 현재 상태를 저장한 다음 DB 인스턴스가 시작될 때 저장된 정보에서 버퍼 풀을 다시 로드하여 MariaDB DB 인스턴스의 성능 향상을 제공할 수 있습니다. 이렇게 하면 보통 데이터베이스 사용에서 "준비"까지의 버퍼 풀에 대한 필요를 무시하고, 대신 알려진 공용 쿼리에 대한 페이지와 함께 버퍼 풀을 미리 로드합니다. 캐시 워밍에 대한 자세한 내용은 MariaDB 설명서에서 Dumping and restoring the buffer pool을 참조하세요.

캐시 워밍은 MariaDB 10.3 이상 DB 인스턴스에서 기본적으로 활성화됩니다. 캐시 워밍을 활성화하려면 DB 인스턴스의 파라미터 그룹에서 innodb_buffer_pool_dump_at_shutdown 및 innodb_buffer_pool_load_at_startup 파라미터를 1로 설정합니다. 파라미터 그룹에서 이들

파라미터 값을 변경하면 파라미터 그룹을 사용하는 모든 MariaDB DB 인스턴스가 영향을 받습니다. 특정 MariaDB DB 인스턴스에 대해 캐시 워밍을 활성화하려면, 이들 DB 인스턴스에 대한 새 파라미터 그룹을 생성해야 할 수도 있습니다. 파라미터 그룹에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하세요.

캐시 워밍은 주로 표준 스토리지를 사용하는 DB 인스턴스를 위해 성능 혜택을 제공합니다. PIOPS 스토리지를 사용하는 경우에는 통상적으로 성능 혜택이 현저하지 않습니다.

Important

MariaDB DB 인스턴스가 정상적으로 종료되지 않는 경우(예: 장애 조치 도중), 버퍼 풀 상태가 디스크에 저장되지 않습니다. 이 경우 MariaDB는 DB 인스턴스가 다시 시작될 때 이용 가능한 모든 버퍼 풀 파일을 로드합니다. 어떤 손상도 발생하지 않지만, 복원된 버퍼 풀은 대부분의 경우 다시 시작하기 이전의 버퍼 풀 최신 상태를 반영하지 못할 수도 있습니다. 시작 시 캐시를 워밍하기 위해 버퍼 풀의 최신 상태를 이용할 수 있게 하려면, "요청 시" 버퍼 풀을 주기적으로 덤프하는 것이 좋습니다. 사용자가 요청 시 버퍼 풀을 덤프 또는 로드할 수 있습니다.
버퍼 풀을 자동으로 그리고 정기적으로 덤프하는 이벤트를 생성할 수 있습니다. 예를 들면, 다음 문은 매 시간마다 버퍼 풀을 덤프하는 이름이 `periodic_buffer_pool_dump`인 이벤트를 생성합니다.

```
CREATE EVENT periodic_buffer_pool_dump
   ON SCHEDULE EVERY 1 HOUR
   DO CALL mysql.rds_innodb_buffer_pool_dump_now();
```

자세한 내용은 MariaDB 설명서에서 Events를 참조하세요.

요청 시 버퍼 풀 덤핑 및 로딩

다음의 저장 프로시저를 사용해 요청 시 캐시를 저장하고 로드할 수 있습니다.

- 버퍼 풀의 현재 상태를 디스크에 덤프하려면 mysql.rds_innodb_buffer_pool_dump_now 저장 프로시저를 호출합니다.
- 디스크에서 저장된 버퍼 풀의 상태를 로드하려면 mysql.rds_innodb_buffer_pool_load_now 저장 프로시저를 호출합니다.
- 진행 중인 로드 작업을 취소하려면 mysql.rds_innodb_buffer_pool_load_abort 저장 프로시저를 호출합니다.

Amazon RDS에서 지원하지 않는 MariaDB 기능

다음 MariaDB 기능은 Amazon RDS에서 지원되지 않습니다.

- S3 스토리지 엔진
- 인증 플러그인 – GSSAPI
- 인증 플러그인 – Unix 소켓
- AWS Key Management 암호화 플러그 인
- 10.6보다 낮은 MariaDB 버전에 대한 지연 복제
- InnoDB 및 Aria에 대한 기본 유휴 시 MariaDB 암호화

  Amazon RDS 리소스 암호화의 지침에 따라 MariaDB DB 인스턴스에 유휴 시 암호화를 활성화할 수 있습니다.
- HandlerSocket
- 10.6보다 낮은 MariaDB 버전에 대한 JSON 테이블 유형
- MariaDB ColumnStore
- MariaDB Galera Cluster
- 멀티 소스 복제
- 10.6보다 낮은 MariaDB 버전용 MyRocks 스토리지 엔진
- 11.4 미만의 MariaDB 버전에 대한 암호 검증 플러그인, `simple_password_check` 및 `cracklib_password_check`
- 스파이더 스토리지 엔진
- Sphinx 스토리지 엔진
- TokuDB 스토리지 엔진
- 스토리지 엔진별 객체 속성(MariaDB 설명서의 Engine-defined New Table/Field/Index Attributes에서 설명).
- 테이블 및 tablespace 암호화
- Hashicorp 키 관리 플러그인
- 두 업그레이드를 병렬로 실행

관리형 서비스 환경을 제공하기 위해 Amazon RDS는 DB 인스턴스에 대해 shell 액세스를 제공하지 않으며, 고급 권한을 필요로 하는 특정 시스템 절차와 테이블에 대한 액세스를 제한합니다. Amazon RDS는 표준 SQL 클라이언트 애플리케이션을 사용하여 DB 인스턴스의 데이터베이스에 대한 액세스를 지

원합니다. Amazon RDS는 Telnet, SSH(보안 셸) 또는 Windows 원격 데스크톱 연결을 사용하여 DB 인스턴스에 대한 직접적인 호스트 액세스를 허용하지 않습니다.

Amazon RDS MariaDB 버전

MariaDB 버전 번호는 버전 X.Y.Z로 구성됩니다. Amazon RDS 용어에서 X.Y는 메이저 버전을 나타내고 Z는 마이너 버전 번호를 나타냅니다. Amazon RDS 구현을 위해서, 메이저 버전 번호가 변경될 경우(예: 버전 10.5에서 10.6으로 변경) 이를 메이저 버전 변경으로 간주합니다. 마이너 버전 번호만 변경된 경우(예: 버전 10.6.14에서 10.6.16로 변경)에는 마이너 버전 변경으로 간주합니다.

주제

- Amazon RDS에서 지원되는 MariaDB 마이너 버전
- Amazon RDS에서 지원되는 MariaDB 메이저 버전
- 데이터베이스 미리 보기 환경 작업
- 데이터베이스 미리 보기 환경의 MariaDB 버전 11.4
- 더 이상 사용되지 않는 Amazon RDS for MariaDB 버전

Amazon RDS에서 지원되는 MariaDB 마이너 버전

Amazon RDS는 현재 다음과 같은 MariaDB 마이너 버전을 지원합니다.

> **Note**
>
> 월과 연도만 있는 날짜는 대략적인 날짜이며 알 수 있는 정확한 날짜로 업데이트됩니다.

다음 표는 현재 Amazon RDS가 지원하는 MariaDB 10.11의 마이너 버전을 보여줍니다.

MariaDB 엔진 버전	커뮤니티 릴리스 날짜	RDS 릴리스 날짜	RDS 표준 지원 종료일
10.11.8	2024년 5월 16일	2024년 6월 14일	2025년 9월
10.11.7	2024년 2월 7일	2024년 2월 26일	2025년 3월
10.11.6	2023년 11월 13일	2023년 12월 12일	2025년 3월
10.11.5	2023년 8월 14일	2023년 9월 7일	2025년 3월
10.11.4	2023년 6월 7일	2023년 8월 21일	2025년 3월

다음 표는 현재 Amazon RDS가 지원하는 MariaDB 10.6의 마이너 버전을 보여줍니다.

MariaDB 엔진 버전	커뮤니티 릴리스 날짜	RDS 릴리스 날짜	RDS 표준 지원 종료일
10.6.18	2024년 5월 16일	2024년 6월 14일	2025년 9월
10.6.17	2024년 2월 7일	2024년 2월 26일	2025년 3월
10.6.16	2023년 11월 13일	2023년 12월 12일	2025년 3월
10.6.15	2023년 8월 14일	2023년 9월 7일	2025년 3월
10.6.14	2023년 6월 7일	2023년 6월 22일	2025년 3월
10.6.13	2023년 5월 10일	2023년 6월 15일	2025년 3월

다음 표는 현재 Amazon RDS가 지원하는 MariaDB 10.5의 마이너 버전을 보여줍니다.

MariaDB 엔진 버전	커뮤니티 릴리스 날짜	RDS 릴리스 날짜	RDS 표준 지원 종료일
10.5.25	2024년 5월 16일	2024년 6월 14일	2025년 9월
10.5.24	2024년 2월 7일	2024년 2월 26일	2025년 3월
10.5.23	2023년 11월 13일	2023년 12월 12일	2025년 3월
10.5.22	2023년 8월 14일	2023년 9월 7일	2025년 3월
10.5.21	2023년 6월 7일	2023년 6월 22일	2025년 3월
10.5.20	2023년 5월 10일	2023년 6월 15일	2025년 3월

다음 표는 현재 Amazon RDS가 지원하는 MariaDB 10.4의 마이너 버전을 보여줍니다.

MariaDB 엔진 버전	커뮤니티 릴리스 날짜	RDS 릴리스 날짜	RDS 표준 지원 종료일
10.4.34	2024년 5월 16일	2024년 6월 14일	2025년 2월
10.4.33	2024년 2월 7일	2024년 2월 26일	2025년 2월

MariaDB 엔진 버전	커뮤니티 릴리스 날짜	RDS 릴리스 날짜	RDS 표준 지원 종료일
10.4.32	2023년 11월 13일	2023년 12월 12일	2025년 2월
10.4.31	2023년 8월 14일	2023년 9월 7일	2025년 2월
10.4.30	2023년 6월 7일	2023년 6월 22일	2025년 2월
10.4.29	2023년 5월 10일	2023년 6월 15일	2025년 2월

새 DB 인스턴스를 생성할 때는 현재 지원되는 모든 MariaDB 버전을 지정할 수 있습니다. MariaDB 10.5와 같이 메이저 버전과 지정된 메이저 버전에 대해 지원되는 모든 마이너 버전을 지정할 수 있습니다. 버전이 지정되지 않은 경우 Amazon RDS는 지원되는 버전(보통 최신 버전)을 기본값으로 설정합니다. 메이저 버전이 지정되었지만 마이너 버전이 지정되지 않은 경우, Amazon RDS는 고객이 지정한 메이저 버전의 최근 릴리스를 기본값으로 설정합니다. 지원되는 버전 목록과 새로 만든 DB 인스턴스의 기본값을 보려면 `describe-db-engine-versions` AWS CLI 명령을 사용합니다.

예를 들어 RDS for MariaDB 버전 목록을 보려면 다음 CLI 명령을 실행합니다.

```
aws rds describe-db-engine-versions --engine mariadb --query "*[].
{Engine:Engine,EngineVersion:EngineVersion}" --output text
```

기본 MariaDB 버전은 AWS 리전에 따라 다를 수 있습니다. 특정 마이너 버전으로 DB 인스턴스를 생성하려면 DB 인스턴스 생성 중에 마이너 버전을 지정합니다. 다음 AWS CLI 명령을 사용하여 AWS 리전의 기본 마이너 버전을 확인할 수 있습니다.

```
aws rds describe-db-engine-versions --default-only --engine mariadb
 --engine-version major-engine-version --region region --query "*[].
{Engine:Engine,EngineVersion:EngineVersion}" --output text
```

*major-engine-version*을 메이저 엔진 버전으로 바꾸고 *##*을 AWS 리전으로 바꿉니다. 예를 들어, 다음 AWS CLI 명령은 10.5 메이저 버전과 미국 서부(오레곤) AWS 리전(us-west-2)에 대한 기본 MariaDB 마이너 엔진 버전을 반환합니다.

```
aws rds describe-db-engine-versions --default-only --engine mariadb --engine-version
 10.5 --region us-west-2 --query "*[].{Engine:Engine,EngineVersion:EngineVersion}" --
output text
```

Amazon RDS에서 지원되는 MariaDB 메이저 버전

RDS for MariaDB 메이저 버전은 해당 커뮤니티 버전에 대한 커뮤니티 수명이 끝날 때까지 사용할 수 있습니다. 다음 날짜를 사용하여 테스트 및 업그레이드 주기를 계획할 수 있습니다. Amazon이 RDS for MariaDB 버전에 대한 지원을 원래 명시일보다 오래 연장할 경우, 이 표를 이후 날짜를 반영하도록 업데이트할 계획입니다.

> **Note**
>
> 월과 연도만 있는 날짜는 대략적인 날짜이며 알 수 있는 정확한 날짜로 업데이트됩니다.

MariaDB 메이저 버전	커뮤니티 릴리스 날짜	RDS 릴리스 날짜	커뮤니티 수명 종료 날짜	RDS 표준 지원 종료일
MariaDB 10.11	2023년 2월 16일	2023년 8월 21일	2028년 2월 16일	2028년 2월
MariaDB 10.6	2021년 7월 6일	2022년 2월 3일	2026년 7월 6일	2026년 7월
MariaDB 10.5	2020년 6월 24일	2021년 1월 21일	2024년 6월 24일	2025년 6월
MariaDB 10.4	2019년 6월 18일	2020년 4월 6일	2024년 6월 18일	2025년 2월

데이터베이스 미리 보기 환경 작업

데이터베이스 미리 보기 환경의 RDS for MariaDB DB 인스턴스는 다른 RDS for MariaDB DB 인스턴스와 기능적으로 유사합니다. 하지만 프로덕션 워크로드에는 데이터베이스 미리 보기 환경을 사용할 수 없습니다.

미리 보기 환경에는 다음과 같은 제한 사항이 적용됩니다.

- Amazon RDS는 생성 시점으로부터 60일이 지난 모든 DB 인스턴스를 백업 및 스냅샷과 함께 삭제합니다.
- 범용 SSD와 프로비저닝된 IOPS SSD 스토리지만 사용할 수 있습니다.
- DB 인스턴스와 관련해서는 지원의 도움을 받을 수 없습니다. 대신 AWS 관리형 Q&A 커뮤니티인 AWS re:Post에 질문을 게시할 수 있습니다.
- DB 인스턴스의 스냅샷을 프로덕션 환경으로 복제할 수 없습니다.

미리 보기에서 지원되는 옵션은 다음과 같습니다.

- db.m6i, db.r6i, db.m6g, db.m5, db.t3, db.r6g, and db.r5 DB 인스턴스 클래스를 사용하여 DB 인스턴스를 생성할 수 있습니다. RDS 인스턴스 클래스에 대한 자세한 내용은 DB 인스턴스 클래스 단원을 참조하세요.
- 단일 AZ 배포와 다중 AZ 배포를 모두 사용할 수 있습니다.
- 표준 MariaDB 덤프 및 로드 함수를 사용하여 데이터베이스 미리 보기 환경에서 데이터베이스를 내보내거나 데이터베이스 미리 보기 환경으로 데이터베이스를 가져올 수 있습니다.

데이터베이스 미리 보기 환경에서 지원하지 않는 기능

다음 기능은 데이터베이스 미리 보기 환경에서 사용할 수 없습니다.

- 리전 간 스냅샷 복제
- 리전 간 읽기 전용 복제본
- RDS 프록시

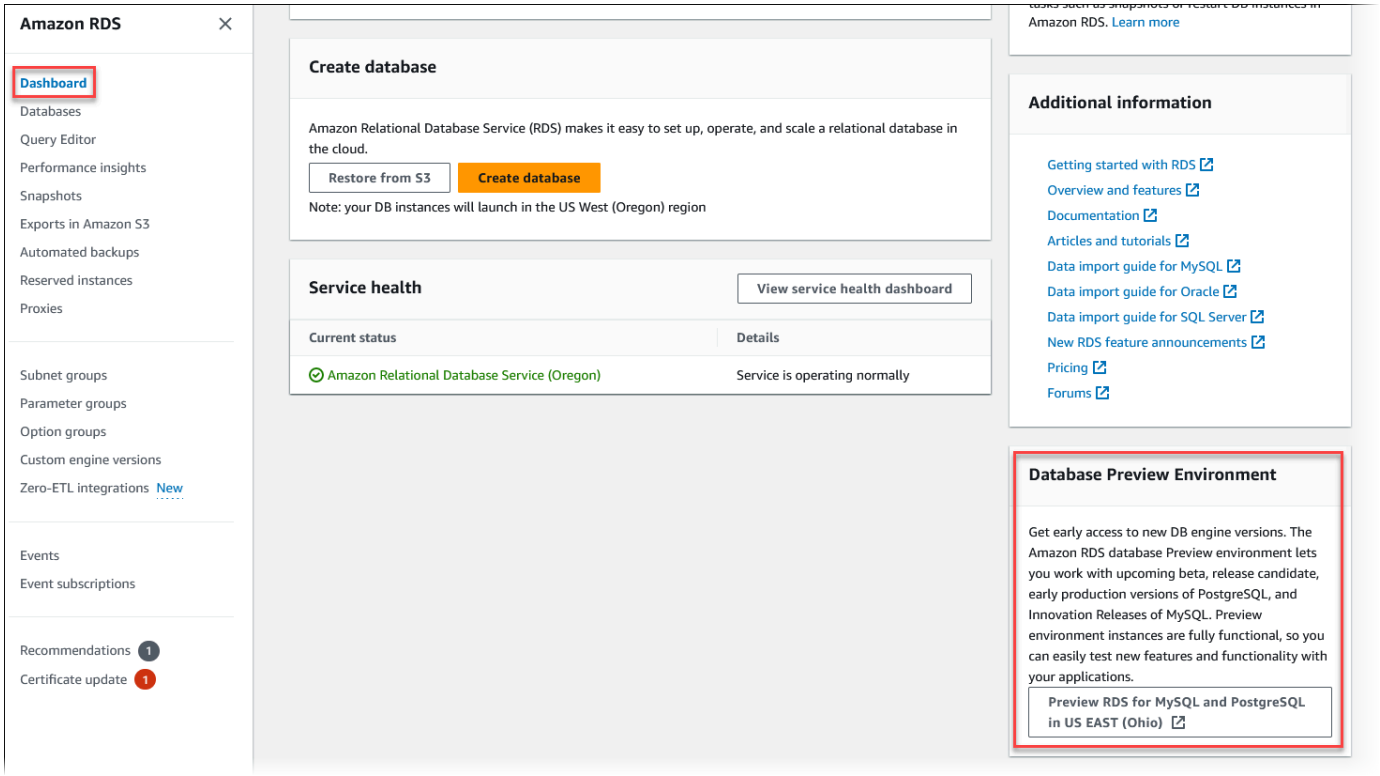
데이터베이스 미리 보기 환경에서 새 DB 인스턴스 생성

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 데이터베이스 미리 보기 환경에서 DB 인스턴스를 생성할 수 있습니다.

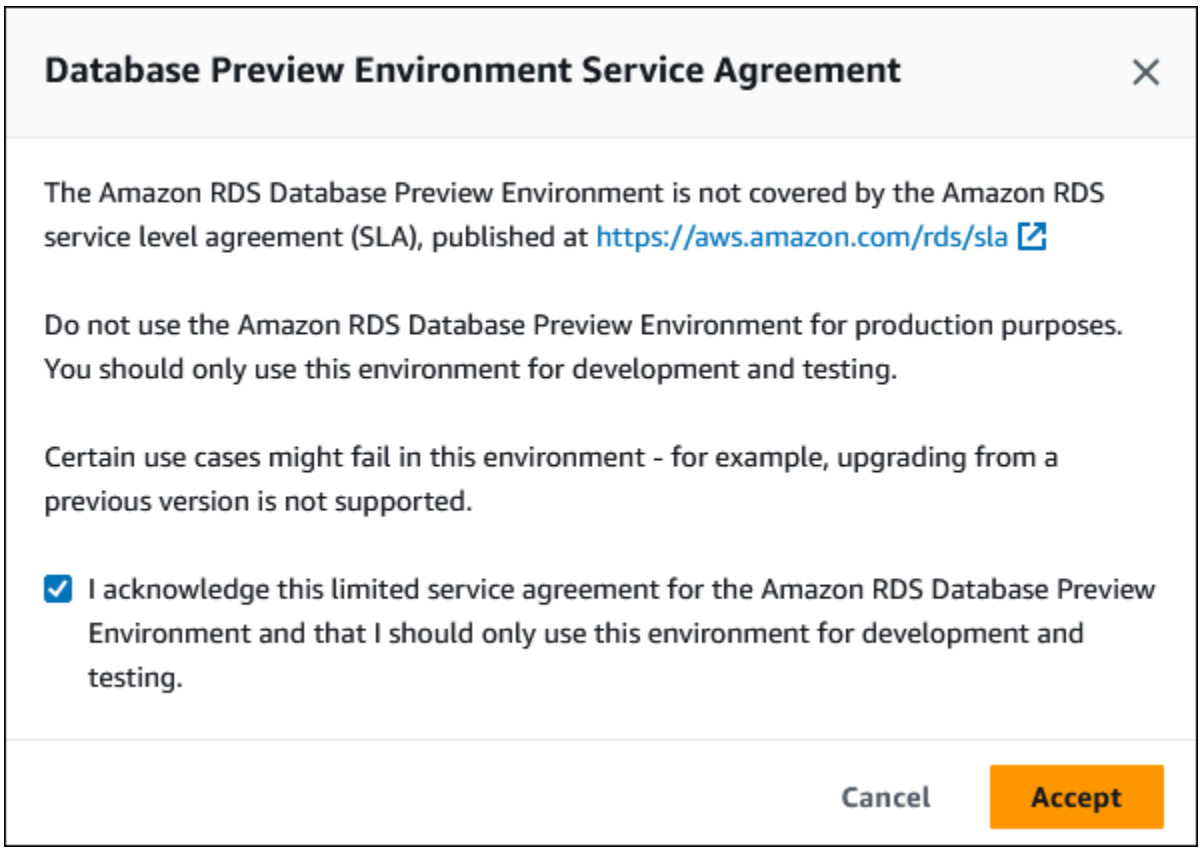
콘솔

데이터베이스 미리 보기 환경에서 DB 인스턴스를 생성하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 대시보드를 선택합니다.
3. 다음 그림과 같이 대시보드 페이지에서 데이터베이스 미리 보기 환경 섹션을 찾습니다.



데이터베이스 미리 보기 환경으로 바로 이동할 수도 있습니다. 계속하려면 먼저 제한 사항을 확인하고 수락해야 합니다.



4. RDS for MariaDB DB 인스턴스를 생성하려면 일반적으로 Amazon RDS DB 인스턴스를 생성할 때와 동일한 프로세스를 따릅니다. 자세한 내용은 DB 인스턴스 생성의 콘솔 절차를 참조하세요.

AWS CLI

AWS CLI를 사용하여 데이터베이스 미리 보기 환경에 DB 인스턴스를 생성하려면 다음 엔드포인트를 사용합니다.

```
rds-preview.us-east-2.amazonaws.com
```

RDS for MariaDB DB 인스턴스를 생성하려면 일반적으로 Amazon RDS DB 인스턴스를 생성할 때와 동일한 프로세스를 따릅니다. 자세한 내용은 DB 인스턴스 생성의 AWS CLI 절차를 참조하세요.

RDS API

RDS API를 사용하여 데이터베이스 미리 보기 환경에 DB 인스턴스를 생성하려면 다음 엔드포인트를 사용합니다.

```
rds-preview.us-east-2.amazonaws.com
```

RDS for MariaDB DB 인스턴스를 생성하려면 일반적으로 Amazon RDS DB 인스턴스를 생성할 때와 동일한 프로세스를 따릅니다. 자세한 내용은 DB 인스턴스 생성의 RDS API 절차를 참조하세요.

데이터베이스 미리 보기 환경의 MariaDB 버전 11.4

이제 MariaDB 버전 11.4를 Amazon RDS 데이터베이스 미리 보기 환경에서 사용할 수 있습니다. MariaDB 버전 11.4에는 MariaDB 11.4의 변경 및 개선 사항에 설명된 여러 개선 사항이 포함되어 있습니다. 데이터베이스 미리 보기 환경을 사용하여 모든 AWS 리전 프로덕션 워크로드에 사용할 수 있게 되기 전에 이 릴리스를 기준으로 워크로드를 테스트할 수 있습니다.

데이터베이스 미리 보기 환경에 대한 자세한 내용은 the section called “데이터베이스 미리 보기 환경” 섹션을 참조하세요. 콘솔에서 미리 보기 환경에 액세스하려면 rds-preview/를 선택합니다.

더 이상 사용되지 않는 Amazon RDS for MariaDB 버전

Amazon RDS for MariaDB 버전 10.0, 10.1, 10.2, 10.3은 더 이상 지원되지 않습니다.

MariaDB에 대한 Amazon RDS 사용 중단 정책에 대한 자세한 내용은 Amazon RDS FAQ를 참조하세요.

MariaDB DB 인스턴스에 연결

Amazon RDS가 DB 인스턴스를 프로비저닝한 후에는 표준 MariaDB 클라이언트 애플리케이션 또는 유틸리티를 사용하여 인스턴스에 연결할 수 있습니다. 연결 문자열에서 DB 인스턴스 엔드포인트의 Domain Name System(DNS) 주소를 호스트 파라미터로 지정합니다. 또한 DB 인스턴스 엔드포인트의 포트 번호를 포트 파라미터로 지정합니다.

Amazon RDS for MariaDB DB 인스턴스는 MySQL 명령줄 클라이언트 같은 도구를 사용하여 연결할 수 있습니다. MySQL 명령줄 클라이언트 사용에 대한 자세한 내용은 MariaDB 문서의 mysql 명령줄 클라이언트를 참조하세요. 연결에 사용할 수 있는 GUI 기반 애플리케이션 중 하나는 Heidi입니다. 자세한 내용은 HeidiSQL 다운로드 페이지를 참조하십시오. MySQL 설치(MySQL 명령줄 클라이언트 포함)에 대한 정보는 MySQL 설치 및 업그레이드를 참조하세요.

대부분의 Linux 배포에는 Oracle MySQL 클라이언트 대신 MariaDB 클라이언트가 포함됩니다. Amazon Linux 2023에서 MySQL 명령줄 클라이언트를 설치하려면 다음 명령을 실행합니다.

```
sudo dnf install mariadb105
```

Amazon Linux 2에서 MySQL 명령줄 클라이언트를 설치하려면 다음 명령을 실행합니다.

```
sudo yum install mariadb
```

대부분의 DEB 기반 Linux 배포판에 MySQL 명령줄 클라이언트를 설치하려면 다음 명령을 실행합니다.

```
apt-get install mariadb-client
```

MySQL 명령줄 클라이언트의 버전을 확인하려면 다음 명령을 실행합니다.

```
mysql --version
```

현재 클라이언트 버전에 대한 MySQL 설명서를 보려면 다음 명령을 실행합니다.

```
man mysql
```

Amazon VPC 기반의 Virtual Private Cloud(VPC) 외부에서 DB 인스턴스에 연결하려면 DB 인스턴스에 공개적으로 액세스할 수 있어야 합니다. 또한 DB 인스턴스 보안 그룹의 인바운드 규칙을 사용하여 액세스 권한을 부여해야 하며, 기타 요구 사항을 충족해야 합니다. 자세한 내용은 Amazon RDS DB 인스턴스에 연결할 수 없음 섹션을 참조하세요.

MariaDB DB 인스턴스에 연결할 때는 SSL 암호화를 사용할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB DB 인스턴스에 대한 SSL/TLS 지원을 참조하세요.

RDS for MariaDB DB 인스턴스를 찾아 연결하려면 다음 주제를 참조하세요.

주제

- MariaDB DB 인스턴스에 대한 연결 정보 찾기
- RDS for MariaDB용 MySQL 명령줄 클라이언트에서 연결(암호화되지 않음)
- AWS JDBC 드라이버 및 AWS Python 드라이버를 사용하여 RDS for MariaDB에 연결
- MariaDB DB 인스턴스에 대한 연결 문제 해결

MariaDB DB 인스턴스에 대한 연결 정보 찾기

DB 인스턴스의 연결 정보에는 엔드포인트, 포트 및 유효한 데이터베이스 사용자(예: 마스터 사용자)가 포함됩니다. 예를 들어 엔드포인트 값이 `mydb.123456789012.us-east-1.rds.amazonaws.com`이라고 가정합니다. 이 경우 포트 값은 `3306`이고 데이터베이스 사용자는 `admin`입니다. 이 정보를 바탕으로 연결 문자열에 다음 값을 지정합니다.

- 호스트 또는 호스트 이름 또는 DNS 이름에 `mydb.123456789012.us-east-1.rds.amazonaws.com`을 지정합니다.
- 포트에 대해 `3306`을 지정합니다.
- 사용자에 `admin`을 지정합니다.

DB 인스턴스에 연결하려면 MariaDB DB 엔진에 대해 임의의 클라이언트를 사용합니다. 예를 들어 MySQL 명령줄 클라이언트 또는 MySQL 워크벤치를 사용할 수 있습니다.

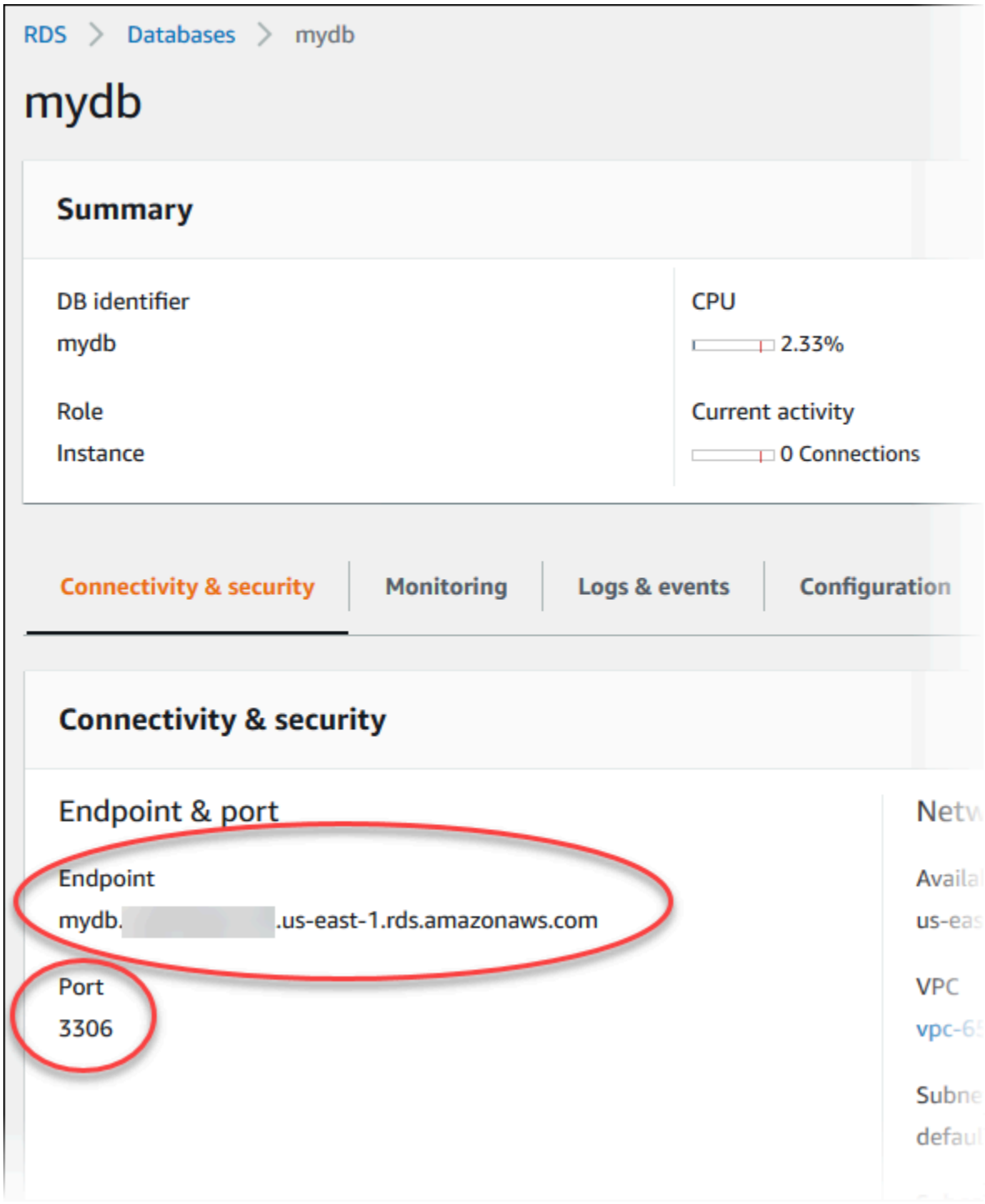
DB 인스턴스에 대한 연결 정보를 찾으려면 AWS Management Console, AWS Command Line Interface(AWS CLI) describe-db-instances 명령 또는 Amazon RDS API DescribeDBInstances 작업을 사용하여 세부 정보를 나열하면 됩니다.

콘솔

AWS Management Console에서 DB 인스턴스에 대한 연결 정보를 찾으려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [데이터베이스(Database)] 를 선택하여 DB 인스턴스 목록을 표시합니다.

3. MariaDB DB 인스턴스의 이름을 선택하여 세부 정보를 표시합니다.
4. Connectivity & security(연결 및 보안) 탭에서 엔드포인트를 복사합니다. 또한 포트 번호를 적어 둡니다. DB 인스턴스에 연결하려면 엔드포인트와 포트 번호가 모두 필요합니다.



5. 마스터 사용자 이름을 찾아야 하는 경우 [구성(Configuration)] 탭을 선택하고 [마스터 사용자 이름(Master username)] 값을 확인합니다.

AWS CLI

AWS CLI를 사용하여 MariaDB DB 인스턴스의 연결 정보를 찾으려면 describe-db-instances 명령을 실행합니다. 이 호출에서 DB 인스턴스 ID, 엔드포인트, 포트 및 마스터 사용자 이름을 쿼리합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-instances \
  --filters "Name=engine,Values=mariadb" \
  --query "*[].[DBInstanceIdentifier,Endpoint.Address,Endpoint.Port,MasterUsername]"
```

Windows의 경우:

```
aws rds describe-db-instances ^
  --filters "Name=engine,Values=mariadb" ^
  --query "*[].[DBInstanceIdentifier,Endpoint.Address,Endpoint.Port,MasterUsername]"
```

다음과 유사하게 출력되어야 합니다.

```
[
    [
        "mydb1",
        "mydb1.123456789012.us-east-1.rds.amazonaws.com",
        3306,
        "admin"
    ],
    [
        "mydb2",
        "mydb2.123456789012.us-east-1.rds.amazonaws.com",
        3306,
        "admin"
    ]
]
```

RDS API

Amazon RDS API를 사용하여 DB 인스턴스의 연결 정보를 찾으려면 DescribeDBInstances 작업을 호출합니다. 출력에서 엔드포인트 주소, 엔드포인트 포트 및 마스터 사용자 이름의 값을 찾습니다.

RDS for MariaDB용 MySQL 명령줄 클라이언트에서 연결(암호화되지 않음)

> **Important**
>
> 클라이언트와 서버가 동일한 VPC에 있고 네트워크를 신뢰할 수 있는 경우에만 암호화되지 않은 MySQL 연결을 사용합니다. 암호화된 연결 사용에 대한 자세한 내용은 MySQL 명령줄 클라이언트(암호화됨)에서 SSL/TLS를 사용하여 Amazon RDS의 MariaDB DB 인스턴스에 연결 섹션을 참조하세요.

MySQL 명령줄 클라이언트를 사용하여 DB 인스턴스에 연결하려면 클라이언트 컴퓨터의 명령 프롬프트에서 다음 명령을 입력합니다. 이렇게 하면 MariaDB DB 인스턴스의 데이터베이스로 연결됩니다. *<endpoint>*는 DB 인스턴스의 DNS 이름(엔드포인트)으로 대체하고, *<mymasteruser>*는 사용된 마스터 사용자 이름으로 대체합니다. 암호를 묻는 메시지가 표시되면 사용한 마스터 암호를 제공합니다.

```
mysql -h <endpoint> -P 3306 -u <mymasteruser> -p
```

사용자에 대한 암호를 입력하면 다음과 유사한 출력이 나타납니다.

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 10.6.10-MariaDB-log Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

AWS JDBC 드라이버 및 AWS Python 드라이버를 사용하여 RDS for MariaDB에 연결

AWS JDBC 드라이버 및 AWS Python 드라이버를 사용하여 RDS for MariaDB DB 인스턴스에 연결합니다. 자세한 내용은 다음 항목을 참조하십시오.

주제

- Amazon Web Services(AWS) JDBC 드라이버를 사용하여 RDS for MariaDB에 연결

- Amazon Web Services(AWS) Python 드라이버를 사용하여 RDS for MariaDB에 연결

Amazon Web Services(AWS) JDBC 드라이버를 사용하여 RDS for MariaDB에 연결

Amazon Web Services(AWS) JDBC 드라이버는 고급 JDBC 래퍼로 설계되었습니다. 이 래퍼는 기존 JDBC 드라이버의 기능을 보완하고 확장합니다. 이 드라이버는 커뮤니티 MySQL Connector/J 드라이버 및 커뮤니티 MariaDB Connector/J 드라이버와 드롭인 호환됩니다.

AWS JDBC 드라이버를 설치하려면 `CLASSPATH` 애플리케이션에 있는 AWS JDBC 드라이버 .jar 파일을 추가하고 해당 커뮤니티 드라이버에 대한 참조를 보관해 두세요. 다음과 같이 해당 연결 URL 접두사를 업데이트하세요.

- `jdbc:mysql://~jdbc:aws-wrapper:mysql://`
- `jdbc:mariadb://~jdbc:aws-wrapper:mariadb://`

AWS JDBC 드라이버에 대한 자세한 내용 및 사용 방법에 대한 전체 지침은 Amazon Web Services (AWS) JDBC Driver GitHub repository를 참조하세요.

Amazon Web Services(AWS) Python 드라이버를 사용하여 RDS for MariaDB에 연결

Amazon Web Services(AWS) Python 드라이버는 고급 Python 래퍼로 설계되었습니다. 이 래퍼는 오픈 소스 Psycopg 드라이버의 기능을 보완하고 확장합니다. AWS Python 드라이버는 Python 버전 3.8 이상을 지원합니다. `pip` 명령을 사용하여 `psycopg` 오픈 소스 패키지와 함께 `aws-advanced-python-wrapper` 패키지를 설치할 수 있습니다.

AWS Python 드라이버에 대한 자세한 내용 및 사용 방법에 대한 전체 지침은 Amazon Web Services (AWS) JDBC Python GitHub repository를 참조하세요.

MariaDB DB 인스턴스에 대한 연결 문제 해결

새로운 DB 인스턴스의 연결 오류가 발생하는 두 가지 공통 원인은 다음과 같습니다.

- 보안 그룹을 사용하여 DB 인스턴스를 생성하였지만 이 보안 그룹이 MariaDB 애플리케이션 또는 유틸리티를 실행 중인 디바이스나 Amazon EC2 인스턴스에서 연결을 승인하지 않은 경우. DB 인스턴스에 연결 권한을 부여하는 VPC 보안 그룹이 있어야 합니다. 자세한 내용은 Amazon VPC 및 Amazon RDS 단원을 참조하십시오.

  보안 그룹에서 인바운드 규칙을 추가하거나 편집할 수 있습니다. 소스에서 내 IP를 선택합니다. 이렇게 하면 브라우저에서 감지된 IP 주소에서 DB 인스턴스에 액세스할 수 있습니다.

- 포트 3306을 사용해 DB 인스턴스를 만들었는데 기업 방화벽 규칙에 따라 기업 네트워크의 디바이스에서 해당 포트에 연결하는 것이 차단된 경우. 이 오류를 수정하려면 인스턴스를 다른 포트로 다시 만들어야 합니다.

연결 문제에 대한 자세한 내용은 Amazon RDS DB 인스턴스에 연결할 수 없음 단원을 참조하십시오.

MariaDB DB 인스턴스 연결 보안

MariaDB DB 인스턴스의 보안을 관리할 수 있습니다.

주제

- Amazon RDS MariaDB 보안
- RDS for MariaDB용 암호 확인 플러그인 사용
- SSL/TLS를 사용하여 Amazon RDS의 MariaDB DB 인스턴스에 대한 클라이언트 연결 암호화
- 새 SSL/TLS 인증서를 사용해 MariaDB 인스턴스에 연결할 애플리케이션 업데이트

Amazon RDS MariaDB 보안

MariaDB DB 인스턴스용 보안은 세 가지 수준에서 관리됩니다.

- AWS Identity and Access Management는 DB 인스턴스에서 Amazon RDS 관리 작업을 수행할 수 있는 사용자를 제어합니다. IAM 자격 증명을 사용하여 AWS에 연결할 때, IAM 계정은 Amazon RDS 관리 작업을 수행하는 데 필요한 권한을 부여하는 IAM 정책을 보유하고 있어야 합니다. 자세한 내용은 Amazon RDS의 자격 증명 및 액세스 관리 단원을 참조하십시오.
- DB 인스턴스를 생성할 때 VPC 보안 그룹을 사용하여 어떤 디바이스와 Amazon EC2 인스턴스가 DB 인스턴스의 엔드포인트 및 포트에 대한 연결을 열 수 있는지를 제어하게 됩니다. 보안 소켓 계층(SSL) 및 전송 계층 보안(TLS)을 사용하여 이러한 연결을 구성할 수 있습니다. 또한 회사의 방화벽 규칙을 통해 회사에서 실행하는 디바이스가 DB 인스턴스에 대한 연결을 열 수 있는지를 제어할 수 있습니다.
- 일단 MariaDB DB 인스턴스에 대한 연결이 활성화되면, 로그인 및 권한에 대한 인증은 MariaDB의 독립 실행형 인스턴스에서와 동일한 방식으로 적용됩니다. `CREATE USER`, `RENAME USER`, `GRANT`, `REVOKE` 및 `SET PASSWORD` 등의 명령은 독립 실행형 데이터베이스에서 작동하는 것과 마찬가지로 작동하며, 데이터베이스 스키마 테이블을 직접 수정할 때도 동일합니다.

Amazon RDS DB 인스턴스를 생성할 때 마스터 사용자는 다음과 같은 기본 권한을 갖습니다.

- `alter`
- `alter routine`
- `create`
- `create routine`
- `create temporary tables`

- create user
- create view
- delete
- drop
- event
- execute
- grant option
- index
- insert
- lock tables
- process
- references
- reload

  MariaDB DB 인스턴스에서 이 권한은 제한됩니다. FLUSH LOGS 또는 FLUSH TABLES WITH READ LOCK 작업에는 액세스할 수 없습니다.
- replication client
- replication slave
- select
- show create routine

  이 권한은 버전 11.4 이상을 실행하는 MariaDB DB 인스턴스에만 적용됩니다.
- show databases
- show view
- trigger
- update

권한에 대한 자세한 내용은 MariaDB 설명서에서 User Account Management를 참조하세요.

> **Note**
>
> DB 인스턴스에서 마스터 사용자를 삭제할 수는 있지만, 마스터 사용자를 삭제하지 않을 것을 권장합니다. 마스터 사용자를 다시 생성하려면 ModifyDBInstance API 또는 modify-db-

instance AWS CLI를 사용하고 적절한 파라미터와 함께 새 마스터 사용자 암호를 지정합니다. 마스터 사용자가 인스턴스에 존재하지 않는 경우 마스터 사용자가 지정된 암호와 함께 생성됩니다.

각 DB 인스턴스에 관리 서비스를 제공하기 위해 DB 인스턴스가 생성될 때 rdsadmin 사용자가 만들어집니다. rdsadmin 계정을 삭제하려고 하거나, 계정 이름 또는 암호를 변경하려고 하거나, 계정 권한을 변경하려고 하면 오류가 발생합니다.

DB 인스턴스의 관리를 허용하기 위해 표준 kill 및 kill_query 명령이 제한되었습니다. MariaDB 및 MySQL에서는 DB 인스턴스에서 사용자 세션 또는 쿼리를 종료할 수 있도록 Amazon RDS 명령 mysql.rds_kill, mysql.rds_kill_query 및 mysql.rds_kill_query_id가 제공됩니다.

RDS for MariaDB용 암호 확인 플러그인 사용

RDS for MariaDB 버전 11.4부터 다음 암호 확인 플러그인을 사용하여 데이터베이스 연결의 보안을 강화할 수 있습니다.

- simple_password_check – 암호에 특정 유형의 특정 개수 이상의 문자가 포함되어 있는지 확인합니다.
- cracklib_password_check – 암호가 CrackLib 라이브러리의 사전 파일에 나타나는지 확인합니다.

이러한 플러그인을 활성화하려면 DB 인스턴스와 연결된 DB 파라미터 그룹에서 파라미터 simple_password_check 또는 cracklib_password_check의 값을 FORCE_PLUS_PERMANENT로 설정합니다. 이 값을 설정하면 런타임 시 UNINSTALL PLUGIN 문을 사용하여 플러그인을 제거할 수 없습니다.

이러한 플러그인을 비활성화하려면 DB 인스턴스와 연결된 DB 파라미터 그룹에서 파라미터 simple_password_check 또는 cracklib_password_check의 값을 OFF로 설정합니다. 이 값을 설정하면 플러그인 검증 규칙이 더 이상 새 암호에 적용되지 않습니다.

파라미터 그룹에서 파라미터 값 설정에 대한 자세한 정보는 the section called “DB 파라미터 그룹의 파라미터 수정” 섹션을 참조하세요.

SSL/TLS를 사용하여 Amazon RDS의 MariaDB DB 인스턴스에 대한 클라이언트 연결 암호화

SSL(Secure Sockets Layer)은 클라이언트와 서버 간의 네트워크 연결을 보호하는 데 사용되는 업계 표준 프로토콜입니다. SSL 버전 3.0 이후로 이름이 전송 계층 보안(TLS)으로 변경되었습니다.

Amazon RDS는 MariaDB DB 인스턴스에 SSL/TLS 암호화를 지원합니다. SSL/TLS를 사용하여 애플리케이션 클라이언트와 MariaDB DB 인스턴스 간의 연결을 암호화할 수 있습니다. SSL/TLS 지원 기능은 모든 AWS 리전에서 사용할 수 있습니다.

Amazon RDS를 사용하면 SSL/TLS로 MariaDB DB 인스턴스에 대한 클라이언트 연결을 암호화하고, MariaDB DB 인스턴스에 대한 모든 연결에 SSL/TLS를 요구하고, MySQL 명령줄 클라이언트에서 SSL/TLS(암호화됨)로 연결하여 전송 중 데이터를 보호할 수 있습니다. 다음 섹션에서는 Amazon RDS의 MariaDB DB 인스턴스에 대한 SSL/TLS 암호화를 구성하고 활용하는 방법에 대한 지침을 제공합니다.

주제

- Amazon RDS의 MariaDB DB 인스턴스에 대한 SSL/TLS 지원
- Amazon RDS의 MariaDB DB 인스턴스에 대한 특정 사용자 계정에 SSL/TLS 요구
- Amazon RDS의 MariaDB DB 인스턴스에 대한 모든 연결에 SSL/TLS 요구
- MySQL 명령줄 클라이언트(암호화됨)에서 SSL/TLS를 사용하여 Amazon RDS의 MariaDB DB 인스턴스에 연결

Amazon RDS의 MariaDB DB 인스턴스에 대한 SSL/TLS 지원

Amazon RDS가 SSL/TLS 인증서를 생성한 후 Amazon RDS가 인스턴스를 프로비저닝할 때 DB 인스턴스에 인증서를 설치합니다. 인증 기관이 서명하는 SSL 인증서에는 SSL/TLS 인증서에는 스푸핑 공격으로부터 보호해주는 SSL/TLS 인증서를 위한 일반 이름(CN)으로 DB 인스턴스 엔드포인트가 포함되어 있습니다.

Amazon RDS에서 생성하는 SSL/TLS 인증서는 신뢰할 수 있는 루트 개체이므로 대부분의 경우에 작동하지만, 애플리케이션에서 인증서 체인을 수락하지 않을 경우 사용하지 못할 수 있습니다. 애플리케이션에서 인증서 체인을 허용하지 않는 경우 중간 인증서로 사용 중인 AWS 리전에 연결해 보세요. 예를 들어, SSL/TLS를 사용하여 AWS GovCloud (US) 리전에 연결하려면 중간 인증서를 사용해야 합니다.

인증서 다운로드에 대한 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 단원을 참조하세요. MySQL에서 SSL/TLS를 사용하는 방법에 대한 자세한 내용은 새 SSL/TLS 인증서를 사용해 MariaDB 인스턴스에 연결할 애플리케이션 업데이트 섹션을 참조하세요.

Amazon RDS for MariaDB는 전송 계층 보안(TLS) 버전 1.3, 1.2, 1.1 및 1,0을 지원합니다. TLS 지원은 MariaDB 마이너 버전에 따라 달라집니다. 다음 테이블은 MariaDB 마이너 버전에 대한 TLS 지원을 보여줍니다.

TLS 버전	MariaDB 11.4	MariaDB 10.11	MariaDB 10.6	MariaDB 10.5	MariaDB 10.4
TLS 1.3	모든 마이너 버전	모든 마이너 버전	모든 마이너 버전	모든 마이너 버전	모든 마이너 버전
TLS 1.2	모든 마이너 버전	모든 마이너 버전	모든 마이너 버전	모든 마이너 버전	모든 마이너 버전
TLS 1.1	지원되지 않음	10.11.6 이하	10.6.16 이하	10.5.23 이하	10.4.32 이하
TLS 1.0	지원되지 않음	10.11.6 이하	10.6.16 이하	10.5.23 이하	10.4.32 이하

Amazon RDS의 MariaDB DB 인스턴스에 대한 특정 사용자 계정에 SSL/TLS 요구

Amazon RDS의 MariaDB DB 인스턴스에 대한 지정된 사용자 계정 연결을 위해 SSL/TLS 암호화가 필요할 수 있습니다. 데이터 기밀성이 우려되는 보안 정책을 적용하려면 민감한 정보를 무단 액세스 또는 가로채기로부터 보호하는 것이 중요합니다.

특정 사용자의 계정에 대해 SSL/TLS 연결을 요구하려면 MySQL 버전에 따라 다음 문 중 하나를 사용하여 사용자 계정 `encrypted_user`에서 SSL/TLS 연결을 요구하세요.

이렇게 하려면 다음 문을 사용하면 됩니다.

```
ALTER USER 'encrypted_user'@'%' REQUIRE SSL;
```

MariaDB와의 SSL/TLS 연결에 대한 자세한 내용은 MariaDB 설명서에서 클라이언트 및 서버에 대한 연결 보안을 참조하세요.

Amazon RDS의 MariaDB DB 인스턴스에 대한 모든 연결에 SSL/TLS 요구

`require_secure_transport` 파라미터를 사용하여 MariaDB DB 인스턴스에 대한 모든 사용자 연결이 SSL/TLS를 사용하도록 요구합니다. 기본적으로 `require_secure_transport` 파라미터는 `OFF`로 설정됩니다. `require_secure_transport` 파라미터를 `ON`으로 설정하면 해당 DB 인스턴스에 대한 연결 시 SSL/TLS를 요구합니다.

> **Note**
>
> `require_secure_transport` 파라미터는 MariaDB 버전 10.5 이상에서만 지원됩니다.

`require_secure_transport` 파라미터 값은 DB 인스턴스의 DB 파라미터 그룹을 업데이트하여 설정할 수 있습니다. 변경 사항을 적용하기 위해 DB 인스턴스를 재부팅할 필요가 없습니다.

DB 클러스터에 대해 `require_secure_transport` 파라미터를 `ON`으로 설정하면 암호화된 연결을 설정할 수 있는 경우 데이터베이스 클라이언트가 인스턴스에 연결할 수 있습니다. 그렇지 않으면 다음과 유사한 오류 메시지가 클라이언트에 반환됩니다.

```
ERROR 1045 (28000): Access denied for user 'USER'@'localhost' (using password: YES |
 NO)
```

파라미터 설정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정을 참조하세요.

`require_secure_transport` 파라미터에 대한 자세한 내용은 MariaDB 설명서를 참조하세요.

MySQL 명령줄 클라이언트(암호화됨)에서 SSL/TLS를 사용하여 Amazon RDS의 MariaDB DB 인스턴스에 연결

`mysql` 클라이언트 프로그램 파라미터는 MySQL 5.7 버전, MySQL 8.0 버전 또는 MariaDB 버전을 사용하는 경우 약간 다릅니다.

사용 중인 버전을 확인하려면 `--version` 옵션을 사용하여 `mysql` 명령을 실행합니다. 다음 예에서는 출력은 클라이언트 프로그램이 MariaDB의 프로그램임을 나타냅니다.

```
$ mysql --version
mysql  Ver 15.1 Distrib 10.5.15-MariaDB, for osx10.15 (x86_64) using readline 5.1
```

Amazon Linux, CentOS, SUSE 및 Debian과 같은 대부분의 Linux 배포판은 MySQL을 MariaDB로 대체했으며 `mysql` 버전은 MariaDB에서 가져온 것입니다.

다음 단계에 따라 SSL/TLS를 사용하여 DB 인스턴스에 연결합니다.

MySQL 명령줄 클라이언트를 사용하여 SSL/TLS를 통해 DB 인스턴스에 연결하려면

1. 모든 AWS 리전에 적용되는 루트 인증서를 다운로드할 수 있습니다.

인증서 다운로드에 대한 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 단원을 참조하세요.

2. MySQL 명령줄 클라이언트를 사용하여 SSL/TLS를 통해 DB 인스턴스에 연결합니다. -h 파라미터의 경우 해당 DB 인스턴스의 DNS 이름(엔드포인트)로 대체합니다. --ssl-ca 파라미터는 해당하는 SSL/TLS 인증서 파일 이름으로 대체합니다. -P 파라미터에는 DB 인스턴스의 포트로 대체합니다. -u 파라미터에는 마스터 사용자와 같이 유효한 데이터베이스 사용자의 사용자 이름으로 대체합니다. 입력 프롬프트가 표시되면 마스터 사용자 암호를 입력합니다.

   다음 예제는 MariaDB 클라이언트를 사용하여 --ssl-ca 파라미터를 통해 클라이언트를 시작하는 방법을 보여줍니다.

   ```
   mysql -h mysql-instance1.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=global-bundle.pem --ssl -P 3306 -u myadmin -p
   ```

   SSL/TLS 연결에서 SSL/TLS 인증서의 엔드포인트와 비교하여 DB 인스턴스 엔드포인트를 확인하도록 요구할 수 있습니다.

   ```
   mysql -h mysql-instance1.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=global-bundle.pem --ssl-verify-server-cert -P 3306 -u myadmin -p
   ```

   다음 예제는 MySQL 5.7 클라이언트 이후 버전의 경우 --ssl-ca 파라미터를 사용하여 클라이언트를 시작하는 방법을 보여줍니다.

   ```
   mysql -h mysql-instance1.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=global-bundle.pem --ssl-mode=REQUIRED -P 3306 -u myadmin -p
   ```

3. 입력 프롬프트가 표시되면 마스터 사용자 암호를 입력합니다.

다음과 유사한 출력 화면이 표시되어야 합니다.

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 10.6.10-MariaDB-log Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
MariaDB [(none)]>
```

새 SSL/TLS 인증서를 사용해 MariaDB 인스턴스에 연결할 애플리케이션 업데이트

2023년 1월 13일부터 Amazon RDS는 보안 소켓 계층(SSL) 또는 전송 계층 보안(TLS)을 사용해 RDS DB 인스턴스에 연결하기 위한 용도의 새 인증 기관(CA) 인증서를 게시하였습니다. 아래에서 새 인증서를 사용하기 위해 애플리케이션을 업데이트하는 방법에 관한 정보를 찾으실 수 있습니다.

이 주제는 애플리케이션이 DB 인스턴스에 연결하기 위해 인증서 확인이 필요한지 판단하는 데 도움이 됩니다.

> **Note**
>
> 어떤 애플리케이션은 서버에서 인증서를 성공적으로 확인할 수 있는 경우에만 MariaDB에 연결하도록 구성되어 있습니다. 이러한 애플리케이션의 경우 클라이언트 애플리케이션 트러스트 스토어를 업데이트하여 새 CA 인증서를 포함해야 합니다.
>
> `disabled`, `preferred` 및 `required`의 SSL 모드를 지정할 수 있습니다. `preferred` SSL 모드를 사용할 때 CA 인증서가 없거나 최신 상태가 아닌 경우 연결이 SSL을 사용하지 않는 것으로 폴백되고 여전히 성공적으로 연결됩니다.
>
> `preferred` 모드는 피하는 것이 좋습니다. `preferred` 모드에서 연결이 잘못된 인증서가 발견되면 모드에서 암호화 사용을 중지하고 암호화되지 않은 상태로 진행합니다.

클라이언트 애플리케이션 트러스트 스토어에서 CA 인증서를 업데이트한 후에는 DB 인스턴스에서 인증서를 교환할 수 있습니다. 이 절차를 프로덕션 환경에서 구현하기 전에 개발 또는 스테이징 환경에서 테스트해볼 것을 적극 권장합니다.

인증서 교환에 대한 자세한 내용은 SSL/TLS 인증서 교체 단원을 참조하십시오. 인증서 다운로드에 대한 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 단원을 참조하십시오. MariaDB DB 인스턴스에서 SSL/TLS를 사용하는 방법에 관한 자세한 내용은 Amazon RDS의 MariaDB DB 인스턴스에 대한 SSL/TLS 지원 단원을 참조하십시오.

주제

- 클라이언트에서 연결을 위해 인증서 확인이 필요한지 여부 확인
- 애플리케이션 트러스트 스토어 업데이트
- SSL 연결 설정을 위한 Java 코드 예시

클라이언트에서 연결을 위해 인증서 확인이 필요한지 여부 확인

JDBC 클라이언트 및 MySQL 클라이언트에서 연결 시 인증서 확인이 필요한지 여부를 확인할 수 있습니다.

JDBC

아래의 MySQL 커넥터/J 8.0 관련 예시에서는 애플리케이션의 JDBC 연결 속성을 확인하여 성공적인 연결을 위해 유효한 인증서가 필요한지 여부를 판단하는 한 가지 방법을 보여줍니다. MySQL에 대한 모든 JDBC 연결 옵션에 대한 자세한 내용은 MySQL 문서의 구성 속성을 참조하십시오.

MySQL 커넥터/J 8.0을 사용 중인 경우 다음 예시와 같이 연결 속성에서 `sslMode`가 `VERIFY_CA` 또는 `VERIFY_IDENTITY`로 설정되어 있다면 SSL 연결 시 서버 CA 인증서에 대한 확인이 필요합니다.

```
Properties properties = new Properties();
properties.setProperty("sslMode", "VERIFY_IDENTITY");
properties.put("user", DB_USER);
properties.put("password", DB_PASSWORD);
```

> **Note**
>
> 데이터베이스에 연결할 때 SSL/TLS를 사용하도록 애플리케이션을 명시적으로 구성하지 않았더라도 MySQL Java Connector v5.1.38 이상 또는 MySQL Java Connector v8.0.9 이상을 사용하여 데이터베이스에 연결하는 경우, 이러한 클라이언트 드라이버는 기본적으로 SSL/TLS를 사용합니다. 또한 SSL/TLS 사용 시 부분 인증서 확인을 수행하고 데이터베이스 서버 인증서가 만료되면 연결에 실패합니다.
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

MySQL

MySQL 클라이언트에 관한 다음 예시에서는 스크립트의 MySQL 연결을 확인하여 성공적인 연결을 위해 유효한 인증서가 필요한지 여부를 판단하는 두 가지 방법을 보여줍니다. MySQL 클라이언트의 모든 연결 옵션에 대한 자세한 내용은 MySQL 문서의 암호화된 연결을 위한 클라이언트 측 구성을 참조하십시오.

MySQL 5.7 또는 MySQL 8.0 클라이언트를 사용 중인 경우 다음 예시와 같이 `--ssl-mode` 옵션에 대해 `VERIFY_CA` 또는 `VERIFY_IDENTITY`을 지정했다면 SSL 연결 시 서버 CA 인증서에 대한 확인이 필요합니다.

```
mysql -h mysql-database.rds.amazonaws.com -uadmin -ppassword --ssl-ca=/tmp/ssl-cert.pem
 --ssl-mode=VERIFY_CA
```

MySQL 5.6 클라이언트를 사용 중인 경우 다음 예시와 같이 `--ssl-verify-server-cert` 옵션을 지정했다면 SSL 연결 시 서버 CA 인증서에 대한 확인이 필요합니다.

```
mysql -h mysql-database.rds.amazonaws.com -uadmin -ppassword --ssl-ca=/tmp/ssl-cert.pem
 --ssl-verify-server-cert
```

애플리케이션 트러스트 스토어 업데이트

MySQL 애플리케이션의 트러스트 스토어 업데이트에 대한 정보는 MariaDB 문서에서 MariaDB Connector/J와 함께 TLS/SSL 사용을 참조하십시오.

루트 인증서 다운로드에 대한 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 단원을 참조하십시오.

인증서를 가져오는 샘플 스크립트는 트러스트 스토어로 인증서를 가져오기 위한 샘플 스크립트 섹션을 참조하세요.

> **Note**
>
> 트러스트 스토어를 업데이트할 때 새 인증서를 추가할 뿐 아니라 이전 인증서를 유지할 수도 있습니다.

애플리케이션에서 MariaDB Connector/J JDBC 드라이버를 사용 중인 경우 애플리케이션에서 다음 속성을 설정하십시오.

```
System.setProperty("javax.net.ssl.trustStore", certs);
System.setProperty("javax.net.ssl.trustStorePassword", "password");
```

애플리케이션을 시작할 때 다음 속성을 설정하십시오.

```
java -Djavax.net.ssl.trustStore=/path_to_truststore/MyTruststore.jks -
Djavax.net.ssl.trustStorePassword=my_truststore_password com.companyName.MyApplication
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

SSL 연결 설정을 위한 Java 코드 예시

다음은 JDBC를 사용하여 SSL 연결을 설정하는 방법을 보여 주는 코드 예입니다.

```
private static final String DB_USER = "admin";

        private static final String DB_USER = "user name";
        private static final String DB_PASSWORD = "password";
        // This key store has only the prod root ca.
        private static final String KEY_STORE_FILE_PATH = "file-path-to-keystore";
        private static final String KEY_STORE_PASS = "keystore-password";

    public static void main(String[] args) throws Exception {
        Class.forName("org.mariadb.jdbc.Driver");

        System.setProperty("javax.net.ssl.trustStore", KEY_STORE_FILE_PATH);
        System.setProperty("javax.net.ssl.trustStorePassword", KEY_STORE_PASS);

        Properties properties = new Properties();
        properties.put("user", DB_USER);
        properties.put("password", DB_PASSWORD);


        Connection connection = DriverManager.getConnection("jdbc:mysql://ssl-mariadb-
public.cni62e2e7kwh.us-east-1.rds.amazonaws.com:3306?useSSL=true",properties);
        Statement stmt=connection.createStatement();
```

```
        ResultSet rs=stmt.executeQuery("SELECT 1 from dual");

        return;
    }
```

Important

데이터베이스 연결에서 SSL/TLS를 사용함을 확인하고 애플리케이션 트러스트 스토어를 업데이트한 후에는 데이터베이스에서 rds-ca-rsa2048-g1 인증서를 사용하도록 업데이트할 수 있습니다. 지침은 DB 인스턴스 또는 클러스터를 수정하여 CA 인증서 업데이트의 3단계를 참조하십시오.
보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

Amazon RDS Optimized Reads로 RDS for MariaDB 쿼리 성능 개선

Amazon RDS Optimized Reads로 RDS for MariaDB의 쿼리 처리 속도를 높일 수 있습니다. RDS Optimized Reads를 사용하는 RDS for MariaDB DB 인스턴스는 이를 사용하지 않는 DB 인스턴스에 비해 쿼리 처리 속도가 최대 2배 더 빠릅니다.

주제

- RDS Optimized Reads 개요
- RDS Optimized Reads 사용 사례
- RDS Optimized Reads 모범 사례
- RDS Optimized Reads 사용
- RDS Optimized Reads를 사용하는 DB 인스턴스 모니터링
- RDS Optimized Reads 제한 사항

RDS Optimized Reads 개요

RDS Optimized Reads가 켜져 있는 RDS for MariaDB DB 인스턴스를 사용하면 DB 인스턴스가 인스턴스 스토어를 사용하여 더 빠른 쿼리 성능을 얻을 수 있습니다. 인스턴스 스토어는 DB 인스턴스에 블록 수준의 임시 스토리지를 제공합니다. 스토리지는 호스트 서버에 물리적으로 연결된 NVMe(Non-Volatile Memory Express) 솔리드 스테이트 드라이브(SSD)에 있습니다. 이 스토리지는 짧은 지연 시간, 높은 임의 I/O 성능, 높은 순차 읽기 처리량(throughput)에 최적화되어 있습니다.

RDS Optimized Reads는 DB 인스턴스가 db.m5d 또는 db.m6gd 같은 DB 인스턴스 클래스를 인스턴스 스토어와 함께 사용할 때 기본적으로 켜집니다. RDS Optimized Reads를 사용하면 일부 임시 객체가 인스턴스 스토어에 저장됩니다. 이러한 임시 객체에는 내부 임시 파일, 내부 온디스크 임시 테이블, 메모리 맵 파일, 이진 로그(binlog) 캐시 파일이 포함됩니다. 인스턴스 스토어 유형에 대한 자세한 내용은 Linux 인스턴스용 Amazon Elastic Compute Cloud 사용 설명서의 Amazon EC2 인스턴스 스토어 섹션을 참조하세요.

쿼리 처리를 위해 MariaDB에서 임시 객체를 생성하는 워크로드는 인스턴스 스토어를 활용하여 쿼리를 더 빠르게 처리할 수 있습니다. 이러한 유형의 워크로드에는 정렬, 해시 집계, 고부하 조인, CTE(Common Table Expression) 및 인덱싱되지 않은 열에 대한 쿼리가 포함됩니다. 이러한 인스턴스 스토어 볼륨은 영구 Amazon EBS 스토리지에 사용되는 스토리지 구성과 관계없이 더 높은 IOPS와 성능을 제공합니다. RDS Optimized Reads는 임시 객체에 대한 작업을 인스턴스 스토어로 오프로드하므로 이제 영구 객체에 대한 작업에 영구 스토리지(Amazon EBS)의 초당 입출력 작업 처리량(IOPS) 또는

처리량(throughput)을 사용할 수 있습니다. 이러한 작업에는 일반 데이터 파일 읽기 및 쓰기와 플러싱 및 버퍼 병합 삽입과 같은 백그라운드 엔진 작업이 포함됩니다.

> **Note**
>
> 수동 및 자동 RDS 스냅샷에는 모두 영구 객체를 위한 엔진 파일만 포함됩니다. 인스턴스 스토어에서 생성된 임시 객체는 RDS 스냅샷에 포함되지 않습니다.

RDS Optimized Reads 사용 사례

쿼리 실행을 위해 내부 테이블이나 파일 같은 임시 객체에 크게 의존하는 워크로드가 있는 경우 RDS Optimized Reads를 켜는 것이 좋습니다. RDS Optimized Reads의 사용 사례는 다음과 같습니다.

- 복잡한 CTE(Common Table Expression), 파생된 테이블, 그룹화 작업을 사용하여 분석 쿼리를 실행하는 애플리케이션
- 최적화되지 않은 쿼리로 많은 읽기 트래픽을 처리하는 읽기 전용 복제본
- `GROUP BY` 및 `ORDER BY` 절이 있는 쿼리와 같이 복잡한 작업이 필요한 온디맨드 또는 동적 보고 쿼리를 실행하는 애플리케이션
- 쿼리 처리를 위해 내부 임시 테이블을 사용하는 워크로드

  엔진 상태 변수 `created_tmp_disk_tables`를 모니터링하여 DB 인스턴스에 생성된 디스크 기반 임시 테이블의 수를 확인할 수 있습니다.
- 직접 또는 절차에 따라 대형 임시 테이블을 생성하여 중간 결과를 저장하는 애플리케이션
- 인덱싱되지 않은 열에서 그룹화 또는 정렬 작업을 수행하는 데이터베이스 쿼리

RDS Optimized Reads 모범 사례

RDS Optimized Reads에 대한 다음 모범 사례를 따릅니다.

- 실행 중 인스턴스 스토어가 가득 차서 쿼리가 실패하는 경우에 대비하여 읽기 전용 쿼리의 재시도 로직을 추가합니다.
- CloudWatch 지표 `FreeLocalStorage`를 사용하여 인스턴스 스토어에서 사용 가능한 스토리지 공간을 모니터링합니다. DB 인스턴스의 워크로드로 인해 인스턴스 스토어가 한도에 도달하면 더 큰 DB 인스턴스 클래스를 사용하도록 DB 인스턴스를 수정하세요.

- DB 인스턴스의 메모리가 충분하지만 여전히 인스턴스 스토어의 스토리지 한도에 도달하면 `binlog_cache_size` 값을 늘려 세션별 binlog 항목을 메모리에 유지합니다. 이 구성은 binlog 항목을 디스크의 임시 binlog 캐시 파일에 쓰는 것을 방지합니다.

  `binlog_cache_size` 파라미터는 세션별로 다릅니다. 새 세션마다 값을 변경할 수 있습니다. 이 파라미터를 설정하면 피크 워크로드 중에 DB 인스턴스의 메모리 사용률을 높일 수 있습니다. 따라서 애플리케이션의 워크로드 패턴과 DB 인스턴스의 가용 메모리를 기반으로 파라미터 값을 높이는 것을 고려해 보세요.
- `binlog_format`에 기본값 `MIXED`를 사용합니다. 트랜잭션 크기에 따라 `binlog_format`을 `ROW`로 설정하면 인스턴스 스토어에 큰 binlog 캐시 파일이 생성될 수 있습니다.
- 단일 트랜잭션에서 대량 변경을 수행하지 마세요. 이러한 유형의 트랜잭션은 인스턴스 스토어에 대용량 binlog 캐시 파일을 생성할 수 있으며, 인스턴스 스토어가 가득 차면 문제가 발생할 수 있습니다. binlog 캐시 파일의 스토리지 사용을 최소화하려면 쓰기를 여러 개의 작은 트랜잭션으로 분할하는 것이 좋습니다.

RDS Optimized Reads 사용

단일 AZ DB 인스턴스 배포 또는 다중 AZ DB 인스턴스 배포에서 다음 DB 인스턴스 클래스 중 하나를 사용하여 RDS for MariaDB DB 인스턴스를 프로비저닝하면 DB 인스턴스는 자동으로 RDS Optimized Reads를 사용합니다.

RDS Optimized Read를 켜려면 다음 중 하나를 수행합니다.

- 이러한 DB 인스턴스 클래스 중 하나를 사용하여 RDS for MariaDB DB 인스턴스를 생성합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 단원을 참조하십시오.
- 이러한 DB 인스턴스 클래스 중 하나를 사용하여 기존 RDS for MariaDB DB 인스턴스를 수정합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

RDS Optimized Reads는 로컬 NVMe SSD 스토리지가 있는 DB 인스턴스 클래스 중 하나 이상이 지원되는 모든 AWS 리전에서 사용 가능합니다. DB 인스턴스 클래스에 대한 자세한 내용은 the section called “DB 인스턴스 클래스” 섹션을 참조하세요.

DB 인스턴스 클래스 가용성은 AWS 리전에 따라 다릅니다. 특정 AWS 리전 클래스에서 DB 인스턴스 클래스가 지원되는지 여부를 확인하려면 the section called “AWS 리전에서 DB 인스턴스 클래스 지원 확인”를 참조하세요.

RDS Optimized Reads를 사용하지 않으려면 기능을 지원하는 DB 인스턴스 클래스를 사용하지 않도록 DB 인스턴스를 수정하세요.

RDS Optimized Reads를 사용하는 DB 인스턴스 모니터링

다음 CloudWatch 지표를 사용하여 RDS Optimized Reads를 사용하는 DB 인스턴스를 모니터링할 수 있습니다.

- `FreeLocalStorage`
- `ReadIOPSLocalStorage`
- `ReadLatencyLocalStorage`
- `ReadThroughputLocalStorage`
- `WriteIOPSLocalStorage`
- `WriteLatencyLocalStorage`
- `WriteThroughputLocalStorage`

이러한 지표는 사용 가능한 인스턴스 스토어 스토리지, IOPS 및 처리량(throughput)에 대한 데이터를 제공합니다. 이러한 지표에 대한 자세한 내용은 Amazon RDS에 대한 Amazon CloudWatch 지표 섹션을 참조하세요.

RDS Optimized Reads 제한 사항

RDS Optimized Reads에는 다음과 같은 제한 사항이 적용됩니다.

- RDS Optimized Reads는 다음 RDS for MariaDB 버전에서 지원됩니다.
  - 10.11.4 이상의 10.11 버전
  - 10.6.7 이상의 10.6 버전
  - 10.5.16 이상의 10.5 버전
  - 10.4.25 이상의 10.4 버전

  RDS for MariaDB 버전에 대한 자세한 내용은 Amazon RDS MariaDB 버전 섹션을 참조하세요.
- RDS Optimized Reads를 지원하는 DB 인스턴스 클래스에서는 임시 객체의 위치를 영구 스토리지(Amazon EBS)로 변경할 수 없습니다.
- DB 인스턴스에서 이진 로깅이 활성화된 경우 최대 트랜잭션 크기는 인스턴스 스토어의 크기에 의해 제한됩니다. MariaDB에서 `binlog_cache_size` 값보다 더 많은 스토리지가 필요한 세션은 인스턴스 스토어에 생성된 임시 binlog 캐시 파일에 트랜잭션 변경 사항을 기록합니다.

- 인스턴스 스토어가 가득 차면 트랜잭션이 실패할 수 있습니다.

Amazon RDS Optimized Writes for MariaDB를 통한 쓰기 성능 개선

RDS Optimized Writes for MariaDB를 사용하여 쓰기 트랜잭션의 성능을 개선할 수 있습니다. RDS for MariaDB 데이터베이스에서 RDS Optimized Writes를 사용하는 경우 쓰기 트랜잭션 처리량을 최대 2배까지 높일 수 있습니다.

주제

- RDS Optimized Writes 개요
- RDS Optimized Writes 사용
- 기존 데이터베이스에서 RDS 최적화된 쓰기 활성화
- RDS Optimized Writes 제한 사항

RDS Optimized Writes 개요

RDS Optimized Writes를 켜면 이중 쓰기 버퍼를 사용할 필요 없이 내구성이 뛰어난 스토리지로 데이터를 플러싱할 때 RDS for MariaDB 데이터베이스가 데이터를 한 번만 씁니다. 데이터베이스는 향상된 성능과 함께 안정적인 데이터베이스 트랜잭션을 위한 ACID 속성 보호를 지속적으로 제공합니다.

MariaDB 같은 관계형 데이터베이스는 신뢰할 수 있는 데이터베이스 트랜잭션을 위한 ACID(원자성, 일관성, 격리, 내구성) 속성을 제공합니다. 이러한 속성을 제공하기 위해 MariaDB는 이중 쓰기 버퍼라는 데이터 스토리지 영역을 사용하여 부분 페이지 쓰기 오류를 방지합니다. 이러한 오류는 정전의 경우처럼 데이터베이스에서 페이지를 업데이트하는 동안 하드웨어 장애가 있을 때 발생합니다. MariaDB 데이터베이스는 부분 페이지 쓰기를 감지하고 이중 쓰기 버퍼에 있는 페이지 복사본으로 복구할 수 있습니다. 이 기술은 보호 기능을 제공하지만 추가 쓰기 작업도 초래합니다. MariaDB 이중 쓰기 버퍼에 대한 자세한 내용은 MariaDB 설명서의 InnoDB 이중 쓰기 버퍼를 참조하세요.

RDS Optimized Writes를 켜면 이중 쓰기 버퍼를 사용하지 않고 내구성이 뛰어난 스토리지로 데이터를 플러싱할 때 RDS for MariaDB 데이터베이스가 데이터를 한 번만 씁니다. RDS Optimized Writes는 RDS for MariaDB 데이터베이스에서 쓰기가 많은 워크로드를 실행하는 경우에 유용합니다. 쓰기 작업이 많은 데이터베이스의 예로는 디지털 결제, 금융 거래, 게임 애플리케이션을 지원하는 데이터베이스가 있습니다.

이러한 데이터베이스는 AWS Nitro System을 사용하는 DB 인스턴스 클래스에서 실행됩니다. 이러한 시스템의 하드웨어 구성으로 인해 데이터베이스는 한 단계로 안정적이고 내구성 있게 16KiB 페이지를 데이터 파일에 직접 쓸 수 있습니다. AWS Nitro 시스템은 RDS Optimized Writes를 가능하게 합니다.

새 데이터베이스 매개 변수 `rds.optimized_writes`를 설정하여 RDS for MariaDB 데이터베이스의 RDS Optimized Writes 기능을 제어할 수 있습니다. RDS for MariaDB의 DB 파라미터 그룹에서 이 파라미터에 액세스할 수 있는 버전은 다음과 같습니다.

- 11.4.3 이상의 11.4 버전
- 10.11.4 이상의 10.11 버전
- 10.6.10 이상의 10.6 버전

다음 값을 사용하여 파라미터를 설정합니다.

- `AUTO` - 데이터베이스에서 지원하는 경우 RDS Optimized Writes를 켭니다. 데이터베이스에서 지원하지 않는 경우 RDS Optimized Writes를 해제합니다. 이 설정이 기본값입니다.
- `OFF` - 데이터베이스에서 지원하더라도 RDS Optimized Writes를 해제합니다.

RDS Optimized Writes를 사용하도록 구성된 RDS for MariaDB 데이터베이스를 해당 기능을 지원하지 않는 DB 인스턴스 클래스로 마이그레이션하는 경우, RDS는 데이터베이스의 RDS Optimized Writes를 자동으로 해제합니다.

RDS Optimized Writes가 해제된 경우 데이터베이스는 MariaDB 이중 쓰기 버퍼를 사용합니다.

RDS for MariaDB 데이터베이스에서 RDS Optimized Writes를 사용하고 있는지 확인하려면 데이터베이스 `innodb_doublewrite` 파라미터의 현재 값을 확인하세요. 데이터베이스에서 RDS Optimized Writes를 사용 중인 경우 이 파라미터는 `FALSE(0)`로 설정됩니다.

RDS Optimized Writes 사용

RDS 콘솔, AWS CLI 또는 RDS API를 사용하여 RDS for MariaDB 데이터베이스를 생성할 때 RDS Optimized Writes를 켤 수 있습니다. 데이터베이스 생성 중에 다음 두 가지 조건이 모두 적용되는 경우 RDS Optimized Writes가 자동으로 켜집니다.

- RDS Optimized Writes를 지원하는 DB 엔진 버전과 DB 인스턴스 클래스를 지정합니다.
  - RDS Optimized Writes는 다음 RDS for MariaDB 버전에서 지원됩니다.
    - 11.4.3 이상의 11.4 버전
    - 10.11.4 이상의 10.11 버전
    - 10.6.10 이상의 10.6 버전

    RDS for MariaDB 버전에 대한 자세한 내용은 Amazon RDS MariaDB 버전 섹션을 참조하세요.

- RDS Optimized Writes는 다음 DB 인스턴스 클래스를 사용하는 RDS for MariaDB 데이터베이스에서 지원됩니다.
  - db.m7i
  - db.m7g
  - db.m6g
  - db.m6gd
  - db.m6i
  - db.m5
  - db.m5d
  - db.r7i
  - db.r7g
  - db.r6g
  - db.r6gd
  - db.r6i
  - db.r5
  - db.r5b
  - db.r5d
  - db.x2idn
  - db.x2iedn

  DB 인스턴스 클래스에 대한 자세한 내용은 the section called “DB 인스턴스 클래스” 섹션을 참조하세요.

  DB 인스턴스 클래스 가용성은 AWS 리전에 따라 다릅니다. 특정 AWS 리전 클래스에서 DB 인스턴스 클래스가 지원되는지 여부를 확인하려면 the section called “AWS 리전에서 DB 인스턴스 클래스 지원 확인”를 참조하세요.
- 데이터베이스에 연결된 파라미터 그룹에서는 `rds.optimized_writes` 파라미터가 `AUTO`로 설정됩니다. 기본 파라미터 그룹에서는 이 파라미터가 항상 `AUTO`로 설정됩니다.

RDS Optimized Writes를 지원하는 DB 엔진 버전과 DB 인스턴스 클래스를 사용하고 싶지만 이 기능은 사용하고 싶지 않을 경우, 데이터베이스를 생성할 때 사용자 지정 파라미터 그룹을 지정하세요. 이 파라미터 그룹에서 `rds.optimized_writes` 파라미터를 `OFF`로 설정합니다. 나중에 데이터베이스에서 RDS Optimized Writes를 사용하도록 하려면 파라미터를 `AUTO`로 설정하여 켤 수 있습니다. 사용자

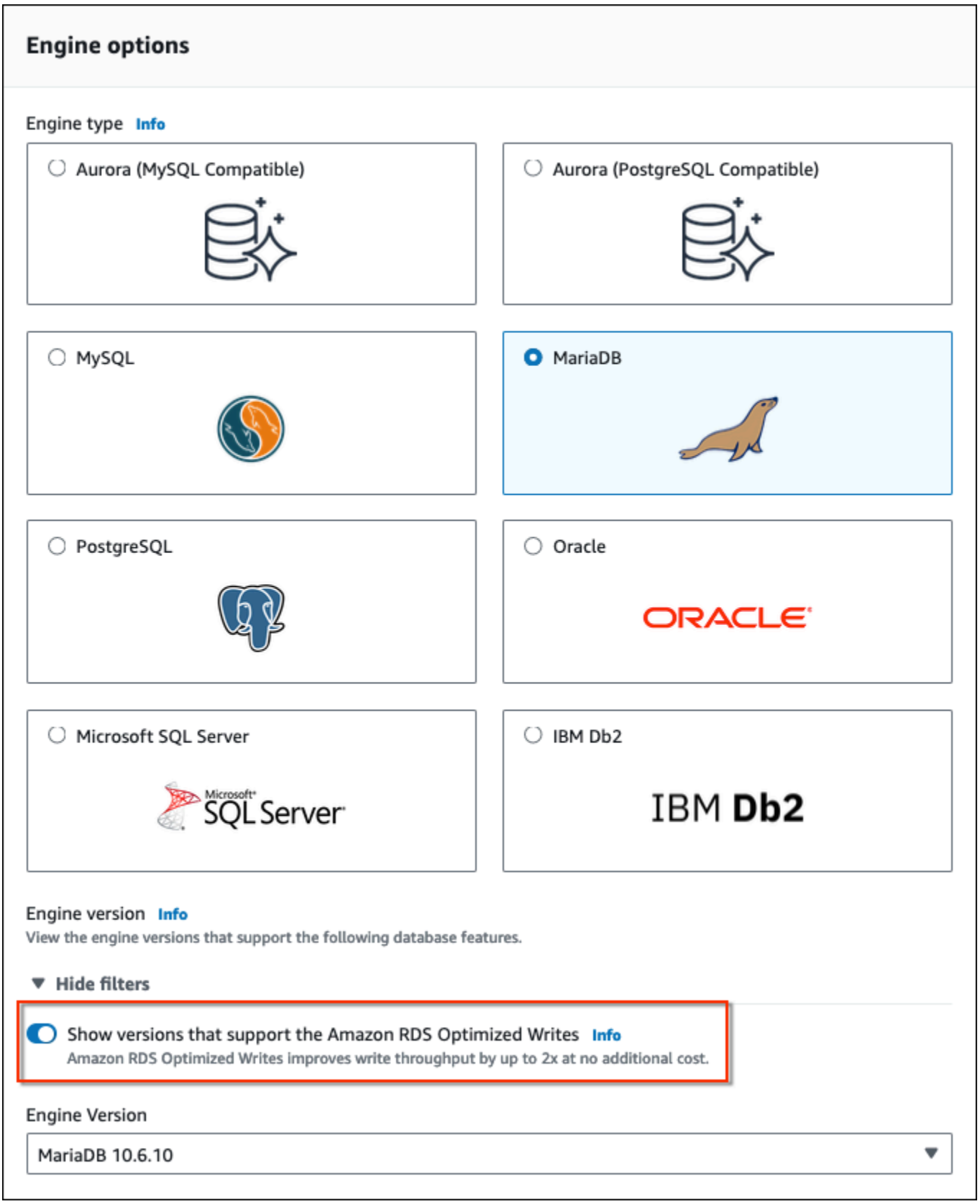
지정 파라미터 그룹 및 파라미터 설정에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 섹션을 참조하세요.

DB 인스턴스 생성에 대한 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.

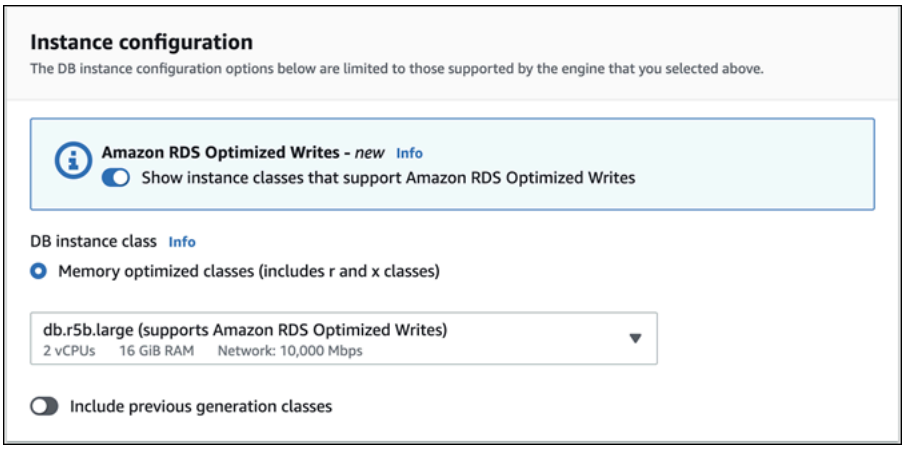
콘솔

RDS 콘솔을 사용하여 RDS for MariaDB 데이터베이스를 생성할 때 RDS Optimized Writes를 지원하는 DB 엔진 버전 및 DB 인스턴스 클래스를 필터링할 수 있습니다. 필터를 켜면 사용 가능한 DB 엔진 버전 및 DB 인스턴스 클래스 중에서 선택할 수 있습니다.

RDS Optimized Writes를 지원하는 DB 엔진 버전을 선택하려면 엔진 버전에서 이를 지원하는 RDS for MariaDB DB 엔진 버전을 필터링한 다음 버전을 선택합니다.


Engine options
Engine type Info
Aurora (MySQL Compatible)
Aurora (PostgreSQL Compatible)
MySQL
MariaDB
PostgreSQL
Oracle
ORACLE
Microsoft SQL Server
Microsoft SQL Server
IBM Db2
IBM Db2
Engine version Info
View the engine versions that support the following database features.
Hide filters
Show versions that support the Amazon RDS Optimized Writes Info
Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.
Engine Version
MariaDB 10.6.10

인스턴스 구성 섹션에서 RDS Optimized Writes를 지원하는 DB 인스턴스 클래스를 필터링한 다음 DB 인스턴스 클래스를 선택합니다.



이러한 선택을 완료한 후 요구 사항에 맞는 다른 설정을 선택하고 콘솔을 사용하여 RDS for MariaDB 데이터베이스 생성을 완료할 수 있습니다.

AWS CLI

AWS CLI를 사용하여 DB 인스턴스를 만들려면 create-db-instance 명령을 실행합니다. `--engine-version` 및 `--db-instance-class` 값이 RDS Optimized Writes를 지원하는지 확인하세요. 또한 DB 인스턴스에 연결된 파라미터 그룹에서 `rds.optimized_writes` 파라미터가 `AUTO`로 설정되어 있는지 확인합니다. 다음 예제에서는 기본 파라미터 그룹을 DB 인스턴스와 연결합니다.

Example RDS Optimized Writes를 사용하는 DB 인스턴스 생성

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --engine mariadb \
    --engine-version 10.6.10 \
    --db-instance-class db.r5b.large \
    --manage-master-user-password \
    --master-username admin \
    --allocated-storage 200
```

Windows의 경우:

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

```
    --engine mariadb ^
    --engine-version 10.6.10 ^
    --db-instance-class db.r5b.large ^
    --manage-master-user-password ^
    --master-username admin ^
    --allocated-storage 200
```

RDS API

CreateDBInstance 작업을 사용하여 DB 인스턴스를 생성할 수 있습니다. 이 작업을 사용할 경우 `EngineVersion` 및 `DBInstanceClass` 값이 RDS Optimized Writes를 지원하는지 확인하세요. 또한 DB 인스턴스에 연결된 파라미터 그룹에서 `rds.optimized_writes` 파라미터가 `AUTO`로 설정되어 있는지 확인합니다.

기존 데이터베이스에서 RDS 최적화된 쓰기 활성화

RDS 최적화된 쓰기를 켜도록 기존 RDS for MariaDB 데이터베이스를 수정하려면 지원되는 DB 엔진 버전 및 DB 인스턴스 클래스로 데이터베이스가 생성된 상태여야 합니다. 또한 RDS 최적화된 쓰기가 출시된 2023년 3월 7일 이후에 생성한 데이터베이스여야 합니다. 필요한 기본 파일 시스템 구성이 릴리스 전에 생성된 데이터베이스의 구성과 호환되지 않기 때문입니다. 이러한 조건이 충족되면 `rds.optimized_writes` 파라미터를 `AUTO`로 설정하여 RDS 최적화된 쓰기를 활성화할 수 있습니다.

지원되는 엔진 버전, 인스턴스 클래스 또는 파일 시스템 구성으로 데이터베이스를 생성하지 않은 경우 RDS 블루/그린 배포를 사용하여 지원되는 구성으로 마이그레이션할 수 있습니다. 블루/그린 배포를 생성하는 동안 다음을 수행하세요.

- 그린 데이터베이스에서 Optimized Writes 활성화를 선택하고 RDS 최적화된 쓰기를 지원하는 엔진 버전과 DB 인스턴스 클래스를 지정합니다. 지원되는 엔진 버전과 인스턴스 클래스의 목록은 the section called “새 데이터베이스와 함께 사용” 섹션을 참조하세요.
- 스토리지에서 스토리지 파일 시스템 구성 업그레이드를 선택합니다. 이 옵션은 데이터베이스를 호환 가능한 기본 파일 시스템 구성으로 업그레이드합니다.

블루/그린 배포를 생성하는 도중 `rds.optimized_writes` 파라미터가 `AUTO`로 설정된 경우 그린 환경에서 RDS 최적화된 쓰기가 자동으로 활성화됩니다. 그런 다음 블루/그린 배포를 전환하여 그린 환경을 새로운 프로덕션 환경으로 승격합니다.

자세한 내용은 the section called “블루/그린 배포 생성” 단원을 참조하십시오.

RDS Optimized Writes 제한 사항

스냅샷에서 RDS for MariaDB 데이터베이스를 복원할 때는 다음 조건이 모두 적용되는 경우에만 데이터베이스의 RDS 최적화된 쓰기를 켤 수 있습니다.

- RDS Optimized Writes를 지원하는 데이터베이스에서 스냅샷이 생성되었습니다.
- RDS Optimized Writes가 릴리스된 이후에 생성한 데이터베이스에서 스냅샷이 생성되었습니다.
- RDS Optimized Writes를 지원하는 데이터베이스로 스냅샷이 복원됩니다.
- 복원된 데이터베이스는 `rds.optimized_writes` 파라미터가 `AUTO`로 설정된 파라미터 그룹에 연결됩니다.

MariaDB DB 엔진 업그레이드

Amazon RDS에서 새 데이터베이스 엔진 버전을 지원하는 경우, DB 인스턴스를 새 버전으로 업그레이드할 수 있습니다. MariaDB DB 인스턴스의 업그레이드에는 메이저 버전 업그레이드와 마이너 버전 업그레이드라는 두 가지 업그레이드가 있습니다.

메이저 버전 업그레이드에는 기존 애플리케이션과 호환되지 않는 데이터베이스 변경 사항이 포함될 수 있습니다. 따라서 DB 인스턴스의 메이저 버전 업그레이드를 수동으로 수행해야 합니다. DB 인스턴스를 수정하여 메이저 버전 업그레이드를 시작할 수 있습니다. 그러나 메이저 버전 업그레이드를 수행하기 전에 RDS for MariaDB 메이저 버전 업그레이드의 지침을 따르는 것이 좋습니다.

반대로 마이너 버전 업그레이드에는 기존 애플리케이션과 호환되는 변경 사항만 포함됩니다. DB 인스턴스를 수정하여 마이너 버전 업그레이드를 수동으로 시작할 수 있습니다. 또는 DB 인스턴스를 생성하거나 수정할 때 마이너 버전 자동 업그레이드 옵션을 활성화할 수 있습니다. 이렇게 하면 Amazon RDS에서 새 버전을 테스트 및 승인한 후 DB 인스턴스가 자동으로 업그레이드됩니다. 업그레이드 수행에 대한 자세한 내용은 DB 인스턴스 엔진 버전 업그레이드 단원을 참조하십시오.

MariaDB DB 인스턴스가 읽기 전용 복제본을 사용하는 경우 소스 인스턴스를 업그레이드하기 전에 읽기 전용 복제본을 모두 업그레이드해야 합니다. DB 인스턴스를 다중 AZ 배포로 생성한 경우 라이터와 예비 복제본이 모두 업그레이드됩니다. 업그레이드가 완료될 때까지 DB 인스턴스를 사용할 수 없습니다.

MariaDB 지원 버전 및 버전 관리에 대한 자세한 내용은 Amazon RDS MariaDB 버전 단원을 참조하십시오.

데이터베이스 엔진 업그레이드에는 다운타임이 필요합니다. 다운타임 시간은 DB 인스턴스의 크기에 따라 다릅니다.

> **Tip**
>
> 블루/그린 배포를 사용하면 DB 인스턴스 업그레이드에 필요한 다운타임을 최소화할 수 있습니다. 자세한 내용은 데이터베이스 업데이트에 Amazon RDS 블루/그린 배포 사용 단원을 참조하십시오.

주제

- MariaDB 업그레이드 고려 사항
- 유효한 업그레이드 대상 찾기
- MariaDB 버전 번호

- RDS for MariaDB의 RDS 버전 번호
- RDS for MariaDB 메이저 버전 업그레이드
- MariaDB DB 인스턴스 업그레이드
- RDS for MariaDB 마이너 버전 자동 업그레이드
- RDS for MariaDB 데이터베이스 업그레이드 시 읽기 전용 복제본을 사용하여 가동 중지 시간 단축

MariaDB 업그레이드 고려 사항

Amazon RDS는 업그레이드 프로세스 중에 DB 스냅샷을 2개 이상 캡처합니다. Amazon RDS는 업그레이드를 변경하기 전에 DB 인스턴스의 스냅샷을 최대 2개까지 캡처합니다. 업그레이드가 데이터베이스에 맞지 않는 경우에는 이 스냅샷을 복구하여 이전 버전을 실행하는 DB 인스턴스를 생성할 수 있습니다. Amazon RDS는 업그레이드가 완료되면 DB 인스턴스의 또 다른 스냅샷을 캡처합니다. Amazon RDS는 AWS Backup에서 DB 인스턴스의 백업을 관리하는지 여부에 관계없이 이러한 스냅샷을 생성합니다.

> **Note**
>
> DB 인스턴스에 대한 백업 보존 기간을 0보다 큰 수로 설정하면 Amazon RDS는 DB 스냅샷만 캡처합니다. 백업 보존 기간을 변경하려면 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

업그레이드가 완료되면 이전 버전의 데이터베이스 엔진으로 되돌릴 수 없습니다. 이때 이전 버전으로 되돌리려면 첫 번째로 캡처한 DB 스냅샷을 복구하여 새로운 DB 인스턴스를 생성해야 합니다.

DB 인스턴스를 Amazon RDS가 지원하는 새 버전으로 업그레이드하는 시기는 사용자가 직접 관리합니다. 이러한 관리 수준은 특정 데이터베이스 버전과 호환성을 유지하거나 프로덕션 환경에 배포하기 전에 애플리케이션을 이용해 새 버전을 테스트하는 데 효과적입니다. 모든 준비를 마치면 일정에 가장 적합한 시기에 버전 업그레이드를 실행할 수 있습니다.

DB 인스턴스가 읽기 전용 복제본을 사용하는 경우 원본 인스턴스를 업그레이드하기 전에 읽기 전용 복제본부터 모두 업그레이드해야 합니다.

DB 인스턴스를 다중 AZ 배포로 생성한 경우에는 기본 DB 인스턴스와 예비 DB 인스턴스가 모두 업그레이드됩니다. 기본 DB 인스턴스와 예비 DB 인스턴스가 모두 동시에 업그레이드되므로 업그레이드가 끝날 때까지 작동 중단을 겪게 됩니다. 중단 시간은 데이터베이스 엔진, 엔진 버전 및 DB 인스턴스의 크기에 따라 달라집니다.

유효한 업그레이드 대상 찾기

AWS Management Console을 사용하여 DB 인스턴스를 업그레이드하면 DB 인스턴스의 유효한 업그레이드 대상이 표시됩니다. 다음 AWS CLI 명령을 실행하여 DB 인스턴스의 유효한 업그레이드 대상을 식별할 수도 있습니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-engine-versions \
  --engine mariadb \
  --engine-version version_number \
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

Windows의 경우:

```
aws rds describe-db-engine-versions ^
  --engine mariadb ^
  --engine-version version_number ^
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

예를 들어 MariaDB 버전 10.5.17 DB 인스턴스의 유효한 업그레이드 대상을 식별하려면 다음 AWS CLI 명령을 실행합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-engine-versions \
  --engine mariadb \
  --engine-version 10.5.17 \
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

Windows의 경우:

```
aws rds describe-db-engine-versions ^
  --engine mariadb ^
  --engine-version 10.5.17 ^
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

MariaDB 버전 번호

MariaDB용 RDS 데이터베이스 엔진의 버전 번호 지정 시퀀스는 major.minor.patch.YYYYMMDD 또는 major.minor.patch 형식(예: 10.11.5.R2.20231201 또는 10.4.30)입니다. 최대 크기는 MariaDB 엔진 버전에 따라 다릅니다.

메이저

버전 번호의 정수 부분과 첫 번째 소수 부분 모두가 메이저 버전 번호입니다(예: 10.11). 메이저 버전 업그레이드는 버전 번호의 메이저 부분이 증가합니다. 예를 들어 10.5.20에서 10.6.12로 업그레이드하는 것은 메이저 버전 업그레이드입니다. 여기서 10.5과 10.6이 메이저 버전 번호입니다.

마이너

마이너 버전 번호는 버전 번호의 세 번째 부분입니다(예: 10.11.5의 5).

패치

패치는 버전 번호의 네 번째 부분입니다(예: 10.11.5.R2의 R2). RDS 패치 버전에는 릴리스 후 마이너 버전에 추가된 중요한 버그 수정이 포함되어 있습니다.

YYYYMMDD(날짜)

날짜는 버전 번호의 다섯 번째 부분입니다(예: 10.11.5.R2.20231201의 20231201). RDS 날짜 버전은 보안 패치로 릴리스 후 마이너 버전에 추가된 중요한 보안 수정이 포함되어 있습니다. 엔진 동작을 변경할 수 있는 수정 사항은 포함되지 않습니다.

다음 표에서는 RDS for MariaDB 버전 10.11의 이름 지정 체계를 설명합니다.

10.11 마이너 버전	이름 지정 체계
≥5	새 DB 인스턴스는 major.minor.patch.YYMMDD를 사용합니다(예: 10.11.5.R2.20231201). 기존 DB 인스턴스는 다음 메이저 또는 마이너 버전 업그레이드 전까지 major.minor.patch(예: 10.11.5.R2)를 사용할 수 있습니다.
5 미만	기존 DB 인스턴스는 major.minor.patch를 사용합니다(예: 10.11.4.R2).

다음 표에서는 RDS for MariaDB 버전 10.6의 이름 지정 체계를 설명합니다.

10.6 마이너 버전	이름 지정 체계
14 이상	새 DB 인스턴스는 major.minor.patch.YYMMDD를 사용합니다(예: 10.6.14.R2.20231201). 기존 DB 인스턴스는 다음 메이저 또는 마이너 버전 업그레이드 전까지 major.minor.patch(예: 10.6.14.R2)를 사용할 수 있습니다.
14 미만	기존 DB 인스턴스는 major.minor.patch를 사용합니다(예: 10.6.13.R2).

다음 표에서는 RDS for MariaDB 버전 10.5의 이름 지정 체계를 설명합니다.

10.5 마이너 버전	이름 지정 체계
21 이상	새 DB 인스턴스는 major.minor.patch.YYMMDD를 사용합니다(예: 10.5.21.R2.20231201). 기존 DB 인스턴스는 다음 메이저 또는 마이너 버전 업그레이드 전까지 major.minor.patch(예: 10.5.21.R2)를 사용할 수 있습니다.
21 미만	기존 DB 인스턴스는 major.minor.patch를 사용합니다(예: 10.5.20.R2).

다음 표에서는 RDS for MariaDB 버전 10.4의 이름 지정 체계를 설명합니다.

10.4 마이너 버전	이름 지정 체계
30 이상	새 DB 인스턴스는 major.minor.patch.YYMMDD를 사용합니다(예: 10.4.30.R2.20231201). 기존 DB 인스턴스는 다음 메이저 또는 마이너 버전 업그레이드 전까지 major.minor.patch(예: 10.4.30.R2)를 사용할 수 있습니다.
30 미만	기존 DB 인스턴스는 major.minor.patch를 사용합니다(예: 10.4.29.R2).

RDS for MariaDB의 RDS 버전 번호

RDS 버전 번호는 *major*.*minor*.*patch* 또는 *major*.*minor*.*patch*.*YYYYMMDD* 명명 체계를 사용합니다. RDS 패치 버전에는 릴리스 후 마이너 버전에 추가된 중요한 버그 수정이 포함되어 있습니다. RDS 날짜 버전(*YYMMDD*)은 보안 패치입니다. 보안 패치에는 엔진 동작을 변경할 수 있는 수정 사항은 포함되지 않습니다.

데이터베이스의 Amazon RDS 버전 번호를 식별하려면 먼저 다음 명령을 사용하여 `rds_tools` 확장을 생성해야 합니다.

```
CREATE EXTENSION rds_tools;
```

다음 SQL 쿼리를 사용하여 RDS for MariaDB 데이터베이스의 RDS 버전 번호를 확인할 수 있습니다.

```
mysql> select mysql.rds_version();
```

예를 들어 RDS for MariaDB 10.6.14 데이터베이스를 쿼리하면 다음 아웃풋이 반환됩니다.

```
+---------------------+
| mysql.rds_version() |
+---------------------+
| 10.6.14.R2.20231201  |
+---------------------+
1 row in set (0.01 sec)
```

RDS for MariaDB 메이저 버전 업그레이드

메이저 버전 업그레이드에는 기존 애플리케이션과 호환되지 않는 데이터베이스 변경 사항이 포함될 수 있습니다. 따라서 Amazon RDS는 자동으로 메이저 버전 업그레이드를 적용하지 않습니다. DB 인스턴스를 수동으로 수정해야 합니다. 모든 업그레이드는 프로덕션 환경의 인스턴스에 적용하기 전에 반드시 철저하게 테스트하는 것이 좋습니다.

Amazon RDS는 MariaDB 데이터베이스 엔진의 다음과 같은 메이저 버전 업그레이드를 지원합니다.

- MariaDB 11.4에서 모든 MariaDB 버전
- MariaDB 10.11에서 모든 MariaDB 버전
- MariaDB 10.6에서 모든 MariaDB 버전
- MariaDB 10.4에서 MariaDB 10.5로

사용자 지정 파라미터 그룹을 사용 중이고 메이전 버전 업그레이드를 수행하는 경우 새 DB 엔진 버전에 대한 기본 파라미터 그룹을 지정하거나 새 DB 엔진 버전에 대한 사용자 지정 파라미터 그룹을 만들어야 합니다. 새 파라미터 그룹을 DB 인스턴스에 연결하려면 업그레이드가 완료된 후 고객이 데이터베이스 재부팅을 시작해야 합니다. 파라미터 그룹 변경 사항을 적용하기 위해 인스턴스를 재부팅해야 하는 경우, 인스턴스의 파라미터 그룹 상태가 `pending-reboot`로 표시됩니다. 인스턴스의 파라미터 그룹 상태는 AWS Management Console에서 확인하거나 'describe' 직접 호출(예: `describe-db-instances`)을 실행하여 확인할 수 있습니다.

MariaDB DB 인스턴스 업그레이드

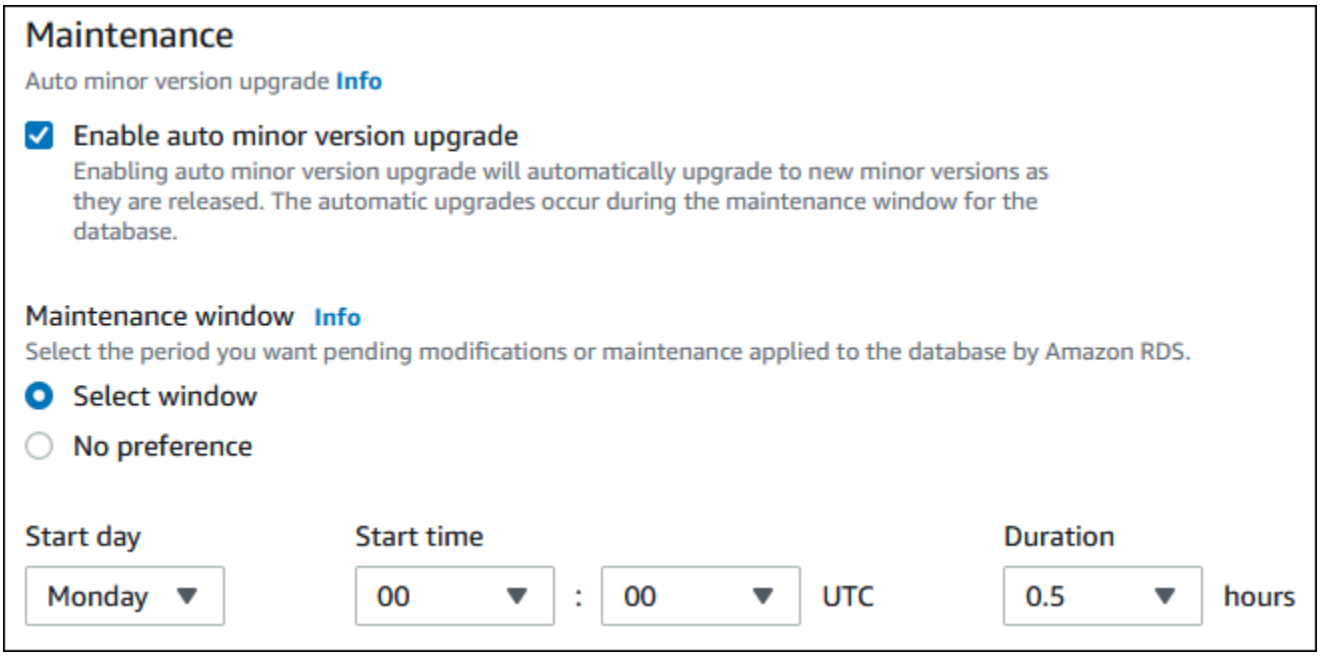
MariaDB DB 인스턴스의 수동 또는 자동 업그레이드에 대한 자세한 내용은 DB 인스턴스 엔진 버전 업그레이드 단원을 참조하십시오.

RDS for MariaDB 마이너 버전 자동 업그레이드

DB 인스턴스를 생성하거나 수정할 때 다음 설정을 지정하면 DB 인스턴스가 자동으로 업그레이드되도록 할 수 있습니다.

- 마이너 버전 자동 업그레이드(Auto minor version upgrade) 설정을 활성화되어 있습니다.
- 백업 보존 기간(Backup retention period) 설정이 0보다 큽니다.

AWS Management Console에서 이 설정은 추가 구성(Additional configuration)에 있습니다. 다음 이미지는 자동 마이너 버전 업그레이드(Auto minor version upgrade) 설정을 보여줍니다.



이러한 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하세요.

일부 AWS 리전에 있는 특정 RDS for MariaDB 메이저 버전의 경우 RDS에서 하나의 마이너 버전을 자동 업그레이드 버전으로 지정합니다. Amazon RDS가 마이너 버전을 테스트하고 승인하면 유지 관리 기간 중에 자동으로 마이너 버전 업그레이드가 실행됩니다. RDS는 자동으로 새로 릴리스된 마이너 버전을 자동 업그레이드 버전으로 설정하지 않습니다. RDS가 더 새로운 자동 업그레이드 버전을 지정하기 전에 다음과 같은 여러 기준이 고려됩니다.

- 알려진 보안 문제
- MariaDB 커뮤니티 버전의 버그
- 마이너 버전 릴리스 이후 전반적인 플릿 안정성

> **Note**
>
> TLS 버전 1.0 및 1.1 사용에 대한 지원은 MariaDB의 특정 마이너 버전부터 제거되었습니다. 지원되는 MariaDB 마이너 버전에 대한 자세한 내용은 the section called “MariaDB에 대한 SSL/TLS 지원” 섹션을 참조하세요.

다음 AWS CLI 명령을 실행하여 특정 AWS 리전의 지정된 MariaDB 마이너 버전에 대한 현재의 자동 마이너 업그레이드 대상 버전을 확인할 수 있습니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-engine-versions \
--engine mariadb \
--engine-version minor_version \
--region region \
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" \
--output text
```

Windows의 경우:

```
aws rds describe-db-engine-versions ^
--engine mariadb ^
--engine-version minor_version ^
--region region ^
```

```
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" ^
--output text
```

예를 들어, 다음 AWS CLI 명령은 미국 동부(오하이오) AWS 리전(us-east-2)의 MariaDB 마이너 버전 10.5.16에 대한 자동 마이너 업그레이드 대상을 결정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-engine-versions \
--engine mariadb \
--engine-version 10.5.16 \
--region us-east-2 \
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" \
--output table
```

Windows의 경우:

```
aws rds describe-db-engine-versions ^
--engine mariadb ^
--engine-version 10.5.16 ^
--region us-east-2 ^
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" ^
--output table
```

다음과 같은 출력이 표시됩니다.

```
---------------------------------
|    DescribeDBEngineVersions    |
+--------------+-----------------+
|  AutoUpgrade |  EngineVersion  |
+--------------+-----------------+
|  True        |  10.5.17        |
|  False       |  10.5.18        |
|  False       |  10.5.19        |
|  False       |  10.6.5         |
|  False       |  10.6.7         |
|  False       |  10.6.8         |
|  False       |  10.6.10        |
|  False       |  10.6.11        |
```

```
|  False       |  10.6.12        |
+--------------+-----------------+
```

이 예제에서 AutoUpgrade 값은 MariaDB 버전 10.5.17의 경우 True입니다. 따라서 자동 마이너 업그레이드 대상은 출력에서 강조 표시된 MariaDB 버전 10.5.17입니다.

MariaDB DB 인스턴스는 다음 기준이 충족되면 유지 관리 기간 중에 자동으로 업그레이드됩니다.

- 마이너 버전 자동 업그레이드(Auto minor version upgrade) 설정을 활성화되어 있습니다.
- 백업 보존 기간(Backup retention period) 설정이 0보다 큽니다.
- DB 인스턴스가 현재 자동 업그레이드 마이너 버전보다 낮은 DB 엔진 버전을 실행 중입니다.

자세한 내용은 마이너 엔진 버전 자동 업그레이드 섹션을 참조하세요.

RDS for MariaDB 데이터베이스 업그레이드 시 읽기 전용 복제본을 사용하여 가동 중지 시간 단축

대부분의 경우 블루/그린 배포는 MariaDB DB 인스턴스를 업그레이드할 때 다운타임을 줄이는 가장 좋은 방법입니다. 자세한 내용은 데이터베이스 업데이트에 Amazon RDS 블루/그린 배포 사용 단원을 참조하십시오.

블루/그린 배포를 사용할 수 없으며 현재 프로덕션 애플리케이션에 MariaDB DB 인스턴스를 사용 중인 경우, 다음 절차를 사용하여 DB 인스턴스의 데이터베이스 버전을 업그레이드할 수 있습니다. 이 절차는 애플리케이션의 가동 중지 시간을 줄일 수 있습니다.

읽기 전용 복제본을 사용하면 대부분의 유지 관리 단계를 미리 수행하고 실제 운영 중단 중에 필요한 변경 사항을 최소화할 수 있습니다. 이 기법을 사용하면 기존 DB 인스턴스를 변경하지 않으면서 새 DB 인스턴스를 테스트하고 준비할 수 있습니다.

이 절차는 MariaDB 버전 10.5를 MariaDB 버전 10.6으로 업그레이드하는 예를 보여줍니다. 동일한 일반 절차를 사용하여 다른 메이저 버전으로 업그레이드할 수 있습니다.

DB 인스턴스를 사용하면서 MariaDB 데이터베이스를 업그레이드하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. MariaDB 10.5 DB 인스턴스의 읽기 전용 복제본을 생성합니다. 이 프로세스에서 업그레이드 가능한 데이터베이스 사본이 만들어집니다. DB 인스턴스의 다른 읽기 전용 복제본도 존재할 수 있습니다.

a. 콘솔에서 데이터베이스(Databases)와 업그레이드하려는 DB 인스턴스를 차례로 선택합니다.

b. 작업에서 읽기 전용 복제본 만들기를 선택합니다.

c. 읽기 전용 복제본의 DB 인스턴스 식별자 값을 제공하고 DB instance class(DB 인스턴스 클래스) 및 기타 설정이 MariaDB 10.5 DB 인스턴스와 일치하는지 확인합니다.

d. [Create read replica]를 선택합니다.

3. (선택 사항) 읽기 전용 복제본이 생성되고 상태(Status)가 사용 가능(Available)으로 표시되면 읽기 전용 복제본을 다중 AZ 배포로 변환하고 백업을 활성화합니다.

   기본적으로 읽기 전용 복제본은 백업이 비활성화된 단일 AZ 배포로 생성됩니다. 읽기 전용 복제본은 궁극적으로 프로덕션 DB 인스턴스가 되기 때문에 지금 다중 AZ 배포를 구성하고 백업을 활성화하는 것이 가장 좋습니다.

   a. 콘솔에서 데이터베이스(Databases)와 방금 생성한 읽기 전용 복제본을 차례로 선택합니다.

   b. 수정을 선택합니다.

   c. 다중 AZ 배포(Multi-AZ deployment)에서 대기 인스턴스 생성(Create a standby instance)을 선택합니다.

   d. Backup Retention Period(백업 보존 기간)로 0이 아닌 양수 값(예: 3일)을 선택한 후 Continue(계속)를 선택합니다.

   e. 수정 사항 예약에 대해 즉시 적용을 선택합니다.

   f. DB 인스턴스 수정을 선택합니다.

4. 읽기 전용 복제본 Status(상태)가 Available(사용 가능)로 표시되면 읽기 전용 복제본을 MariaDB 10.6으로 업그레이드합니다.

   a. 콘솔에서 데이터베이스(Databases)와 방금 생성한 읽기 전용 복제본을 차례로 선택합니다.

   b. 수정을 선택합니다.

   c. DB engine version(DB 엔진 버전)에서 업그레이드할 MariaDB 10.6 버전을 선택한 후 Continue(계속)를 선택합니다.

   d. 수정 사항 예약에 대해 즉시 적용을 선택합니다.

   e. [Modify DB instance]를 선택하여 업그레이드를 시작합니다.

5. 업그레이드가 완료되고 Status(상태)가 Available(사용 가능)로 표시되면 업그레이드한 읽기 전용 복제본이 소스 MariaDB 10.5 DB 인스턴스로 업데이트되는지 확인합니다. 확인하려면 읽기 전용 복제본에 연결하고 `SHOW REPLICA STATUS` 명령을 실행합니다. `Seconds_Behind_Master` 필드가 `0`이면 복제본이 최신 상태입니다.

> **Note**
>
> 이전 버전의 MariaDB에는 `SHOW SLAVE STATUS` 대신 `SHOW REPLICA STATUS`가 사용되었습니다. 10.6 이전 MariaDB 버전을 사용하는 경우 `SHOW SLAVE STATUS`를 사용합니다.

6. (선택 사항) 읽기 전용 복제본의 읽기 전용 복제본을 생성합니다.

   DB 인스턴스가 독립형 DB 인스턴스로 승격된 후 읽기 전용 복제본을 갖도록 하려면 지금 읽기 전용 복제본을 생성하면 됩니다.

   a. 콘솔에서 데이터베이스(Databases)와 방금 업그레이드한 읽기 전용 복제본을 차례로 선택합니다.

   b. 작업에서 읽기 전용 복제본 만들기를 선택합니다.

   c. 읽기 전용 복제본의 DB 인스턴스 식별자 값을 제공하고 DB instance class(DB 인스턴스 클래스) 및 기타 설정이 MariaDB 10.5 DB 인스턴스와 일치하는지 확인합니다.

   d. [Create read replica]를 선택합니다.

7. (선택 사항) 읽기 전용 복제본에 대한 사용자 지정 DB 파라미터 그룹을 구성합니다.

   DB 인스턴스가 독립형 DB 인스턴스로 승격된 후 사용자 지정 파라미터 그룹을 사용하도록 하려면 지금 DB 파라미터 그룹을 생성하여 읽기 전용 복제본과 연결하면 됩니다.

   a. MariaDB 10.6에 대한 사용자 지정 DB 파라미터 그룹을 생성합니다. 지침은 Amazon RDS에서 DB 파라미터 그룹 생성 섹션을 참조하세요.

   b. 방금 생성한 DB 파라미터 그룹에서 변경하려는 파라미터를 수정합니다. 지침은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요.

   c. 콘솔에서 데이터베이스(Databases)와 읽기 전용 복제본을 차례로 선택합니다.

   d. 수정을 선택합니다.

   e. DB parameter group(DB 파라미터 그룹)에서 방금 생성한 MariaDB 10.6 DB 파라미터 그룹을 선택한 후 Continue(계속)를 선택합니다.

   f. 수정 사항 예약에 대해 즉시 적용을 선택합니다.

   g. [Modify DB instance]를 선택하여 업그레이드를 시작합니다.

8. MariaDB 10.6 읽기 전용 복제본을 독립형 DB 인스턴스로 만듭니다.

> **Important**
>
> MariaDB 10.6 읽기 전용 복제본을 독립형 DB 인스턴스로 승격하면 더 이상 MariaDB 10.5 DB 인스턴스의 복제본이 아닙니다. 소스 MariaDB 10.5 DB 인스턴스가 읽기 전용 모드이고, 모든 쓰기 작업이 일시 중단되는 유지 관리 기간 동안 MariaDB 10.6 읽기 전용 복제본을 승격하는 것이 좋습니다. 승격이 완료되면 쓰기 연산을 MariaDB 10.6 DB 인스턴스에서 실행하여 쓰기 연산이 손실되는 것을 막을 수 있습니다.
>
> 그밖에도 MariaDB 10.6 읽기 전용 복제본을 승격하기 전에 MariaDB 10.6 읽기 전용 복제본에서 필요한 모든 DDL(데이터 정의 언어) 작업을 수행하는 것이 좋습니다. 인덱스 생성을 예로 들 수 있습니다. 그러면 승격 후에도 MariaDB 10.6 읽기 전용 복제본의 성능에 미치는 부정적인 영향을 방지할 수 있습니다. 읽기 전용 복제본을 승격하려면 다음 절차를 사용하십시오.

a. 콘솔에서 데이터베이스(Databases)와 방금 업그레이드한 읽기 전용 복제본을 차례로 선택합니다.

b. 작업에서 Promote(승격)를 선택합니다.

c. 읽기 전용 복제본 인스턴스에 대해 자동 백업을 활성화하려면 예를 선택합니다. 자세한 내용은 백업 소개 섹션을 참조하세요.

d. [Continue]를 선택합니다.

e. [Promote Read Replica]를 선택합니다.

9. 이제 MariaDB 데이터베이스 버전이 업그레이드되었습니다. 이제 애플리케이션을 새로운 MariaDB 10.6 DB 인스턴스로 리디렉션할 수 있습니다.

MariaDB DB 스냅샷 엔진 버전 업그레이드

Amazon RDS를 사용하여 MariaDB DB 인스턴스의 스토리지 볼륨 DB 스냅샷을 만들 수 있습니다. DB 스냅샷 생성 시, 생성되는 스냅샷은 DB 인스턴스에서 사용하는 엔진 버전에 기반합니다. DB 스냅샷의 엔진 버전을 업그레이드할 수 있습니다.

RDS for MariaDB의 경우 사용 가능한 모든 엔진 버전으로 업그레이드할 수 있습니다. 암호화되거나 암호화되지 않은 DB 스냅샷을 업그레이드할 수 있습니다.

RDS for MariaDB DB 스냅샷에 사용 가능한 엔진 버전을 보려면 다음 AWS CLI 예제를 사용합니다.

```
aws rds describe-db-engine-versions --engine mariadb --include-all --engine-
version example-engine-version --query "DBEngineVersions[*].ValidUpgradeTarget[*].
{EngineVersion:EngineVersion}" --output text
```

스냅샷에 대한 결과가 표시되지 않으면 엔진 버전이 더 이상 사용되지 않을 수 있습니다. 엔진 버전이 더 이상 지원되지 않는 경우 최신 메이저 버전 업그레이드 대상 또는 해당 버전에 대해 사용 가능한 다른 업그레이드 대상 중 하나로 업그레이드하는 것이 좋습니다. 자세한 내용은 RDS for MariaDB에서 지원되지 않는 엔진 버전이 있는 DB 스냅샷에 대한 업그레이드 옵션 섹션을 참조하세요.

새 엔진 버전으로 업그레이드된 DB 스냅샷을 복원한 후에는 업그레이드가 성공적이었는지 테스트해야 합니다. 메이저 버전 업그레이드에 대한 자세한 내용은 the section called “MariaDB DB 엔진 업그레이드” 단원을 참조하십시오. DB 스냅샷을 복원하는 방법은 the section called “DB 인스턴스 복원” 단원을 참조하십시오.

> **Note**
>
> 자동 백업 과정에서 생성된 자동 DB 스냅샷은 업그레이드할 수 없습니다.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB 스냅샷을 업그레이드할 수 있습니다.

Console

AWS Management Console을 사용하여 DB 스냅샷 엔진 버전을 업그레이드하려면 다음 절차를 사용합니다.

DB 스냅샷을 업그레이드하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.
3. 업그레이드할 스냅샷을 선택합니다.
4. 작업에서 Upgrade snapshot(스냅샷 업그레이드)을 선택합니다. Upgrade snapshot(스냅샷 업그레이드) 페이지가 표시됩니다.
5. 업그레이드할 New engine version(새 엔진 버전)을 선택합니다.
6. 스냅샷을 업그레이드하려면 변경 내용 저장을 선택합니다.

   업그레이드 중에는 이 DB 스냅샷의 모든 스냅샷 작업이 비활성화됩니다. 또한 DB 스냅샷 상태가 사용 가능에서 업그레이드 중으로 바뀐 다음 완료되면 활성으로 바뀝니다 스냅샷 손상 문제로 인해 DB 스냅샷을 업그레이드할 수 없는 경우, 상태가 사용할 수 없음으로 바뀝니다. 이 상태로부터 스냅샷을 복구할 수는 없습니다.

   > **Note**
   >
   > DB 스냅샷 업그레이드에 실패하면 스냅샷이 원래 버전의 원래 상태로 롤백됩니다.

AWS CLI

DB 스냅샷을 새 데이터베이스 엔진 버전으로 업그레이드하려면 AWS CLI modify-db-snapshot 명령을 실행합니다.

옵션

- `--db-snapshot-identifier` – 업그레이드할 DB 스냅샷의 식별자입니다. 식별자는 고유의 Amazon 리소스 이름(ARN)이어야 합니다. 자세한 내용은 Amazon RDS의 Amazon 리소스 이름(ARN) 섹션을 참조하세요.
- `--engine-version` – DB 스냅샷을 업그레이드할 엔진 버전입니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-snapshot \
```

```
    --db-snapshot-identifier my_db_snapshot \
    --engine-version new_version
```

Windows의 경우:

```
aws rds modify-db-snapshot ^
    --db-snapshot-identifier my_db_snapshot ^
    --engine-version new_version
```

Amazon RDS API

DB 스냅샷을 새 데이터베이스 엔진 버전으로 업그레이드하려면 RDS API ModifyDBSnapshot 작업을 직접 호출하세요.

파라미터

- `DBSnapshotIdentifier` – 업그레이드할 DB 스냅샷의 식별자입니다. 식별자는 고유의 Amazon 리소스 이름(ARN)이어야 합니다. 자세한 내용은 Amazon RDS의 Amazon 리소스 이름(ARN) 섹션을 참조하세요.
- `EngineVersion` – DB 스냅샷을 업그레이드할 엔진 버전입니다.

RDS for MariaDB에서 지원되지 않는 엔진 버전이 있는 DB 스냅샷에 대한 업그레이드 옵션

다음 표에는 RDS for MariaDB DB 스냅샷에 대해 지원되지 않는 엔진 버전에서 업그레이드할 수 있는 엔진 버전이 나와 있습니다.

> **Note**
>
> 선택한 엔진 버전으로 업그레이드하려면 DB 스냅샷을 두 번 이상 업그레이드해야 할 수 있습니다.

DB 스냅샷 엔진 버전	업그레이드 가능한 엔진 버전
10.0.17, 10.0.24, 10.0.28, 10.0.31, 10.0.32, 10.0.34, 10.0.35	10.0.38, 10.1.48, 10.6.19, 10.6.20, 10.11.9, 10.11.10, 11.4.3, 11.4.4
10.1.14, 10.1.16, 10.1.19, 10.1.23, 10.1.26, 10.1.31, 10.1.34	10.1.48, 10.2.44, 10.6.19, 10.6.20, 10.11.9, 10.11.10, 11.4.3, 11.4.4
10.2.11, 10.2.12, 10.2.15, 10.2.21, 10.2.32, 10.2.37, 10.2.39, 10.2.40, 10.2.41, 10.2.43	10.2.44, 10.3.39, 10.6.19, 10.6.20, 10.6.21, 10.11.9, 10.11.10, 10.11.11, 11.4.3, 11.4.4
10.3.8, 10.3.13, 10.3.20, 10.3.23, 10.3.28, 10.3.31, 10.3.32	10.3.39, 10.4.34, 10.6.19, 10.6.20 10.6.21, 10.11.9, 10.11.10, 10.11.11, 11.4.3, 11.4.4
10.4.8	10.4.34, 10.5.16, 10.5.17, 10.5.18, 10.5.20, 10.5.21, 10.5.22, 10.5.23, 10.5.24, 10.5.25, 10.5.26, 10.5.27, 10.5.28, 10.6.8, 10.6.10, 10.6.11, 10.6.13, 10.6.14, 10.6.15, 10.6.16, 10.6.18, 10.6.19, 10.6.20, 10.6.21, 10.11.4, 10.6.17, 10.11.5, 10.11.6, 10.11.7, 10.11.8, 10.11.9, 10.11.10, 10.11.11

Amazon RDS for MariaDB DB 인스턴스로 데이터 가져오기

RDS for MariaDB DB 인스턴스로 데이터를 가져오는 기법에는 몇 가지가 있습니다. 가장 좋은 방법은 다음과 같은 여러 요인에 따라 달라집니다.

- 데이터 원본
- 데이터 분량
- 일회성 혹은 지속적
- 가동 중지 시간 길이

데이터와 함께 애플리케이션을 마이그레이션하는 경우라면 감당할 수 있는 가동 중지 시간도 고려해야 합니다.

다음 표에는 RDS for MariaDB DB 인스턴스로 데이터를 가져오는 기법이 나와 있습니다.

> **Note**
>
> Amazon RDS는 Amazon S3에서 RDS for MySQL DB 인스턴스로 가져오기만 지원합니다. `mariadb-backup`을 사용하여 생성된 백업을 RDS for MariaDB로 가져오는 것은 현재 지원되지 않습니다.

소스	데이터 분량	일회성 혹은 지속적	애플리케이션 가동 중지	기술	추가 정보
온프레미스 또는 Amazon EC2에 있는 기존 MariaDB	임의	지속적	최소화	기존 MariaDB 데이터베이스가 복제 소스가 되도록 복제본을 구성합니다. 외부 인스턴스가 MariaDB 버전 10.0.24 이상인 경우 MariaDB 글로벌 트랜잭션 식별자(GTID)를 사용하거나 10.0.24 이전 버전의 MariaDB 인스턴스인 경우 바이너리 로그 좌표를 사용하여 MariaDB DB 인스턴스로의 복제를 구성할 수 있습니다. MariaDB GTID는 MySQL GTID와 다르게 구현되며,	외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성

소스	데이터 분량	일회성 혹은 지속적	애플리케이션 가동 중지	기술	추가 정보
데이터베이스				MySQL GTID는 Amazon RDS에서 지원되지 않습니다.	가동 중지 시간을 줄이면서 Amazon RDS for MariaDB DB 인스턴스로 데이터 가져오기

소스	데이터 분량	일회성 혹은 지속적	애플리케이션 가동 중지	기술	추가 정보
기존의 모든 데이터베이스	모두 선택	일회성 혹은 지속적	최소화	AWS Database Migration Service을 사용하면 가동 중지 시간을 최소화하면서 데이터베이스를 마이그레이션할 수 있으며 대부분의 DB 엔진에서는 지속적으로 복제를 계속할 수 있습니다.	AWS Database Migration Service란? 및 AWS Database Migration Service 사용 설명서의 AWS DMS에서 MySQL 호환 데이터베이스를 대상으로 사용
기존 MariaDB DB 인스턴스	임의	일회성 혹은 지속적	최소화	지속적인 복제를 위한 읽기 전용 복제본을 생성합니다. 새 DB 인스턴스를 한 번만 생성하도록 읽기 전용 복제본을 승격시킵니다.	DB 인스턴스 읽기 전용 복제본 작업

소스	데이터 분량	일회성 혹은 지속적	애플리케이션 가동 중지	기술	추가 정보
기존 MariaDB 데이터베이스	스몰	한 번만	약간	명령줄 유틸리티를 사용하여 MariaDB DB 인스턴스에 바로 데이터를 복제합니다.	외부 MariaDB 데이터베이스에서 Amazon RDS for MariaDB DB 인스턴스로 데이터 가져오기
기존 데이터베이스에 저장되지 않은 데이터	Medium	한 번만	약간	플랫 파일을 만들고 MariaDB `LOAD DATA LOCAL INFILE` 문을 이용하여 가져옵니다.	원하는 소스에서 Amazon RDS for MariaDB DB 인스턴스로 데이터 가져오기

> **Note**
>
> `mysql` 시스템 데이터베이스에는 DB 인스턴스에 로그인하고 데이터에 액세스하는 데 필요한 인증 및 권한 부여 정보가 포함되어 있습니다. DB 인스턴스에 있는 `mysql` 데이터베이스의 각종 테이블, 데이터 또는 기타 콘텐츠를 삭제하거나 변경하거나 이름을 바꾸거나 자르면 오류가

발생하여 DB 인스턴스와 데이터에 액세스할 수 없게 될 수 있습니다. 이 문제가 발생할 경우 AWS CLI restore-db-instance-from-db-snapshot 명령을 사용하여 DB 인스턴스를 스냅샷에서 복원할 수 있습니다. restore-db-instance-to-point-in-time 명령을 사용하여 DB 인스턴스를 복구할 수 있습니다.

MariaDB에 대한 데이터 가져오기 고려 사항

다음 내용에는 MariaDB로 데이터를 로드하는 것과 관련된 기술 정보가 포함되어 있습니다. 이 콘텐츠는 MariaDB 서버 아키텍처에 익숙한 사용자를 대상으로 합니다.

이진 로깅

이진 로깅을 활성화하면 로깅을 비활성화했을 때와 비교해 데이터 로드 성능이 저하되고 최대 4배의 추가 디스크 공간이 필요합니다. 데이터를 로드하는 데 사용되는 트랜잭션 크기는 시스템 성능 및 디스크 공간 요구 사항에 직접적인 영향을 미칩니다. 트랜잭션이 클수록 더 많은 리소스가 필요합니다.

트랜잭션 크기

트랜잭션 크기는 MariaDB 데이터 로드의 다음 측면에 영향을 미칩니다.

- 리소스 소비
- 디스크 공간 사용률
- 프로세스 재개
- 복구 시간
- 입력 형식(플랫 파일 또는 SQL)

이 섹션에서는 트랜잭션 크기가 이진 로깅에 미치는 영향을 설명하고 큰 데이터를 로드하는 중에 이진 로깅을 비활성화하는 이유를 논증합니다. Amazon RDS 자동 백업 보존 기간을 설정하여 이진 로깅을 활성화 및 비활성화할 수 있습니다. 0이 아닌 값으로 설정하면 이진 로깅이 활성화되고 0으로 설정하면 비활성화됩니다. 자세한 내용은 백업 보관 기간 섹션을 참조하세요.

이 섹션에서는 대규모 트랜잭션이 InnoDB에 미치는 영향과 트랜잭션 크기를 작게 유지하는 것이 중요한 이유도 설명합니다.

작은 트랜잭션

작은 트랜잭션의 경우, 이진 로깅을 사용하면 데이터 로드에 필요한 디스크 쓰기 작업 수가 배가됩니다. 이 결과 다른 데이터베이스 세션의 성능이 심각하게 저하되고 데이터 로딩 시간이 증가할 수 있습니다. 발생하는 성능 저하는 부분적으로 다음 요인에 따라 달라집니다.

- 업로드 속도
- 로드 중에 발생하는 기타 데이터베이스 활동
- Amazon RDS DB 인스턴스의 용량

또한, 이진 로그는 로그가 백업 및 제거될 때까지 로드된 데이터의 양과 대략적으로 같은 양의 디스크 공간을 사용합니다. Amazon RDS는 이진 로그를 자주 백업하고 제거하는 방법으로 이 문제를 최소화합니다.

대규모 트랜잭션

대규모 트랜잭션의 경우 다음과 같은 이유로 이진 로깅이 사용하는 IOPS 및 디스크가 3배로 늘어납니다.

- 이진 로그 캐시는 트랜잭션 데이터를 일시적으로 디스크에 저장합니다.
- 이 캐시는 디스크 공간을 소비하는 트랜잭션 크기에 따라 증가합니다.
- 트랜잭션(커밋 또는 롤백)이 완료되면 시스템은 캐시를 이진 로그에 복사합니다.

이 프로세스는 다음과 같은 데이터 복사본 세 개를 만듭니다.

- 원본 데이터
- 디스크의 캐시
- 최종 이진 로그 항목

각 쓰기 작업에는 추가 IO가 발생하여 성능에 더욱 영향을 미칩니다.

따라서 이진 로깅에는 로깅을 비활성화했을 때에 비해 디스크 공간이 3배 필요합니다. 예를 들어 10GiB의 데이터를 단일 트랜잭션으로 로드하면 다음과 같은 세 개의 복사본이 만들어집니다.

- 테이블 데이터 10GiB
- 이진 로그 캐시 10GiB
- 이진 로그 파일 10GiB

필요한 총 임시 디스크 공간은 30GiB입니다.

중요한 디스크 공간 고려 사항:

- 캐시 파일은 세션이 종료되거나 새 트랜잭션이 다른 캐시를 만들 때까지 유지됩니다.
- 이진 로그는 백업될 때까지 유지되며, 잠재적으로 20GiB(캐시 및 로그)를 장기간 차지할 수 있습니다.

`LOAD DATA LOCAL INFILE`을 사용하여 데이터를 로드하는 경우 데이터 복구는 로드 전에 수행된 백업에서 데이터베이스를 복구해야 하는 경우를 대비하여 네 번째 복사본을 만듭니다. 복원 중에 MariaDB가 이진 로그에서 플랫 파일로 데이터를 추출합니다. 그런 다음 MariaDB는 `LOAD DATA LOCAL INFILE`을 실행합니다. 이전 예시의 내용을 바탕으로 봤을 때, 이 복구를 수행하려면 테이블, 캐시, 로그 및 로컬 파일에 대해 각각 10GiB씩 총 40GiB의 임시 디스크 공간이 필요합니다. 최소 40GiB의 여유 디스크 공간이 없으면 복구가 실패합니다.

대규모 데이터 로드 최적화

대규모 데이터 로드의 경우 이진 로깅을 비활성화하여 오버헤드 및 디스크 공간 요구 사항을 줄입니다. 백업 보존 기간을 0으로 설정하여 이진 로깅을 비활성화할 수 있습니다. 로드가 완료되면 백업 보존 기간을 0이 아닌 적절한 값으로 복원합니다. 자세한 정보는 Amazon RDS DB 인스턴스 수정 및 설정 표의 백업 보존 기간을 참조하세요.

> **Note**
>
> DB 인스턴스가 읽기 전용 복제본의 원본 DB 인스턴스인 경우에는 백업 보존 기간을 0으로 설정할 수 없습니다.

데이터를 로드하기 전에 DB 스냅샷을 만드는 것이 좋습니다. 자세한 내용은 수동 백업 관리 섹션을 참조하세요.

InnoDB

실행 취소 로깅 및 복구 옵션에 대한 다음 정보는 데이터베이스 성능을 최적화하기 위해 InnoDB 트랜잭션을 작게 유지하는 데 도움이 됩니다.

InnoDB 실행 취소 로깅 이해

실행 취소는 트랜잭션 롤백을 활성화하고 다중 버전 동시성 제어(MVCC)를 지원하는 로깅 메커니즘입니다.

MariaDB 10.11 이하 버전의 경우 실행 취소 로그는 InnoDB 시스템 테이블스페이스(일반적으로 ibdata1)에 저장되며 제거 스레드가 로그를 제거할 때까지 유지됩니다. 따라서 대규모 데이터 로드 트랜잭션은 시스템 테이블스페이스가 상당히 커지는 원인이 될 수 있고, 이때 사용되는 디스크 공간은 데이터베이스를 다시 만들어야 회수할 수 있습니다.

모든 MariaDB 버전에서 제거 스레드는 가장 오래된 활성 트랜잭션이 커밋되거나 롤백될 때까지 실행 취소 로그를 제거할 수 없습니다. 데이터베이스가 로드 중에 다른 트랜잭션을 처리 중인 경우, 이들 트랜잭션의 실행 취소 로그 역시 누적되며 트랜잭션이 커밋되고 MVCC에 대해 실행 취소 로그가 필요한 다른 트랜잭션이 없더라도 실행 취소 로그를 제거할 수 없습니다. 이 경우 읽기 전용 트랜잭션을 포함한 모든 트랜잭션이 느려집니다. 이 속도 저하는 로드 트랜잭션뿐만 아니라 모든 트랜잭션이 변경하는 모든 행에 모든 트랜잭션이 액세스하기 때문에 발생합니다. 실제로 트랜잭션은 장기 실행 로드 트랜잭션으로 인해 실행 취소 로그 정리 중에 제거되지 않는 실행 취소 로그를 검사해야 합니다. 이는 수정된 행에 액세스하는 모든 작업의 성능에 영향을 줍니다.

InnoDB 트랜잭션 복구 옵션

InnoDB는 커밋 작업을 최적화하지만 대규모 트랜잭션 롤백은 느립니다. 더 빠른 복구를 위해 시점 복구를 수행하거나 DB 스냅샷을 복원합니다. 자세한 내용은 시점 복구 및 DB 인스턴스 복원(을)를 참조하세요.

데이터 가져오기 형식

MariaDB는 플랫 파일과 SQL이라는 두 가지 데이터 가져오기 형식을 지원합니다. 각 형식에 대한 정보를 검토하여 요구 사항에 가장 적합한 옵션을 결정합니다.

플랫 파일

소규모 트랜잭션의 경우 `LOAD DATA LOCAL INFILE`을 사용하여 플랫 파일을 로드합니다. 이 데이터 가져오기 형식은 SQL 사용에 비해 다음과 같은 이점을 제공할 수 있습니다.

- 더 낮은 네트워크 트래픽
- 데이터 전송 비용 절감
- 데이터베이스 처리 오버헤드 감소
- 더 빠른 처리

`LOAD DATA LOCAL INFILE`은 전체 플랫 파일을 하나의 트랜잭션으로 로드합니다. 다음과 같은 이점을 위해 개별 파일의 크기를 작게 유지합니다.

- 재개 기능 – 로드된 파일을 추적할 수 있습니다. 로드 중에 문제가 발생하면 중단된 부분부터 다시 시작할 수 있습니다. 일부 데이터를 Amazon RDS로 다시 전송해야 할 수도 있지만, 파일이 작으면 재전송되는 양이 최소화됩니다.
- 병렬 데이터 로드 - 단일 파일 로드에 충분한 IOPS와 네트워크 대역폭이 있는 경우 병렬로 로드하면 시간이 절약될 수 있습니다.
- 로드 속도 제어 - 데이터 로드가 다른 프로세스에 부정적인 영향을 미치는 경우 파일 간 간격을 늘려 로드 속도를 제어할 수 있습니다.

대규모 트랜잭션은 `LOAD DATA LOCAL INFILE`을 사용하여 데이터를 가져올 때의 이점을 줄입니다. 대량의 데이터를 더 작은 파일로 나눌 수 없는 경우 SQL을 사용하는 것을 고려하세요.

SQL

SQL은 플랫 파일에 비해 한 가지 중요한 장점이 있는데, 그것은 바로 트랜잭션 크기를 작게 유지하기 쉽다는 점입니다. 그러나 SQL은 플랫 파일보다 로드하는 데 훨씬 더 오래 걸릴 수 있습니다. 또한 장애 발생 후 재개 위치를 결정하기 어려울 수 있으며, mariadb-dump 파일을 다시 시작할 수 없습니다. mariadb-dump 파일을 로드하는 동안 오류가 발생하는 경우 이 파일을 수정하거나 바꾸어야 로드를 재개할 수 있습니다. 또는 장애의 원인을 수정한 후 파일을 로드하고 재전송하기 전의 특정 시점으로 복원할 수 있습니다. 자세한 내용은 시점 복구 섹션을 참조하세요.

데이터베이스 체크포인트에 Amazon RDS DB 스냅샷 사용

이진 로깅 없이 몇 시간 또는 며칠 등의 긴 기간 동안 데이터를 로드하는 경우 DB 스냅샷을 사용하여 데이터 안전을 위한 주기적 체크포인트를 제공하세요. 각 DB 스냅샷은 시스템 장애 또는 데이터 손상 이벤트 중에 복구 시점 역할을 하는 데이터베이스 인스턴스의 일관된 사본을 만듭니다. DB 스냅샷은 빠르기 때문에 체크포인트를 자주 제공해도 로드 성능에 미치는 영향이 최소화됩니다. 데이터베이스 내구성이나 복구 기능에 영향을 주지 않고 이전 DB 스냅샷을 삭제할 수 있습니다. DB 스냅샷에 대한 자세한 내용은 수동 백업 관리 섹션을 참조하세요.

데이터베이스 로드 시간 단축

다음 항목은 로드 시간을 줄이기 위한 추가 팁입니다.

- MariaDB 데이터베이스로 데이터를 로드하기 전에 모든 보조 인덱스를 만듭니다. 다른 데이터베이스 시스템과 달리 MariaDB는 보조 인덱스를 추가하거나 수정할 때 전체 테이블을 다시 빌드합니다. 이 프로세스는 인덱스 변경 사항이 있는 새 테이블을 만들고, 모든 데이터를 복사하고, 원본 테이블을 삭제합니다.

- 프라이머리 키 순서로 데이터를 로드합니다. InnoDB 테이블의 경우 이렇게 하면 로드 시간을 75%~80% 줄이고 데이터 파일 크기를 50% 줄일 수 있습니다.
- `foreign_key_checks`를 `0`으로 설정하여 외래 키 제약 조건을 비활성화합니다. `LOAD DATA LOCAL INFILE`을 통해 로드되는 플랫 파일의 경우 이 단계는 많은 경우에 필수입니다. 모든 로드에서 외래 키 검사를 비활성화하면 데이터 로드가 가속화됩니다. 로드가 완료된 후 `foreign_key_checks`를 `1`로 설정하고 데이터를 확인하여 제약 조건을 다시 활성화합니다.
- 리소스 한도가 가까워지지 않은 한 데이터를 병렬로 로드합니다. 여러 테이블 세그먼트에서 동시 로드를 활성화하기 위해 적절한 경우 파티셔닝된 테이블을 사용합니다.
- SQL 실행 오버헤드를 줄이려면 여러 `INSERT` 문을 단일 다중 값 `INSERT` 작업으로 결합합니다.`mariadb-dump`는 이 최적화를 자동으로 구현합니다.
- `innodb_flush_log_at_trx_commit`을 `0`으로 설정하여 InnoDB 로그 IO 작업을 줄입니다. 로드가 완료되면 `innodb_flush_log_at_trx_commit`을 `1`로 복원합니다.

> **Warning**
>
> `innodb_flush_log_at_trx_commit`을 `0`으로 설정하면 InnoDB가 커밋이 발생할 때마다가 아닌 1초마다 로그를 플러시합니다. 이 설정은 성능을 높이지만 시스템 장애 발생 시 트랜잭션이 손실될 위험이 있습니다.

- 읽기 전용 복제본이 없는 DB 인스턴스로 데이터를 로드하는 경우 `sync_binlog`를 `0`으로 설정합니다. 로드가 완료되면 `sync_binlog parameter`를 `1`로 복원합니다.
- DB 인스턴스를 다중 AZ 배포로 변환하기 전에 단일 AZ DB 인스턴스로 데이터를 로드합니다. DB 인스턴스가 이미 다중 AZ 배포를 사용하는 경우 데이터 로드를 위해 단일 AZ 배포로 전환하지 않는 것이 좋습니다. 이렇게 해도 아주 작은 부분만이 개선됩니다.

외부 MariaDB 데이터베이스에서 Amazon RDS for MariaDB DB 인스턴스로 데이터 가져오기

기존 MariaDB 또는 MySQL 데이터베이스에서 RDS for MariaDB 또는 RDS for MySQL DB 인스턴스로 데이터를 가져올 수 있습니다. 이렇게 하려면 mysqldump를 사용하여 데이터베이스를 복사하고 RDS for MariaDB 또는 RDS for MySQL DB 인스턴스에 바로 파이프합니다. `mysqldump` 명령줄 유틸리티는 한 MySQL 또는 MariaDB 서버에서 다른 MySQL 또는 MariaDB 서버로 데이터를 전송하고 백업본을 만드는 데 흔히 사용됩니다. 이 유틸리티는 MySQL 및 MariaDB 클라이언트 소프트웨어와 함께 포함되어 있습니다.

MariaDB 10.5에서는 이 클라이언트를 mariadb-dump라고 합니다. MariaDB 11.0.1부터 `mysqldump` 대신 `mariadb-dump`를 사용해야 합니다.

> **Note**
>
> MySQL DB 인스턴스에 많은 양의 데이터를 가져와서 내보내는 경우 `xtrabackup` 백업 파일 및 Amazon S3를 사용하여 Amazon RDS 내부 및 외부로 데이터를 더 빠르고 안정적으로 이동할 수 있습니다. 자세한 내용은 MySQL DB 인스턴스로 백업 복원 섹션을 참조하세요.
> Amazon RDS는 `xtrabackup` for MariaDB 또는 `mariabackup`을 지원하지 않습니다.
> Amazon RDS는 Amazon S3 for MySQL에서만 가져오기를 지원합니다.

외부 데이터베이스에서 Amazon RDS DB 인스턴스로 데이터를 이동하는 일반적인 `mysqldump` 명령은 다음 예와 같습니다. 값을 실제 정보로 바꿉니다. MariaDB 11.0.1 이상 버전의 경우 `mysqldump`를 `mariadb-dump`로 교체하고 `mysql`을 `mariadb`로 교체하세요.

```
mysqldump -u local_user \
    --databases database_name \
    --single-transaction \
    --compress \
    --order-by-primary  \
    --routines=0 \
    --triggers=0 \
    --events=0 \
    -plocal_password | mysql -u RDS_user \
        --port=port_number \
        --host=host_name \
        -pRDS_password
```

> **Important**
>
> `-p` 옵션과 입력한 암호 사이에 공백이 없어야 합니다.
> 보안 모범 사례로 예시에 표시된 프롬프트 이외의 자격 증명을 지정하는 것이 좋습니다.

다음 권장 사항과 고려 사항을 잘 파악하고 있어야 합니다.

- 덤프 파일에서 다음 스키마를 제외합니다.
  - `sys`

  - performance_schema
  - information_schema

  mysqldump 및 mariadb-dump 유틸리티는 기본적으로 이러한 스키마를 제외합니다.
- 사용자 및 권한을 마이그레이션해야 하는 경우 이를 다시 생성하는 데이터 제어 언어(DCL)를 생성하는 도구 사용을 고려합니다. 예를 들어 pt-show-grants 유틸리티가 있습니다.
- 가져오기를 수행하려면 사용자가 DB 인스턴스에 액세스할 수 있어야 합니다. 자세한 내용은 보안 그룹을 통한 액세스 제어 섹션을 참조하세요.

사용되는 파라미터는 다음과 같습니다.

- -u *local_user* – 사용자 이름을 지정하기 위해 사용합니다. 이 파라미터를 처음 사용할 때, 로컬 MySQL 또는 MariaDB 데이터베이스에서 --databases 파라미터로 식별되는 사용자 계정 이름을 지정합니다.
- --databases *database_name* – 로컬 MySQL 또는 MariaDB 인스턴스에서 Amazon RDS로 가져오려는 데이터베이스 이름을 지정합니다.
- --single-transaction – 로컬 데이터베이스에서 로드한 모든 데이터가 단일 시점에서 일치하는지 확인하기 위해 사용합니다. mysqldump가 데이터를 읽는 동안 데이터를 변경하는 다른 프로세스가 있는 경우, 이 파라미터를 사용하여 데이터 무결성을 유지합니다.
- --compress – 데이터를 Amazon RDS로 전송하기 전에 로컬 데이터베이스에서 데이터를 압축하여 네트워크 대역폭 사용을 줄이기 위해 사용합니다.
- --order-by-primary – 기본 키를 기준으로 각 테이블의 데이터를 정렬하여 로드 시간을 줄이기 위해 사용합니다.
- --routines - 복사하려는 데이터베이스에 저장 프로시저 또는 함수와 같은 루틴이 있는 경우 사용합니다. 파라미터를 0으로 설정하면 가져오기 프로세스 중에 루틴이 제외됩니다. 그런 다음 나중에 Amazon RDS 데이터베이스에서 루틴을 수동으로 다시 만듭니다.
- --triggers - 복사 중인 데이터베이스에 트리거가 있는 경우 사용합니다. 파라미터를 0으로 설정하면 가져오기 프로세스 중에 트리거가 제외됩니다. 그런 다음 Amazon RDS 데이터베이스에서 트리거를 수동으로 다시 만듭니다.
- --events - 복사하려는 데이터베이스에 이벤트가 있는 경우 사용합니다. 파라미터를 0으로 설정하면 가져오기 프로세스 중에 이벤트가 제외됩니다. 그런 다음 Amazon RDS 데이터베이스에서 이벤트를 수동으로 다시 만듭니다.
- -p*local_password* – 암호를 지정하기 위해 사용합니다. 이 파라미터를 처음 사용할 때, 첫 번째 -u 파라미터로 식별되는 사용자 계정 암호를 지정합니다.

- `-u` *`RDS_user`* – 사용자 이름을 지정하기 위해 사용합니다. 이 파라미터를 두 번째로 사용할 때, `--host` 파라미터로 식별되는 MariaDB 또는 MySQL DB 인스턴스에 대한 기본 데이터베이스의 사용자 계정 이름을 지정합니다.
- `--port` *`port_number`* - MySQL 또는 MariaDB DB 인스턴스의 포트를 지정하기 위해 사용합니다. DB 인스턴스를 만들 때 값을 변경하지 않는 한, 기본값은 3306입니다.
- `--host` *`host_name`* - Amazon RDS DB 인스턴스 엔드포인트의 도메인 이름 시스템(DNS) 이름을 지정하기 위해 사용합니다(예:`myinstance.123456789012.us-east-1.rds.amazonaws.com`). Amazon RDS 콘솔의 인스턴스 세부 정보에서 엔드포인트 값을 찾을 수 있습니다.
- `-p`*`RDS_password`* – 암호를 지정하기 위해 사용합니다. 이 파라미터를 두 번째로 사용할 때, 두 번째 `-u` 파라미터로 식별되는 사용자 계정 암호를 지정합니다.

Amazon RDS 데이터베이스에서 저장 프로시저, 트리거, 함수 또는 이벤트를 수동으로 만들어야 합니다. 복사 중인 데이터베이스에 이런 객체가 하나라도 있는 경우에는 `mysqldump` 또는 `mariadb-dump`를 실행할 때 이런 객체를 제외합니다. 이렇게 하려면 `mysqldump` 또는 `mariadb-dump` 명령에 다음 파라미터를 포함합니다.

- `--routines=0`
- `--triggers=0`
- `--events=0`

예제

다음 예제에서는 로컬 호스트에 있는 `world` 샘플 데이터베이스를 RDS for MySQL DB 인스턴스로 복사합니다. 값을 실제 정보로 바꿉니다. 샘플 데이터베이스를 RDS for MariaDB DB 인스턴스에 복사하려면 `mysqldump`를 `mariadb-dump`로 교체하고 `mysql`을 `mariadb`로 교체합니다.

대상 LinuxmacOS, 또는Unix:

```
sudo mysqldump -u local_user \
    --databases world \
    --single-transaction \
    --compress \
    --order-by-primary  \
    --routines=0 \
    --triggers=0 \
    --events=0 \
    -plocal_password | mysql -u rds_user \
```

```
        --port=3306 \
        --host=my_instance.123456789012.us-east-1.rds.amazonaws.com \
        -pRDS_password
```

Windows의 경우:

다음 명령은 Windows 프로그램 메뉴에서 명령 프롬프트를 마우스 오른쪽 버튼으로 클릭하고 관리자 권한으로 실행을 선택하여 열린 명령 프롬프트에서 실행해야 합니다. 값을 실제 정보로 바꿉니다. 샘플 데이터베이스를 RDS for MariaDB DB 인스턴스에 복사하려면 `mysqldump`를 `mariadb-dump`로 교체하고 `mysql`을 `mariadb`로 교체합니다.

```
mysqldump -u local_user ^
    --databases world ^
    --single-transaction ^
    --compress ^
    --order-by-primary  ^
    --routines=0 ^
    --triggers=0 ^
    --events=0 ^
    -plocal_password | mysql -u RDS_user ^
        --port=3306 ^
        --host=my_instance.123456789012.us-east-1.rds.amazonaws.com ^
        -pRDS_password
```

> **Note**
>
> 보안 모범 사례로 예시에 표시된 프롬프트 이외의 자격 증명을 지정하는 것이 좋습니다.

가동 중지 시간을 줄이면서 Amazon RDS for MariaDB DB 인스턴스로 데이터 가져오기

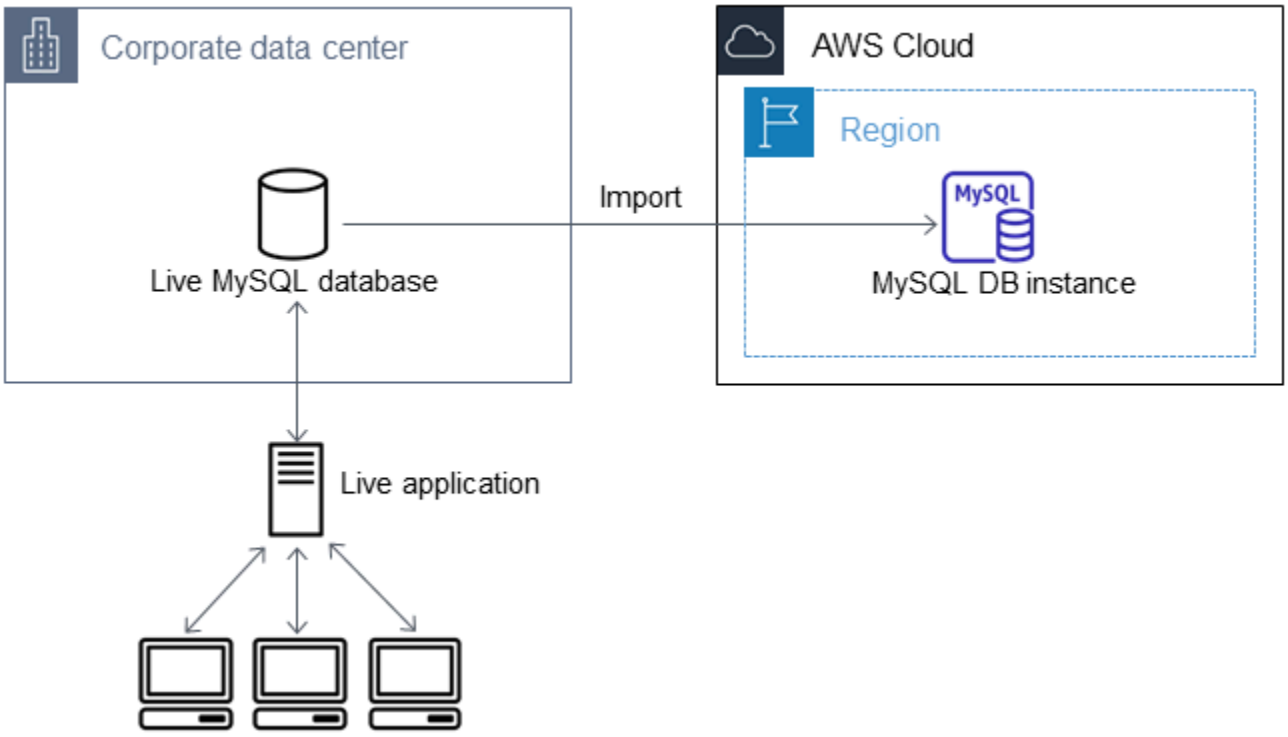
라이브 애플리케이션을 지원하는 외부 MariaDB 또는 MySQL 데이터베이스에서 RDS for MariaDB 인스턴스, RDS for MySQL 인스턴스 또는 RDS for MySQL 다중 AZ DB 클러스터로 데이터를 가져와야 할 경우가 있습니다. 다음 절차를 통해 애플리케이션 가용성에 미치는 영향을 최소화하세요. 이 절차는 대규모 데이터베이스로 작업하는 경우에도 유용합니다. 이 절차를 통해 네트워크를 거쳐 AWS로 전달되는 데이터의 양을 줄여 가져오기 비용을 줄일 수 있습니다.

이 절차에서는 데이터베이스 데이터의 복사본을 Amazon EC2 인스턴스로 전송하고 데이터를 새 Amazon RDS 데이터베이스로 가져옵니다. 그런 다음, 애플리케이션을 Amazon RDS 데이터베이스

로 리디렉션하기 전에 복제를 사용하여 Amazon RDS 데이터베이스를 라이브 외부 인스턴스에 맞춰 최신 상태로 업데이트합니다. 외부 인스턴스가 MariaDB 10.0.24 이상이고 대상 인스턴스가 RDS for MariaDB일 경우에는 글로벌 트랜잭션 식별자(GTID)를 기반으로 MariaDB 복제를 구성합니다. 그렇지 않으면 이진 로그 좌표를 기반으로 복제를 구성합니다. GTID 기반 복제가 더욱 신뢰성이 높은 방법이므로 외부 데이터베이스에서 지원된다면 GTID 기반 복제를 사용하는 것이 좋습니다. 자세한 내용은 MariaDB 설명서에서 Global Transaction ID 단원을 참조하세요.

> **Note**
>
> RDS for MySQL DB 인스턴스에 데이터를 가져오고자 하며 시나리오가 이 방법을 지원하는 경우 백업 파일 및 Amazon S3를 사용하여 Amazon RDS 내부 및 외부로 데이터를 이동하는 것이 좋습니다. 자세한 내용은 MySQL DB 인스턴스로 백업 복원 섹션을 참조하세요.

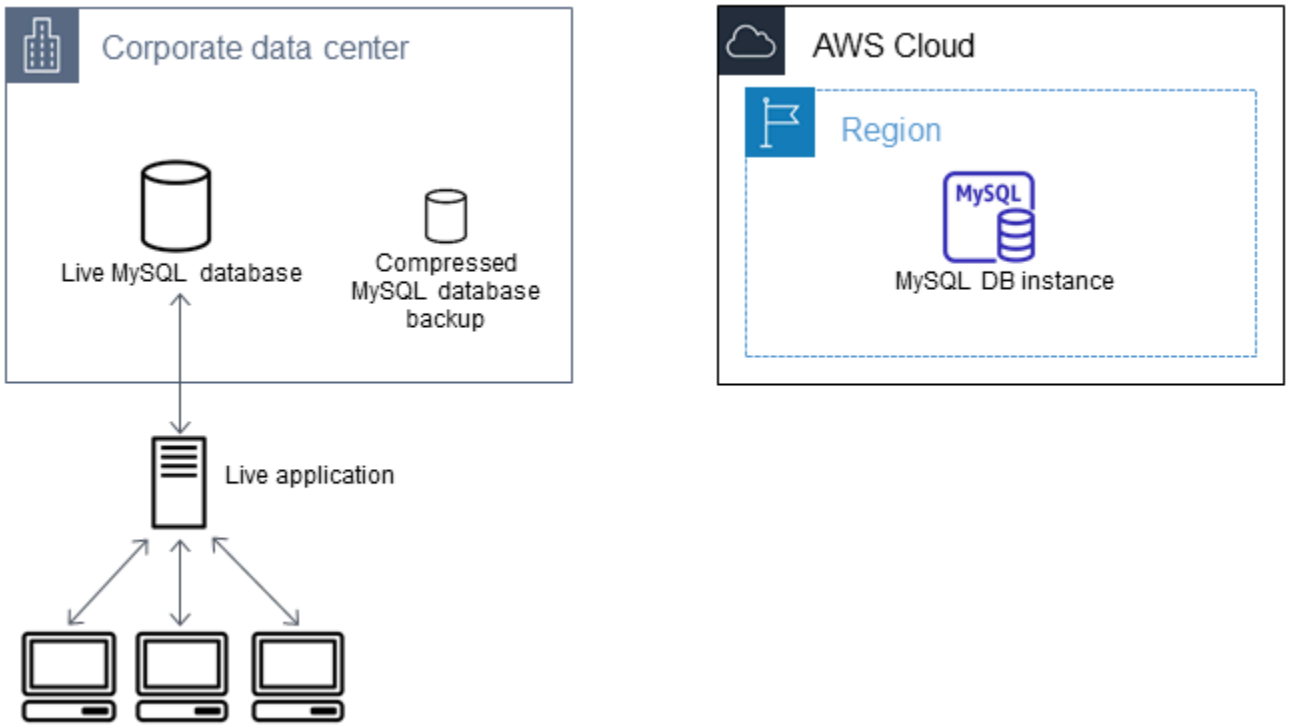
다음 다이어그램은 외부 MySQL 데이터베이스를 Amazon RDS의 MySQL 데이터베이스로 가져오는 것을 보여 줍니다.



작업 1: 기존 데이터베이스의 복사본 만들기

최소한의 가동 중지 시간으로 대량의 데이터를 RDS for MariaDB 또는 RDS for MySQL 데이터베이스로 마이그레이션하는 프로세스에서 첫 번째 단계는 원본 데이터의 복사본을 만드는 것입니다.

다음 다이어그램은 MySQL 데이터베이스의 백업을 만드는 방법을 보여 줍니다.



mysqldump 유틸리티를 사용하여 SQL 또는 구분 기호로 분리된 텍스트 형식으로 데이터베이스 백업본을 만들 수 있습니다. MariaDB 10.5에서는 이 클라이언트를 mariadb-dump라고 합니다. MariaDB 11.0.1부터 mysqldump 대신 mariadb-dump를 사용해야 합니다. 비프로덕션 환경에서 각각의 형식으로 테스트 실행을 통해 어떤 방법을 사용해야 mysqldump 실행 시간이 최소화되는지 확인하는 것을 권장합니다.

또한, 로드를 위해 구분 기호로 분리된 텍스트 형식을 사용할 때의 이점에 대해 mysqldump 성능상의 이점을 비교 검토하는 것을 권장합니다. 구분 기호로 분리된 텍스트 형식을 사용하는 백업에서는 덤프되는 각 테이블에 대해 탭으로 구분된 텍스트 파일이 생성됩니다. 데이터베이스를 가져오는 데 필요한 시간을 줄이기 위해 LOAD DATA LOCAL INFILE 명령을 사용하여 이런 파일을 병렬로 로드할 수 있습니다. 자세한 내용은 Load the data for MariaDB 또는 Load the data for MySQL을 참조하세요.

백업 작업을 시작하기 전에 Amazon RDS로 복사할 MySQL 또는 MariaDB 데이터베이스에서 복제 옵션을 설정해야 합니다. 복제 옵션에는 이진 로깅 활성화와 고유한 서버 ID 설정이 포함됩니다. 이러한 옵션을 설정하면 서버가 데이터베이스 트랜잭션 로깅을 시작하고 이 프로세스의 후반부에 소스 복제 인스턴스가 되도록 서버를 준비시킵니다.

다음 권장 사항과 고려 사항을 잘 파악하고 있어야 합니다.

- 데이터베이스의 일관된 상태를 덤프하기 때문에 `--single-transaction`와 함께 `mysqldump` 옵션을 사용하세요. 덤프 파일이 올바른지 확인하려면 `mysqldump`가 실행되는 동안 데이터 정의 언어(DDL) 문을 실행하지 마시기 바랍니다. 이러한 작업에 대한 유지 관리 기간을 예약할 수 있습니다.
- 덤프 파일에서 다음 스키마를 제외합니다.
  - `sys`
  - `performance_schema`
  - `information_schema`

  `mysqldump` 및 `mariadb-dump` 유틸리티는 기본적으로 이러한 스키마를 제외합니다.
- 사용자 및 권한을 마이그레이션해야 하는 경우 이를 다시 생성하는 데이터 제어 언어(DCL)를 생성하는 도구 사용을 고려합니다. 예를 들어 pt-show-grants 유틸리티가 있습니다.

복제 옵션을 설정하려면

1. `my.cnf` 파일을 편집합니다. 이 파일은 보통 `/etc` 아래에 있습니다.

   ```
   sudo vi /etc/my.cnf
   ```

   `log_bin` 및 `server_id` 옵션을 `[mysqld]` 섹션에 추가합니다. `log_bin` 옵션은 이진 로그 파일에 대한 파일 이름 식별자를 제공합니다. `server_id` 옵션은 소스-복제본 관계에서 서버의 고유 식별자를 제공합니다.

   다음 예제에서는 `my.cnf` 파일의 업데이트된 `[mysqld]` 섹션을 보여줍니다.

   ```
   [mysqld]
   log-bin=mysql-bin
   server-id=1
   ```

   자세한 내용은 MySQL 설명서의 Setting the Replication Source Configuration을 참조하세요.
2. 다중 AZ DB 클러스터를 사용한 복제의 경우 `ENFORCE_GTID_CONSISTENCY` 및 `GTID_MODE` 파라미터를 `ON`으로 설정합니다.

   ```
   mysql> SET @@GLOBAL.ENFORCE_GTID_CONSISTENCY = ON;
   ```

   ```
   mysql> SET @@GLOBAL.GTID_MODE = ON;
   ```

DB 인스턴스를 사용한 복제에는 이러한 설정이 필요하지 않습니다.

3. mysql 서비스를 다시 시작합니다.

```
sudo service mysqld restart
```

기존 데이터베이스의 백업 복사본 만들기

1. SQL 또는 구분 기호로 분리된 텍스트 형식을 지정하는 mysqldump 유틸리티 또는 mariadb-dump 유틸리티를 사용하여 데이터 백업을 만듭니다.

   MySQL 8.0.25 이하 버전에서는 서버 간 복제를 시작할 때 사용할 수 있는 백업 파일을 만들려면 --master-data=2를 지정합니다. MySQL 8.0.26 이상 버전에서는 서버 간 복제를 시작할 때 사용할 수 있는 백업 파일을 만들려면 --source-data=2를 지정합니다. 자세한 내용은 MySQL 설명서의 mysqldump — A Database Backup Program을 참조하세요.

   성능을 개선하고 데이터 무결성을 보장하려면 mysqldump에 --order-by-primary 및 --single-transaction 옵션을 사용합니다.

   백업에 MySQL 시스템 데이터베이스가 포함되지 않도록 하려면 mysqldump에 --all-databases 옵션을 사용하지 마세요. 자세한 내용은 MySQL 설명서의 Creating a Data Snapshot Using mysqldump를 참조하세요.

   필요한 경우 chmod를 사용하여 백업 파일이 만들어질 디렉터리가 쓰기 가능한 디렉터리가 되도록 하세요.

   > **Important**
   >
   > Windows에서는 명령 창을 관리자 권한으로 실행하십시오.

   - SQL 출력을 표시하려면 다음 명령을 사용합니다.

     대상 LinuxmacOS, 또는Unix:

     ```
     sudo mysqldump \
         --databases database_name \
         --master-data=2  \
         --single-transaction \
     ```

```
    --order-by-primary \
    -r backup.sql \
    -u local_user \
    -ppassword
```

> **Note**
>
> 보안 모범 사례로 예시에 표시된 프롬프트 이외의 자격 증명을 지정하는 것이 좋습니다.

Windows의 경우:

```
mysqldump ^
    --databases database_name ^
    --master-data=2  ^
    --single-transaction ^
    --order-by-primary ^
    -r backup.sql ^
    -u local_user ^
    -ppassword
```

> **Note**
>
> 보안 모범 사례로 예시에 표시된 프롬프트 이외의 자격 증명을 지정하는 것이 좋습니다.

- 구분 기호로 분리된 텍스트 출력을 표시하려면 다음 명령을 사용합니다.

  대상 LinuxmacOS, 또는Unix:

```
sudo mysqldump \
    --tab=target_directory \
    --fields-terminated-by ',' \
    --fields-enclosed-by '"' \
    --lines-terminated-by 0x0d0a \
    database_name \
    --master-data=2 \
    --single-transaction \
    --order-by-primary \
    -ppassword
```

Windows의 경우:

```
mysqldump ^
    --tab=target_directory ^
    --fields-terminated-by "," ^
    --fields-enclosed-by """ ^
    --lines-terminated-by 0x0d0a ^
    database_name ^
    --master-data=2 ^
    --single-transaction ^
    --order-by-primary ^
    -ppassword
```

> **Note**
>
> 보안 모범 사례로 예시에 표시된 프롬프트 이외의 자격 증명을 지정하는 것이 좋습니다. Amazon RDS 데이터베이스에서 저장 프로시저, 트리거, 함수 또는 이벤트를 수동으로 만들어야 합니다. 복사 중인 데이터베이스에 이런 객체가 하나라도 있는 경우에는 `mysqldump`를 실행할 때 이런 객체를 제외합니다. 이렇게 하려면 `mysqldump` 명령에 다음 인수를 포함하세요.
>
> - `--routines=0`
> - `--triggers=0`
> - `--events=0`

MySQL 8.0.22 이하 버전 및 MariaDB 버전의 경우 `mysqldump`를 실행하고 구분된 텍스트 형식을 지정하면 `CHANGE MASTER TO` 설명이 반환됩니다. 이 설명에는 마스터 로그 파일 이름과 위치가 포함됩니다. MySQL 8.0.23 이상 버전의 경우 구분된 텍스트 형식을 사용하여 `mysqldump`를 실행하면 `CHANGE REPLICATION SOURCE TO` 설명이 반환됩니다. 이 설명에는 소스 로그 파일 이름과 위치가 포함됩니다. 외부 인스턴스가 MariaDB 10.0.24 이상 버전 또는 MySQL 8.0.23 이상 버전이 아닌 경우 `MASTER_LOG_FILE` 및 `MASTER_LOG_POS`의 값을 확인합니다. 복제를 설정할 때 이러한 값이 필요합니다.

MariaDB 버전과 MySQL 8.0.22 이하 버전에 대해 다음 출력이 반환됩니다.

```
-- Position to start replication or point-in-time recovery from
--
```

```
-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin-changelog.000031',
 MASTER_LOG_POS=107;
```

MySQL 8.0.23 이상 버전에 대해 다음 출력이 반환됩니다.

```
-- Position to start replication or point-in-time recovery from
--
-- CHANGE SOURCE TO SOURCE_LOG_FILE='mysql-bin-changelog.000031',
 SOURCE_LOG_POS=107;
```

MySQL 8.0.22 이하 버전에서 SQL 형식을 사용하는 경우 백업 파일의 CHANGE MASTER TO 설명에서 마스터 로그 파일 이름 및 위치를 가져올 수 있습니다. MySQL 8.0.23 이상 버전에서 SQL 형식을 사용하는 경우 백업 파일의 CHANGE REPLICATION SOURCE TO 설명에서 소스 로그 파일 이름 및 위치를 가져올 수 있습니다. 외부 인스턴스가 MariaDB 버전 10.0.24 이상일 경우에는 GTID를 다음 단계에서 가져올 수 있습니다.

2. 사용하는 외부 인스턴스가 MariaDB 버전 10.0.24 이상일 경우 GTID 기반 복제를 사용합니다. 외부 MariaDB 인스턴스에서 SHOW MASTER STATUS를 실행하여 이진 로그 파일 이름 및 위치를 가져온 다음, 외부 MariaDB 인스턴스에서 BINLOG_GTID_POS를 실행하여 GTID로 변환합니다.

```
SELECT BINLOG_GTID_POS('binary_log_file_name', binary_log_file_position);
```

반환된 GTID를 기록해 둡니다. 복제를 구성하는 데 GTID가 필요합니다.

3. 복사된 데이터를 압축하여 데이터를 Amazon RDS 데이터베이스로 복사하는 데 필요한 네트워크 리소스의 양을 줄입니다. 백업 파일의 크기를 기록해 둡니다. Amazon EC2 인스턴스를 얼마나 크게 만들지 결정할 때 이 정보가 필요합니다. 모두 완료되었으면, GZIP 또는 선호하는 압축 유틸리티를 사용하여 백업 파일을 압축합니다.

   - SQL 출력을 압축하려면 다음 명령을 사용합니다.

   ```
   gzip backup.sql
   ```

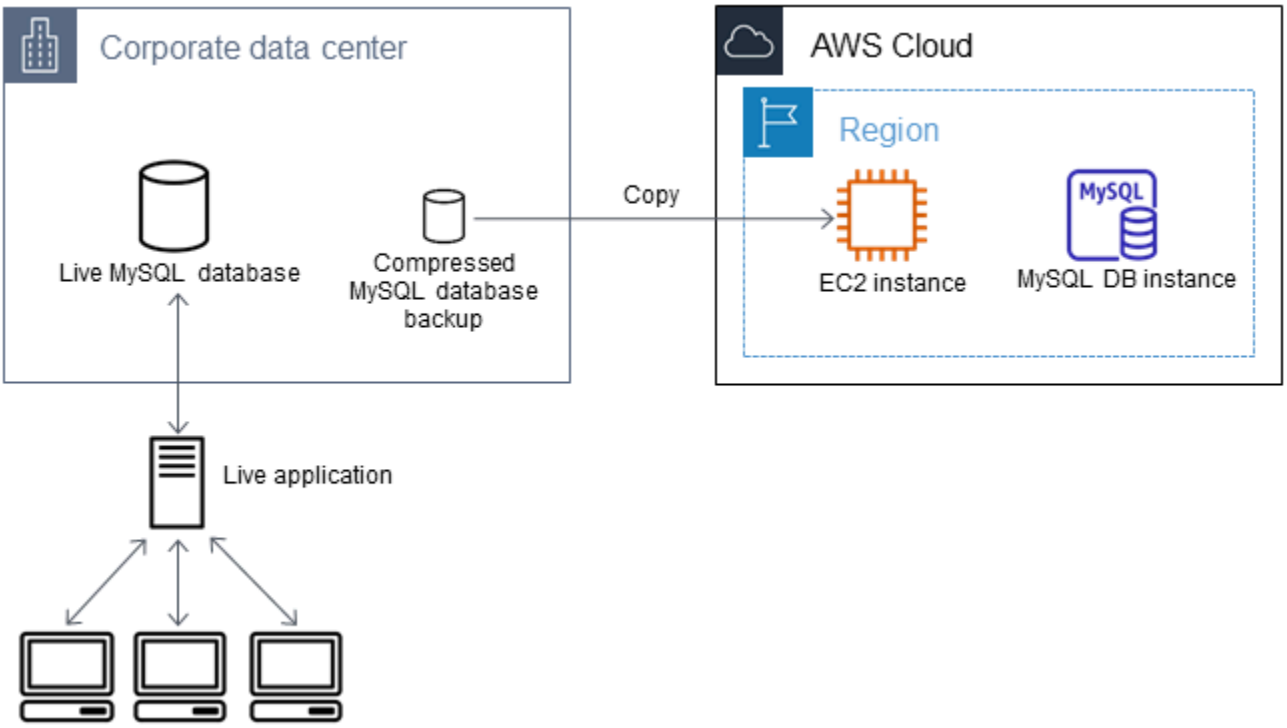
   - 구분 기호로 분리된 텍스트 출력을 압축하려면 다음 명령을 사용합니다.

   ```
   tar -zcvf backup.tar.gz target_directory
   ```

작업 2: Amazon EC2 인스턴스 만들기 및 압축된 데이터베이스 복사

압축된 데이터베이스 백업 파일을 Amazon EC2 인스턴스로 복사할 때는 데이터베이스 인스턴스 사이에서 압축되지 않은 데이터를 직접 복사할 때보다 네트워크 리소스를 덜 사용합니다. 데이터가 Amazon EC2에 있으면 MySQL 또는 MariaDB 데이터베이스로 바로 데이터를 복사할 수 있습니다. 네트워크 리소스의 비용을 절약하려면 Amazon EC2 인스턴스가 Amazon RDS DB 인스턴스와 같은 AWS 리전에 있어야 합니다. Amazon EC2 인스턴스를 Amazon RDS 데이터베이스와 같은 AWS 리전에 두면 가져오기 도중의 네트워크 대기 시간도 줄어듭니다.

다음 다이어그램은 데이터베이스 백업을 Amazon EC2 인스턴스로 복사하는 것을 보여줍니다.



Amazon EC2 인스턴스를 만들고 데이터를 복사하려면

1. Amazon RDS 데이터베이스를 만들려는 AWS 리전에서 가상 프라이빗 클라우드(VPC), VPC 보안 그룹 및 VPC 서브넷을 만듭니다. VPC 보안 그룹의 인바운드 규칙에서 애플리케이션에 필요한 IP 주소가 AWS에 연결하도록 허용하는지 확인합니다. IP 주소의 범위(예: `203.0.113.0/24`) 또는 다른 VPC 보안 그룹을 지정할 수 있습니다. Amazon VPC 콘솔을 사용하여 VPC, 서브넷 및 보안 그룹을 만들고 관리할 수 있습니다. 자세한 내용은 Amazon Virtual Private Cloud 사용 설명서의 Getting started with Amazon VPC를 참조하세요.
2. Amazon EC2 콘솔을 열고 Amazon EC2 인스턴스와 Amazon RDS 데이터베이스를 모두 포함하는 AWS 리전을 선택합니다. 1단계에서 만든 VPC, 서브넷 및 보안 그룹을 사용하여 Amazon EC2 인스턴스를 시작합니다. 데이터베이스 백업 파일이 압축되어 있지 않을 때는 이 파일을 위한 충분한 스

토리지 공간을 가진 인스턴스 유형을 선택해야 합니다. Amazon EC2 인스턴스에 관한 세부 정보는 Amazon Elastic Compute Cloud 사용 설명서의 Getting started with Amazon EC2를 참조하세요.

3. Amazon EC2 인스턴스에서 Amazon RDS 데이터베이스로 연결하려면 VPC 보안 그룹을 편집하세요. EC2 인스턴스의 프라이빗 IP 주소를 지정하는 인바운드 규칙을 추가합니다. EC2 콘솔 창에 있는 인스턴스 창의 세부 정보 탭에서 프라이빗 IP 주소를 찾을 수 있습니다. VPC 보안 그룹을 편집하고 인바운드 규칙을 추가하려면 EC2 콘솔 탐색 창에서 보안 그룹(Security Groups)를 선택하고 보안 그룹을 선택한 다음 EC2 인스턴스의 프라이빗 IP 주소를 지정하는 MySQL 또는 Aurora에 대한 인바운드 규칙을 추가합니다. 인바운드 규칙을 VPC 보안 그룹에 추가하는 방법을 알아보려면 Amazon Virtual Private Cloud 사용 설명서의 Security group rules를 참조하세요.
4. 로컬 시스템에서 Amazon EC2 인스턴스로 압축된 데이터베이스 백업 파일을 복사합니다. 필요한 경우 chmod를 사용하여 Amazon EC2 인스턴스의 대상 디렉터리에 대해 쓰기 권한이 있는지 확인하세요. scp 또는 Secure Shell(SSH) 클라이언트를 사용하여 파일을 복사할 수 있습니다. 다음은 예제 scp 명령입니다.

```
scp -r -i key pair.pem backup.sql.gz ec2-user@EC2 DNS:/target_directory/backup.sql.gz
```

> **Important**
>
> 민감한 데이터는 반드시 보안 네트워크 전송 프로토콜을 사용하여 복사해야 합니다.

5. Amazon EC2 인스턴스에 연결하고 다음 명령을 사용하여 최신 업데이트와 MySQL 클라이언트 도구를 설치합니다.

```
sudo yum update -y
sudo yum install mysql -y
```

자세한 내용은 Amazon Elastic Compute Cloud 사용 설명서의 Connect to your instance를 참조하세요.

> **Important**
>
> 이 예제에서는 Amazon Linux AMI 배포의 Amazon Machine Image(AMI)에 MySQL 클라이언트를 설치합니다. 이 예제에서는 Ubuntu 또는 Red Hat Enterprise Linux 등 다른 배포에 MySQL 클라이언트를 설치하지 않습니다. MySQL 설치에 대한 자세한 내용은 MySQL 설명서의 Installing MySQL을 참조하세요.

6. Amazon EC2 인스턴스에 연결되어 있는 상태에서 데이터베이스 백업 파일의 압축을 풉니다. 다음 명령은 예시입니다.

   - SQL 출력을 압축 해제하려면 다음 명령을 사용합니다.

     ```
     gzip backup.sql.gz -d
     ```

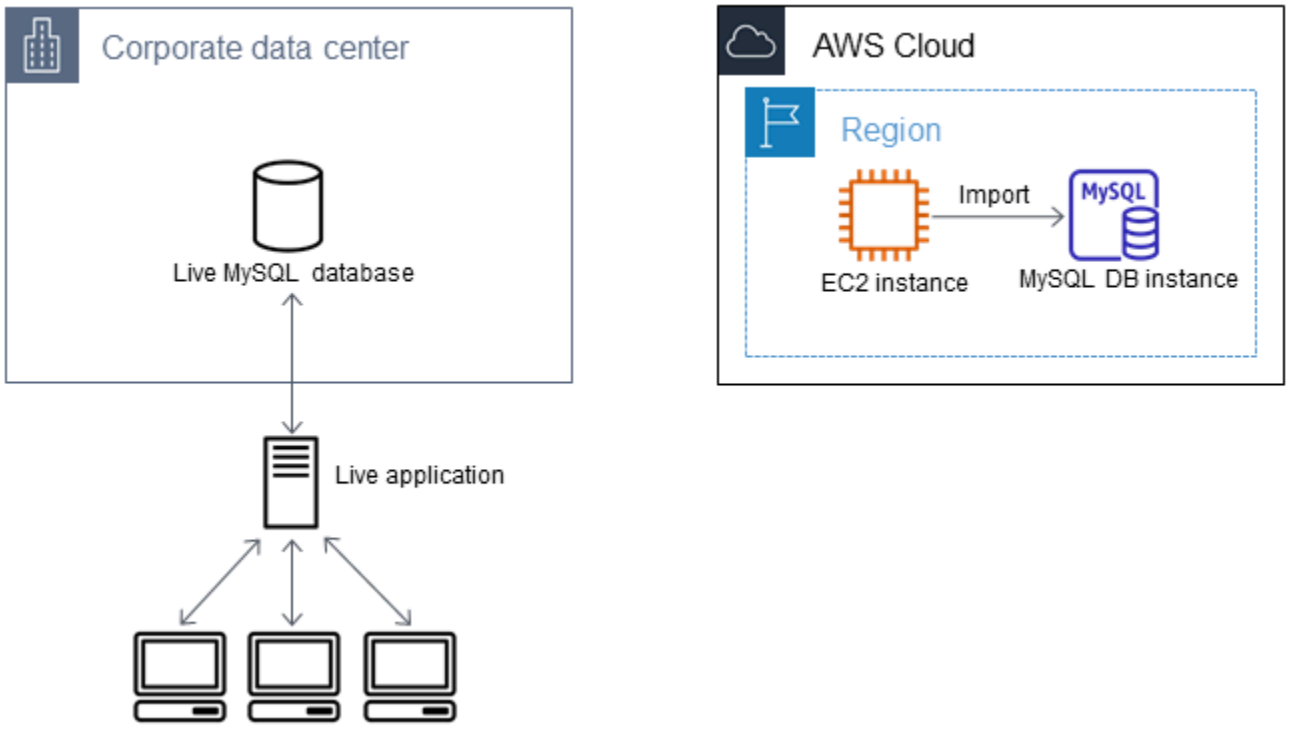
   - 구분 기호로 분리된 텍스트 출력을 압축 해제하려면 다음 명령을 사용합니다.

     ```
     tar xzvf backup.tar.gz
     ```

작업 3: MySQL 또는 MariaDB 데이터베이스 만들기 및 Amazon EC2 인스턴스에서 데이터 가져오기

Amazon EC2 인스턴스와 같은 AWS 리전에 RDS for MariaDB DB 인스턴스, RDS for MySQL DB 인스턴스 또는 RDS for MySQL 다중 AZ DB 클러스터를 만들면 인터넷에서 가져오는 것보다 빠른 속도로 Amazon EC2에서 데이터베이스 백업 파일을 가져올 수 있습니다.

다음 다이어그램은 Amazon EC2 인스턴스에서 MySQL 데이터베이스로 백업을 가져오는 것을 보여줍니다.



MariaDB 또는 MySQL 데이터베이스를 만들고 데이터 가져오기

1. 이 Amazon RDS 데이터베이스에 대해 예상되는 워크로드를 지원하기 위해 필요한 DB 인스턴스 클래스와 스토리지 공간의 양을 결정합니다. 이 프로세스의 일환으로 데이터 로드 절차를 위해 충분한 공간과 처리 용량이 어느 정도인지 결정합니다. 또한 프로덕션 워크로드를 처리하는 데 필요한 사항을 결정합니다. 원본 MySQL 또는 MariaDB 데이터베이스의 크기와 리소스를 바탕으로 평가할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요.
2. Amazon EC2 인스턴스가 포함된 AWS 리전에서 DB 인스턴스 또는 다중 AZ DB 클러스터를 만듭니다.

   RDS for MySQL 다중 AZ DB 클러스터를 만들려면 Amazon RDS용 다중 AZ DB 클러스터 생성 섹션의 지침을 따르세요.

   RDS for MariaDB 또는 RDS for MySQL DB 인스턴스를 생성하려면 Amazon RDS DB 인스턴스 생성의 지침과 다음 지침을 따르세요.

   - 소스 DB 인스턴스와 호환되는 DB 엔진 버전을 지정합니다.
     - 소스 인스턴스가 MySQL 5.5.x인 경우 Amazon RDS DB 인스턴스는 MySQL이어야 합니다.
     - 소스 인스턴스가 MySQL 5.6.x 또는 5.7.x인 경우 Amazon RDS DB 인스턴스는 MySQL 또는 MariaDB여야 합니다.
     - 원본 인스턴스가 MySQL 8.0.x인 경우 Amazon RDS DB 인스턴스는 MySQL 8.0.x여야 합니다.
     - 원본 인스턴스가 MySQL 8.4.x인 경우 Amazon RDS DB 인스턴스는 MySQL 8.4.x여야 합니다.
     - 소스 인스턴스가 MariaDB 5.5 이상인 경우 Amazon RDS DB 인스턴스는 MariaDB여야 합니다.
   - Amazon EC2 인스턴스에서와 동일한 Virtual Private Cloud(VPC) 및 VPC 보안 그룹을 지정합니다. 이 접근 방식에서는 Amazon EC2 인스턴스와 Amazon RDS 인스턴스가 네트워크를 통해 서로를 볼 수 있습니다. DB 인스턴스가 공개적으로 액세스 가능한지 확인합니다. 다음 섹션에 설명한 대로 소스 데이터베이스를 사용하여 복제를 설정하려면 DB 인스턴스에 공개적으로 액세스할 수 있어야 합니다.
   - 데이터베이스 백업을 가져온 후에만 여러 가용 영역, 백업 보존 또는 읽기 전용 복제본을 구성해야 합니다 가져오기가 완료되면 프로덕션 인스턴스에 대해 다중 AZ 및 백업 보존을 설정할 수 있습니다.
3. Amazon RDS 데이터베이스에 대한 기본 구성 옵션을 검토합니다. 데이터베이스에 대한 기본 파라미터 그룹에 원하는 구성 옵션이 없는 경우, 원하는 구성 옵션이 있는 다른 파라미터 그룹

을 찾거나 새 파라미터 그룹을 생성합니다. 파라미터 그룹을 생성하는 것에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 섹션을 참조하세요.

4. 마스터 사용자로 Amazon RDS 데이터베이스에 연결합니다. DB 인스턴스에 액세스해야 하는 서비스, 애플리케이션 및 관리자를 지원하는 데 필요한 사용자를 만듭니다. Amazon RDS 데이터베이스의 호스트 이름은 포트 번호를 포함하지 않은 DB 인스턴스의 엔드포인트 값입니다(예: `mysampledb.123456789012.us-west-2.rds.amazonaws.com`). Amazon RDS 콘솔의 데이터베이스 세부 정보에서 엔드포인트 값을 찾을 수 있습니다.
5. Amazon EC2 인스턴스에 연결합니다. 자세한 내용은 Amazon Elastic Compute Cloud 사용 설명서의 Connect to your instance를 참조하세요.
6. `mysql` 명령을 사용하여 Amazon EC2 인스턴스에서 원격 호스트로 Amazon RDS 데이터베이스에 연결합니다. 다음 명령은 예제입니다.

   ```
   mysql -h host_name -P 3306 -u db_master_user -p
   ```

   *host_name*은 Amazon RDS 데이터베이스 엔드포인트입니다.
7. `mysql` 프롬프트에서 `source` 명령을 실행하고 데이터베이스 덤프 파일의 이름을 전달합니다. 이 명령은 데이터를 Amazon RDS DB 인스턴스로 로드합니다.

   - SQL 형식인 경우 다음 명령을 사용합니다.

     ```
     mysql> source backup.sql;
     ```

   - 구분 기호로 분리된 텍스트 형식의 경우 Amazon RDS 데이터베이스를 설정할 때 만든 기본 데이터베이스가 아니라면 먼저 데이터베이스를 만듭니다.

     ```
     mysql> create database database_name;
     mysql> use database_name;
     ```

     그런 다음, 테이블을 생성합니다.

     ```
     mysql> source table1.sql
     mysql> source table2.sql
     etc...
     ```

     그런 다음, 데이터를 가져옵니다.

```
mysql> LOAD DATA LOCAL INFILE 'table1.txt' INTO TABLE table1 FIELDS TERMINATED BY
 ',' ENCLOSED BY '"' LINES TERMINATED BY '0x0d0a';
mysql> LOAD DATA LOCAL INFILE 'table2.txt' INTO TABLE table2 FIELDS TERMINATED BY
 ',' ENCLOSED BY '"' LINES TERMINATED BY '0x0d0a';
etc...
```

성능을 개선하려면 여러 연결에서 이들 작업을 병렬로 수행하여 모든 테이블이 만들어진 다음에 동시에 로드되도록 할 수 있습니다.

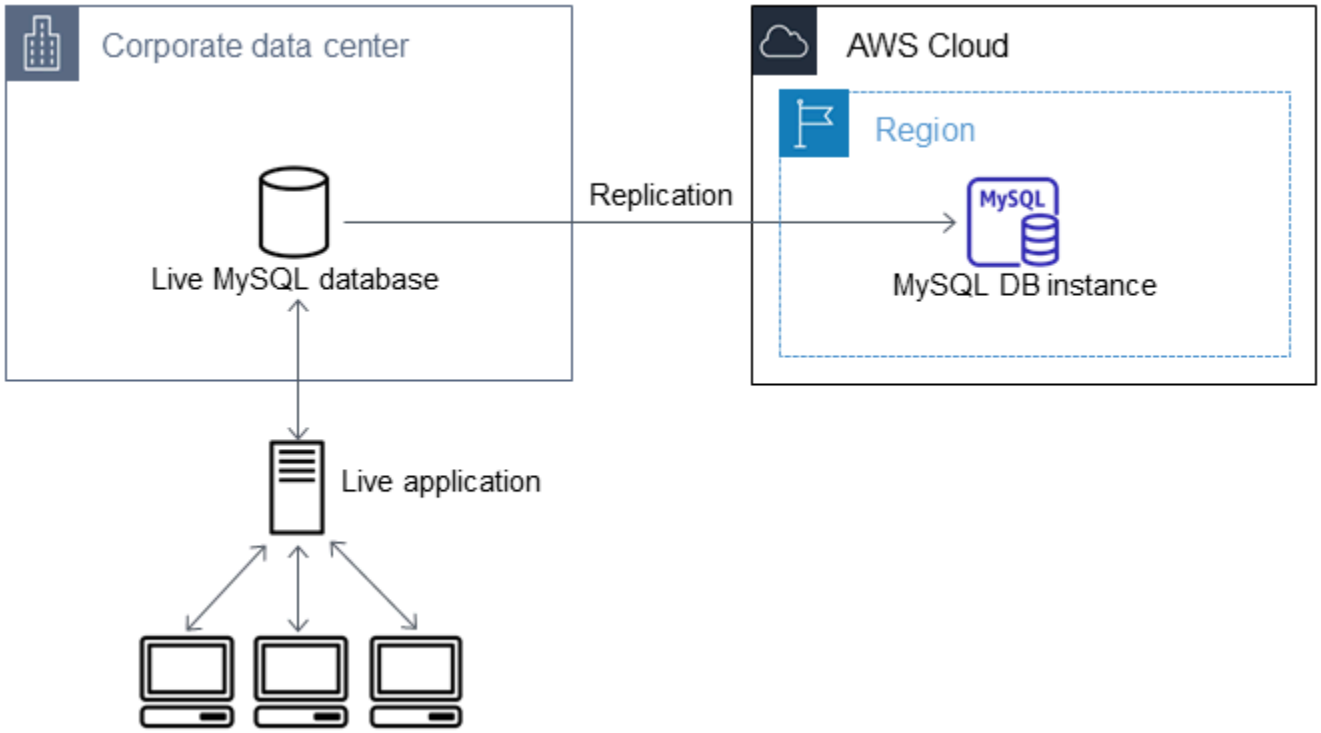
> **Note**
>
> 처음에 테이블을 덤프할 때 `mysqldump`와 함께 데이터 서식 옵션을 사용했다면, `LOAD DATA LOCAL INFILE`과 같은 옵션을 사용하여 데이터 파일 내용을 올바로 해석해야 합니다.

8. 가져온 데이터베이스에 있는 테이블 중 1개 또는 2개에 대해 간단한 `SELECT` 쿼리를 실행하여 가져오기에 성공했는지 확인합니다.

이 절차에 사용되는 Amazon EC2 인스턴스가 더 이상 필요하지 않은 경우 EC2 인스턴스를 종료하여 AWS 리소스 사용량을 줄여야 합니다. EC2 인스턴스를 종료하려면 Amazon Elastic Compute Cloud 사용 설명서의 Terminate an instance를 참조하세요.

작업 4: 외부 데이터베이스에서 새 Amazon RDS 데이터베이스로 데이터 복제

MariaDB 또는 MySQL 데이터베이스로 데이터를 복사하고 전송하는 데 걸린 시간 중에 원본 데이터베이스가 업데이트되었을 것입니다. 따라서 복제를 사용하여 복사된 데이터베이스를 원본 데이터베이스에 맞춰 최신 상태로 업데이트합니다.



Amazon RDS 데이터베이스에서 복제를 시작하는 데 필요한 권한은 제한되고 Amazon RDS 마스터 사용자는 사용할 수 없습니다. 따라서 메이저 엔진 버전에 적합한 다음의 Amazon RDS 저장 프로시저를 사용합니다.

- mysql.rds_set_external_master(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하)
- mysql.rds_set_external_source(RDS for MySQL 메이저 버전 8.4 이상)
- 복제를 구성하려면 mysql.rds_set_external_master_gtid, 복제를 시작하려면 mysql.rds_start_replication

복제를 시작하려면

작업 1에서 복제 옵션을 설정할 때 이진 로깅을 활성화하고 소스 데이터베이스의 고유한 서버 ID를 설정합니다. 이제 라이브 데이터베이스를 소스 복제 인스턴스로 포함하고 있는 복제본으로서 Amazon RDS 데이터베이스를 설정할 수 있습니다.

1. Amazon RDS 콘솔에서 Amazon RDS 데이터베이스에 대한 VPC 보안 그룹에 원본 데이터베이스를 호스팅하는 서버의 IP 주소를 추가합니다. VPC 보안 그룹 구성에 관한 자세한 내용은 Amazon Virtual Private Cloud 사용 설명서의 Configure security group rules를 참조하세요.

원본 인스턴스와 통신할 수 있도록, Amazon RDS 데이터베이스 IP 주소에서의 연결을 허용하도록 로컬 네트워크를 구성해야 할 수도 있습니다. Amazon RDS 데이터베이스의 IP 주소를 확인하려면 host 명령을 사용합니다.

```
host host_name
```

*host_name*은 Amazon RDS 데이터베이스 엔드포인트의 DNS 이름입니다(예: myinstance.123456789012.us-east-1.rds.amazonaws.com). Amazon RDS 콘솔의 인스턴스 세부 정보에서 엔드포인트 값을 찾을 수 있습니다.

2. 선택한 클라이언트를 사용하여 원본 인스턴스에 연결하고 복제에 사용될 사용자를 만듭니다. 이 계정은 오직 복제용으로만 사용되며 보안 향상을 위해 사용자의 도메인으로 제한되어야 합니다. 다음 명령은 예제입니다.

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 보안 인증 정보를 지정하는 것이 좋습니다.

3. 원본 인스턴스의 경우 복제 사용자에게 REPLICATION CLIENT 및 REPLICATION SLAVE 권한을 부여합니다. 예를 들어 도메인의 'repl_user' 사용자를 위해 모든 데이터베이스에서 REPLICATION CLIENT 및 REPLICATION SLAVE 권한을 부여하려면 다음 명령을 실행합니다.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
```

4. SQL 형식을 사용하여 백업 파일을 만들었고 외부 인스턴스가 MariaDB 10.0.24 이상이 아닌 경우 다음 명령을 실행하여 해당 파일의 내용을 살펴봅니다.

```
cat backup.sql
```

이 파일에는 마스터 로그 파일 이름과 위치를 포함한 CHANGE MASTER TO 설명이 포함됩니다. 이 설명은 --master-data와 함께 mysqldump 옵션을 사용할 때 백업 파일에 포함됩니다. MASTER_LOG_FILE 및 MASTER_LOG_POS에 대한 값을 확인합니다.

```
--
-- Position to start replication or point-in-time recovery from
```

```
--

-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin-changelog.000031', MASTER_LOG_POS=107;
```

구분 기호로 분리된 텍스트 형식을 사용하여 백업 파일을 만들었고 외부 인스턴스가 MariaDB 10.0.24 이상이 아닌 경우, 작업 1의 '존재하는 데이터베이스의 백업 복사본 만들기' 절차 1단계에서 확보한 바이너리 로그 좌표가 이미 있을 것입니다.

외부 인스턴스가 MariaDB 10.0.24 이상인 경우 복제를 시작할 수 있는 GTID를 작업 1의 '존재하는 데이터베이스의 백업 복사본 만들기' 절차 2단계에서 이미 확보했을 것입니다.

5. Amazon RDS 데이터베이스를 복제본으로 만듭니다. 외부 인스턴스가 MariaDB 10.0.24 이상이 아닌 경우 마스터 사용자로 Amazon RDS 데이터베이스에 연결하고 적절한 Amazon RDS 저장 프로시저를 사용하여 소스 데이터베이스를 소스 복제 인스턴스로 식별합니다.

   - mysql.rds_set_external_master(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하)
   - mysql.rds_set_external_source(RDS for MySQL 메이저 버전 8.4 이상)

   SQL 형식 백업 파일이 있는 경우 4단계에서 확인한 마스터 로그 파일 이름과 마스터 로그 위치를 사용합니다. 구분 기호로 분리된 텍스트 형식을 사용한 경우 백업 파일을 만들 때 확인한 이름과 위치를 사용합니다. 다음 명령은 예시입니다.

   MySQL 8.4 이상 버전

   ```
   CALL mysql.rds_set_external_source ('myserver.mydomain.com', 3306,
       'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 1);
   ```

   MariaDB 및 MySQL 8.0 이하 버전

   ```
   CALL mysql.rds_set_external_master ('myserver.mydomain.com', 3306,
       'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 1);
   ```

   > **Note**
   >
   > 보안 모범 사례로 여기에 표시된 프롬프트 이외의 보안 인증 정보를 지정하는 것이 좋습니다.

   외부 인스턴스가 MariaDB 10.0.24 이상인 경우 마스터 사용자로 Amazon RDS 데이터베이스에 연결하고 mysql.rds_set_external_master_gtid 명령을 사용하여 소스 데이터베이스를 소스 복제 인

스턴스로 식별합니다. 작업 1의 '기존 데이터베이스의 백업 복사본 만들기' 절차 2단계에서 확인된 GTID를 사용합니다. 다음 명령은 예제입니다.

```
CALL mysql.rds_set_external_master_gtid ('source_server_ip_address', 3306,
 'ReplicationUser', 'password', 'GTID', 1);
```

*source_server_ip_address*는 소스 복제 인스턴스의 IP 주소입니다. EC2 프라이빗 DNS 주소는 현재 지원되지 않습니다.

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 보안 인증 정보를 지정하는 것이 좋습니다.

6. Amazon RDS 데이터베이스에서 다음 mysql.rds_start_replication 명령을 실행하여 복제를 시작합니다.

```
CALL mysql.rds_start_replication;
```

7. Amazon RDS 데이터베이스에서 SHOW REPLICA STATUS 명령을 실행하여 복제본이 소스 복제 인스턴스에 맞게 최신 상태가 되는 시점을 파악합니다. `SHOW REPLICA STATUS` 명령의 결과에는 `Seconds_Behind_Master` 필드가 포함됩니다. `Seconds_Behind_Master` 필드가 0을 반환하면 복제본이 소스 복제 인스턴스에 맞게 최신 상태인 것입니다.

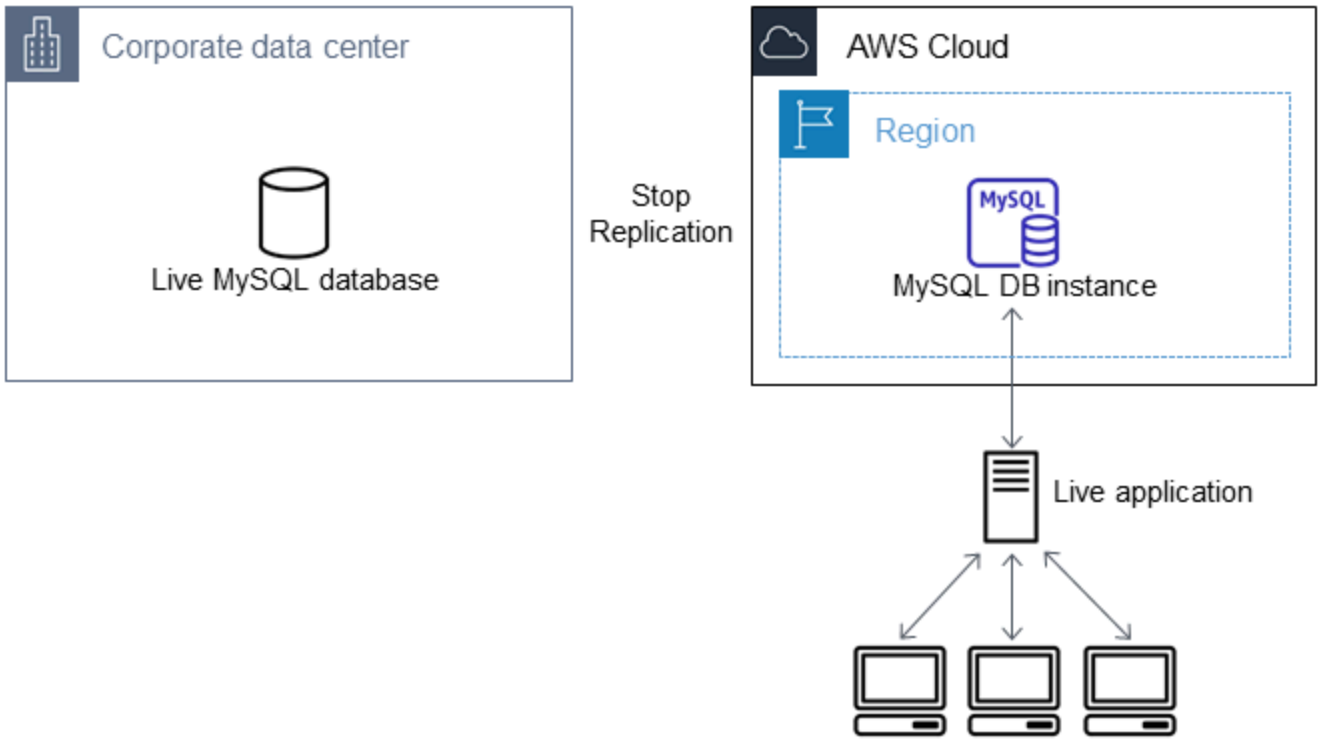
> **Note**
>
> 이전 버전의 MySQL에는 `SHOW SLAVE STATUS` 대신 `SHOW REPLICA STATUS`가 사용되었습니다. 8.0.23 이전 MySQL 버전을 사용하는 경우 `SHOW SLAVE STATUS`를 사용합니다.

MariaDB 10.5, 10.6 또는 10.11 DB 인스턴스의 경우 MySQL 명령 대신 mysql.rds_replica_status 저장 프로시저를 실행합니다.

8. Amazon RDS 데이터베이스가 최신 상태로 업데이트된 후, 필요할 경우 그 데이터베이스를 복원할 수 있도록 자동 백업을 켭니다. Amazon RDS 콘솔을 사용하여 Amazon RDS 데이터베이스에 대한 자동 백업을 켜거나 수정할 수 있습니다. 자세한 내용은 백업 소개 섹션을 참조하세요.

작업 5: 라이브 애플리케이션을 Amazon RDS 인스턴스로 리디렉션

MariaDB 또는 MySQL 데이터베이스가 소스 복제 인스턴스에 맞게 최신 상태로 업데이트되면 라이브 애플리케이션을 업데이트하여 Amazon RDS 인스턴스를 사용할 수 있습니다.



라이브 애플리케이션을 MariaDB 또는 MySQL 데이터베이스로 리디렉션하고 복제 중지

1. Amazon RDS 데이터베이스에 대한 VPC 보안 그룹을 추가하려면 애플리케이션을 호스팅하는 서버의 IP 주소를 추가합니다. VPC 보안 그룹 수정에 대한 자세한 내용은 Amazon Virtual Private Cloud 사용 설명서의 Configure security group rules를 참조하세요.
2. SHOW REPLICA STATUS 명령 결과에서 `Seconds_Behind_Master` 필드가 0인지 확인합니다. 이는 복제본이 소스 복제 인스턴스에 맞게 최신 상태라는 의미입니다.

```
SHOW REPLICA STATUS;
```

> **Note**
>
> 이전 버전의 MySQL에는 `SHOW SLAVE STATUS` 대신 `SHOW REPLICA STATUS`가 사용되었습니다. 8.0.23 이전 MySQL 버전을 사용하는 경우 `SHOW SLAVE STATUS`를 사용합니다.

MariaDB 10.5, 10.6, 10.11 DB 인스턴스의 경우 MySQL 대신 mysql.rds_replica_status 프로시저를 실행합니다.

3. 트랜잭션이 완료되면 소스에 대한 모든 연결을 닫습니다.
4. Amazon RDS 데이터베이스를 사용할 수 있도록 애플리케이션을 업데이트합니다. 이 업데이트에는 일반적으로 Amazon RDS 데이터베이스의 호스트 이름과 포트, 연결할 사용자 계정과 암호, 사용할 데이터베이스를 식별하기 위해 연결 설정을 변경하는 과정이 포함됩니다.
5. DB 인스턴스에 연결합니다.

   다중 AZ DB 클러스터의 경우에는 라이터 DB 인스턴스에 연결합니다.
6. 다음 mysql.rds_stop_replication 명령을 사용하여 Amazon RDS 인스턴스에 대한 복제를 중지합니다.

```
CALL mysql.rds_stop_replication;
```

7. 이 인스턴스가 더 이상 복제본으로 식별되지 않도록 Amazon RDS 데이터베이스에서 적절한 Amazon RDS 저장 프로시저를 실행하여 복제 구성을 재설정합니다.
   - mysql.rds_reset_external_master(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하)
   - mysql.rds_reset_external_source(RDS for MySQL 메이저 버전 8.4 이상)

   MySQL 8.4 이상 버전

```
CALL mysql.rds_reset_external_source;
```

   MariaDB 및 MySQL 8.0 이하 버전

```
CALL mysql.rds_reset_external_master;
```

8. 다중 AZ 지원 및 읽기 전용 복제본과 같은 Amazon RDS 기능을 추가로 활성화합니다. 자세한 내용은 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 및 DB 인스턴스 읽기 전용 복제본 작업 단원을 참조하세요.

원하는 소스에서 Amazon RDS for MariaDB DB 인스턴스로 데이터 가져오기

다음 단계에서는 Amazon RDS DB 인스턴스로 데이터를 가져오는 방법을 안내합니다.

1단계: 로드할 데이터를 포함한 플랫 파일 만들기

쉼표로 구분된 값(CSV)과 같은 일반적인 형식을 사용하여 로드할 데이터를 저장합니다. 각 테이블에는 자체 파일이 있어야 하며, 여러 테이블에 대한 데이터를 같은 파일에 결합할 수 없습니다. 각 파일의 이름은 그에 상응하는 테이블과 같은 이름으로 지정합니다. 파일 확장명은 원하는 대로 정할 수 있습니다. 예를 들어, 테이블 이름이 `sales`인 경우 파일 이름은 `sales.csv` 또는 `sales.txt`일 수 있습니다.

가능하면 로드되는 테이블의 프라이머리 키를 기준으로 데이터를 정렬하세요. 그러면 로드 시간이 대폭 단축되고 데이터 스토리지 요구 사항이 최소화됩니다.

이 절차의 수행 속도와 효율성은 파일의 크기를 작게 유지하느냐에 좌우됩니다. 개별 파일의 압축되지 않은 크기가 1GiB보다 큰 경우에는 파일을 여러 개의 파일로 분할하고 각각 하나씩 따로 로드합니다.

Unix와 같은 시스템(Linux 포함)에서는 `split` 명령을 사용합니다. 예를 들어 다음 명령을 실행하면 `sales.csv` 파일이 1GiB 미만의 여러 파일로 분할되며, 줄 바꿈에서만 분할됩니다(`-C 1024m`). 새 파일의 이름에는 오름차순 숫자 접미사가 포함됩니다. 다음 명령은 `sales.part_00` 및 `sales.part_01`과 같은 이름의 파일을 생성합니다.

```
split -C 1024m -d sales.csv sales.part_
```

다른 운영 체제에서는 이와 유사한 유틸리티를 사용할 수 있습니다.

플랫 파일은 어디에나 저장할 수 있습니다. 그러나 5단계에서 데이터를 로드할 때는 파일이 있는 위치와 동일한 위치에서 `mysql` 쉘을 간접적으로 호출하거나 `LOAD DATA LOCAL INFILE`을 실행할 때 파일의 절대 경로를 사용해야 합니다.

2단계: 대상 DB 인스턴스에 액세스 중인 애플리케이션을 모두 중지

대량 로드를 시작하기 전에 로드하려는 대상 DB 인스턴스에 액세스하는 모든 애플리케이션의 활동을 중단합니다. 로드 중인 테이블 혹은 참조 테이블을 다른 세션이 수정하는 경우에는 더욱 더 중단해야 합니다. 그러면 로드 중에 발생하는 제약 조건 위반의 위험이 줄어들고 로드 성능이 향상됩니다. 또한 로드에 관여하지 않는 프로세스에 의한 변경 내용이 손실되지 않고 로드 직전의 시점으로 DB 인스턴스를 복원할 수 있습니다.

물론, 이것이 가능하지 않거나 유용하지 않을 수도 있습니다. 로드 전에 애플리케이션이 DB 인스턴스에 액세스하지 못하게 막을 수 없다면, 데이터의 가용성과 무결성을 보장하기 위한 일련의 단계를 수행하세요. 필요한 구체적인 단계는 특정 사용 사례와 사이트 요구 사항에 따라 크게 달라집니다.

3단계: DB 스냅샷 만들기

아무런 데이터도 없는 새 DB 인스턴스로 데이터를 로드할 경우에는 이 단계를 건너뛰어도 됩니다. 아니면 데이터 로드 전후에 대상 Amazon RDS DB 인스턴스의 DB 스냅샷을 만드는 것이 좋습니다. Amazon RDS DB 스냅샷은 DB 인스턴스를 알려진 상태로 복원하는 데 사용할 수 있는 DB 인스턴스의 완전한 백업입니다. DB 스냅샷을 시작하면 데이터베이스가 백업되는 동안 DB 인스턴스에 대한 I/O 작업이 일시 중단됩니다.

필요한 경우 로드 직전에 DB 스냅샷을 만들면 데이터베이스를 로드 이전의 상태로 복원할 수 있습니다. 로드 직후에 생성된 DB 스냅샷은 사고 발생 시 데이터를 다시 로드하지 않아도 됩니다. 또한 로드 후 DB 스냅샷을 사용하여 데이터를 새 데이터베이스 인스턴스로 가져올 수 있습니다.

아래 예시에서는 AWS CLI create-db-snapshot 명령을 실행하여 `AcmeRDS` 인스턴스의 DB 스냅샷을 만들고 DB 스냅샷에 `"preload"` 식별자를 지정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-snapshot \
    --db-instance-identifier AcmeRDS \
    --db-snapshot-identifier preload
```

Windows의 경우:

```
aws rds create-db-snapshot ^
    --db-instance-identifier AcmeRDS ^
    --db-snapshot-identifier preload
```

DB 스냅샷 기능에서 복원하는 기능을 사용하여 테스트 실행을 위한 테스트 DB 인스턴스를 만들거나 로드 중의 변경 사항을 실행 취소할 수도 있습니다.

DB 스냅샷에서 데이터베이스를 복원하면 모든 DB 인스턴스와 마찬가지로 고유 식별자와 엔드포인트가 있는 새 DB 인스턴스가 생성된다는 점을 꼭 명심해야 합니다. 엔드포인트를 변경하지 않고 DB 인스턴스를 복원해야 한다면, 우선 엔드포인트를 다시 사용할 수 있도록 DB 인스턴스를 삭제해야 합니다.

예를 들어 테스트 실행 또는 다른 테스트를 위한 DB 인스턴스를 만들려면 DB 인스턴스에 고유 식별자를 지정합니다. 예제에서 `AcmeRDS-2"`는 식별자입니다. 이 예제는 `AcmeRDS-2`와 연결된 엔드포인트를 사용하여 DB 인스턴스에 연결합니다. 자세한 정보는 restore-db-instance-from-db-snapshot을 참조하세요.

대상 LinuxmacOS, 또는Unix:

```
aws rds restore-db-instance-from-db-snapshot \
    --db-instance-identifier AcmeRDS-2 \
    --db-snapshot-identifier preload
```

Windows의 경우:

```
aws rds restore-db-instance-from-db-snapshot ^
    --db-instance-identifier AcmeRDS-2 ^
    --db-snapshot-identifier preload
```

기존 엔드포인트를 재사용하려면 우선 DB 인스턴스를 삭제한 다음, 복원된 데이터베이스에 같은 식별자를 지정해야 합니다. 자세한 내용은 delete-db-instance를 참조하세요.

다음 예제에서는 또한 DB 인스턴스의 최종 DB 스냅샷을 생성한 후에 삭제합니다. 이 단계는 선택 사항이지만 권장됩니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds delete-db-instance \
    --db-instance-identifier AcmeRDS \
    --final-db-snapshot-identifier AcmeRDS-Final

aws rds restore-db-instance-from-db-snapshot \
    --db-instance-identifier AcmeRDS \
    --db-snapshot-identifier preload
```

Windows의 경우:

```
aws rds delete-db-instance ^
    --db-instance-identifier AcmeRDS ^
    --final-db-snapshot-identifier AcmeRDS-Final

aws rds restore-db-instance-from-db-snapshot ^
    --db-instance-identifier AcmeRDS ^
    --db-snapshot-identifier preload
```

4단계(선택 사항): Amazon RDS 자동 백업 끄기

> **Warning**
>
> 시점 복구를 수행할 필요가 있는 경우에는 자동 백업을 끄지 마세요.

자동 백업을 끄는 것은 성능 최적화 수단이며 데이터 로드에 필수적이지는 않습니다. 자동 백업을 끄면 기존 백업이 모두 지워집니다. 따라서 자동 백업을 끈 후에는 시점 복구를 수행할 수 없습니다. 자동 백업을 비활성화해도 수동 DB 스냅샷에는 영향을 주지 않습니다. 기존의 모든 수동 DB 스냅샷을 계속 복원 작업에 사용할 수 있습니다.

자동 백업을 비활성화하면 로드 시간이 약 25% 단축되고 로드 중에 필요한 스토리지 공간의 양이 줄어듭니다. 아무런 데이터도 없는 새 DB 인스턴스로 데이터를 로드할 경우에는 백업을 비활성화하는 것이 로드 속도를 높이고 백업에 필요한 추가 스토리지의 사용을 피하는 손쉬운 방법입니다. 하지만 이미 데이터가 있는 DB 인스턴스로 로드할 경우도 있습니다. 그렇다면 특정 시점으로 복구를 수행할 능력을 상실하는 데 따른 영향에 비해 백업을 해제할 때의 이점을 비교합니다.

DB 인스턴스에서는 기본적으로 자동 백업이 활성화되어 있습니다(보존 기간은 하루). 자동 백업을 비활성화하기 위해 백업 보존 기간을 0으로 설정합니다. 로드 후에는 백업 보존 기간을 0이 아닌 값으로 설정하여 백업을 다시 활성화할 수 있습니다. 백업을 켜거나 끄기 위해 Amazon RDS가 DB 인스턴스를 종료했다가 다시 시작하여 MariaDB 또는 MySQL 로깅을 켜거나 끕니다.

AWS CLI `modify-db-instance` 명령을 실행하여 백업 보존 기간을 0으로 설정하고 변경 사항을 즉시 적용합니다. 보존 기간을 0으로 설정하려면 DB 인스턴스를 다시 시작해야 하므로, 다시 시작 프로세스가 완료될 때까지 기다린 후 계속 진행합니다. 자세한 내용은 modify-db-instance를 참조하세요.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier AcmeRDS \
    --apply-immediately \
    --backup-retention-period 0
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier AcmeRDS ^
    --apply-immediately ^
```

```
    --backup-retention-period 0
```

AWS CLI describe-db-instances 명령으로 DB 인스턴스의 상태를 확인할 수 있습니다. 다음 예제에서는 AcmeRDS DB 인스턴스의 DB 인스턴스 상태를 표시합니다.

```
aws rds describe-db-instances --db-instance-identifier AcmeRDS --query "*[].
{DBInstanceStatus:DBInstanceStatus}"
```

DB 인스턴스 상태가 available이면 다음 단계로 계속할 준비가 된 것입니다.

5단계: 데이터 로드

플랫 파일의 행을 데이터베이스 테이블로 읽어들이려면 MySQL LOAD DATA LOCAL INFILE 문을 사용합니다.

> **Note**
>
> 플랫 파일이 있는 위치와 동일한 위치에서 mysql 쉘을 호출하거나 LOAD DATA LOCAL INFILE을 실행할 때 파일의 절대 경로를 사용해야 합니다.

다음 예시는 이름이 sales.txt인 파일에서 데이터베이스의 Sales 테이블로 데이터를 로드하는 방법을 보여줍니다.

```
mysql> LOAD DATA LOCAL INFILE 'sales.txt' INTO TABLE Sales FIELDS TERMINATED BY ' '
 ENCLOSED BY '' ESCAPED BY '\\';
Query OK, 1 row affected (0.01 sec)
Records: 1  Deleted: 0  Skipped: 0  Warnings: 0
```

LOAD DATA 문에 대한 자세한 내용은 MySQL 설명서의 LOAD DATA Statement를 참조하세요.

6단계: Amazon RDS 자동 백업 다시 켜기

4단계에서 Amazon RDS 자동 백업을 껐다면 로드가 완료된 후 백업 보존 기간을 로드 이전의 값으로 다시 설정하여 자동 백업을 켭니다. 4단계에서 언급한 것처럼, Amazon RDS가 DB 인스턴스를 다시 시작하므로 잠시 작동이 중단되는 상황에 대비하세요.

다음 예시에서는 AWS CLI modify-db-instance 명령을 실행하여 AcmeRDS DB 인스턴스에서 자동 백업을 켜고 보존 기간을 1일로 설정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier AcmeRDS \
    --backup-retention-period 1 \
    --apply-immediately
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier AcmeRDS ^
    --backup-retention-period 1 ^
    --apply-immediately
```

Amazon RDS에서 MariaDB 복제 작업

일반적으로 읽기 전용 복제본을 사용하여 Amazon RDS DB 인스턴스 간 복제를 구성합니다. 읽기 전용 복제본에 대한 일반적인 정보는 DB 인스턴스 읽기 전용 복제본 작업 단원을 참조하십시오. Amazon RDS for MariaDB의 읽기 복제본 작업에 대한 자세한 내용은 MariaDB 읽기 전용 복제본 작업 단원을 참조하세요.

또한 MariaDB DB 인스턴스에 대해 바이너리 로그 좌표를 기반으로 복제를 구성할 수 있습니다. MariaDB 인스턴스의 경우에는 전역 트랜잭션 ID(GTID)를 기반으로 복제를 구성할 수도 있습니다. 그러면 충돌 안정성이 개선됩니다. 자세한 내용은 외부 소스 인스턴스를 사용하여 GTID 기반 복제 구성 단원을 참조하십시오.

다음은 RDS for MariaDB에서 사용 가능한 다른 복제 옵션입니다.

- RDS for MySQL 또는 MariaDB DB 인스턴스와 Amazon RDS 외부에 있는 MariaDB 인스턴스 간의 복제를 설정할 수 있습니다. 외부 소스를 사용하여 복제를 구성하는 방법에 대한 자세한 내용은 외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성 단원을 참조하십시오.
- Amazon RDS 외부에 있는 MySQL 또는 MariaDB 인스턴스에서 데이터베이스를 가져오거나 그런 인스턴스로 데이터베이스를 내보내도록 복제를 구성할 수 있습니다. 자세한 내용은 가동 중지 시간을 줄이면서 Amazon RDS for MariaDB DB 인스턴스로 데이터 가져오기 및 복제를 사용하여 MySQL DB 인스턴스에서 데이터 내보내기 단원을 참조하십시오.

이러한 복제 옵션의 경우 행 기반 복제, 문 기반 복제 또는 혼합 복제를 사용할 수 있습니다. 행 기반 복제는 SQL 문으로 인해 변경된 행만 복제합니다. 문 기반 복제는 전체 SQL 문을 복제합니다. 혼합 복제는 가능한 경우 문 기반 복제를 사용하지만, 문 기반 복제에 안전하지 않은 SQL 문이 실행될 경우 행 기반 복제로 전환합니다. 대부분의 경우 혼합 복제가 권장됩니다. DB 인스턴스의 이진 로그 형식은 복제가 행 기반인지, 문 기반인지, 혼합인지 결정합니다. 이진 로그 형식 설정에 대한 자세한 내용은 MariaDB 바이너리 로깅 구성 단원을 참조하십시오.

MariaDB 버전 간의 복제 호환성에 대한 자세한 내용은 MariaDB 설명서의 Replication Compatibility를 참조하세요.

주제

- MariaDB 읽기 전용 복제본 작업
- 외부 소스 인스턴스를 사용하여 GTID 기반 복제 구성
- 외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성

MariaDB 읽기 전용 복제본 작업

다음으로, Amazon RDS for MariaDB에서의 읽기 복제본 작업에 대한 특정 정보를 찾을 수 있습니다. 읽기 전용 복제본에 대한 일반적인 정보와 사용 지침은 DB 인스턴스 읽기 전용 복제본 작업 단원을 참조하십시오.

- MariaDB를 사용한 복제 필터 구성
- MariaDB를 사용한 지연 복제 구성
- MariaDB를 사용한 읽기 전용 복제본 업데이트
- MariaDB를 사용한 다중 AZ 읽기 전용 복제본 배포 작업
- RDS for MariaDB에서의 계단식 읽기 전용 복제본 사용
- MariaDB 읽기 전용 복제본 모니터링
- MariaDB 읽기 전용 복제본을 사용한 복제 시작 및 중지
- MariaDB 읽기 전용 복제본의 문제 해결

MariaDB를 사용한 읽기 전용 복제본 구성

MariaDB DB 인스턴스가 복제 소스 역할을 하기 전에 백업 보존 기간을 0이 아닌 값으로 설정하여 소스 DB 인스턴스에서 자동 백업을 켜야 합니다. 이 요구 사항은 다른 읽기 전용 복제본의 원본 DB 인스턴스인 읽기 전용 복제본에도 적용됩니다.

동일 리전 내의 DB 인스턴스 하나에서 최대 15개까지 읽기 전용 복제본을 생성할 수 있습니다. 효과적인 복제를 위해서는 읽기 전용 복제본도 각각 원본 DB 인스턴스와 동일한 양의 컴퓨팅 및 스토리지 리소스를 가져야 합니다. 원본 DB 인스턴스를 확장하는 경우 읽기 전용 복제본도 확장합니다.

RDS for MariaDB는 계단식 읽기 전용 복제본을 지원합니다. 읽기 전용 복제본을 계단식으로 구성하는 방법을 알아보려면 RDS for MariaDB에서의 계단식 읽기 전용 복제본 사용 단원을 참조하세요.

동일한 원본 DB 인스턴스를 참조하는 여러 읽기 전용 복제본 생성 및 삭제 작업을 동시에 실행할 수 있습니다. 이러한 작업을 수행할 때 각 원본 인스턴스의 읽기 전용 복제본 한도 15개를 넘지 않아야 합니다.

MariaDB를 사용한 복제 필터 구성

복제 필터를 사용하여 읽기 복제본과 함께 복제할 데이터베이스와 테이블을 지정할 수 있습니다. 복제 필터는 데이터베이스와 테이블을 복제에 포함하거나 복제에서 제외할 수 있습니다.

다음은 복제 필터의 몇 가지 사용 사례입니다.

- 읽기 복제본의 크기를 줄이는 방법. 복제 필터링을 사용하면 읽기 복제본에 필요하지 않은 데이터베이스와 테이블을 제외할 수 있습니다.
- 보안상의 이유로 읽기 복제본에서 데이터베이스와 테이블을 제외하는 방법.
- 다른 읽기 복제본에서 특정 사용 사례에 대해 서로 다른 데이터베이스와 테이블을 복제하는 방법. 예를 들어 분석 또는 샤딩에 특정 읽기 복제본을 사용할 수 있습니다.
- 여러 AWS 리전에 읽기 전용 복제본이 있는 DB 인스턴스의 경우 서로 다른 AWS 리전에 있는 다른 데이터베이스 또는 테이블을 복제합니다.

> **Note**
>
> 복제 필터를 사용하여 인바운드 복제 토폴로지에서 복제본으로 구성된 기본 MariaDB DB 인스턴스로 복제할 데이터베이스 및 테이블을 지정할 수도 있습니다. 이 구성에 대한 자세한 내용은 외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성 단원을 참조하십시오.

주제

- Amazon RDS for MariaDB에 대한 복제 필터링 파라미터 설정
- RDS for MariaDB에 대한 복제 필터링 제한 사항
- RDS for MariaDB에 대한 복제 필터링 예제
- 읽기 복제본에 대한 복제 필터 보기

Amazon RDS for MariaDB에 대한 복제 필터링 파라미터 설정

복제 필터를 구성하려면 읽기 복제본에서 다음 복제 필터링 파라미터를 설정합니다.

- `replicate-do-db` – 지정된 데이터베이스에 변경 내용을 복제합니다. 읽기 복제본에 대해 이 파라미터를 설정하면 파라미터에 지정된 데이터베이스만 복제됩니다.
- `replicate-ignore-db` – 지정된 데이터베이스에 변경 내용을 복제하지 마세요. 읽기 복제본에 대해 `replicate-do-db` 파라미터가 설정되면 이 파라미터는 평가되지 않습니다.
- `replicate-do-table` – 지정된 테이블에 변경 사항을 복제합니다. 읽기 복제본에 대해 이 파라미터를 설정하면 파라미터에 지정된 테이블만 복제됩니다. 또한 `replicate-do-db` 또는 `replicate-ignore-db` 파라미터가 설정되면 지정된 테이블을 포함하는 데이터베이스가 읽기 복제본의 복제에 포함되어야 합니다.

- `replicate-ignore-table` – 지정된 테이블에 변경 내용을 복제하지 마세요. 읽기 복제본에 대해 `replicate-do-table` 파라미터가 설정되면 이 파라미터는 평가되지 않습니다.
- `replicate-wild-do-table` – 지정된 데이터베이스와 테이블 이름 패턴을 기반으로 테이블을 복제합니다. `%` 및 `_` 와일드카드 문자가 지원됩니다. `replicate-do-db` 또는 `replicate-ignore-db` 파라미터가 설정되면 복제에 지정된 테이블이 포함된 데이터베이스를 읽기 복제본과 함께 포함해야 합니다.
- `replicate-wild-ignore-table` – 지정된 데이터베이스와 테이블 이름 패턴을 기반으로 테이블을 복제하지 마세요 `%` 및 `_` 와일드카드 문자가 지원됩니다. 읽기 복제본에 대해 `replicate-do-table` 또는 `replicate-wild-do-table` 파라미터가 설정되면 이 파라미터는 평가되지 않습니다.

파라미터는 나열된 순서대로 평가됩니다. 이러한 파라미터의 작동 방식에 대한 자세한 내용은 MariaDB 설명서를 참조하세요.

기본적으로 이러한 각 파라미터에는 빈 값이 있습니다. 각 읽기 복제본에서 이러한 파라미터를 사용하여 복제 필터를 설정, 변경 및 삭제할 수 있습니다. 이러한 파라미터 중 하나를 설정할 때 각 필터를 쉼표로 구분합니다.

`%` 및 `_` 파라미터에 `replicate-wild-do-table` 및 `replicate-wild-ignore-table` 와일드카드 문자를 사용할 수 있습니다. `%` 와일드카드는 원하는 수의 문자를 찾으며 `_` 와일드카드는 한 문자만 찾습니다.

원본 DB 인스턴스의 이진 로깅 형식은 데이터 변경 기록을 결정하므로 복제에 중요합니다. `binlog_format` 파라미터 설정에 따라 복제가 행 기반인지 또는 문 기반인지가 결정됩니다. 자세한 내용은 MariaDB 바이너리 로깅 구성 섹션을 참조하세요.

> **Note**
>
> 원본 DB 인스턴스의 `binlog_format` 설정과 관계없이, 모든 DDL(데이터 정의어) 문은 문으로 복제됩니다.

RDS for MariaDB에 대한 복제 필터링 제한 사항

RDS for MariaDB에 대한 복제 필터링에 다음과 같은 제한 사항이 적용됩니다.

- 각 복제 필터링 파라미터에는 2,000자 제한이 있습니다.
- 복제 필터에서는 쉼표가 지원되지 않습니다.

- 이진 로그 필터링에 대해서는 MariaDB `binlog_do_db` 및 `binlog_ignore_db` 옵션이 지원되지 않습니다.
- 복제 필터링은 XA 트랜잭션을 지원하지 않습니다.

  자세한 내용은 MySQL 설명서에서 XA 트랜잭션에 대한 제한 사항을 참조하세요.
- RDS for MariaDB 버전 10.2에 대해서는 복제 필터링이 지원되지 않습니다.

RDS for MariaDB에 대한 복제 필터링 예제

읽기 복제본에 대한 복제 필터링을 구성하려면 읽기 복제본과 연결된 파라미터 그룹에서 복제 필터링 파라미터를 수정합니다.

> **Note**
>
> 기본 파라미터 그룹을 수정할 수 없습니다. 읽기 복제본에서 기본 파라미터 그룹을 사용 중인 경우 새 파라미터 그룹을 생성하여 읽기 복제본과 연결합니다. DB 파라미터 그룹에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 파라미터 그룹에서 파라미터를 설정할 수 있습니다. 파라미터 설정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요. 파라미터 그룹에서 파라미터를 설정하면 파라미터 그룹과 연결된 모든 DB 인스턴스가 파라미터 설정을 사용합니다. 파라미터 그룹에서 복제 필터링 파라미터를 설정하는 경우, 파라미터 그룹이 읽기 복제본에만 연결되어 있는지 확인합니다. 원본 DB 인스턴스에 대해 복제 필터링 파라미터를 비워 둡니다.

다음 예제에서는 AWS CLI를 사용하여 파라미터를 설정합니다. 이 예제는 CLI 명령이 완료된 직후에 파라미터가 변경되도록 `ApplyMethod`을(를) `immediate`(으)로 설정합니다. 읽기 복제본이 재부팅된 후 보류 중인 변경 사항을 적용하려면 `ApplyMethod`을(를) `pending-reboot`(으)로 설정합니다.

다음 예제에서는 복제 필터를 설정합니다.

- Including databases in replication
- Including tables in replication
- Including tables in replication with wildcard characters
- Escaping wildcard characters in names
- Excluding databases from replication

- Excluding tables from replication
- Excluding tables from replication using wildcard characters

Example 복제에 데이터베이스 포함

다음 예제에서는 복제에 mydb1 및 mydb2 데이터베이스가 포함되어 있습니다. 읽기 복제본에 대해 replicate-do-db을(를) 설정하면 파라미터에 지정된 데이터베이스만 복제됩니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "[{"ParameterName": "replicate-do-db", "ParameterValue": "mydb1,mydb2",
 "ApplyMethod":"immediate"}]"
```

Windows의 경우:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
  --parameters "[{"ParameterName": "replicate-do-db", "ParameterValue": "mydb1,mydb2",
 "ApplyMethod":"immediate"}]"
```

Example 복제에 테이블 포함

다음 예제에서는 복제의 table1 데이터베이스에 table2 및 mydb1 테이블이 포함되어 있습니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "[{"ParameterName": "replicate-do-table", "ParameterValue":
 "mydb1.table1,mydb1.table2", "ApplyMethod":"immediate"}]"
```

Windows의 경우:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
```

```
  --parameters "[{"ParameterName": "replicate-do-table", "ParameterValue":
 "mydb1.table1,mydb1.table2", "ApplyMethod":"immediate"}]"
```

Example 와일드카드 문자를 사용하여 복제에 테이블 포함

다음 예제에서는 복제의 데이터베이스 `orders`에 `returns` 및 `mydb`(으)로 시작하는 이름을 가진 테이블이 포함되어 있습니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "[{"ParameterName": "replicate-wild-do-table", "ParameterValue":
 "mydb.orders%,mydb.returns%", "ApplyMethod":"immediate"}]"
```

Windows의 경우:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
  --parameters "[{"ParameterName": "replicate-wild-do-table", "ParameterValue":
 "mydb.orders%,mydb.returns%", "ApplyMethod":"immediate"}]"
```

Example 이름에서 와일드카드 문자 이스케이프

다음 예제에서는 이스케이프 문자 `\`을(를) 사용하여 이름의 일부인 와일드카드 문자를 이스케이프하는 방법을 보여줍니다.

`mydb1`(으)로 시작하는 데이터베이스 `my_table`에 여러 테이블 이름이 있다고 가정하고, 이러한 테이블을 복제에 포함하고자 합니다. 테이블 이름에 와일드카드 문자이기도 한 밑줄이 포함되어 있으므로 테이블 이름에서 밑줄을 이스케이프합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "[{"ParameterName": "replicate-wild-do-table", "ParameterValue": "my
\_table%", "ApplyMethod":"immediate"}]"
```

Windows의 경우:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
  --parameters "[{"ParameterName": "replicate-wild-do-table", "ParameterValue": "my
\_table%", "ApplyMethod":"immediate"}]"
```

Example 복제에서 데이터베이스 제외

다음 예제에서는 mydb1 및 mydb2 데이터베이스를 복제에서 제외합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "[{"ParameterName": "replicate-ignore-db", "ParameterValue":
 "mydb1,mydb2", "ApplyMethod":"immediate"}]"
```

Windows의 경우:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
  --parameters "[{"ParameterName": "replicate-ignore-db", "ParameterValue":
 "mydb1,mydb2", "ApplyMethod":"immediate"}]"
```

Example 복제에서 테이블 제외

다음 예제에서는 데이터베이스 table1의 table2 및 mydb1 테이블을 복제에서 제외합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "[{"ParameterName": "replicate-ignore-table", "ParameterValue":
 "mydb1.table1,mydb1.table2", "ApplyMethod":"immediate"}]"
```

Windows의 경우:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
  --parameters "[{"ParameterName": "replicate-ignore-table", "ParameterValue":
 "mydb1.table1,mydb1.table2", "ApplyMethod":"immediate"}]"
```

Example 와일드카드 문자를 사용하여 복제에서 테이블 제외

다음 예제에서는 데이터베이스 `orders`에서 `returns` 및 `mydb`(으)로 시작하는 이름을 가진 테이블을 복제에서 제외합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "[{"ParameterName": "replicate-wild-ignore-table", "ParameterValue":
 "mydb.orders%,mydb.returns%", "ApplyMethod":"immediate"}]"
```

Windows의 경우:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
  --parameters "[{"ParameterName": "replicate-wild-ignore-table", "ParameterValue":
 "mydb.orders%,mydb.returns%", "ApplyMethod":"immediate"}]"
```

읽기 복제본에 대한 복제 필터 보기

다음과 같은 방법으로 읽기 복제본에 대한 복제 필터를 볼 수 있습니다.

- 읽기 복제본과 연결된 파라미터 그룹에서 복제 필터링 파라미터 설정을 확인합니다.

  지침은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 값 보기 섹션을 참조하세요.
- MariaDB 클라이언트에서 읽기 전용 복제본에 연결하고 `SHOW REPLICA STATUS` 문을 실행합니다.

  출력 시, 다음 필드에서 읽기 복제본에 대한 복제 필터를 보여줍니다.
  - `Replicate_Do_DB`
  - `Replicate_Ignore_DB`
  - `Replicate_Do_Table`
  - `Replicate_Ignore_Table`

- `Replicate_Wild_Do_Table`
- `Replicate_Wild_Ignore_Table`

이러한 필드에 대한 자세한 내용은 MySQL 설명서의 복제 상태 확인을 참조하세요.

> **Note**
>
> 이전 버전의 MariaDB에는 `SHOW SLAVE STATUS` 대신 `SHOW REPLICA STATUS`가 사용되었습니다. 10.5 이전 MariaDB 버전을 사용하는 경우 `SHOW SLAVE STATUS`를 사용합니다.

MariaDB를 사용한 지연 복제 구성

지연 복제를 재해 복구를 위한 전략으로 사용할 수 있습니다. 지연된 복제를 사용하여 원본에서 읽기 전용 복제본으로의 복제를 지연할 최소 시간(초)을 지정합니다. 재해 발생 시(예: 실수로 테이블 삭제) 다음 단계를 완료하여 재해로부터 빠르게 복구할 수 있습니다.

- 재해를 일으킨 변경 사항이 읽기 전용 복제본으로 전송되기 이전에 읽기 전용 복제본에 대한 복제를 중지합니다.

  mysql.rds_stop_replication 저장 프로시저를 사용하여 복제를 중지합니다.
- 읽기 전용 복제본을 독립 DB 인스턴스로 승격의 지침에 따라 읽기 전용 복제본을 새 원본 DB 인스턴스로 승격합니다.

> **Note**
>
> - 지연된 복제는 MariaDB 10.6 이상에서 지원됩니다.
> - 저장 프로시저를 사용하여 지연된 복제를 구성합니다. AWS Management Console, AWS CLI 또는 Amazon RDS API를 사용하여 지연 복제를 구성할 수 없습니다.
> - 지연된 복제 구성에서 전역 트랜잭션 ID(GTID)를 기반으로 하는 복제를 사용할 수 있습니다.

주제

- 읽기 전용 복제본 생성 중 지연 복제 구성
- 기존 읽기 전용 복제본에 대한 지연 복제 수정
- 읽기 전용 복제본 승격

읽기 전용 복제본 생성 중 지연 복제 구성

DB 인스턴스에서 향후에 생성되는 읽기 전용 복제본에 대한 지연된 복제를 구성하려면 mysql.rds_set_configuration 파라미터와 함께 `target delay` 저장 프로시저를 실행합니다.

읽기 전용 복제본을 생성하는 동안 지연된 복제를 구성하려면

1. MariaDB 클라이언트를 사용하여 마스터 사용자로 읽기 전용 복제본에 대한 원본이 될 MariaDB DB 인스턴스에 연결합니다.
2. mysql.rds_set_configuration 파라미터와 함께 `target delay` 저장 프로시저를 실행합니다.

   예를 들어, 현재 DB 인스턴스에서 생성되는 모든 읽기 전용 복제본에 대해 1시간(3,600초) 이상 복제를 지연하도록 지정하려면 다음 저장 프로시저를 실행합니다.

   ```
   call mysql.rds_set_configuration('target delay', 3600);
   ```

   > **Note**
   >
   > 이 저장 프로시저를 실행한 후 AWS CLI 또는 Amazon RDS API를 사용하여 생성하는 모든 읽기 전용 복제본은 지정된 시간(초)만큼 복제를 지연하도록 구성됩니다.

기존 읽기 전용 복제본에 대한 지연 복제 수정

기존 읽기 전용 복제본에 대한 지연된 복제를 수정하려면 mysql.rds_set_source_delay 저장 프로시저를 실행합니다.

기존 읽기 전용 복제본에 대한 지연된 복제를 수정하려면

1. MariaDB 클라이언트를 사용하여 마스터 사용자로 읽기 전용 복제본에 연결합니다.
2. mysql.rds_stop_replication 저장 프로시저를 사용하여 복제를 중지합니다.
3. mysql.rds_set_source_delay 저장 프로시저를 실행합니다.

   예를 들어, 읽기 전용 복제본에 대한 복제가 1시간(3,600초) 이상 지연되도록 지정하려면 다음 저장 프로시저를 실행합니다.

   ```
   call mysql.rds_set_source_delay(3600);
   ```

4. mysql.rds_start_replication 저장 프로시저를 사용하여 복제를 시작합니다.

읽기 전용 복제본 승격

복제가 중지된 후 재해 복구 시나리오에서 읽기 전용 복제본을 새 원본 DB 인스턴스로 승격할 수 있습니다. 읽기 전용 복제본 승격에 대한 자세한 내용은 읽기 전용 복제본을 독립 DB 인스턴스로 승격 단원을 참조하십시오.

MariaDB를 사용한 읽기 전용 복제본 업데이트

읽기 전용 복제본은 읽기 쿼리를 지원하도록 설계되었지만 경우에 따라 업데이트가 필요할 수 있습니다. 예를 들어 특정 유형의 쿼리가 복제본에 액세스하는 속도를 높이기 위해 인덱스를 추가해야 할 경우가 있습니다. 읽기 전용 복제본의 DB 파라미터 그룹에서 `read_only` 파라미터를 0으로 설정하여 업데이트를 활성화할 수 있습니다.

MariaDB를 사용한 다중 AZ 읽기 전용 복제본 배포 작업

단일 AZ 또는 다중 AZ DB 인스턴스 배포를 통해 읽기 전용 복제본을 생성할 수 있습니다. 다중 AZ 배포는 중요 데이터의 내구성과 가용성을 개선하는 데 효과적이지만 읽기 전용 쿼리를 실행하는 데 보조로 사용할 수는 없습니다. 대신 트래픽이 많은 다중 AZ DB 인스턴스에서 읽기 전용 쿼리를 오프로드할 목적으로 읽기 전용 복제본을 생성할 수 있습니다. 다중 AZ 배포의 원본 인스턴스가 보조 인스턴스로 장애 조치되는 경우 연결된 모든 읽기 전용 복제본이 자동으로 전환되어 보조(이제는 기본) 인스턴스를 복제 원본으로 사용합니다. 자세한 내용은 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 섹션을 참조하세요.

다중 AZ DB 인스턴스에서 읽기 전용 복제본을 생성할 수 있습니다. Amazon RDS는 복제본에 대한 장애 조치 지원을 위해 대기 복제본을 다른 가용 영역에 생성합니다. 읽기 전용 복제본을 다중 AZ DB 인스턴스로 생성하는 작업은 원본 데이터베이스가 다중 AZ DB 인스턴스인지 여부와는 무관합니다.

RDS for MariaDB에서의 계단식 읽기 전용 복제본 사용

RDS for MariaDB는 계단식 읽기 전용 복제본을 지원합니다. 계단식 읽기 전용 복제본을 사용하면 소스 RDS for MariaDB DB 인스턴스에 오버헤드를 추가하지 않고도 읽기 전용 복제본 크기를 조정할 수 있습니다.

계단식 읽기 전용 복제본을 사용하면 RDS for MariaDB DB 인스턴스가 데이터를 체인의 첫 번째 읽기 전용 복제본으로 전송합니다. 뒤이어 해당 읽기 전용 복제본이 데이터를 체인의 두 번째 복제본으로 전송하는 식으로 이루어집니다. 결과적으로 체인의 모든 읽기 전용 복제본이 소스 DB 인스턴스에만 오버헤드가 발생하는 일 없이 RDS for MariaDB DB 인스턴스에서 변경됩니다.

소스 RDS for MariaDB DB 인스턴스에서 체인에 최대 3개의 읽기 전용 복제본을 생성할 수 있습니다. 예를 들어 RDS MariaDB DB 인스턴스, `mariadb-main`이 있다고 가정해봅니다. 다음을 수행할 수 있습니다.

- mariadb-main부터 시작해서 체인에 첫 번째 읽기 전용 복제본 read-replica-1을 생성합니다.
- 다음으로 read-replica-1에서 체인에 다음 읽기 전용 복제본 read-replica-2를 생성합니다.
- 마지막으로 read-replica-2에서 체인에 세 번째 읽기 전용 복제본 read-replica-3을 생성합니다.

체인에서 mariadb-main에 대한 세 번째 계단식 읽기 전용 복제본 다음으로 또 다른 읽기 전용 복제본을 생성할 수 없습니다. RDS for MariaDB 소스 DB 인스턴스부터 계단식 읽기 전용 복제본 체인의 마지막에 이르는 전체 인스턴스는 최대 4개의 DB 인스턴스로 구성될 수 있습니다.

읽기 전용 복제본을 계단식으로 실행하려면 각 소스 RDS for MariaDB DB 인스턴스에 자동 백업이 켜져 있어야 합니다. 읽기 전용 복제본에서 자동 백업을 켜려면 먼저 읽기 전용 복제본을 생성한 다음 자동 백업이 켜지도록 읽기 전용 복제본을 수정합니다. 자세한 내용은 읽기 전용 복제본 생성 단원을 참조하십시오.

모든 읽기 전용 복제본과 마찬가지로 계단식 구성에 포함된 읽기 전용 복제본을 승격할 수 있습니다. 읽기 전용 복제본 체인의 한 읽기 전용 복제본을 승격하면 체인에서 해당 복제본이 제거됩니다. 예를 들어 mariadb-main DB 인스턴스의 일부 워크로드를 회계 부서에서만 사용할 수 있도록 새 인스턴스로 옮기려고 합니다. 이 예제에서 3개의 읽기 전용 복제본 체인이 있다고 가정하고 read-replica-2를 승격하기로 결정합니다. 체인은 다음과 같이 변화합니다.

- read-replica-2를 승격하면 복제 체인에서 제거됩니다.
  - 이제 전체 읽기/쓰기 DB 인스턴스가 됩니다.
  - 승격 전과 마찬가지로 read-replica-3으로 계속 복제합니다.
- mariadb-main은 read-replica-1로 계속 복제를 진행합니다.

읽기 전용 복제본 승격에 대한 자세한 내용은 읽기 전용 복제본을 독립 DB 인스턴스로 승격 섹션을 참조하세요.

MariaDB 읽기 전용 복제본 모니터링

MariaDB 읽기 전용 복제본의 경우 Amazon RDS ReplicaLag 지표를 보면서 Amazon CloudWatch의 복제 지연 시간을 모니터링할 수 있습니다. ReplicaLag 메트릭은 Seconds_Behind_Master 명령의 SHOW REPLICA STATUS 필드의 값을 보고합니다.

> **Note**
>
> 이전 버전의 MariaDB에는 `SHOW SLAVE STATUS` 대신 `SHOW REPLICA STATUS`가 사용되었습니다. 10.5 이전 MariaDB 버전을 사용하는 경우 `SHOW SLAVE STATUS`를 사용합니다.

이렇게 MariaDB에서 복제 지연이 발생하는 공통 원인은 다음과 같습니다.

- 네트워크 중단.
- 읽기 전용 복제본의 인덱스가 있는 테이블에 쓰기 작업 중일 때. 읽기 전용 복제본에서 `read_only` 파라미터가 0으로 설정되어 있지 않으면 복제가 중단될 수 있습니다.
- MyISAM과 같은 비트랜잭션 스토리지 엔진 사용. 복제는 MariaDB의 InnoDB 스토리지 엔진에서만 지원됩니다.

`ReplicaLag` 지표가 0에 도달하면 복제본이 원본 DB 인스턴스를 따라잡은 것입니다. `ReplicaLag` 지표가 -1을 반환하는 경우 복제가 현재 활성이 아닙니다. `ReplicaLag` = -1은 `Seconds_Behind_Master` = `NULL`과 동등합니다.

MariaDB 읽기 전용 복제본을 사용한 복제 시작 및 중지

Amazon RDS DB 인스턴스에서는 시스템에 저장된 프로시저인 mysql.rds_stop_replication과(와) mysql.rds_start_replication을 호출하여 복제 프로세스를 종료하거나 재시작할 수 있습니다. 대용량 인덱스를 생성하는 등 오랜 시간이 걸리는 작업에서 두 Amazon RDS 인스턴스를 서로 복제할 때도 이런 방법이 가능합니다. 또한 데이터베이스를 가져오거나 내보낼 때도 복제를 종료하거나 시작할 필요가 있습니다. 자세한 내용은 가동 중지 시간을 줄이면서 Amazon RDS for MySQL 데이터베이스로 데이터 가져오기 및 복제를 사용하여 MySQL DB 인스턴스에서 데이터 내보내기 단원을 참조하십시오.

수동으로 또는 복제 오류로 인해 연속하여 30일 이상 복제가 중단된 경우에는 Amazon RDS가 소스 DB 인스턴스와 모든 읽기 전용 복제본 사이의 복제를 종료합니다. 이렇게 하는 이유는 소스 DB 인스턴스에 대한 스토리지 요건 증가와 장애 조치의 장기화를 방지하기 위해서입니다. 읽기 전용 복제본 DB 인스턴스는 계속 사용할 수 있습니다. 그러나 복제가 종료된 후 읽기 전용 복제본에 필요한 이진 로그가 원본 DB 인스턴스에서 삭제되므로 복제를 재개할 수 없습니다. 원본 DB 인스턴스에서 복제를 재설정하려면 새 읽기 전용 복제본을 생성해야 합니다.

MariaDB 읽기 전용 복제본의 문제 해결

MariaDB의 복제본 기술은 비동기적입니다. 간혹 원본 DB 인스턴스에서 `BinLogDiskUsage`가 증가하면 읽기 전용 복제본의 `ReplicaLag`가 예상되는 이유도 비동기식이기 때문입니다. 예를 들어 원본

DB 인스턴스에 대해 대량의 쓰기 작업이 동시에 발생할 수 있습니다. 반대로 읽기 전용 복제본에 대한 쓰기 작업은 단일 I/O 스레드를 사용하기 때문에 연이어 차례로 발생합니다. 이로 인해 원본 인스턴스와 읽기 전용 복제본 사이에 지연 시간이 있기 마련입니다. 읽기 전용 복제본에 대한 자세한 내용은 MariaDB 설명서에서 복제 개요를 참조하십시오.

원본 DB 인스턴스와 뒤이어 일어나는 읽기 전용 복제본의 업데이트 간 지연 시간을 줄일 수 있는 방법에는 다음과 같이 몇 가지가 있습니다.

- 읽기 전용 복제본의 크기를 조정하여 원본 DB 인스턴스에 버금가는 스토리지 크기와 DB 인스턴스 클래스를 할당합니다.
- 원본 DB 인스턴스와 읽기 전용 복제본에 사용되는 DB 파라미터 그룹의 파라미터 설정이 서로 호환되는지 확인합니다. 자세한 정보와 예는 이번 섹션 후반의 `max_allowed_packet` 파라미터 관련 설명을 참조하십시오.

Amazon RDS는 읽기 전용 복제본의 복제 상태를 모니터링하고, 어떤 이유로든 복제가 중지되는 경우 읽기 전용 복제본 인스턴스의 `Replication State` 필드를 `Error`로 업데이트합니다. 읽기 전용 복제본에서 실행되는 DML 쿼리가 원본 DB 인스턴스의 업데이트와 충돌하는 경우가 한 예가 될 수 있습니다.

MariaDB 엔진에서 발생하는 관련 오류에 대한 세부 정보는 `Replication Error` 필드에서 다시 확인할 수 있습니다. RDS-EVENT-0045, RDS-EVENT-0046 및 RDS-EVENT-0047을 포함하여 읽기 전용 복제본의 상태를 나타내는 이벤트도 생성됩니다. 이벤트와 이벤트 구독에 대한 자세한 내용은 Amazon RDS 이벤트 알림 작업 단원을 참조하십시오. MariaDB 오류 메시지가 반환되는 경우 MariaDB 오류 메시지 문서에 설명되어 있는 오류를 검토하십시오.

복제 오류의 원인이 되는 공통적인 문제를 하나 꼽으라고 하면 읽기 전용 복제본의 `max_allowed_packet` 파라미터 값이 원본 DB 인스턴스의 `max_allowed_packet` 파라미터 값보다 작을 때입니다. `max_allowed_packet` 파라미터는 DB 파라미터 그룹에서 설정할 수 있는 사용자 지정 파라미터로, 데이터베이스에서 실행할 수 있는 DML 코드의 최대 크기를 지정하는 데 사용됩니다. 경우에 따라 원본 DB 인스턴스와 연결된 DB 파라미터 그룹의 `max_allowed_packet` 파라미터 값이 원본의 읽기 전용 복제본과 연결된 DB 파라미터 그룹의 `max_allowed_packet` 파라미터 값보다 작습니다. 이러한 경우에는 복제 프로세스에서 오류(패킷이 'max_allowed_packet' 바이트보다 큼)가 발생하여 복제가 중단될 수도 있습니다. 원본 및 읽기 전용 복제본이 동일한 `max_allowed_packet` 파라미터 값을 가진 DB 파라미터 그룹을 사용하도록 하여 이 오류를 해결할 수 있습니다.

이밖에 복제 오류의 원인이 되는 공통적인 상황은 다음과 같습니다.

- 읽기 전용 복제본의 테이블에 쓰기 작업 중일 때. 읽기 전용 복제본에서 인덱스를 생성 중인 경우 `read_only` 파라미터를 0으로 설정하여 인덱스를 생성해야 합니다. 읽기 전용 복제본의 테이블에 쓰기 작업 중인 경우 복제가 중단될 수 있습니다.
- MyISAM과 같은 비트랜잭션 스토리지 엔진을 사용할 때. 읽기 전용 복제본에 트랜잭션 스토리지 엔진이 필요합니다. 복제는 MariaDB의 InnoDB 스토리지 엔진에서만 지원됩니다.
- `SYSDATE()`와 같이 안전하지 않은 비결정적 쿼리를 사용하는 경우. 자세한 내용은 바이너리 로깅에서 안전한 문과 안전하지 않은 문 결정을 참조하세요.

오류를 건너뛰어도 안전하다고 판단될 경우에는 RDS for MySQL에 대한 현재 복제 오류 건너뛰기에 설명한 단계를 따르십시오. 그 밖에 읽기 전용 복제본을 삭제하고 엔드포인트가 이전 읽기 전용 복제본의 엔드포인트와 동일하게 유지되도록 동일한 DB 인스턴스 식별자를 사용하여 인스턴스를 생성할 수도 있습니다. 복제 오류가 해결되면 `Replication State`가 replicating으로 변경됩니다.

MariaDB DB 인스턴스에서는 경우에 따라 읽기 전용 복제본을 보조 인스턴스로 전환하지 못할 수도 있습니다. 이는 일부 이진 로그(binlog) 이벤트가 오류로 인해 플러시되지 않기 때문입니다. 이러한 경우 수동으로 읽기 전용 복제본을 삭제한 후 재생성해야 합니다. `sync_binlog=1` 및 `innodb_flush_log_at_trx_commit=1` 파라미터 값을 설정하여 발생하는 이러한 가능성을 줄일 수 있습니다. 단, 이 설정은 성능 감소의 원인이 될 수도 있으므로 변경 사항을 프로덕션 환경에 적용하기 전에 그 효과를 테스트하는 것이 좋습니다.

외부 소스 인스턴스를 사용하여 GTID 기반 복제 구성

버전 10.0.24 이상의 외부 MariaDB 인스턴스에서 RDS for MariaDB DB 인스턴스로 가는 전역 트랜잭션 ID(GTID)를 기반으로 복제를 설정할 수 있습니다. Amazon RDS에서 외부 소스 인스턴스 및 복제본을 설정할 때 다음 지침을 따르십시오.

- 사용자의 복제본인 RDS for MariaDB DB 인스턴스에 대한 장애 조치 이벤트를 모니터링합니다. 장애 조치가 발생할 경우에는 사용자의 복제본인 DB 인스턴스가 다른 네트워크 주소를 가진 새 호스트에서 다시 생성될 수도 있습니다. 장애 조치 이벤트를 모니터링하는 자세한 방법은 Amazon RDS 이벤트 알림 작업 단원을 참조하십시오.
- 이진 로그(binlog)가 복제본에 적용되었음을 확인할 때까지는 소스 인스턴스에서 binlog를 유지 관리하십시오. 이렇게 유지 관리해야 오류 발생 시 소스 인스턴스를 복원할 수 있습니다.
- Amazon RDS 상의 MariaDB DB 인스턴스에서 자동 백업을 활성화합니다. 자동 백업을 활성화하면 소스 인스턴스 및 복제본을 다시 동기화할 필요가 있을 때 복제본을 특정 시점으로 복원할 수 있습니다. 백업과 특정 시점으로 복원에 대한 자세한 내용은 다음(데이터 백업, 복원 및 내보내기)단원을 참조하십시오.

> **Note**
>
> MariaDB DB 인스턴스에서 복제를 시작하는 데 필요한 권한은 제한되고 Amazon RDS 마스터 사용자는 사용할 수 없습니다. 따라서 Amazon RDS mysql.rds_set_external_master_gtid 및 mysql.rds_start_replication 명령을 사용하여 라이브 데이터베이스와 RDS for MariaDB 데이터베이스 사이의 복제를 설정해야 합니다.

외부 소스 인스턴스와 Amazon RDS의 MariaDB DB 인스턴스 간에 복제를 시작하려면 다음 절차를 수행하십시오.

복제를 시작하려면

1. 원본 MariaDB 인스턴스를 읽기 전용으로 설정합니다.

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SET GLOBAL read_only = ON;
```

2. 외부 MariaDB 인스턴스의 현재 GTID를 가져옵니다. 이를 위해 `mysql` 또는 원하는 쿼리 편집기를 사용하여 `SELECT @@gtid_current_pos;`를 실행할 수 있습니다.

   GTID는 `<domain-id>-<server-id>-<sequence-id>` 형식입니다. 일반적인 GTID는 **`0-1234510749-1728`**과 비슷한 형식입니다. GTID 및 그 구성 요소에 대한 자세한 내용은 MariaDB 설명서의 전역 트랜잭션 ID를 참조하십시오.

3. `mysqldump`를 사용하여 외부 MariaDB 인스턴스에서 MariaDB DB 인스턴스로 데이터베이스를 복사합니다. 매우 큰 데이터베이스의 경우, 가동 중지 시간을 줄이면서 Amazon RDS for MySQL 데이터베이스로 데이터 가져오기에서 이 절차를 사용하고 싶을 것입니다.

   대상 LinuxmacOS, 또는Unix:

```
mysqldump \
    --databases database_name \
    --single-transaction \
    --compress \
    --order-by-primary \
    -u local_user \
    -plocal_password | mysql \
        --host=hostname \
        --port=3306 \
        -u RDS_user_name \
```

```
        -pRDS_password
```

Windows의 경우:

```
mysqldump ^
    --databases database_name ^
    --single-transaction ^
    --compress ^
    --order-by-primary \
    -u local_user \
    -plocal_password | mysql ^
        --host=hostname ^
        --port=3306 ^
        -u RDS_user_name ^
        -pRDS_password
```

> **Note**
>
> -p 옵션과 입력한 암호 사이에 공백이 없어야 합니다.
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

MariaDB DB 인스턴스에 연결하기 위해 호스트 이름, 사용자 이름, 포트 및 암호를 지정하려면 `--host` 명령에서 `--user (-u)`, `--port`, `-p` 및 `mysql` 옵션을 사용합니다. 호스트 이름은 MariaDB DB 인스턴스 엔드포인트의 DNS 이름입니다(예: `myinstance.123456789012.us-east-1.rds.amazonaws.com`). Amazon RDS Management Console의 인스턴스 세부 정보에서 엔드포인트 값을 찾을 수 있습니다.

4. 원본 MariaDB 인스턴스를 다시 쓰기 가능한 상태로 만듭니다.

```
mysql> SET GLOBAL read_only = OFF;
mysql> UNLOCK TABLES;
```

5. Amazon RDS Management Console에서 MariaDB DB 인스턴스에 대한 VPC 보안 그룹에 외부 MariaDB 데이터베이스를 호스팅하는 서버의 IP 주소를 추가합니다. VPC 보안 그룹 수정에 관한 자세한 내용은 Amazon Virtual Private Cloud 사용 설명서에서 VPC의 보안 그룹 단원을 참조하십시오.

   다음 조건이 충족되면 IP 주소가 바뀔 수 있습니다.

- 외부 소스 인스턴스와 DB 인스턴스 간 통신에 퍼블릭 IP 주소를 사용하는 경우
- 외부 소스 인스턴스가 중지되었다가 다시 시작된 경우

위 두 가지 조건이 충족되면 추가하기 전에 IP 주소를 확인하십시오.

외부 MariaDB 인스턴스와 통신할 수 있도록, MariaDB DB 인스턴스 IP 주소에서의 연결을 허용하도록 로컬 네트워크를 구성해야 할 수도 있습니다. MariaDB DB 인스턴스의 IP 주소를 찾으려면 `host` 명령을 사용합니다.

```
host db_instance_endpoint
```

호스트 이름은 MariaDB DB 인스턴스 엔드포인트의 DNS 이름입니다.

6. 선택한 클라이언트를 사용하여 외부 MariaDB 인스턴스에 연결하고 복제에 사용될 MariaDB 사용자를 만듭니다. 이 계정은 오직 복제용으로만 사용되며 보안 향상을 위해 사용자의 도메인으로 제한되어야 합니다. 다음은 예입니다.

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

7. 외부 MariaDB 인스턴스의 경우 복제 사용자에게 `REPLICATION CLIENT` 및 `REPLICATION SLAVE` 권한을 부여합니다. 예를 들어 도메인의 'REPLICATION CLIENT' 사용자를 위해 모든 데이터베이스에서 `REPLICATION SLAVE` 및 `repl_user` 권한을 부여하려면 다음 명령을 실행합니다.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
```

8. MariaDB DB 인스턴스를 복제본으로 만듭니다. 마스터 사용자로서 MariaDB DB 인스턴스에 연결하고 mysql.rds_set_external_master_gtid 명령을 사용하여 외부 MariaDB 데이터베이스를 복제 소스 인스턴스로 식별합니다. 2단계에서 확인된 GTID를 사용하십시오. 다음은 예입니다.

```
CALL mysql.rds_set_external_master_gtid ('mymasterserver.mydomain.com', 3306,
 'repl_user', 'password', 'GTID', 1);
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

9. MariaDB DB 인스턴스에서 mysql.rds_start_replication 명령을 실행하여 복제를 시작합니다.

```
CALL mysql.rds_start_replication;
```

외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성

바이너리 로그 파일 복제를 사용하여 RDS for MySQL 또는 MariaDB DB 인스턴스와 Amazon RDS 외부에 있는 MySQL 또는 MariaDB 인스턴스 간에 복제를 설정할 수 있습니다.

주제

- 시작하기 전에
- 외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성

시작하기 전에

복제된 트랜잭션의 이진 로그 파일 위치를 사용하여 복제를 구성할 수 있습니다.

Amazon RDS DB 인스턴스에서 복제를 시작하는 데 필요한 권한은 제한되고 Amazon RDS 마스터 사용자는 사용할 수 없습니다. 이 때문에 Amazon RDS mysql.rds_set_external_master(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하) 또는 mysql.rds_set_external_source(RDS for MySQL 메이저 버전 8.4 이상) 및 mysql.rds_start_replication 명령을 사용하여 라이브 데이터베이스와 Amazon RDS 데이터베이스 사이의 복제를 설정해야 합니다.

MySQL 또는 MariaDB 데이터베이스의 이진 로깅 형식을 설정하려면 `binlog_format` 파라미터를 업데이트합니다. DB 인스턴스가 기본 DB 인스턴스 파라미터 그룹을 사용하는 경우, `binlog_format` 파라미터를 수정하려면 새로운 DB 파라미터 그룹을 만듭니다. MariaDB 및 MySQL 8.0 이하 버전에서는 `binlog_format`의 기본값이 `MIXED`입니다. 그러나 특정한 이진 로그(binlog) 형식이 필요하다면 `binlog_format`을 `ROW` 또는 `STATEMENT`로 설정할 수도 있습니다. 변경 사항을 적용하려면 DB 인스턴스를 재부팅합니다. MySQL 8.4 이상 버전에서는 `binlog_format`의 기본값이 `ROW`입니다.

`binlog_format` 파라미터 설정에 대한 자세한 내용은 단일 AZ 데이터베이스의 RDS for MySQL 이진 로깅 구성 단원을 참조하십시오. 다양한 MySQL 복제 유형에 대한 자세한 내용은 MySQL 설명서의 문 기반 및 행 기반 복제의 장/단점을 참조하십시오.

외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성

Amazon RDS에서 외부 소스 인스턴스 및 복제본을 설정할 때 다음 지침을 따르십시오.

- 사용자의 복제본인 Amazon RDS DB 인스턴스에 대한 장애 조치 이벤트를 모니터링합니다. 장애 조치가 발생할 경우에는 사용자의 복제본인 DB 인스턴스가 다른 네트워크 주소를 가진 새 호스트에서 다시 생성될 수도 있습니다. 장애 조치 이벤트를 모니터링하는 자세한 방법은 Amazon RDS 이벤트 알림 작업 단원을 참조하십시오.
- binlog가 복제본에 적용된 것으로 확인될 때까지는 소스 인스턴스에서 binlog를 유지 관리합니다. 이렇게 유지 관리해야 오류 발생 시 소스 인스턴스를 복원할 수 있습니다.
- Amazon RDS DB 인스턴스에서 자동 백업을 활성화합니다. 자동 백업을 활성화하면 소스 인스턴스 및 복제본을 다시 동기화할 필요가 있을 때 복제본을 특정 시점으로 복원할 수 있습니다. 백업 및 특정 시점으로 복원에 대한 자세한 내용은 데이터 백업, 복원 및 내보내기 단원을 참조하십시오.

외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제를 구성하려면

1. 원본 MySQL 또는 MariaDB 인스턴스를 읽기 전용으로 설정합니다.

   ```
   mysql> FLUSH TABLES WITH READ LOCK;
   mysql> SET GLOBAL read_only = ON;
   ```

2. 원본 MySQL 또는 MariaDB 인스턴스에서 `SHOW MASTER STATUS` 명령을 실행하여 binlog 위치를 확인합니다.

   다음 예제와 비슷한 출력 결과를 얻습니다.

   ```
   File                          Position
   ------------------------------------
    mysql-bin-changelog.000031      107
   ------------------------------------
   ```

3. `mysqldump`를 사용하여 외부 인스턴스에서 Amazon RDS DB 인스턴스로 데이터베이스를 복사합니다. 매우 큰 데이터베이스의 경우, 가동 중지 시간을 줄이면서 Amazon RDS for MySQL 데이터베이스로 데이터 가져오기에서 이 절차를 사용하고 싶을 것입니다.

   대상 LinuxmacOS, 또는Unix:

   ```
   mysqldump --databases database_name \
       --single-transaction \
   ```

```
    --compress \
    --order-by-primary \
    -u local_user \
    -plocal_password | mysql \
        --host=hostname \
        --port=3306 \
        -u RDS_user_name \
        -pRDS_password
```

Windows의 경우:

```
mysqldump --databases database_name ^
    --single-transaction ^
    --compress ^
    --order-by-primary ^
    -u local_user ^
    -plocal_password | mysql ^
        --host=hostname ^
        --port=3306 ^
        -u RDS_user_name ^
        -pRDS_password
```

> **Note**
>
> -p 옵션과 입력한 암호 사이에 공백이 없어야 합니다.

Amazon RDS DB 인스턴스에 연결하기 위해 호스트 이름, 사용자 이름, 포트 및 암호를 지정하려면 `--host` 명령에 `--user (-u)`, `--port`, `-p`, `mysql` 옵션을 사용합니다. 호스트 이름은 Amazon RDS DB 인스턴스 엔드포인트의 DNS(Domain Name Service) 이름입니다(예: `myinstance.123456789012.us-east-1.rds.amazonaws.com`). AWS Management Console의 인스턴스 세부 정보에서 엔드포인트 값을 확인할 수 있습니다.

4. 원본 MySQL 또는 MariaDB 인스턴스를 다시 쓰기 가능한 상태로 만듭니다.

```
mysql> SET GLOBAL read_only = OFF;
mysql> UNLOCK TABLES;
```

복제에 사용할 백업을 만드는 방법에 대한 자세한 내용은 MySQL 설명서를 참조하십시오.

5. AWS Management Console에서 Amazon RDS DB 인스턴스에 대한 Virtual Private Cloud(VPC) 보안 그룹에 외부 데이터베이스를 호스팅하는 서버의 IP 주소를 추가합니다. VPC 보안 그룹 수정에 대한 자세한 내용은 Amazon Virtual Private Cloud 사용 설명서의 VPC용 보안 그룹을 참조하십시오.

   다음 조건이 충족되면 IP 주소가 바뀔 수 있습니다.

   - 외부 소스 인스턴스와 DB 인스턴스 간 통신에 퍼블릭 IP 주소를 사용하는 경우
   - 외부 소스 인스턴스가 중지되었다가 다시 시작된 경우

   위 두 가지 조건이 충족되면 추가하기 전에 IP 주소를 확인하십시오.

   Amazon RDS DB 인스턴스의 IP 주소로부터의 연결을 허용하도록 로컬 네트워크를 구성해야 할 수도 있습니다. 이렇게 하면 로컬 네트워크가 외부 MySQL 또는 MariaDB 인스턴스와 통신할 수 있습니다. Amazon RDS DB 인스턴스의 IP 주소를 확인하려면 `host` 명령을 사용합니다.

   ```
   host db_instance_endpoint
   ```

   호스트 이름은 Amazon RDS DB 인스턴스 엔드포인트의 DNS 이름입니다.

6. 선택한 클라이언트를 사용하여 외부 인스턴스에 연결하고 복제에 사용할 사용자를 만듭니다. 이 계정을 복제용으로만 사용하고, 보안을 강화하기 위해 해당 도메인으로만 제한하십시오. 다음은 예입니다.

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
   ```

   > **Note**
   >
   > 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

7. 외부 인스턴스의 경우 복제 사용자에게 `REPLICATION CLIENT` 및 `REPLICATION SLAVE` 권한을 부여합니다. 예를 들어 도메인의 'REPLICATION CLIENT' 사용자를 위해 모든 데이터베이스에서 `REPLICATION SLAVE` 및 `repl_user` 권한을 부여하려면 다음 명령을 실행합니다.

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
   ```

8. Amazon RDS DB 인스턴스를 복제본으로 만듭니다. 이렇게 하려면 먼저 마스터 사용자로 Amazon RDS DB 인스턴스에 연결합니다. 그런 다음 mysql.rds_set_external_source(RDS for

MySQL 메이저 버전 8.4 이상) 또는 mysql.rds_set_external_master(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하) 명령을 사용하여 외부 MySQL 또는 MariaDB 데이터베이스를 소스 인스턴스로 식별합니다. 2단계에서 확인한 마스터 로그 파일 이름과 마스터 로그 위치를 사용합니다. 다음 명령은 예시입니다.

MySQL 8.4

```
CALL mysql.rds_set_external_source ('mysourceserver.mydomain.com', 3306,
 'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 1);
```

MariaDB 및 MySQL 8.0 및 5.7

```
CALL mysql.rds_set_external_master ('mymasterserver.mydomain.com', 3306,
 'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 1);
```

> **Note**
>
> RDS for MySQL의 경우 mysql.rds_set_external_source_with_delay(RDS for MySQL 메이저 버전 8.4 이상) 또는 mysql.rds_set_external_master_with_delay(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하) 저장 프로시저를 대신 실행하여 지연 복제를 사용하도록 선택할 수 있습니다. RDS for MySQL에서 지연 복제를 사용하는 이유 중 하나는 mysql.rds_start_replication_until 저장 프로시저를 사용하여 재해를 복구할 수 있기 때문입니다. 현재 RDS for MariaDB에서는 지연된 복제를 지원하지만 `mysql.rds_start_replication_until` 절차에서는 지원하지 않습니다.

9. Amazon RDS DB 인스턴스에서 mysql.rds_start_replication 명령을 실행하여 복제를 시작합니다.

```
CALL mysql.rds_start_replication;
```

MariaDB 데이터베이스 엔진을 위한 옵션

아래에는 MariaDB DB 엔진을 실행하는 Amazon RDS 인스턴스에 사용 가능한 옵션 또는 추가 기능에 대한 설명이 나와 있습니다. 이러한 옵션을 설정하려면 사용자 지정 옵션 그룹에 추가한 다음 옵션 그룹과 DB 인스턴스를 연결해야 합니다. 옵션 그룹 작업에 대한 자세한 내용은 옵션 그룹 작업 단원을 참조하세요.

Amazon RDS는 다음의 MariaDB 옵션을 지원합니다.

옵션 ID	엔진 버전
MARIADB_AUDIT_PLUGIN	MariaDB 10.3 이상

MariaDB 감사 플러그인 지원

Amazon RDS는 MariaDB 데이터베이스 인스턴스에서의 MariaDB 감사 플러그인 사용을 지원합니다. MariaDB 감사 플러그인은 사용자의 데이터베이스 로그온, 데이터베이스에 대해 실행되는 쿼리 등의 데이터베이스 활동을 기록합니다. 데이터베이스 활동 기록은 로그 파일에 저장됩니다.

감사 플러그인 옵션 설정

Amazon RDS는 MariaDB 감사 플러그인 옵션의 다음 설정을 지원합니다.

> **Note**
>
> RDS 콘솔에서 옵션 설정을 구성하지 않을 경우 RDS는 기본 설정을 사용합니다.

옵션 설정	유효한 값	기본값	설명
SERVER_AUDIT_FILE_PATH	/rdsdbdata/log/audit/	/rdsdbdata/log/audit/	로그 파일의 위치. 로그 파일에는 SERVER_AUDIT_EVENTS 에서 지정된 활동 기록이 포함되어 있습니다. 자세한 내용은 데이터베이스 로그 파일 보기 및 나열 및 MariaDB 데이터베이스 로그 파일 단원을 참조하십시오.

옵션 설정	유효한 값	기본값	설명
SERVER_AUDIT_FILE_ROTATE_SIZE	1–1000000000	1000000	도달 시 파일 로테이션을 초래하는 바이트 크기. 자세한 내용은 MariaDB의 로그 교체 및 보존 섹션을 참조하세요.
SERVER_AUDIT_FILE_ROTATIONS	0–100	9	server_audit_output_type=file 일 경우 저장할 로그 교체 수입니다. 0으로 설정하면 로그 파일이 교체되지 않습니다. 자세한 내용은 MariaDB의 로그 교체 및 보존 및 데이터베이스 로그 파일 다운로드 단원을 참조하세요.
SERVER_AUDIT_EVENTS	CONNECT, QUERY, TABLE, QUERY_DDL , QUERY_DML , QUERY_DML_NO_SELECT , QUERY_DCL	CONNECT, QUERY	로그에 기록할 활동 유형. MariaDB 감사 플러그인 설치 자체가 로깅됩니다. • CONNECT: 성공/실패한 데이터베이스 연결과 데이터베이스 연결 해제를 로깅합니다. • QUERY: 데이터베이스에 대해 실행된 모든 쿼리의 텍스트를 로깅합니다. • TABLE: 데이터베이스에 대해 쿼리가 실행될 때 쿼리의 영향을 받는 테이블을 로깅합니다. • QUERY_DDL : QUERY 이벤트와 유사하지만 데이터 정의 언어(DDL) 쿼리(CREATE, ALTER 등)만 반환합니다. • QUERY_DML : QUERY 이벤트와 유사하지만 데이터 조작 언어(DML) 쿼리(INSERT, UPDATE 등과 SELECT)만 반환합니다. • QUERY_DML_NO_SELECT : QUERY_DML 이벤트와 유사하지만 SELECT 쿼리를 로깅하지 않습니다. • QUERY_DCL : QUERY 이벤트와 유사하지만 데이터 제어 언어(DCL) 쿼리(GRANT, REVOKE 등)만 반환합니다.

옵션 설정	유효한 값	기본값	설명
SERVER_AUDIT_INCL_USERS	복수의 쉼표로 분리된 값	없음	지정된 사용자들의 활동만을 포함하십시오. 기본적으로 활동은 모든 사용자에 대해 기록됩니다. SERVER_AUDIT_INCL_USERS 및 SERVER_AUDIT_EXCL_USERS 는 상호 배타적입니다. SERVER_AUDIT_INCL_USERS 에 값을 추가하는 경우 SERVER_AUDIT_EXCL_USERS 에 값이 추가되지 않았는지 확인합니다.
SERVER_AUDIT_EXCL_USERS	복수의 쉼표로 분리된 값	없음	지정된 사용자들의 활동을 제외하십시오. 기본적으로 활동은 모든 사용자에 대해 기록됩니다. SERVER_AUDIT_INCL_USERS 및 SERVER_AUDIT_EXCL_USERS 는 상호 배타적입니다. SERVER_AUDIT_EXCL_USERS 에 값을 추가하는 경우 SERVER_AUDIT_INCL_USERS 에 값이 추가되지 않았는지 확인합니다. rdsadmin 사용자는 데이터베이스 상태를 확인하기 위해 1초마다 데이터베이스에 쿼리를 요청합니다. 다른 설정에 따라 이 활동은 로그 파일의 크기를 아주 빨리 대폭 증가시킬 수 있습니다. 이 활동을 기록할 필요가 없는 경우, rdsadmin 사용자를 SERVER_AUDIT_EXCL_USERS 목록에 추가하십시오. **Note** CONNECT 활동은 해당 사용자가 이 옵션 설정에 지정되었다 해도 모든 사용자에 대해 기록됩니다.

옵션 설정	유효한 값	기본값	설명
SERVER_AUDIT_LOGGING	ON	ON	로깅이 활성화되었습니다. 유일한 유효 값은 ON입니다. Amazon RDS는 로깅 비활성화를 지원하지 않습니다. 로깅을 비활성화하려면 MariaDB 감사 플러그인을 제거하십시오. 자세한 내용은 MariaDB 감사 플러그인 제거하기 섹션을 참조하세요.
SERVER_AUDIT_QUERY_LOG_LIMIT	0–2147483647	1024	레코드에서 쿼리 문자열의 길이 제한.

MariaDB 감사 플러그인 추가하기

MariaDB 감사 플러그인을 DB 인스턴스에 추가하는 일반적인 프로세스는 다음과 같습니다.

1. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 복사 또는 수정합니다.
2. [] 옵션을 옵션 그룹에 추가합니다.
3. 옵션 그룹을 DB 인스턴스에 연동시킵니다.

MariaDB 감사 플러그인을 추가한 후 DB 인스턴스를 재시작할 필요가 없습니다. 옵션 그룹이 활성화되자마자 감사가 즉시 시작됩니다.

MariaDB 감사 플러그인을 추가하려면,

1. 사용할 옵션 그룹을 결정합니다. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 사용합니다. 기존 옵션 그룹을 사용하려면 다음 단계로 건너뛰십시오. 그렇지 않다면 사용자 지정 DB 옵션 그룹을 생성합니다. Engine(엔진)에서 mariadb를 선택하고 Major engine version(메이저 엔진 버전)에서 10.3 이상을 선택합니다. 자세한 내용은 옵션 그룹 생성 단원을 참조하십시오.
2. MARIADB_AUDIT_PLUGIN 옵션을 옵션 그룹에 추가하고 옵션 설정을 구성하십시오. 옵션 추가에 대한 자세한 내용은 옵션 그룹에 옵션 추가 섹션을 참조하세요. 각 설정에 대한 자세한 내용은 감사 플러그인 옵션 설정 단원을 참조하십시오.
3. 옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스에 적용합니다.

- 새 DB 인스턴스의 경우, 인스턴스를 시작할 때 옵션 그룹을 적용합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
- 기존 DB 인스턴스의 경우 DB 인스턴스를 수정하고 새 옵션 그룹을 연결하여 옵션 그룹을 적용합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

MariaDB 감사 플러그인 로그 보기 및 다운로드

MariaDB 감사 플러그인을 활성화한 후 다른 텍스트 기반 로그 파일에 액세스하는 것과 동일한 방식으로 로그 파일의 결과에 액세스할 수 있습니다. 감사 로그 파일은 `/rdsdbdata/log/audit/`에 있습니다. 콘솔에서 로그 파일 보기에 대한 자세한 내용은 데이터베이스 로그 파일 보기 및 나열을(를) 참조하십시오. 로그 파일 다운로드에 대한 자세한 내용은 데이터베이스 로그 파일 다운로드을(를) 참조하십시오.

MariaDB 감사 플러그인 설정 수정

MariaDB 감사 플러그인을 활성화한 후 플러그인 설정을 수정할 수 있습니다. 옵션 설정을 변경하는 방법에 대한 자세한 내용은 옵션 설정 수정을(를) 참조하십시오. 각 설정에 대한 자세한 내용은 감사 플러그인 옵션 설정 단원을 참조하십시오.

MariaDB 감사 플러그인 제거하기

Amazon RDS는 MariaDB 감사 플러그인에서의 로깅 끄기를 지원하지 않습니다. 다만 DB 인스턴스에서 플러그인을 제거할 수는 있습니다. MariaDB 감사 플러그인을 제거할 때 DB 인스턴스가 자동으로 재시작하여 감사가 중지됩니다.

MariaDB 감사 플러그인을 DB 인스턴스에서 제거하려면 다음 중 하나를 실행하십시오.

- MariaDB 감사 플러그인이 속한 옵션 그룹에서 MariaDB 감사 플러그인 옵션을 제거합니다. 이 변경은 해당 옵션 그룹을 사용하는 모든 DB 인스턴스에 영향을 미칩니다. 자세한 내용은 옵션 그룹에서 옵션 제거 섹션을 참조하세요.
- DB 인스턴스를 수정하고, 플러그인이 포함되지 않은 다른 옵션 그룹을 지정합니다. 이 변경은 단일 DB 인스턴스에 영향을 미칩니다. 기본(빈) 옵션 그룹을 지정하거나 다른 사용자 지정 옵션 그룹을 지정할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

MariaDB에 대한 파라미터

기본적으로, MariaDB DB 인스턴스는 MariaDB 데이터베이스에만 해당되는 DB 파라미터 그룹을 사용합니다. 이 파라미터 그룹에는 MySQL 데이터베이스 엔진용 Amazon RDS DB 파라미터 그룹의 파라미터 중 일부만 포함되어 있습니다. 또한 몇 개의 새로운 MariaDB 고유 파라미터도 포함되어 있습니다. 파라미터 그룹 작업 및 파라미터 설정에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.

MariaDB 파라미터 보기

RDS for MariaDB 파라미터는 사용자가 선택한 스토리지 엔진의 기본값으로 설정됩니다. MariaDB 파라미터에 대한 자세한 내용은 MariaDB 설명서를 참조하세요. MariaDB 스토리지 엔진에 대한 자세한 내용은 Amazon RDS MariaDB에 대해 지원되는 스토리지 엔진 섹션을 참조하세요.

RDS 콘솔이나 AWS CLI를 사용하여 특정 RDS for MariaDB 버전에 대해 사용할 수 있는 파라미터를 볼 수 있습니다. RDS 콘솔의 파라미터 그룹에서 MariaDB 파라미터 보기에 대한 자세한 정보는 Amazon RDS에서 DB 파라미터 그룹의 파라미터 값 보기 섹션을 참조하세요.

AWS CLI로 `describe-engine-default-parameters` 명령을 실행하여 RDS for MariaDB 버전의 파라미터를 볼 수 있습니다. `--db-parameter-group-family` 옵션에 대해 다음 값 중 하나를 지정할 수 있습니다.

- `mariadb10.11`
- `mariadb10.6`
- `mariadb10.5`
- `mariadb10.4`
- `mariadb10.3`

예를 들어 RDS for MariaDB 버전 10.6에 대한 파라미터를 보려면 다음 명령을 실행합니다.

```
aws rds describe-engine-default-parameters --db-parameter-group-family mariadb10.6
```

출력 결과는 다음과 비슷합니다.

```
{
    "EngineDefaults": {
        "Parameters": [
```

```
            {
                "ParameterName": "alter_algorithm",
                "Description": "Specify the alter table algorithm.",
                "Source": "engine-default",
                "ApplyType": "dynamic",
                "DataType": "string",
                "AllowedValues": "DEFAULT,COPY,INPLACE,NOCOPY,INSTANT",
                "IsModifiable": true
            },
            {
                "ParameterName": "analyze_sample_percentage",
                "Description": "Percentage of rows from the table ANALYZE TABLE will
 sample to collect table statistics.",
                "Source": "engine-default",
                "ApplyType": "dynamic",
                "DataType": "float",
                "AllowedValues": "0-100",
                "IsModifiable": true
            },
            {
                "ParameterName": "aria_block_size",
                "Description": "Block size to be used for Aria index pages.",
                "Source": "engine-default",
                "ApplyType": "static",
                "DataType": "integer",
                "AllowedValues": "1024-32768",
                "IsModifiable": false
            },
            {
                "ParameterName": "aria_checkpoint_interval",
                "Description": "Interval in seconds between automatic checkpoints.",
                "Source": "engine-default",
                "ApplyType": "dynamic",
                "DataType": "integer",
                "AllowedValues": "0-4294967295",
                "IsModifiable": true
            },
        ...
```

RDS for MariaDB 버전 10.6에 대한 수정 가능 파라미터를 나열하려면 다음 명령을 실행합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-engine-default-parameters --db-parameter-group-family mariadb10.6 \
```

```
   --query 'EngineDefaults.Parameters[?IsModifiable==`true`]'
```

Windows의 경우:

```
aws rds describe-engine-default-parameters --db-parameter-group-family mariadb10.6 ^
   --query "EngineDefaults.Parameters[?IsModifiable==`true`]"
```

사용할 수 없는 MySQL 파라미터입니다.

다음의 MySQL 파라미터는 MariaDB 고유의 DB 파라미터 그룹에서 사용할 수 없습니다.

- bind_address
- binlog_error_action
- binlog_gtid_simple_recovery
- binlog_max_flush_queue_time
- binlog_order_commits
- binlog_row_image
- binlog_rows_query_log_events
- binlogging_impossible_mode
- block_encryption_mode
- core_file
- default_tmp_storage_engine
- div_precision_increment
- end_markers_in_json
- enforce_gtid_consistency
- eq_range_index_dive_limit
- explicit_defaults_for_timestamp
- gtid_executed
- gtid-mode
- gtid_next
- gtid_owned
- gtid_purged
- log_bin_basename

- log_bin_index
- log_bin_use_v1_row_events
- log_slow_admin_statements
- log_slow_slave_statements
- log_throttle_queries_not_using_indexes
- master-info-repository
- optimizer_trace
- optimizer_trace_features
- optimizer_trace_limit
- optimizer_trace_max_mem_size
- optimizer_trace_offset
- relay_log_info_repository
- rpl_stop_slave_timeout
- slave_parallel_workers
- slave_pending_jobs_size_max
- slave_rows_search_algorithms
- storage_engine
- table_open_cache_instances
- timed_mutexes
- transaction_allow_batching
- validate-password
- validate_password_dictionary_file
- validate_password_length
- validate_password_mixed_case_count
- validate_password_number_count
- validate_password_policy
- validate_password_special_char_count

MySQL 파라미터에 대한 자세한 내용은 MySQL 설명서를 참조하세요.

MySQL DB 스냅샷에서 MariaDB DB 인스턴스로 데이터 마이그레이션

AWS Management Console, AWS CLI 또는 Amazon RDS API를 사용하여 MariaDB를 실행하는 새 DB 인스턴스로 RDS for MySQL DB 스냅샷을 마이그레이션할 수 있습니다. MySQL 5.6 또는 5.7을 실행하는 Amazon RDS DB 인스턴스에서 생성된 DB 스냅샷을 사용해야 합니다. RDS for MySQL DB 스냅샷을 생성하는 방법은 Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성 섹션을 참조하세요.

스냅샷을 마이그레이션해도 스냅샷이 생성된 원본 DB 인스턴스에는 영향을 주지 않습니다. 원래 DB 인스턴스를 대체하려고 트래픽을 전환하기 전에 새 DB 인스턴스를 테스트하고 검증할 수 있습니다.

MySQL에서 MariaDB로 마이그레이션하면 MariaDB DB 인스턴스는 기본 DB 파라미터 그룹 및 옵션 그룹과 연동됩니다. DB 스냅샷을 복원한 후에는 새 DB 인스턴스와 사용자 지정 DB 파라미터 그룹을 연동할 수 있습니다. 그러나 MariaDB 파라미터 그룹에는 구성 가능한 다양한 시스템 변수 집합이 있습니다. MySQL 및 MariaDB 시스템 변수의 차이에 대한 정보는 MariaDB 및 MySQL 간 시스템 변수 차이점을 참조하세요. DB 파라미터 그룹에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오. 옵션 그룹에 대한 자세한 내용은 옵션 그룹 작업 단원을 참조하십시오.

마이그레이션 수행

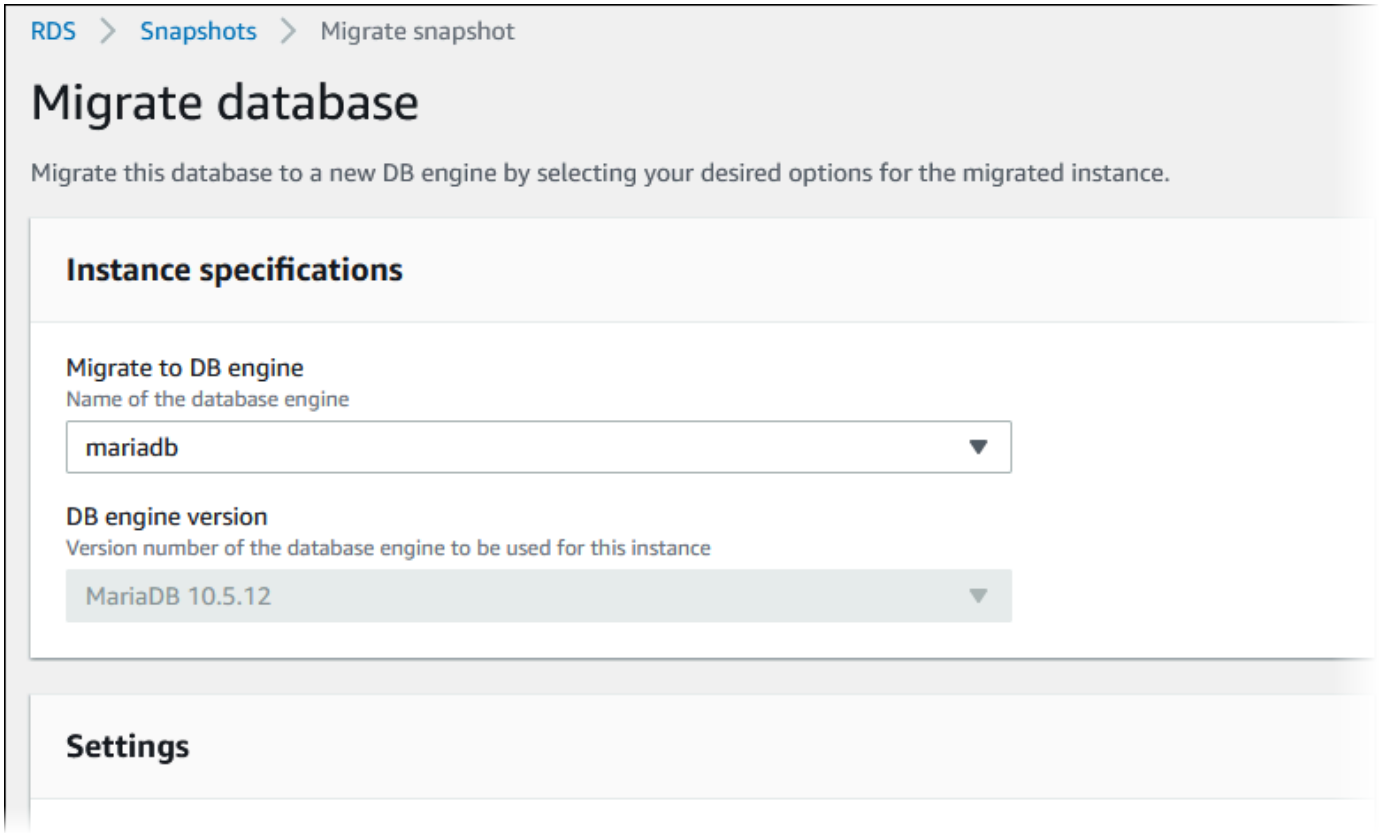
AWS Management Console, AWS CLI 또는 RDS API를 사용하여 RDS for MySQL DB 스냅샷을 새 MariaDB DB 인스턴스로 마이그레이션할 수 있습니다.

콘솔

MySQL DB 스냅샷에서 MariaDB DB 인스턴스로 마이그레이션하려면

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택한 다음 마이그레이션하려는 MySQL DB 스냅샷을 선택합니다.
3. 작업(Actions)에서 스냅샷 마이그레이션(Migrate snapshot)을 선택합니다. 데이터베이스 마이그레이션 페이지가 표시됩니다.
4. [Migrate to DB Engine]에서 [mariadb]를 선택합니다.

   Amazon RDS에서 DB 엔진 버전(DB engine version)을 자동으로 선택합니다. DB 엔진 버전은 변경할 수 없습니다.



5. 나머지 섹션에서 DB 인스턴스 설정을 지정합니다. 각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하세요.
6. [Migrate]를 선택합니다.

AWS CLI

MySQL DB 스냅샷에서 MariaDB DB 인스턴스로 데이터를 마이그레이션하려면 AWS CLI `restore-db-instance-from-db-snapshot` 명령을 다음 옵션과 함께 실행합니다.

- --db-instance-identifier – DB 스냅샷에서 생성할 DB 인스턴스의 이름.
- --db-snapshot-identifier – 복구할 DB 스냅샷에 대한 식별자.
- --engine – 새 인스턴스에 사용할 데이터베이스 엔진.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds restore-db-instance-from-db-snapshot \
    --db-instance-identifier newmariadbinstance \
```

```
    --db-snapshot-identifier mysqlsnapshot \
    --engine mariadb
```

Windows의 경우:

```
aws rds restore-db-instance-from-db-snapshot ^
    --db-instance-identifier newmariadbinstance ^
    --db-snapshot-identifier mysqlsnapshot ^
    --engine mariadb
```

API

MySQL DB 스냅샷에서 MariaDB DB 인스턴스로 데이터를 마이그레이션하려면 Amazon RDS API 작업 RestoreDBInstanceFromDBSnapshot을 호출합니다.

MariaDB와 MySQL 간의 호환성 관련 문제

MariaDB와 MySQL 간의 비호환성 관련 문제로는 다음과 같은 것들이 있습니다.

- MySQL 8.0으로 생성된 DB 스냅샷은 MariaDB로 마이그레이션할 수 없습니다.
- 원본 MySQL 데이터베이스에서 SHA256 암호 해시를 사용하는 경우, MariaDB 데이터베이스에 연결하려면 먼저 SHA256 해시 처리된 사용자 암호를 재설정해야 합니다. 다음 코드는 SHA256 해시 처리된 암호를 재설정하는 방법을 보여줍니다.

  ```
  SET old_passwords = 0;
  UPDATE mysql.user SET plugin = 'mysql_native_password',
  Password = PASSWORD('new_password')
  WHERE (User, Host) = ('master_user_name', %);
  FLUSH PRIVILEGES;
  ```

- RDS 기본 사용자 계정에서 SHA-256 암호 해시를 사용하는 경우 AWS Management Console, modify-db-instance AWS CLI 명령 또는 ModifyDBInstance RDS API 작업을 통해 암호를 재설정해야 합니다. DB 인스턴스 수정에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하세요.
- MariaDB는 Memcached 플러그 인을 지원하지 않습니다. 그러나 Memcached 플러그 인이 사용하는 데이터는 InnoDB 테이블 형식으로 저장됩니다. MySQL DB 스냅샷을 마이그레이션한 후에는 SQL을 사용하여 Memcached 플러그인이 사용하는 데이터에 액세스할 수 있습니다. innodb_memcache 데이터베이스에 대한 자세한 내용은 InnoDB memcached Plugin Internals를 참조하십시오.

Amazon RDS SQL의 MariaDB 참조

아래에는 MariaDB DB 엔진을 실행 중인 Amazon RDS 인스턴스에 사용할 수 있는 시스템 저장 프로시저에 대한 설명이 나와 있습니다.

MySQL DB 인스턴스 및 MariaDB DB 인스턴스에 사용할 수 있는 시스템 저장 프로시저를 사용할 수 있습니다. 이러한 저장 프로시저는 RDS for MySQL 저장 프로시저 참조에 소개되어 있습니다. MariaDB DB 인스턴스는 `mysql.rds_start_replication_until` 및 `mysql.rds_start_replication_until_gtid`를 제외한 모든 저장 프로시저를 지원합니다.

또한 다음은 MariaDB를 실행 중인 Amazon RDS DB 인스턴스에 대해서만 지원되는 시스템 저장 프로시저입니다.

- mysql.rds_replica_status
- mysql.rds_set_external_master_gtid
- mysql.rds_kill_query_id

mysql.rds_replica_status

MariaDB 읽기 전용 복제본의 복제 상태를 표시합니다.

읽기 전용 복제본에서 이 프로시저를 호출하여 복제본 스레드의 필수 파라미터에 대한 상태 정보를 표시합니다.

구문

```
CALL mysql.rds_replica_status;
```

사용 노트

이 프로시저는 MariaDB 버전 10.5 이상을 실행하는 MariaDB DB 인스턴스에 대해서만 지원됩니다.

이 프로시저는 `SHOW REPLICA STATUS` 명령과 동일합니다. MariaDB 버전 10.5 이상 DB 인스턴스에서는 이 명령이 지원되지 않습니다.

이전 버전의 MariaDB에서 상응하는 `SHOW SLAVE STATUS` 명령에는 `REPLICATION SLAVE` 권한이 필요했습니다. MariaDB 버전 10.5 이상에서는 `REPLICATION REPLICA ADMIN` 권한이 필요합니다.

MariaDB 10.5 이상 DB 인스턴스의 RDS 관리 권한을 보호하기 위해 이 새 권한은 RDS 마스터 사용자에게 부여되지 않습니다.

예시

다음 예에서는 MariaDB 읽기 전용 복제본의 상태를 보여 줍니다.

```
call mysql.rds_replica_status;
```

응답은 다음과 유사합니다.

```
*************************** 1. row ***************************
               Replica_IO_State: Waiting for master to send event
                    Source_Host: XX.XX.XX.XXX
                    Source_User: rdsrepladmin
                    Source_Port: 3306
                  Connect_Retry: 60
                Source_Log_File: mysql-bin-changelog.003988
            Read_Source_Log_Pos: 405
                 Relay_Log_File: relaylog.011024
                  Relay_Log_Pos: 657
          Relay_Source_Log_File: mysql-bin-changelog.003988
             Replica_IO_Running: Yes
            Replica_SQL_Running: Yes
                Replicate_Do_DB:
            Replicate_Ignore_DB:
             Replicate_Do_Table:
         Replicate_Ignore_Table:
mysql.rds_sysinfo,mysql.rds_history,mysql.rds_replication_status
        Replicate_Wild_Do_Table:
    Replicate_Wild_Ignore_Table:
                     Last_Errno: 0
                     Last_Error:
                   Skip_Counter: 0
            Exec_Source_Log_Pos: 405
                Relay_Log_Space: 1016
                Until_Condition: None
                 Until_Log_File:
                  Until_Log_Pos: 0
             Source_SSL_Allowed: No
             Source_SSL_CA_File:
             Source_SSL_CA_Path:
                Source_SSL_Cert:
```

```
               Source_SSL_Cipher:
                  Source_SSL_Key:
           Seconds_Behind_Master: 0
   Source_SSL_Verify_Server_Cert: No
                   Last_IO_Errno: 0
                   Last_IO_Error:
                  Last_SQL_Errno: 0
                  Last_SQL_Error:
     Replicate_Ignore_Server_Ids:
                Source_Server_Id: 807509301
                  Source_SSL_Crl:
              Source_SSL_Crlpath:
                      Using_Gtid: Slave_Pos
                     Gtid_IO_Pos: 0-807509301-3980
         Replicate_Do_Domain_Ids:
     Replicate_Ignore_Domain_Ids:
                   Parallel_Mode: optimistic
                       SQL_Delay: 0
             SQL_Remaining_Delay: NULL
       Replica_SQL_Running_State: Reading event from the relay log
              Replica_DDL_Groups: 15
Replica_Non_Transactional_Groups: 0
    Replica_Transactional_Groups: 3658
1 row in set (0.000 sec)

Query OK, 0 rows affected (0.000 sec)
```

mysql.rds_set_external_master_gtid

Amazon RDS 외부에서 실행 중인 MariaDB 인스턴스에서 MariaDB DB 인스턴스로의 GTID 기반 복제를 구성합니다. 이 프로시저는 외부 MariaDB 인스턴스가 버전 10.0.24 이상인 경우에만 지원됩니다. 두 인스턴스 중 하나 또는 모두가 MariaDB 전역 트랜잭션 ID(GTID)를 지원하지 않는 복제를 설정하는 경우 다음(mysql.rds_set_external_master(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하))을 사용하십시오.

복제 시 GTID를 사용하면 이진 로그 복제에서는 제공하지 않는 충돌 안정성 기능을 제공하므로 복제 인스턴스가 지원하는 경우에는 GTID 기반 복제를 사용하는 것이 좋습니다.

구문

```
CALL mysql.rds_set_external_master_gtid(
```

```
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , gtid
  , ssl_encryption
);
```

파라미터

host_name

문자열. 소스 인스턴스가 될 Amazon RDS 외부에서 실행 중인 MariaDB 인스턴스의 호스트 이름 또는 IP 주소입니다.

host_port

정수. 소스 인스턴스로 구성될 Amazon RDS 외부에서 실행 중인 MariaDB 인스턴스에서 사용하는 포트입니다. 네트워크 구성에 포트 번호를 변환하는 SSH 포트 복제가 포함되는 경우 SSH에 의해 공개되는 포트 이름을 지정하십시오.

replication_user_name

문자열. 읽기 전용 복제본으로 구성될 MariaDB DB 인스턴스에서 `REPLICATION SLAVE` 권한이 있는 사용자의 ID입니다.

replication_user_password

문자열. `replication_user_name`에 지정된 사용자 ID의 암호입니다.

gtid

문자열. 복제를 시작해야 하는 소스 인스턴스의 전역 트랜잭션 ID.

복제를 구성하는 동안 소스 인스턴스가 잠긴다면 `@@gtid_current_pos`를 사용하여 현재 GTID를 가져올 수 있습니다. 그러므로 GTID를 가져오는 시점과 복제가 시작하는 시점 사이에서 이진 로그가 변경되지 않습니다.

또는 복제를 시작하기 전에 `mysqldump` 버전 10.0.13 이상을 사용하여 복제본 인스턴스를 채우는 경우 `--master-data` 또는 `--dump-slave` 옵션을 사용하여 출력 내 GTID 위치를 가져올 수 있습니다. `mysqldump` 버전 10.0.13 이상을 사용하지 않는 경우에는 `SHOW MASTER STATUS`를 실행하거나 동일한 `mysqldump` 옵션을 사용하여 이진 로그 파일 이름 및 위치를 가져온 다음, 외부 MariaDB 인스턴스에서 `BINLOG_GTID_POS`옵션을 실행해 GTID로 변환할 수 있습니다.

```
SELECT BINLOG_GTID_POS('<binary log file name>', <binary log file position>);
```

GTID의 MariaDB 구현에 대한 자세한 내용은 MariaDB 설명서에서 전역 트랜잭션 ID를 참조하십시오.

ssl_encryption

복제 연결에 보안 소켓 계층(SSL) 암호화를 사용할지 여부를 지정하는 값입니다. 1은 SSL 암호화 사용, 0은 암호화 사용 안 함입니다. 기본값은 0입니다.

> **Note**
>
> MASTER_SSL_VERIFY_SERVER_CERT 옵션은 지원되지 않습니다. 이 옵션은 0으로 설정되어 있는데, 이는 연결이 암호화되었지만 인증서는 확인되지 않았음을 의미합니다.

사용 노트

`mysql.rds_set_external_master_gtid` 프로시저는 마스터 사용자가 실행해야 합니다. Amazon RDS 외부에서 실행 중인 MariaDB DB 인스턴스의 복제본으로 구성하려는 MariaDB DB 인스턴스에서 실행해야 합니다. `mysql.rds_set_external_master_gtid`를 실행하기 전에 Amazon RDS 외부에서 실행 중인 MariaDB의 인스턴스를 소스 인스터스로 구성해야 합니다. 자세한 내용은 Amazon RDS for MariaDB DB 인스턴스로 데이터 가져오기 섹션을 참조하세요.

> **Warning**
>
> `mysql.rds_set_external_master_gtid`를 사용하여 두 Amazon RDS DB 인스턴스 간 복제를 관리하지 마십시오. RDS 외부에서 실행 중인 MariaDB 인스턴스를 복제할 때만 사용합니다. Amazon RDS DB 인스턴스 간 복제 관리에 대한 자세한 내용은 DB 인스턴스 읽기 전용 복제본 작업 주제를 참조하십시오.

`mysql.rds_set_external_master_gtid`를 호출하여 Amazon RDS DB 인스턴스를 읽기 전용 복제본으로 구성한 후 복제본에서 mysql.rds_start_replication을 호출하여 복제 프로세스를 시작할 수 있습니다. mysql.rds_reset_external_master(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하)를 호출하여 읽기 전용 복제본 구성을 제거할 수 있습니다.

`mysql.rds_set_external_master_gtid`가 호출되면 Amazon RDS에서 `mysql.rds_history` 및 `mysql.rds_replication_status` 테이블에 시간, 사용자 및 "set master" 작업을 기록합니다.

예시

MariaDB DB 인스턴스에서 다음 예제를 실행하면 Amazon RDS 외부에서 실행 중인 MariaDB 인스턴스의 복제본으로 구성됩니다.

```
call mysql.rds_set_external_master_gtid
 ('Sourcedb.some.com',3306,'ReplicationUser','SomePassW0rd','0-123-456',0);
```

mysql.rds_kill_query_id

장기 실행 또는 문제가 있는 쿼리를 종료하기 위해 MariaDB 서버에 대해 실행되는 쿼리를 종료합니다. 쿼리 ID를 식별하고 특정 쿼리를 효과적으로 중지하여 성능 문제를 해결하고 최적의 데이터베이스 운영을 유지할 수 있습니다.

구문

```
CALL mysql.rds_kill_query_id(queryID);
```

파라미터

queryID

정수. 종료할 쿼리의 ID입니다.

사용 노트

MariaDB 서버에서 실행 중인 쿼리를 중지하려면 `mysql.rds_kill_query_id` 프로시저를 사용하여 해당 쿼리의 ID를 전달합니다. 쿼리 ID를 가져오려면, 다음과 같이 MariaDB Information Schema PROCESSLIST 테이블을 쿼리합니다

```
SELECT USER, HOST, COMMAND, TIME, STATE, INFO, QUERY_ID FROM
                INFORMATION_SCHEMA.PROCESSLIST WHERE USER = '<user name>';
```

MariaDB 서버와의 연결은 유지됩니다.

예시

다음 예제는 쿼리 ID가 230040인 쿼리를 종료합니다.

```
call mysql.rds_kill_query_id(230040);
```

MariaDB DB 인스턴스의 현지 시간대

기본적으로 MariaDB DB 인스턴스의 시간대는 협정 세계시(UTC)입니다. 대신 DB 인스턴스의 시간대를 애플리케이션의 현지 시간대로 설정할 수 있습니다.

DB 인스턴스의 현지 시간대를 설정하려면 DB 인스턴스의 파라미터 그룹에서 `time_zone` 파라미터를 이 섹션의 뒤에 나오는 지원되는 값 중 하나로 설정합니다. 파라미터 그룹에 대한 `time_zone` 파라미터를 설정하면 해당 파라미터 그룹을 사용 중인 모든 DB 인스턴스와 읽기 전용 복제본이 새로운 현지 시간대를 사용하도록 변경됩니다. 파라미터 그룹에서 파라미터를 설정하는 방법에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하세요.

현지 시간대를 설정하면 데이터베이스에 대한 모든 새 연결에 변경 사항이 반영됩니다. 현지 시간대를 변경할 때 데이터베이스에 대해 열린 연결이 있는 경우 연결을 닫고 새 연결을 열어야 현지 시간대 업데이트가 표시됩니다.

DB 인스턴스와 하나 이상의 읽기 전용 복제본에 대해 서로 다른 현지 시간대를 설정할 수 있습니다. 이렇게 하려면 DB 인스턴스와 복제본에 대해 서로 다른 파라미터 그룹을 사용하고 각 파라미터 그룹에서 `time_zone` 파라미터를 다른 현지 시간대로 설정합니다.

AWS 리전 간 복제 중인 경우 소스 DB 인스턴스와 읽기 전용 복제본은 서로 다른 파라미터 그룹을 사용합니다. 파라미터 그룹은 AWS 리전마다 고유합니다. 각 인스턴스에 대해 동일한 현지 시간대를 사용하려면 인스턴스의 파라미터 그룹과 읽기 전용 복제본의 파라미터 그룹에서 `time_zone` 파라미터를 설정해야 합니다.

DB 스냅샷에서 DB 인스턴스를 복원할 경우 현지 시간대가 UTC로 설정됩니다. 복원이 완료된 후 시간대를 현지 시간대로 업데이트할 수 있습니다. DB 인스턴스를 특정 시점으로 복원할 경우 복원된 DB 인스턴스의 현지 시간대는 복원된 DB 인스턴스의 파라미터 그룹에서 설정한 시간대입니다.

IANA(Internet Assigned Numbers Authority)에서는 https://www.iana.org/time-zones에서 일 년에 여러 번 새로운 표준 시간대를 게시합니다. RDS에서 MariaDB의 새로운 마이너 유지 관리 릴리스를 릴리스할 때마다 릴리스 시점의 최신 표준 시간대 데이터가 함께 제공됩니다. 최신 RDS for MariaDB 버전을 사용하면 RDS의 최신 표준 시간대 데이터를 갖게 됩니다. DB 인스턴스에 최신 표준 시간대 데이터가 있는지 확인하려면 상위 DB 엔진 버전으로 업그레이드하는 것이 좋습니다. 또는 MariaDB DB 인스턴스의 표준 시간대 테이블을 수동으로 수정할 수 있습니다. 이렇게 하려면 SQL 명령을 사용하거나 SQL 클라이언트에서 mysql_tzinfo_to_sql 도구를 실행할 수 있습니다. 표준 시간대 데이터를 수동으로 업데이트한 후 변경 사항을 적용하려면 DB 인스턴스를 재부팅합니다. RDS는 실행 중인 DB 인스턴스의 표준 시간대 데이터를 수정하거나 재설정하지 않습니다. 새 표준 시간대 데이터는 데이터베이스 엔진 버전 업그레이드를 수행할 때만 설치됩니다.

현지 시간대를 다음 값 중 하나로 설정할 수 있습니다.

영역	시간대
아프리카	Africa/Cairo, Africa/Casablanca, Africa/Harare, Africa/Monrovia, Africa/Nairobi, Africa/Tripoli, Africa/Windhoek
아메리카	America/Araguaina, America/Asuncion, America/Bogota, America/Buenos_Aires, America/Caracas, America/Chihuahua, America/Cuiaba, America/Denver, America/Fortaleza, America/Guatemala, America/Halifax, America/Manaus, America/Matamoros, America/Monterrey, America/Montevideo, America/Phoenix, America/Santiago, America/Tijuana
아시아	Asia/Amman, Asia/Ashgabat, Asia/Baghdad, Asia/Baku, Asia/Bangkok, Asia/Beirut, Asia/Calcutta, Asia/Damascus, Asia/Dhaka, Asia/Irkutsk, Asia/Jerusalem, Asia/Kabul, Asia/Karachi, Asia/Kathmandu, Asia/Krasnoyarsk, Asia/Magadan, Asia/Muscat, Asia/Novosibirsk, Asia/Riyadh, Asia/Seoul, Asia/Shanghai, Asia/Singapore, Asia/Taipei, Asia/Tehran, Asia/Tokyo, Asia/Ulaanbaatar, Asia/Vladivostok, Asia/Yakutsk, Asia/Yerevan
대서양	Atlantic/Azores
호주	Australia/Adelaide, Australia/Brisbane, Australia/Darwin, Australia/Hobart, Australia/Perth, Australia/Sydney
브라질	Brazil/DeNoronha, Brazil/East
캐나다	Canada/Newfoundland, Canada/Saskatchewan, Canada/Yukon
유럽	Europe/Amsterdam, Europe/Athens, Europe/Dublin, Europe/Helsinki, Europe/Istanbul, Europe/Kaliningrad Europe/Moscow, Europe/Paris, Europe/Prague, Europe/Sarajevo
태평양	Pacific/Auckland, Pacific/Fiji, Pacific/Guam, Pacific/Honolulu, Pacific/Samoa
US	US/Alaska, US/Central, US/East-Indiana, US/Eastern, US/Pacific
UTC	UTC

RDS for MariaDB에 대해 알려진 문제 및 제한 사항

다음 항목은 RDS for MariaDB를 사용할 때 알려진 문제 및 제한 사항입니다.

> **Note**
>
> 단, 이 목록이 전부는 아닙니다.

주제

- Amazon RDS의 MariaDB 파일 크기 제한
- InnoDB 예약어
- 사용자 지정 포트
- 성능 개선 도우미

Amazon RDS의 MariaDB 파일 크기 제한

MariaDB DB 인스턴스의 경우 InnoDB 테이블당 파일 테이블스페이스를 사용할 때 테이블의 최대 크기는 16TB입니다. 또한 이 제한은 시스템 테이블스페이스를 최대 16TB의 크기로 제한합니다. MariaDB DB 인스턴스에서는 테이블이 각각 자체 테이블스페이스에 들어 있는 InnoDB 테이블당 파일 테이블스페이스가 기본적으로 설정됩니다. 이 한도는 MariaDB DB 인스턴스의 최대 스토리지 한도와 관련이 없습니다. 스토리지 한도에 대한 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 섹션을 참조하세요.

애플리케이션에 따라 InnoDB 테이블당 파일 테이블스페이스 사용에 대한 장점과 단점은 서로 다릅니다. 애플리케이션에 가장 적합한 접근 방식을 확인하려면 MySQL 설명서의 테이블당 파일 테이블스페이스를 참조하세요.

테이블을 최대 파일 크기로 늘리도록 허용하는 것은 권장하지 않습니다. 일반적으로 모범 사례는 성능 및 복구 시간을 향상할 수 있도록 데이터를 더 작은 테이블로 분할하는 것입니다.

대형 테이블을 여러 개의 작은 테이블로 분할하는 데 사용할 수 있는 한 가지 옵션으로는 파티셔닝이 있습니다. 파티셔닝을 수행하면 사용자가 지정하는 규칙에 따라 라지 테이블의 일부가 개별 파일로 배포됩니다. 예를 들어, 트랜잭션을 날짜별로 저장하는 경우 파티셔닝을 사용하여 이전 트랜잭션을 개별 파일로 배포하는 파티셔닝 규칙을 생성할 수 있습니다. 이렇게 하면 애플리케이션에서 즉시 사용할 필요가 없는 이전 트랜잭션 데이터를 주기적으로 보관할 수 있습니다. 자세한 내용은 MySQL 설명서의 파티셔닝을 참조하세요.

모든 InnoDB 테이블스페이스의 크기를 확인하려면

- 다음 SQL 명령을 사용하여 크기가 너무 커서 파티셔닝을 수행해야 하는 테이블이 있는지 확인합니다.

> **Note**
>
> MariaDB 10.6 이상의 경우 이 쿼리는 InnoDB 시스템 테이블스페이스의 크기도 반환합니다.
> 10.6 이전의 MariaDB 버전에서는 시스템 테이블을 쿼리하여 InnoDB 시스템 테이블스페이스의 크기를 확인할 수 없습니다. 이후 버전으로 업그레이드하는 것이 좋습니다.

```
SELECT SPACE,NAME,ROUND((ALLOCATED_SIZE/1024/1024/1024), 2)
as "Tablespace Size (GB)"
FROM information_schema.INNODB_SYS_TABLESPACES ORDER BY 3 DESC;
```

비 InnoDB 사용자 테이블의 크기를 확인하려면

- 다음 SQL 명령을 사용하여 비 InnoDB 사용자 테이블이 너무 큰지 확인합니다.

```
SELECT TABLE_SCHEMA, TABLE_NAME, round(((DATA_LENGTH + INDEX_LENGTH+DATA_FREE)
/ 1024 / 1024/ 1024), 2) As "Approximate size (GB)" FROM information_schema.TABLES
WHERE TABLE_SCHEMA NOT IN ('mysql', 'information_schema', 'performance_schema')
and ENGINE<>'InnoDB';
```

InnoDB 테이블당 파일 테이블스페이스를 활성화하는 방법

- DB 인스턴스의 파라미터 그룹에서 `innodb_file_per_table` 파라미터를 `1`로 설정합니다.

InnoDB 테이블당 파일 테이블스페이스를 비활성화하는 방법

- DB 인스턴스의 파라미터 그룹에서 `innodb_file_per_table` 파라미터를 `0`로 설정합니다.

파라미터 그룹 업데이트에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹을(를) 참조하세요.

InnoDB 테이블당 파일 테이블스페이스를 활성화하거나 비활성화하면 ALTER TABLE 명령을 실행할 수 있습니다. 이 명령을 사용하여 전역 테이블스페이스에서 자체 테이블스페이스로 테이블을 이동할 수 있습니다. 또는 자체 테이블스페이스에서 전역 테이블스페이스로 테이블을 이동할 수 있습니다. 다음은 그 한 예입니다.

```
ALTER TABLE table_name ENGINE=InnoDB, ALGORITHM=COPY;
```

InnoDB 예약어

InnoDB는 RDS for MariaDB의 예약어입니다. MariaDB 데이터베이스에는 이 이름을 사용할 수 없습니다.

사용자 지정 포트

Amazon RDS는 MariaDB 엔진의 사용자 지정 포트 33060으로의 연결을 차단합니다. MariaDB 엔진에 사용할 다른 포트를 선택합니다.

성능 개선 도우미

MariaDB 커뮤니티에서 더 이상 InnoDB 카운터를 지원하지 않기 때문에 RDS for MariaDB 버전 10.11용 성능 개선 도우미에는 InnoDB 카운터가 표시되지 않습니다.

Amazon RDS for Microsoft SQL Server

Amazon RDS는 Microsoft SQL Server의 여러 버전 및 에디션을 지원합니다. 다음 테이블은 각 메이저 버전에서 지원되는 최신 마이너 버전을 보여줍니다. 지원되는 버전, 에디션 및 RDS 엔진 버전의 전체 목록은 Amazon RDS의 Microsoft SQL Server 버전 단원을 참조하십시오.

메이저 버전	서비스 팩/GDR	누적 업데이트	마이너 버전	기술 자료 문서	릴리스 날짜
SQL Server 2022	GDR	CU13	16.0.4131.2	KB5040939	2024년 8월 12일
SQL Server 2019	해당 사항 없음	CU27	15.0.4382.1	KB5040948	2024년 8월 12일
SQL Server 2017	GDR	CU31	14.0.3471.2	KB5040940	2024년 8월 12일
SQL Server 2016	SP3 GDR	해당 사항 없음	13.0.6441.1	KB5040946	2024년 8월 12일

SQL Server 라이선스에 대한 자세한 내용은 Amazon RDS의 Microsoft SQL Server 라이선싱 단원을 참조하십시오. SQL Server 빌드에 대한 자세한 내용은 최신 SQL Server 빌드에 대한 정보를 찾을 수 있는 위치에 대한 Microsoft 지원 문서를 참조하세요.

Amazon RDS를 통해 DB 인스턴스 및 DB 스냅샷, 특정 시점으로 복구 및 자동 또는 수동 백업을 만들 수 있습니다. SQL Server를 실행 중인 DB 인스턴스를 VPC 내에서 사용할 수 있습니다. 보안 소켓 계층(SSL)을 사용하여 SQL 서버를 실행하는 DB 인스턴스에 연결할 수도 있고 투명 데이터 암호화(TDE)를 사용하여 유휴 데이터를 암호화할 수도 있습니다. Amazon RDS는 현재 SQL Server 데이터베이스 미러링(DBM) 또는 상시 가용성 그룹(AG)을 고가용성의 장애 조치 솔루션으로 사용하여 SQL Server용 다중 AZ 배포를 지원합니다.

관리되는 서비스 환경을 제공하기 위해 Amazon RDS는 DB 인스턴스에 대해 셸 액세스를 제공하지 않으며, 고급 권한을 필요로 하는 특정 시스템 절차와 테이블에 대한 액세스를 제한합니다. Amazon RDS는 Microsoft SQL Server Management Studio와 같은 표준 SQL 클라이언트 애플리케이션을 사용하여 DB 인스턴스에서 데이터베이스에 대한 액세스를 지원합니다. Amazon RDS에서는 Telnet,

SSH(Secure Shell) 또는 Windows 원격 데스크톱 연결을 통해 DB 인스턴스에 대한 직접 호스트 액세스를 허용하지 않습니다. DB 인스턴스를 생성할 때 마스터 사용자는 해당 인스턴스의 모든 사용자 데이터베이스에 대한 db_owner 역할을 할당받으며, 백업에 사용된 권한을 제외한 모든 데이터베이스 수준의 권한을 갖습니다. Amazon RDS는 백업을 관리합니다.

첫 번째 DB 인스턴스를 생성하기 전에 이 설명서의 설정 섹션에 나오는 단계를 완료해야 합니다. 자세한 내용은 Amazon RDS 환경 설정 섹션을 참조하세요.

주제

- Amazon RDS에서 Microsoft SQL Server에 대한 공통 관리 작업
- Microsoft SQL Server DB 인스턴스의 한도
- Microsoft SQL Server를 위한 DB 인스턴스 클래스 지원
- Microsoft SQL Server 보안
- Microsoft SQL Server DB 인스턴스에 대한 규정 준수 프로그램 지원
- Microsoft SQL Server 인스턴스를 위한 SSL 지원
- Amazon RDS의 Microsoft SQL Server 버전
- Amazon RDS의 버전 관리
- Amazon RDS의 Microsoft SQL Server 기능
- Microsoft SQL Server DB 인스턴스에 대한 변경 데이터 캡처 지원
- 지원되지 않는 기능과 지원이 제한된 기능
- Microsoft SQL Server 데이터베이스 미러링 또는 상시 가동 가용성 그룹을 사용하여 다중 AZ 배포
- Transparent Data Encryption을 사용하여 유휴 데이터 암호화
- Amazon RDS for Microsoft SQL Server에 대한 함수 및 저장 프로시저
- Microsoft SQL Server DB 인스턴스의 현지 시간대
- Amazon RDS의 Microsoft SQL Server 라이선싱
- Microsoft SQL Server DB 인스턴스에 연결
- RDS for SQL Server를 사용하여 Active Directory 작업
- 새 SSL/TLS 인증서를 사용해 Microsoft SQL Server DB 인스턴스에 연결할 애플리케이션을 업데이트
- Microsoft SQL Server DB 엔진 업그레이드
- 기본 백업 및 복원 기능을 사용하여 SQL Server 데이터베이스 가져오기 및 내보내기
- Amazon RDS에서 Microsoft SQL Server용 읽기 전용 복제본 작업
- Amazon RDS for Microsoft SQL Server의 다중 AZ 배포

- Amazon RDS의 Microsoft SQL Server 추가 기능
- Microsoft SQL Server 데이터베이스 엔진의 옵션
- Amazon RDS for Microsoft SQL Server에 대한 일반 DBA 작업

Amazon RDS에서 Microsoft SQL Server에 대한 공통 관리 작업

다음은 Amazon RDS for SQL Server DB 인스턴스로 수행하는 일반적인 관리 작업과 각 작업에 해당하는 문서 링크입니다.

작업 영역	관련 설명서
인스턴스 클래스, 스토리지 및 PIOPS 프로덕션 목적으로 DB 인스턴스를 만들 경우에는 Amazon RDS에서 인스턴스 클래스, 스토리지 유형 및 프로비저닝된 IOPS이 작동하는 방식을 이해해야 합니다.	Microsoft SQL Server를 위한 DB 인스턴스 클래스 지원 Amazon RDS 스토리지 유형
다중 AZ 배포 프로덕션 DB 인스턴스에서는 다중 AZ 배포를 사용해야 합니다. 다중 AZ 배포는 DB 인스턴스를 위해 향상된 가용성, 데이터 내구성 및 내결함성을 제공합니다. SQL Server용 다중 AZ 배포는 SQL Server의 기본 DBM 또는 AG 기술을 사용하여 구현됩니다.	Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 Microsoft SQL Server 데이터베이스 미러링 또는 상시 가동 가용성 그룹을 사용하여 다중 AZ 배포
Amazon Virtual Private Cloud(VPC) AWS 계정에 기본 VPC가 있는 경우에는 DB 인스턴스가 기본 VPC 내부에 자동으로 생성됩니다. 계정에 기본 VPC가 없는데 VPC 안에 DB 인스턴스를 만들려면 VPC와 서브넷 그룹을 만든 후 DB 인스턴스를 만들어야 합니다.	VPC에서 DB 인스턴스를 사용한 작업
보안 그룹 기본적으로, DB 인스턴스와 함께 인스턴스에 대한 액세스를 막는 방화벽도 생성됩니다. 따라서 DB 인스턴스에 액세스하기 위한 알맞은 IP 주소와 네트워크 구성으로 보안 그룹을 만들어야 합니다.	보안 그룹을 통한 액세스 제어

작업 영역	관련 설명서
파라미터 그룹 수(Parameter groups) DB 인스턴스에 특정 데이터베이스 파라미터가 필요할 경우, 파라미터 그룹을 만든 후 DB 인스턴스를 만들어야 합니다.	Amazon RDS의 파라미터 그룹
옵션 그룹 수 DB 인스턴스에 특정 데이터베이스 옵션이 필요할 경우, 옵션 그룹을 만든 후 DB 인스턴스를 만들어야 합니다.	Microsoft SQL Server 데이터베이스 엔진의 옵션
DB 인스턴스에 연결 보안 그룹을 만들고 이를 DB 인스턴스에 연결한 후, Microsoft SQL Server Management Studio와 같은 표준 SQL 클라이언트 애플리케이션을 사용하여 DB 인스턴스에 연결할 수 있습니다.	Microsoft SQL Server DB 인스턴스에 연결
백업 및 복원 DB 인스턴스를 생성할 때 자동 백업을 하도록 구성할 수 있습니다. 또한 전체 백업 파일(.bak 파일)을 사용하여 데이터베이스를 수동으로 백업 및 복원할 수도 있습니다.	백업 소개 기본 백업 및 복원 기능을 사용하여 SQL Server 데이터베이스 가져오기 및 내보내기
모니터링(Monitoring) CloudWatch Amazon RDS 측정치, 이벤트 및 향상된 모니터링 기능을 통해 SQL Server DB 인스턴스를 모니터링할 수 있습니다.	Amazon RDS 콘솔에서 지표 보기 Amazon RDS 이벤트 보기
로그 파일 SQL Server DB 인스턴스의 로그 파일에 액세스할 수 있습니다.	Amazon RDS 로그 파일 모니터링 Amazon RDS for Microsoft SQL Server 데이터베이스 로그 파일

SQL Server DB 인스턴스 작업을 위한 고급 관리 작업도 있습니다. 자세한 내용은 다음 설명서를 참조하세요.

- Amazon RDS for Microsoft SQL Server에 대한 일반 DBA 작업.
- RDS for SQL Server를 사용하여 AWS 관리형 Active Directory 작업
- tempdb 데이터베이스에 액세스

Microsoft SQL Server DB 인스턴스의 한도

DB 인스턴스에서 Amazon RDS의 Microsoft SQL Server를 구현하려면 다음과 같은 몇 가지 제한 사항을 정확히 파악하고 있어야 합니다.

- DB 인스턴스에 지원되는 최대 데이터베이스 수는 인스턴스 클래스 유형과 가용성 모드—단일 AZ, 다중 AZ 데이터베이스 미러링(DBM) 또는 다중 AZ 가용성 그룹(AG)에 따라 다릅니다. Microsoft SQL Server 시스템 데이터베이스는 이 제한에 포함되지 않습니다.

  다음 표는 각 인스턴스 클래스 유형 및 가용성 모드에 대해 지원되는 최대 데이터베이스 수를 보여줍니다. 이 표를 사용하면 하나의 인스턴스 클래스 유형에서 다른 유형으로 또는 한 가용성 모드에서 다른 모드로 이동할 수 있는지 알 수 있습니다. 원본 DB 인스턴스에 대상 인스턴스 클래스 유형이나 가용성 모드에서 지원할 수 있는 것보다 많은 데이터베이스가 있으면 DB 인스턴스 수정에 실패합니다. 이벤트 창에서 요청 상태를 확인할 수 있습니다.

인스턴스 클래스 유형	단일 AZ	DBM 기능이 있는 다중 AZ	상시 가동 AG 기능이 있는 다중 AZ
db.*.micro~db.*.medium	30	해당 사항 없음	해당 사항 없음
db.*.large	30	30	30
db.*.xlarge~db.*.16xlarge	100	50	75
db.*.24xlarge	100	50	100

  *서로 다른 인스턴스 클래스 유형을 나타냅니다.

  예를 들어 DB 인스턴스가 단일 AZ를 사용한 db.*.16xlarge에서 실행되고 76개의 데이터베이스가 있다고 가정해봅시다. 다중 AZ 상시 가동 AG를 사용하여 업그레이드할 DB 인스턴스를 수정합니다. DB 인스턴스에 대상 구성에서 지원할 수 있는 것보다 많은 데이터베이스가 포함되어 있으므로 이 업

그레이드는 실패합니다. 대신 인스턴스 클래스 유형을 db.*.24xlarge로 업그레이드하면 수정이 완료됩니다.

업그레이드에 실패하면 다음과 유사한 이벤트 및 메시지가 표시됩니다.

- 데이터베이스 인스턴스 클래스를 수정할 수 없습니다. 인스턴스에는 76개의 데이터베이스가 있지만 변환 후에는 75개만 지원됩니다.
- DB 인스턴스를 다중 AZ로 변환할 수 없습니다. 인스턴스에 76개의 데이터베이스가 있지만 변환 후에는 75개만 지원합니다.

특정 시점으로 복구 또는 스냅샷 복원에 실패하면 다음과 유사한 이벤트 및 메시지가 표시됩니다.

- 데이터베이스 인스턴스가 복원 호환 장애 상태로 전환됩니다. 인스턴스에는 76개의 데이터베이스가 있지만 변환 후에는 75개만 지원됩니다.

- 다음 포트는 Amazon RDS용으로 예약되어 있고 DB 인스턴스 `1234`, `1434`, `3260`, `3343`, `3389`, `47001`, 및 `49152-49156`을 생성할 때 사용할 수 없습니다.
- 169.254.0.0/16 범위의 IP 주소에서 클라이언트 연결은 허용되지 않습니다. 이는 로컬 링크 주소 지정에 사용되는 APIPA(Automatic Private IP Addressing) 범위입니다.
- DB 인스턴스에 소프트웨어 제한(24코어, 4소켓 및 128GB RAM)보다 많은 프로세서가 있는 경우 SQL Server Standard Edition은 사용 가능한 프로세서의 하위 집합만 사용합니다. 이러한 예로는 db.m5.24xlarge 및 db.r5.24xlarge 인스턴스 클래스가 있습니다.

  자세한 내용은 Microsoft 설명서의 SQL Server 2019(15.x)의 버전과 지원하는 기능에서 확장 제한 표를 참조하세요.
- Amazon RDS for SQL Server는 msdb 데이터베이스로 데이터 가져오기를 지원하지 않습니다.
- SQL Server 다중 AZ 배포의 DB 인스턴스에 있는 데이터베이스의 이름은 바꿀 수 없습니다.
- RDS for SQL Server에서 다음 DB 파라미터를 설정할 때 다음 지침을 따라야 합니다.
  - `max server memory (mb)` >= 256MB
  - `max worker threads` >= (논리적 CPU 수 * 7)

  DB 파라미터 설정에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 섹션을 참조하세요.
- SQL Server DB 인스턴스를 위한 최대 스토리지 크기는 다음과 같습니다.
  - 범용(SSD) 스토리지 – 모든 에디션에서 16TiB
  - 프로비저닝된 IOPS 스토리지 – 모든 에디션에서 16TiB
  - 마그네틱 스토리지 – 모든 에디션에서 1TiB

대량의 스토리지가 필요한 상황에서는 여러 DB 인스턴스에서 샤딩을 사용하여 이 제한을 우회할 수 있습니다. 이 접근 방식에서는 샤딩된 시스템에 연결하는 애플리케이션에 데이터 종속적인 라우팅 논리가 필요합니다. 기존 샤딩 프레임워크를 사용하거나 샤딩을 활성화하는 사용자 지정 코드를 작성할 수 있습니다. 기존 프레임워크를 사용하는 경우, 이 프레임워크는 DB 인스턴스와 같은 서버에 어떤 구성 요소도 설치할 수 없습니다.

- SQL Server DB 인스턴스를 위한 최소 스토리지 크기는 다음과 같습니다.
  - 범용(SSD) 스토리지 – Enterprise, Standard, Web 및 Express Edition의 경우 20GiB
  - 프로비저닝된 IOPS 스토리지 – Enterprise, Standard, Web 및 Express Edition의 경우 20GiB
  - 마그네틱 스토리지 – Enterprise, Standard, Web 및 Express Edition의 경우 20GiB
- Amazon RDS는 RDS DB 인스턴스와 동일한 서버에서 이러한 서비스를 실행하는 것을 지원하지 않습니다.
  - 데이터 품질 서비스
  - 마스터 데이터 서비스

  이러한 기능을 사용하려면 Amazon EC2 인스턴스에 SQL Server를 설치하거나 온프레미스 SQL Server 인스턴스를 사용하는 것이 좋습니다. 이러한 경우 EC2 또는 SQL Server 인스턴스는 Amazon RDS에서 SQL Server DB 인스턴스의 마스터 데이터 서비스 서버 역할을 합니다. Microsoft 라이선싱 정책에 따라 SQL Server를 Amazon EBS 스토리지가 포함된 Amazon EC2 인스턴스에 설치할 수 있습니다.
- Microsoft SQL Server의 제한 사항 때문에, `DROP DATABASE`의 성공적인 실행 이전의 시점으로 복원해도 해당 시점에서 해당 데이터베이스의 상태가 반영되지 않을 수 있습니다. 예를 들어, 삭제된 데이터베이스는 일반적으로 `DROP DATABASE` 명령이 실행되기 전 최대 5분까지의 상태로 복원됩니다. 이 유형의 복원은 끊긴 데이터베이스에서 몇 분 동안 수행 된 트랜잭션을 복원할 수 없음을 뜻합니다. 이 문제를 피하려면 복원 작업이 완료된 후 `DROP DATABASE` 명령을 다시 실행할 수 있습니다. 데이터베이스를 삭제하면 그 데이터베이스에 대한 트랜잭션 로그가 삭제됩니다.
- SQL Server의 경우, DB 인스턴스를 만든 후에 데이터베이스를 만듭니다. 데이터베이스 이름은 일반적인 SQL Server 명명 규칙을 따르지만 다음과 같은 차이점이 있습니다.
  - `rdsadmin`로 시작할 수 없습니다.
  - 공백 또는 탭으로 시작하거나 끝날 수 없습니다.
  - 새로운 줄을 생성하는 문자가 포함될 수 없습니다.
  - 작은 따옴표(')가 포함될 수 없습니다.
  - 현재 RDS for SQL Server는 자동 마이너 버전 업데이트를 지원하지 않습니다. 자세한 내용은 Amazon RDS의 버전 관리 단원을 참조하십시오.

- SQL Server Web Edition에서는 새 RDS for SQL Server DB 인스턴스를 생성할 때만 개발 및 테스트 템플릿을 사용할 수 있습니다.

Microsoft SQL Server를 위한 DB 인스턴스 클래스 지원

DB 인스턴스의 계산 및 메모리 용량은 해당 DB 인스턴스 클래스에 의해 결정됩니다. 필요한 DB 인스턴스 클래스는 DB 인스턴스의 처리력 및 메모리 요구 사항에 따라 다릅니다. 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요.

편의를 위해 Microsoft SQL Server에 지원되는 다음 DB 인스턴스 클래스 목록이 여기에서 제공됩니다. 최신 목록은 RDS console: https://console.aws.amazon.com/rds/를 참조하세요.

지원되는 SQL Server 마이너 버전에서 일부 DB 인스턴스 클래스는 사용할 수 없습니다. 예를 들어 db.r6i와 같은 일부 최신 DB 인스턴스 클래스는 이전 마이너 버전에서는 지원되지 않습니다. describe-orderable-db-instance-options AWS CLI 명령을 사용하여 SQL Server 에디션 및 버전에 사용할 수 있는 DB 인스턴스 클래스를 확인할 수 있습니다.

SQL Server Edition	2022 지원 범위	2019 지원 범위	2017 및 2016 지원 범위
Enterprise Edition	db.t3.xlarge –db.t3.2xlarge db.r5.large –db.r5.24xlarge db.r5b.large –db.r5b.24xlarge db.r5d.large –db.r5d.24xlarge db.r6i.large –db.r6i.32xlarge	db.t3.xlarge –db.t3.2xlarge db.r5.xlarge –db.r5.24xlarge db.r5b.xlarge –db.r5b.24xlarge db.r5d.xlarge –db.r5d.24xlarge db.r6i.xlarge –db.r6i.32xlarge	db.t3.xlarge –db.t3.2xlarge db.r5.xlarge –db.r5.24xlarge db.r5b.xlarge –db.r5b.24xlarge db.r5d.xlarge –db.r5d.24xlarge db.r6i.xlarge –db.r6i.32xlarge

SQL Server Edition	2022 지원 범위	2019 지원 범위	2017 및 2016 지원 범위
	db.m5.large –db.m5.24xlarge	db.m5.xlarge –db.m5.24xlarge	db.m5.xlarge –db.m5.24xlarge
	db.m5d.large –db.m5d.24xlarge	db.m5d.xlarge –db.m5d.24xlarge	db.m5d.xlarge –db.m5d.24xlarge
	db.m6i.large –db.m6i.32xlarge	db.m6i.xlarge –db.m6i.32xlarge	db.m6i.xlarge –db.m6i.32xlarge
	db.x2iedn.xlarge –db.x2iedn.32xlarge	db.x1.16xlarge –db.x1.32xlarge	db.x1.16xlarge –db.x1.32xlarge
	db.z1d.large –db.z1d.12xlarge	db.x1e.xlarge –db.x1e.32xlarge	db.x1e.xlarge –db.x1e.32xlarge
		db.x2iedn.xlarge –db.x2iedn.32xlarge	db.x2iedn.xlarge –db.x2iedn.32xlarge
		db.z1d.xlarge –db.z1d.12xlarge	db.z1d.xlarge –db.z1d.12xlarge

SQL Server Edition	2022 지원 범위	2019 지원 범위	2017 및 2016 지원 범위
Standard Edition	db.t3.xlarge –db.t3.2xlarge db.r5.large –db.r5.24xlarge db.r5b.large –db.r5b.8xlarge db.r5d.large –db.r5d.24xlarge db.r6i.large –db.r6i.8xlarge db.m5.large –db.m5.24xlarge db.m5d.large –db.m5d.24xlarge db.m6i.large –db.m6i.8xlarge db.x2iedn.xlarge –db.x2iedn.8xlarge db.z1d.large –db.z1d.12xlarge	db.t3.xlarge –db.t3.2xlarge db.r5.large –db.r5.24xlarge db.r5b.large –db.r5b.24xlarge db.r5d.large –db.r5d.24xlarge db.r6i.large –db.r6i.8xlarge db.m5.large –db.m5.24xlarge db.m5d.large –db.m5d.24xlarge db.m6i.large –db.m6i.8xlarge db.x1.16xlarge –db.x1.32xlarge db.x1e.xlarge –db.x1e.32xlarge	db.t3.xlarge –db.t3.2xlarge db.r5.large –db.r5.24xlarge db.r5b.large –db.r5b.24xlarge db.r5d.large –db.r5d.24xlarge db.r6i.large –db.r6i.8xlarge db.m5.large –db.m5.24xlarge db.m5d.large –db.m5d.24xlarge db.m6i.large –db.m6i.8xlarge db.x1.16xlarge –db.x1.32xlarge db.x1e.xlarge –db.x1e.32xlarge

SQL Server Edition	2022 지원 범위	2019 지원 범위	2017 및 2016 지원 범위
		db.x2iedn .xlarge –db.x2iedn .32xlarge db.z1d.la rge –db.z1d.12 xlarge	db.x2iedn .xlarge –db.x2iedn .32xlarge db.z1d.la rge –db.z1d.12 xlarge
Web Edition	db.t3.sma ll –db.t3.xlarge db.r5.lar ge –db.r5.4xlarge db.r5b.la rge –db.r5b.4xlarge db.r5d.la rge –db.r5d.4xlarge db.r6i.la rge –db.r6i.4xlarge db.m5.lar ge –db.m5.4xlarge db.m5d.la rge –db.m5d.4xlarge db.m6i.la rge –db.m6i.4xlarge db.z1d.la rge –db.z1d.13 xlarge	db.t3.sma ll –db.t3.2xlarge db.r5.lar ge –db.r5.4xlarge db.r5b.la rge –db.r5b.4xlarge db.r5d.la rge –db.r5d.4xlarge db.r6i.la rge –db.r6i.4xlarge db.m5.lar ge –db.m5.4xlarge db.m5d.la rge –db.m5d.4xlarge db.m6i.la rge –db.m6i.4xlarge db.z1d.la rge –db.z1d.3xlarge	db.t3.sma ll –db.t3.2xlarge db.r5.lar ge –db.r5.4xlarge db.r5b.la rge –db.r5b.4xlarge db.r5d.la rge –db.r5d.4xlarge db.r6i.la rge –db.r6i.4xlarge db.m5.lar ge –db.m5.4xlarge db.m5d.la rge –db.m5d.4xlarge db.m6i.la rge –db.m6i.4xlarge db.z1d.la rge –db.z1d.3xlarge

SQL Server Edition	2022 지원 범위	2019 지원 범위	2017 및 2016 지원 범위
Express Edition	db.t3.mic ro –db.t3.xlarge	db.t3.mic ro –db.t3.xlarge	db.t3.mic ro –db.t3.xlarge

Microsoft SQL Server 보안

Microsoft SQL Server 데이터베이스 엔진은 역할 기반 보안을 사용합니다. DB 인스턴스를 생성할 때 지정하는 마스터 사용자 이름은 `processadmin`, `public` 및 `setupadmin` 고정 서버 역할의 구성원인 SQL Server 인증 로그인입니다.

데이터베이스를 만드는 사용자는 누구든 해당 데이터베이스에 대한 db_owner 역할에 할당되며, 백업에 사용되는 권한을 제외한 모든 데이터베이스 수준의 권한을 갖게 됩니다. Amazon RDS는 백업을 관리합니다.

Amazon RDS for SQL Server에서는 다음과 같은 서버 수준 역할을 사용할 수 없습니다.

- bulkadmin
- dbcreator
- diskadmin
- securityadmin
- serveradmin
- sysadmin

RDS for SQL Server DB 인스턴스에서는 다음 서버 수준 권한을 사용할 수 없습니다.

- ALTER ANY DATABASE
- ALTER ANY EVENT NOTIFICATION
- ALTER RESOURCES
- ALTER SETTINGS(DB 파라미터 그룹 API 작업을 사용하여 파라미터를 수정할 수 있습니다. 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.)
- AUTHENTICATE SERVER
- CONTROL_SERVER

- CREATE DDL EVENT NOTIFICATION
- CREATE ENDPOINT
- CREATE SERVER ROLE
- CREATE TRACE EVENT NOTIFICATION
- DROP ANY DATABASE
- EXTERNAL ACCESS ASSEMBLY
- SHUTDOWN(RDS 재부팅 옵션을 대신 사용할 수 있음)
- UNSAFE ASSEMBLY
- 모든 가용성 그룹 변경
- 가용성 그룹 생성

Microsoft SQL Server DB 인스턴스에 대한 규정 준수 프로그램 지원

AWS 범위 내 서비스는 외부 감사 기관의 철저한 평가를 거쳐 인증, 규정 준수 증명 또는 운영 권한(ATO)을 받았습니다. 자세한 내용은 규정 준수 프로그램 제공 범위 내 AWS 서비스를 참조하세요.

Microsoft SQL Server DB 인스턴스에 대한 HIPAA 지원

Amazon RDS for Microsoft SQL Server 데이터베이스를 사용하여 HIPAA 인증 애플리케이션을 개발할 수 있습니다. 예를 들어 AWS와 체결한 비즈니스 제휴 계약(BAA)에 따라 보호 대상 건강 정보(PHI)를 비롯한 의료 관련 정보를 저장할 수 있습니다. 자세한 내용은 HIPAA 규정 준수를 참조하십시오.

Amazon RDS for SQL Server는 다음과 같은 버전 및 에디션에서 HIPAA를 지원합니다.

- SQL Server 2022 Enterprise, Standard 및 Web Edition
- SQL Server 2019 Enterprise, Standard 및 Web Edition
- SQL Server 2017 Enterprise, Standard 및 Web Edition
- SQL Server 2016 Enterprise, Standard 및 Web Edition

DB 인스턴스에서 HIPAA 지원을 활성화하려면 다음과 같이 세 가지 구성 요소를 설정해야 합니다.

구성 요소	세부 정보
감사	

구성 요소	세부 정보
	감사를 설정하려면 파라미터 `rds.sqlserver_audit` 을 값 `fedramp_hipaa` 로 설정합니다. DB 인스턴스가 아직 사용자 지정 DB 파라미터 그룹을 사용하지 않는 경우에는 `rds.sqlserver_audit` 파라미터를 수정하기 전에 먼저 사용자 그룹 파라미터를 생성한 후 DB 인스턴스에 연결해야 합니다. 자세한 내용은 Amazon RDS의 파라미터 그룹 섹션을 참조하세요.
전송 데이터 암호화	전송 데이터 암호화를 설정하려면 모든 DB 인스턴스 연결에 강제로 SSL(Secure Sockets Layer)을 사용해야 합니다. 자세한 내용은 ??? 섹션을 참조하세요.
유휴 시 암호화	저장 데이터 암호화의 설정은 다음과 같이 두 가지 옵션이 있습니다. 1. SQL Server 2016–2022 Enterprise Edition 또는 2022 Standard Edition을 실행하는 경우 투명한 데이터 암호화(TDE)를 사용하여 저장 데이터 암호화를 구현할 수 있습니다. 자세한 내용은 SQL Server에서 TDE(투명한 데이터 암호화) 지원 단원을 참조하십시오. 2. 저장 데이터 암호화는 AWS Key Management Service(AWS KMS) 암호화 키를 사용하여 설정할 수 있습니다. 자세한 내용은 Amazon RDS 리소스 암호화 섹션을 참조하세요.

Microsoft SQL Server 인스턴스를 위한 SSL 지원

SSL을 사용하여 애플리케이션과 Microsoft SQL Server를 실행하는 Amazon RDS DB 인스턴스 사이의 연결을 암호화할 수 있습니다. 또한 DB 인스턴스에 대한 모든 연결에서 SSL을 사용하도록 지정할 수도 있습니다. 연결이 SSL을 사용하도록 지정하면 클라이언트에 투명하게 발생하며, 클라이언트는 SSL 사용을 위해 작업을 수행할 필요가 없습니다.

SSL은 지원되는 모든 SQL Server 버전에 대해 모든 AWS 리전에서 사용할 수 있습니다. 자세한 내용은 ??? 섹션을 참조하세요.

Amazon RDS의 Microsoft SQL Server 버전

새 DB 인스턴스를 생성할 때는 현재 지원되는 모든 Microsoft SQL Server 버전을 지정할 수 있습니다. Microsoft SQL Server 메이저 버전(예: Microsoft SQL Server 14.00) 및 지정된 메이저 버전에 대해 지원되는 모든 마이너 버전을 지정할 수 있습니다. 버전이 지정되지 않은 경우 Amazon RDS는 지원되는 버전(보통 최신 버전)을 기본값으로 설정합니다. 메이저 버전이 지정되었지만 마이너 버전이 지정되지 않은 경우, Amazon RDS는 고객이 지정한 메이저 버전의 최근 릴리스를 기본값으로 설정합니다.

다음 표에는 명시된 경우를 제외하고 모든 버전 및 모든 AWS 리전에서 지원되는 버전이 나와 있습니다. `describe-db-engine-versions` AWS CLI 명령을 사용하여 지원되는 버전 목록과 새로 만든 DB 인스턴스의 기본값을 볼 수도 있습니다. 다음 표는 RDS에서 지원되는 SQL Server 버전을 보여줍니다.

메이저 버전	마이너 버전	RDS API **EngineVersion** 및 CLI **engine-version**
SQL Server 2022	16.00.4131.2(CU13)	`16.00.4131.2.v1`
	16.00.4125.3(CU13)	`16.00.4125.3.v1`
	16.00.4120.1(CU12 GDR)	`16.00.4120.1.v1`
	16.00.4115.5(CU12)	`16.00.4115.5.v1`
	16.00.4105.2(CU11)	`16.00.4105.2.v1`
	16.00.4095.4(CU10)	`16.00.4095.4.v1`
	16.00.4085.2(CU9)	`16.00.4085.2.v1`
SQL Server 2019	15.00.4382.1(CU27)	`15.00.4382.1.v1`
	15.00.4375.4(CU27)	`15.00.4375.4.v1`
	15.00.4365.2(CU26)	`15.00.4365.2.v1`
	15.00.4355.3(CU25)	`15.00.4355.3.v1`
	15.00.4345.5(CU24)	`15.00.4345.5.v1`
	15.00.4335.1(CU23)	`15.00.4335.1.v1`

메이저 버전	마이너 버전	RDS API **EngineVersion** 및 CLI **engine-version**
	15.00.4322.2(CU22)	15.00.4322.2.v1
	15.00.4316.3(CU21)	15.00.4316.3.v1
	15.00.4312.2(CU20)	15.00.4312.2.v1
	15.00.4236.7(CU16)	15.00.4236.7.v1
	15.00.4198.2(CU15)	15.00.4198.2.v1
	15.00.4153.1(CU12)	15.00.4153.1.v1
	15.00.4073.23(CU8)	15.00.4073.23.v1
	15.00.4043.16(CU5)	15.00.4043.16.v1
SQL Server 2017	14.00.3471.2(CU31)	14.00.3471.2.v1
	14.00.3465.1(CU31)	14.00.3465.1.v1
	14.00.3460.9(CU31)	14.00.3460.9.v1
	14.00.3451.2(CU30)	14.00.3451.2.v1
	14.00.3421.10(CU27)	14.00.3421.10.v1
	14.00.3401.7(CU25)	14.00.3401.7.v1
	14.00.3381.3(CU23)	14.00.3381.3.v1
	14.00.3356.20(CU22)	14.00.3356.20.v1
	14.00.3294.2(CU20)	14.00.3294.2.v1
	14.00.3281.6(CU19)	14.00.3281.6.v1

메이저 버전	마이너 버전	RDS API **EngineVersion** 및 CLI **engine-version**
SQL Server 2016	13.00.6441.1(GDR)	`13.00.6441.1.v1`
	13.00.6435.1(GDR)	`13.00.6435.1.v1`
	13.00.6430.49(GDR)	`13.00.6430.49.v1`
	13.00.6419.1(SP3 + Hotfix)	`13.00.6419.1.v1`
	13.00.6300.2(SP3)	`13.00.6300.2.v1`

Amazon RDS의 버전 관리

Amazon RDS에는 DB 인스턴스가 패치되거나 업그레이드되는 시기와 방법을 제어할 수 있는 유연한 버전 관리가 포함됩니다. 이렇게 하면 DB 엔진에 대해 다음을 수행할 수 있습니다.

- 데이터베이스 엔진 패치 버전과의 호환성을 유지합니다.
- 새로운 패치 버전을 테스트하여 프로덕션에 배포하기 전에 애플리케이션에서 잘 작동하는지 확인합니다.
- 서비스 수준 계약 및 타이밍 요구 사항을 충족하도록 버전 업그레이드 계획 및 수행합니다.

Amazon RDS의 Microsoft SQL Server 엔진 패치

Amazon RDS는 공식 Microsoft SQL Server 데이터베이스 패치를 정기적으로 Amazon RDS에 해당하는 DB 인스턴스 엔진 버전으로 집계합니다. 각 엔진 버전의 Microsoft SQL Server 패치에 대한 자세한 내용은 Amazon RDS 기반 버전 및 기능 지원을 참조하십시오.

현재 DB 인스턴스에서 모든 엔진 업그레이드를 수동으로 수행합니다. 자세한 내용은 Microsoft SQL Server DB 엔진 업그레이드 섹션을 참조하세요.

Amazon RDS 기반 Microsoft SQL Server의 메이저 엔진 버전의 사용 중단 일정

다음 표에는 Microsoft SQL Server에 예정된 메이저 엔진 버전의 사용 중단 일정이 표시됩니다.

날짜	정보
2024년 7월 9일	Microsoft는 SQL Server 2014에 대한 중요 패치 업데이트를 중단합니다. 자세한 내용은 Microsoft SQL Server 2014를 참조하세요.
2024년 6월 1일	Amazon RDS는 RDS for SQL Server에서 Microsoft SQL Server 2014의 지원을 종료합니다. 종료 시 나머지 인스턴스는 SQL Server 2016으로 마이그레이션되도록 예약됩니다(최신 CU 사용 가능). 자세한 내용은 공지 사항: SQL Server 2014 메이저 버전에 대한 Amazon RDS for SQL Server 지원 종료를 참조하세요. Microsoft SQL Server 2014에서 자동으로 업그레이드하지 않으려면 편리한 시간에 업그레이드할 수 있습니다. 자세한 내용은 DB 인스턴스 엔진 버전 업그레이드 단원을 참조하십시오.
2022년 7월 12일	Microsoft는 SQL Server 2012에 대한 중요 패치 업데이트를 중단합니다. 자세한 내용은 Microsoft SQL Server 2012를 참조하세요.
2022년 6월 1일	Amazon RDS는 SQL Server에 대한 Microsoft SQL Server 2012 on RDS의 지원을 종료합니다. 종료 시 나머지 인스턴스는 SQL Server 2014로 마이그레이션되도록 예약됩니다(최신 CU 사용 가능). 자세한 내용은 공지 사항: SQL Server 2012 메이저 버전에 대한 Amazon RDS for SQL Server 지원 종료를 참조하세요. Microsoft SQL Server 2012에서 자동으로 업그레이드하지 않으려면 편리하게 한 번에 업그레이드할 수 있습니다. 자세한 내용은 DB 인스턴스 엔진 버전 업그레이드 단원을 참조하세요.
2021년 9월 1일	Amazon RDS는 Microsoft SQL Server 2012를 사용한 SQL Server DB 인스턴스용 신규 생성을 중단하기 시작합니다. 자세한 내용은 공지 사항: SQL Server 2012 메이저 버전에 대한 Amazon RDS for SQL Server 지원 종료를 참조하세요.
2019년 7월 12일	Amazon RDS 팀은 Microsoft SQL Server 2008 R2에 대한 지원을 2019년 6월부터 중단합니다. 종료 시 SQL Server 2012 R2의 나머지 인스턴스는 SQL Server 2012로 마이그레이션되고 있습니다(최신 CU 사용 가능). Microsoft SQL Server 2008 R2에서 자동으로 업그레이드하지 않으려면 편리하게 한 번에 업그레이드할 수 있습니다. 자세한 내용은 DB 인스턴스 엔진 버전 업그레이드 섹션을 참조하세요.
2019년 4월 25일	2019년 4월 말 이전에는 더 이상 Microsoft SQL Server 2008 R2를 사용하여 새로운 Amazon RDS for SQL Server 데이터베이스 인스턴스를 만들 수 없습니다.

Amazon RDS의 Microsoft SQL Server 기능

Amazon RDS에서 지원되는 SQL 서버 버전에는 다음과 같은 기능이 포함되어 있습니다. 일반적으로 Microsoft 설명서에 다른 언급이 없는 경우 각 버전에는 이전 버전의 기능도 포함됩니다.

주제

- Microsoft SQL Server 2022 기능
- Microsoft SQL Server 2019 기능
- Microsoft SQL Server 2017 기능
- Microsoft SQL Server 2016 기능
- Amazon RDS에서 Microsoft SQL Server 2014 지원 종료
- Amazon RDS에서 Microsoft SQL Server 2012 지원 종료
- Amazon RDS에서 Microsoft SQL Server 2008 R2 지원 종료

Microsoft SQL Server 2022 기능

SQL Server 2022는 다음과 같은 새 기능을 많이 제공합니다.

- 파라미터를 중시하는 계획 최적화 - 파라미터화된 단일 문에 대해 여러 개의 캐시된 계획을 허용하여 파라미터 스니핑과 관련된 문제를 잠재적으로 줄일 수 있습니다.
- SQL Server 원장 - 데이터가 승인 없이 변경되지 않았음을 암호로 증명할 수 있는 기능을 제공합니다.
- 트랜잭션 로그 파일 증가 이벤트에 대한 즉각적인 파일 초기화 - TDE가 활성화된 데이터베이스를 포함하여 최대 64MB의 로그 증가 이벤트를 더 빠르게 실행할 수 있습니다.
- 시스템 페이지 래치 동시성 향상 - 데이터 페이지 및 범위를 할당 및 할당 해제하는 동안 페이지 래치 경합을 줄여 과중한 `tempdb` 워크로드의 성능을 크게 향상시킵니다.

SQL Server 2022의 전체 기능 목록은 Microsoft 설명서의 What's new in SQL Server 2022(16.x) 섹션을 참조하세요.

지원되지 않는 기능 목록은 지원되지 않는 기능과 지원이 제한된 기능 단원을 참조하세요.

Microsoft SQL Server 2019 기능

SQL Server 2019는 다음과 같은 새 기능을 많이 제공합니다.

- 가속 데이터베이스 복구(ADR) – 재시작 또는 장기 실행 트랜잭션 롤백 후 크래시 복구 시간을 줄입니다.
- 지능형 쿼리 처리(IQP):
  - 행 모드 메모리 부여 피드백 – 과도한 부여를 자동으로 수정합니다. 그렇지 않으면 메모리가 낭비되고 동시성이 감소합니다.
  - Rowstore의 배치 모드 – Columnstore 인덱스를 요구하지 않고 분석 워크로드에 대해 배치 모드 실행을 활성화합니다.
  - 테이블 변수 지연 컴파일 – 테이블 변수를 참조하는 쿼리의 계획 품질 및 전반적인 성능을 개선합니다.
- 지능적인 성능:
  - `OPTIMIZE_FOR_SEQUENTIAL_KEY` 인덱스 옵션 – 인덱스에 대한 동시성이 높은 삽입의 처리량을 개선합니다.
  - 향상된 간접 체크포인트 확장성 – DML 워크로드가 많은 데이터베이스를 지원합니다.
  - 동시 페이지 여유 공간(PFS) 업데이트 – 전용 래치가 아닌 공유 래치로 처리할 수 있습니다.
- 모니터링 개선:
  - `WAIT_ON_SYNC_STATISTICS_REFRESH` 대기 유형 – 동기식 통계 새로 고침 작업에 소요된 누적된 인스턴스 수준 시간을 표시합니다.
  - 데이터베이스 범위 구성 –에는 `LIGHTWEIGHT_QUERY_PROFILING` 및 `LAST_QUERY_PLAN_STATS`가 포함됩니다.
  - 동적 관리 기능(DMF) –에는 `sys.dm_exec_query_plan_stats` 및 `sys.dm_db_page_info`가 포함합니다.
- 상세 표시 잘림 경고 – 데이터 잘림 오류 메시지는 기본적으로 테이블 및 열 이름과 잘려진 값을 포함합니다.
- 다시 시작 가능한 온라인 인덱스 생성 – SQL Server 2017에서는 다시 시작 가능한 온라인 인덱스 재구축만 지원됩니다.

SQL Server 2019의 전체 기능 목록은 Microsoft 설명서의 What's new in SQL Server 2019(15.x) 섹션을 참조하세요.

지원되지 않는 기능 목록은 지원되지 않는 기능과 지원이 제한된 기능 단원을 참조하십시오.

Microsoft SQL Server 2017 기능

SQL Server 2017은 다음과 같은 새 기능을 많이 제공합니다.

- 적응형 쿼리 처리
- 자동 계획 수정(자동 튜닝 기능)
- 그래프DB
- 다시 시작 가능한 인덱스 재작성

SQL Server 2017의 전체 기능 목록은 Microsoft 설명서의 What's new in SQL Server 2017 단원을 참조하십시오.

지원되지 않는 기능 목록은 지원되지 않는 기능과 지원이 제한된 기능 단원을 참조하십시오.

Microsoft SQL Server 2016 기능

Amazon RDS는 다음과 같은 SQL Server 2016 기능을 지원합니다.

- 항상 암호화
- JSON 지원
- 운영 분석
- 쿼리 저장
- 임시 테이블

SQL Server 2016의 전체 기능 목록은 Microsoft 설명서의 What's new in SQL Server 2016 단원을 참조하십시오.

Amazon RDS에서 Microsoft SQL Server 2014 지원 종료

SQL Server 2014는 Amazon RDS에서 지원이 종료되었습니다.

RDS는 여전히 SQL Server 2014를 사용하는 모든 기존 DB 인스턴스를 SQL Server 2016의 최신 마이너 버전으로 업그레이드하고 있습니다. 자세한 내용은 Amazon RDS의 버전 관리 단원을 참조하십시오.

Amazon RDS에서 Microsoft SQL Server 2012 지원 종료

Amazon RDS에서 Microsoft SQL Server 2012 지원 종료

RDS는 여전히 SQL Server 2012를 사용하는 모든 기존 DB 인스턴스를 SQL Server 2016의 최신 마이너 버전으로 업그레이드하고 있습니다. 자세한 내용은 Amazon RDS의 버전 관리 단원을 참조하십시오.

Amazon RDS에서 Microsoft SQL Server 2008 R2 지원 종료

SQL Server 2008 R2는 Amazon RDS에서 지원이 종료되었습니다.

RDS는 여전히 SQL Server 2008 R2를 사용하는 모든 기존 DB 인스턴스를 SQL Server 2012의 최신 마이너 버전으로 업그레이드하고 있습니다. 자세한 내용은 Amazon RDS의 버전 관리 단원을 참조하세요.

Microsoft SQL Server DB 인스턴스에 대한 변경 데이터 캡처 지원

Amazon RDS는 Microsoft SQL Server를 실행하는 DB 인스턴스에 대해 CDC(변경 데이터 캡처)를 지원합니다. CDC는 테이블의 데이터에 수행된 변경 사항을 캡처하고 각 변경 사항에 대해 나중에 액세스할 수 있는 메타데이터를 저장합니다. 자세한 내용은 Microsoft 설명서의 변경 데이터 캡처를 참조하십시오.

Amazon RDS는 다음 SQL Server 에디션과 버전에 대해 CDC를 지원합니다.

- Microsoft SQL Server Enterprise Edition(모든 버전)
- Microsoft SQL Server Standard Edition:
  - 2022
  - 2019
  - 2017
  - 2016 버전 13.00.4422.0 SP1 CU2 이상

Amazon RDS DB 인스턴스에 CDC를 사용하려면 먼저 RDS의 저장 프로시저를 사용하여 데이터베이스 수준에서 CDC를 활성화하거나 비활성화합니다. 그런 후에는 해당 데이터베이스에 대해 `db_owner` 역할이 있는 모든 사용자가 기본 Microsoft 저장 프로시저를 사용하여 해당 데이터베이스에서 CDC를 제어할 수 있습니다. 자세한 내용은 Amazon RDS for SQL Server에 변경 데이터 캡처 사용 섹션을 참조하세요.

CDC와 AWS Database Migration Service를 사용하여 SQL Server DB 인스턴스에서 지속적 복제를 활성화할 수 있습니다.

지원되지 않는 기능과 지원이 제한된 기능

다음 Microsoft SQL Server 기능은 Amazon RDS에서 지원되지 않습니다.

- Microsoft Azure Blob Storage로 백업
- 버퍼 풀 확장
- 사용자 지정 암호 정책
- 데이터 품질 서비스
- 데이터베이스 로그 전달
- 데이터베이스 스냅샷(Amazon RDS는 DB 인스턴스 스냅샷만 지원)
- xp_cmdshell을 포함한 저장 프로시저 확장
- FILESTREAM 지원
- 파일 테이블
- Machine Learning 및 R 서비스(설치를 위해 OS 액세스 필요)
- 유지 관리 계획
- 성능 데이터 수집기
- 정책 기반 관리
- PolyBase
- 복제
- 리소스 거버너
- 서버 수준 트리거
- 서비스 브로커 엔드포인트
- Stretch 데이터베이스
- 신뢰할 수 있는 데이터베이스 속성(sysadmin 역할 필요)
- T-SQL 엔드포인트(CREATE ENDPOINT를 사용하는 모든 작업을 사용할 수 없음)
- WCF 데이터 서비스

다음 Microsoft SQL Server 기능은 Amazon RDS에서 지원이 제한적입니다.

- 분산 쿼리/연결된 서버 자세한 내용은 Implementing Linked Servers with Amazon RDS for Microsoft SQL Server를 참조하세요.
- CLR(공용 런타임 언어). RDS for SQL Server 2016 이하 버전에서 CLR은 SAFE 모드에서 지원되며 어셈블리 비트만 사용합니다. CLR은 RDS for SQL Server 2017 이상 버전에서 지원되지 않습니다. 자세한 내용은 Microsoft 설명서의 공용 런타임 언어 통합을 참조하세요.

- Amazon RDS for SQL Server에서 Oracle OLEDB와 서버를 연결합니다. 자세한 내용은 Amazon RDS for SQL Server의 Oracle OLEDB 포함 연결된 서버 지원 단원을 참조하십시오.

다음 기능은 SQL Server 2022가 설치된 Amazon RDS에서는 지원되지 않습니다.

- 스냅샷을 위해 데이터베이스 일시 중지
- 외부 데이터 소스
- S3 호환 객체 스토리지로 백업 및 복원
- 객체 스토어 통합
- TLS 1.3 및 MS-TDS 8.0
- QAT를 통한 백업 압축 오프로드
- SSAS(SQL Server Analysis Services)
- 다중 AZ 배포에서 데이터베이스 미러링. 다중 AZ 배포에서 지원되는 유일한 방법은 SQL Server Always On입니다.

Microsoft SQL Server 데이터베이스 미러링 또는 상시 가동 가용성 그룹을 사용하여 다중 AZ 배포

Amazon RDS는 SQL Server 데이터베이스 미러링(DBM) 또는 상시 가동 가용성 그룹(AG)을 사용하여 Microsoft SQL Server 기반 DB 인스턴스의 다중 AZ 배포를 지원합니다. 다중 AZ 배포는 DB 인스턴스를 위해 향상된 가용성, 데이터 내구성 및 내결함성을 제공합니다. 계획된 데이터베이스 유지 관리 또는 예기치 않은 서비스 중단이 발생할 경우 Amazon RDS가 최신 보조 복제본으로 자동으로 장애 조치를 수행하므로 수동 개입 없이 데이터베이스 작업을 빠르게 재개할 수 있습니다. 기본 인스턴스 및 보조 인스턴스는 동일한 엔드포인트를 사용합니다. 이 엔드포인트의 물리적 네트워크 주소는 장애 조치 프로세스의 일환으로 수동 보조 복제본으로 전환됩니다. 장애 조치가 발생하는 경우 애플리케이션을 다시 구성할 필요가 없습니다.

Amazon RDS는 다중 AZ 배포를 능동적으로 모니터링하면서 기본 인스턴스에 문제가 발생할 때 장애 조치를 시작하여 장애 조치를 관리합니다. 대기 및 기본 인스턴스가 완벽히 동기화되어 있지 않는 한 장애 조치가 발생하지 않습니다. Amazon RDS는 비정상 DB 인스턴스를 자동으로 복구하고 동기식 복제를 다시 설정하여 다중 AZ 배포를 자동으로 유지합니다. 사용자가 따로 관리할 것이 없습니다. 따라서 Amazon RDS가 기본 인스턴스, 감시 인스턴스, 대기 인스턴스를 자동으로 처리합니다. SQL Server 다중 AZ를 설정하면 RDS가 인스턴스의 모든 데이터베이스에 대해 수동 보조 인스턴스를 구성합니다.

자세한 내용은 Amazon RDS for Microsoft SQL Server의 다중 AZ 배포 섹션을 참조하세요.

Transparent Data Encryption을 사용하여 유휴 데이터 암호화

Amazon RDS는 저장되어 있는 데이터를 자동으로 암호화할 수 있는 Microsoft SQL Server Transparent Data Encryption(TDE)을 지원합니다. Amazon RDS는 옵션 그룹을 사용하여 이러한 기능을 활성화하고 구성합니다. TDE 옵션에 대한 자세한 내용은 SQL Server에서 TDE(투명한 데이터 암호화) 지원 단원을 참조하십시오.

Amazon RDS for Microsoft SQL Server에 대한 함수 및 저장 프로시저

아래에서 SQL Server 태스크를 자동화하는 데 도움이 되는 Amazon RDS 함수 및 저장 프로시저 목록을 확인할 수 있습니다.

작업 유형	프로시저 또는 함수	사용 위치
관리 작업	`rds_drop_database`	Amazon RDS for Microsoft SQL Server DB 인스턴스 데이터베이스 삭제
	`rds_failover_time`	Amazon RDS for SQL Server의 마지막 장애 조치 시간 결정
	`rds_modify_db_name`	다중 AZ 배포의 Amazon RDS for Microsoft SQL Server 데이터베이스 이름 변경
	`rds_read_error_log`	오류 및 에이전트 로그 보기
	`rds_set_configuration`	이 작업은 다양한 DB 인스턴스 구성을 설정하는 데 사용됩니다. • 다중 AZ 인스턴스에 대한 변경 데이터 캡처 • 추적 및 덤프 파일의 보존 기간 설정 • 백업 파일 압축

작업 유형	프로시저 또는 함수	사용 위치
	`rds_set_database_online`	오프라인에서 온라인으로 Amazon RDS for SQL Server 데이터베이스 전환
	`rds_set_system_database_sync_objects`	SQL Server 에이전트 작업 복제 켜기
	`rds_fn_get_system_database_sync_objects`	
	`rds_fn_server_object_last_sync_time`	
	`rds_show_configuration`	`rds_set_configuration` 을 사용하여 설정된 값을 보려면 해당 주제를 살펴보세요. • 다중 AZ 인스턴스에 대한 변경 데이터 캡처 • 추적 및 덤프 파일의 보존 기간 설정
	`rds_shrink_tempdbfile`	tempdb 데이터베이스 축소
CDC(변경 데이터 캡처)	`rds_cdc_disable_db`	CDC 비활성화

작업 유형	프로시저 또는 함수	사용 위치
	`rds_cdc_enable_db`	CDC 활성화
데이터베이스 메일	`rds_fn_sysmail_allitems`	메시지, 로그 및 첨부 파일 보기
	`rds_fn_sysmail_event_log`	메시지, 로그 및 첨부 파일 보기
	`rds_fn_sysmail_mailattachments`	메시지, 로그 및 첨부 파일 보기
	`rds_sysmail_control`	이 작업은 메일 대기열을 시작하고 중지하는 데 사용됩니다. • 메일 대기열 시작 • 메일 대기열 중지
	`rds_sysmail_delete_mailitems_sp`	메시지 삭제
기본 백업 및 복원	`rds_backup_database`	데이터베이스 백업
	`rds_cancel_task`	작업 취소

작업 유형	프로시저 또는 함수	사용 위치
	`rds_finish_restore`	데이터베이스 복원 마무리
	`rds_restore_database`	데이터베이스 복원
	`rds_restore_log`	로그 복원
Amazon S3 파일 전송	`rds_delete_from_filesystem`	RDS DB 인스턴스의 파일 삭제
	`rds_download_from_s3`	Amazon S3 버킷의 파일을 SQL Server DB 인스턴스로 다운로드
	`rds_gather_file_details`	RDS DB 인스턴스의 파일 나열
	`rds_upload_to_s3`	SQL Server DB 인스턴스의 파일을 Amazon S3 버킷으로 업로드
MSDTC(Microsoft Distributed Transaction Coordinator)	`rds_msdtc_transaction_tracing`	트랜잭션 추적 사용
SQL Server Audit	`rds_fn_get_audit_file`	감사 로그 보기

작업 유형	프로시저 또는 함수	사용 위치

투명한 데이터 암호화	`rds_backup_tde_certificate`
	
	`rds_drop_tde_certificate`
	
	`rds_restore_tde_certificate`
	
	`rds_fn_list_user_tde_certificates`
		SQL Server에서 TDE(투명한 데이터 암호화) 지원

작업 유형	프로시저 또는 함수	사용 위치
MSBI(Microsoft Business Intelligence)	`rds_msbi_task`	이 작업은 SQL Server Analysis Services(SSAS)와 함께 사용됩니다. • Amazon RDS에 SSAS 프로젝트 배포 • 도메인 사용자를 데이터베이스 관리자로 추가 • SSAS 데이터베이스 백업 • SSAS 데이터베이스 복원 이 작업은 SQL Server Integration Services(SSIS)에서도 사용됩니다. • SSISDB에 대한 관리 권한 • SSIS 프로젝트 배포 이 작업은 SQL Server Reporting Services(SSRS)에서도 사용됩니다. • 도메인 사용자에게 액세스 권한 부여 • 시스템 수준 권한 취소
	`rds_fn_task_status`	이 작업은 MSBI 태스크의 상태를 표시합니다. • SSAS: 배포 작업의 상태 모니터링 • SSIS: 배포 작업의 상태 모니터링 • SSRS: 작업의 상태 모니터링
SSIS	`rds_drop_ssis_database`	SSISDB 데이터베이스 삭제
	`rds_sqlagent_proxy`	SSIS 프록시 생성

작업 유형	프로시저 또는 함수	사용 위치
SSRS	`rds_drop_ssrs_databases`	SSRS 데이터베이스 삭제

Microsoft SQL Server DB 인스턴스의 현지 시간대

Microsoft SQL Server를 실행 중인 Amazon RDS DB 인스턴스의 시간대가 기본적으로 설정되어 있습니다. 현재 기본값은 협정 세계시(UTC)입니다. DB 인스턴스의 시간대를 애플리케이션의 시간대와 일치하도록 현지 시간대로 설정할 수 있습니다.

DB 인스턴스를 처음 만들 때 시간대를 설정합니다. AWS Management Console, Amazon RDS API CreateDBInstance 작업 또는 AWS CLI create-db-instance 명령을 사용하여 DB 인스턴스를 생성할 수 있습니다.

DB 인스턴스가 다중 AZ 배포의 일부인 경우(SQL Server DBM 또는 AG 사용) 장애 조치 중에 시간대가 설정된 현지 시간대로 유지됩니다. 자세한 내용은 Microsoft SQL Server 데이터베이스 미러링 또는 상시 가동 가용성 그룹을 사용하여 다중 AZ 배포 섹션을 참조하세요.

시점 복원을 요청할 경우 복원 시간을 지정합니다. 시간은 현지 시간대로 표시됩니다. 자세한 내용은 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 섹션을 참조하세요.

다음은 DB 인스턴스에 대해 현지 시간대를 설정할 때 적용되는 제한 사항입니다.

- 기존 SQL Server DB 인스턴스의 시간대를 수정할 수 없습니다.
- DB 인스턴스의 스냅샷을 다른 시간대의 DB 인스턴스로 복원할 수 없습니다.
- 한 표준 시간대의 백업 파일을 다른 표준 시간대로 복원하지 않는 것이 좋습니다. 한 표준 시간대의 백업 파일을 다른 표준 시간대로 복원하는 경우 쿼리와 애플리케이션을 감사하여 표준 시간대 변경의 영향을 확인해야 합니다. 자세한 내용은 기본 백업 및 복원 기능을 사용하여 SQL Server 데이터베이스 가져오기 및 내보내기 섹션을 참조하세요.

지원되는 시간대

현지 시간대를 다음 표에 나열된 값 중 하나로 설정할 수 있습니다.

시간대	표준 시간 오프셋	설명	참고
아프가니스탄 표준시	(UTC+04:30)	카불	이 시간대는 일광 절약 시간을 준수하지 않습니다.
알래스카 표준시	(UTC–09:00)	알래스카	
알류샨 표준시	(UTC–10:00)	알류샨 열도	
알타이 표준시	(UTC+07:00)	바르나울, 고르노알타이스크	
아랍 표준시	(UTC+03:00)	쿠웨이트, 리야드	이 시간대는 일광 절약 시간을 준수하지 않습니다.
아라비아 표준시	(UTC+04:00)	아부다비, 무스카트	
아랍 표준시	(UTC+03:00)	바그다드	이 시간대는 일광 절약 시간을 준수하지 않습니다.
아르헨티나 표준시	(UTC–03:00)	부에노스아이레스	이 시간대는 일광 절약 시간을 준수하지 않습니다.
아스트라한 표준시	(UTC+04:00)	아스트라한, 울랴노브스크	
대서양 표준시	(UTC–04:00)	대서양 표준시(캐나다)	
AUS 중부 표준시	(UTC+09:30)	다윈	이 시간대는 일광 절약 시간을 준수하지 않습니다.
오스트레일리아 중부 표준시	(UTC+08:45)	유클라	

시간대	표준 시간 오프셋	설명	참고
AUS 동부 표준시	(UTC+10:00)	캔버라, 멜버른, 시드니	
아제르바이잔 표준시	(UTC+04:00)	바쿠	
아조레스 표준시	(UTC–01:00)	아조레스	
바이아 표준시	(UTC–03:00)	살바도르	
방글라데시 표준시	(UTC+06:00)	다카	이 시간대는 일광 절약 시간을 준수하지 않습니다.
벨라루스 표준시	(UTC+03:00)	민스크	이 시간대는 일광 절약 시간을 준수하지 않습니다.
부건빌 표준시	(UTC+11:00)	부겐빌 섬	
캐나다 중부 표준시	(UTC–06:00)	서스캐처원	이 시간대는 일광 절약 시간을 준수하지 않습니다.
카포베르데 표준시	(UTC–01:00)	카포베르데 섬	이 시간대는 일광 절약 시간을 준수하지 않습니다.
코카서스 표준시	(UTC+04:00)	예레반	
중부 오스트레일리아 표준시	(UTC+09:30)	애들레이드	
중앙 아메리카 표준시	(UTC–06:00)	중앙 아메리카	이 시간대는 일광 절약 시간을 준수하지 않습니다.
중앙 아시아 표준시	(UTC+06:00)	아스타나	이 시간대는 일광 절약 시간을 준수하지 않습니다.

시간대	표준 시간 오프셋	설명	참고
브라질 중부 표준시	(UTC–04:00)	쿠이아바	
중앙 유럽 표준시	(UTC+01:00)	베오그라드, 브라티슬라바, 부다페스트, 류블랴나, 프라하	
중앙 유럽 표준시	(UTC+01:00)	사라예보, 스코페, 바르샤바, 자그레브	
중앙 태평양 표준시	(UTC+11:00)	솔로몬 제도, 뉴칼레도니아	이 시간대는 일광 절약 시간을 준수하지 않습니다.
중부 표준시	(UTC–06:00)	중부 표준시(미국과 캐나다)	
중부 표준시(멕시코)	(UTC–06:00)	과달라하라, 멕시코시티, 몬테레이	
채텀 섬 표준시	(UTC+12:45)	채텀 섬	
중국 표준시	(UTC+08:00)	베이징, 충칭, 홍콩 특별 행정구, 우루무치	이 시간대는 일광 절약 시간을 준수하지 않습니다.
쿠바 표준시	(UTC–05:00)	하바나	
날짜 변경선 표준시	(UTC–12:00)	날짜 변경선 서쪽	이 시간대는 일광 절약 시간을 준수하지 않습니다.
E. 아프리카 표준시	(UTC+03:00)	나이로비	이 시간대는 일광 절약 시간을 준수하지 않습니다.

시간대	표준 시간 오프셋	설명	참고
E. 오스트레일리아 표준시	(UTC+10:00)	브리즈번	이 시간대는 일광 절약 시간을 준수하지 않습니다.
E. 유럽 표준시	(UTC+02:00)	키시나우	
E. 남아메리카 표준시	(UTC–03:00)	브라질리아	
이스터 섬 표준시	(UTC–06:00)	이스터 섬	
동부 표준시	(UTC–05:00)	동부 표준시(미국과 캐나다)	
동부 표준시(멕시코)	(UTC–05:00)	체투말	
이집트 표준시	(UTC+02:00)	카이로	
예카테린부르크 표준시	(UTC+05:00)	예카테린부르크	
피지 표준시	(UTC+12:00)	피지	
FLE 표준시	(UTC+02:00)	헬싱키, 키예프, 리가, 소피아, 탈린, 빌뉴스	
그루지야 표준시	(UTC+04:00)	트빌리시	이 시간대는 일광 절약 시간을 준수하지 않습니다.
GMT 표준시	(UTC)	더블린, 에든버러, 리스본, 런던	이 시간대는 그리니치 표준시(GMT)와 다릅니다. 이 시간대는 일광 절약 시간을 준수합니다.
그린란드 표준시	(UTC–03:00)	그린란드	

시간대	표준 시간 오프셋	설명	참고
그리니치 표준시	(UTC)	몬로비아, 레이캬비크	이 시간대는 일광 절약 시간을 준수하지 않습니다.
GTB 표준시	(UTC+02:00)	아테네, 부쿠레슈티	
아이티 표준시	(UTC–05:00)	아이티	
하와이 표준시	(UTC–10:00)	하와이	
인도 표준시	(UTC+05:30)	첸나이, 콜카타, 뭄바이, 뉴델리	이 시간대는 일광 절약 시간을 준수하지 않습니다.
이란 표준시	(UTC+03:30)	테헤란	
이스라엘 표준시	(UTC+02:00)	예루살렘	
요르단 표준시	(UTC+02:00)	암만	
칼리닌그라드 표준시	(UTC+02:00)	칼리닌그라드	
캄차카 표준시	(UTC+12:00)	페트로파블로프스크-캄차스키 – 이전	
대한민국 표준시	(UTC+09:00)	서울	이 시간대는 일광 절약 시간을 준수하지 않습니다.
리비아 표준시	(UTC+02:00)	트리폴리	
라인 제도 표준시	(UTC+14:00)	키리티마티 섬	
로드하우 표준시	(UTC+10:30)	로드하우 섬	
마가단 표준시	(UTC+11:00)	마가단	이 시간대는 일광 절약 시간을 준수하지 않습니다.

시간대	표준 시간 오프셋	설명	참고
마가야네스 표준시	(UTC–03:00)	푼타 아레나스	
마키저스 표준시	(UTC–09:30)	마르케사스 제도	
모리셔스 표준시	(UTC+04:00)	포트루이스	이 시간대는 일광 절약 시간을 준수하지 않습니다.
중동 표준시	(UTC+02:00)	베이루트	
몬테비데오 표준시	(UTC–03:00)	몬테비데오	
모로코 표준시	(UTC+01:00)	카사블랑카	
산지 표준시	(UTC–07:00)	산지 표준시(미국과 캐나다)	
산지 표준시(멕시코)	(UTC–07:00)	치와와, 라파스, 마사틀란	
미얀마 표준시	(UTC+06:30)	양곤(랑군)	이 시간대는 일광 절약 시간을 준수하지 않습니다.
N. 중앙 아시아 표준시	(UTC+07:00)	노보시비르스크	
나미비아 표준시	(UTC+02:00)	빈트후크	
네팔 표준시	(UTC+05:45)	카트만두	이 시간대는 일광 절약 시간을 준수하지 않습니다.
뉴질랜드 표준시	(UTC+12:00)	오클랜드, 웰링턴	
뉴펀들랜드 표준시	(UTC–03:30)	뉴펀들랜드	
노퍽 표준시	(UTC+11:00)	노퍽 섬	
북아시아 동부 표준시	(UTC+08:00)	이르쿠츠크	

시간대	표준 시간 오프셋	설명	참고
북아시아 표준시	(UTC+07:00)	크라스노야르스크	
북한 표준시	(UTC+09:00)	평양	
옴스크 표준시	(UTC+06:00)	옴스크	
태평양 SA 표준시	(UTC–03:00)	산티아고	
태평양 표준시	(UTC–08:00)	태평양 표준시(미국과 캐나다)	
태평양 표준시(멕시코)	(UTC–08:00)	바하 캘리포니아	
파키스탄 표준시	(UTC+05:00)	이슬라마바드, 카라치	이 시간대는 일광 절약 시간을 준수하지 않습니다.
파라과이 표준시	(UTC–04:00)	아순시온	
로맨스 표준시	(UTC+01:00)	브뤼셀, 코펜하겐, 마드리드, 파리	
러시아 표준 시간대 10	(UTC+11:00)	초쿠르다흐	
러시아 표준 시간대 11	(UTC+12:00)	아나디리, 페트로파블로프스크-캄차스키	
러시아 표준 시간대 3	(UTC+04:00)	이젭스크, 사마라	
러시아 표준시	(UTC+03:00)	모스크바, 상트페테르부르크, 볼고그라드	이 시간대는 일광 절약 시간을 준수하지 않습니다.
SA 동부 표준시	(UTC–03:00)	카옌, 포르탈레자	이 시간대는 일광 절약 시간을 준수하지 않습니다.

시간대	표준 시간 오프셋	설명	참고
태평양 SA 표준시	(UTC–05:00)	보고타, 리마, 키토, 리오 브랑코	이 시간대는 일광 절약 시간을 준수하지 않습니다.
SA 서부 표준시	(UTC–04:00)	조지타운, 라파스, 마노스, 산후안	이 시간대는 일광 절약 시간을 준수하지 않습니다.
생피에르 표준시	(UTC–03:00)	세인트 피에르 미켈론	
사할린 표준시	(UTC+11:00)	사할린	
사모아 표준시	(UTC+13:00)	사모아	
상투메 표준시	(UTC+01:00)	상투메	
사라토프 표준시	(UTC+04:00)	사라토프	
동남아시아 표준시	(UTC+07:00)	방콕, 하노이, 자카르타	이 시간대는 일광 절약 시간을 준수하지 않습니다.
싱가포르 표준시	(UTC+08:00)	쿠알라룸푸르, 싱가포르	이 시간대는 일광 절약 시간을 준수하지 않습니다.
남아프리카 표준시	(UTC+02:00)	하라레, 프리토리아	이 시간대는 일광 절약 시간을 준수하지 않습니다.
스리랑카 표준시	(UTC+05:30)	스리자야와르데네푸라	이 시간대는 일광 절약 시간을 준수하지 않습니다.
수단 표준시	(UTC+02:00)	하르툼	
시리아 표준시	(UTC+02:00)	다마스쿠스	

시간대	표준 시간 오프셋	설명	참고
타이베이 표준시	(UTC+08:00)	타이베이	이 시간대는 일광 절약 시간을 준수하지 않습니다.
태즈메이니아 표준시	(UTC+10:00)	호바트	
토칸틴스 표준시	(UTC–03:00)	아라구아이나	
도쿄 표준시	(UTC+09:00)	오사카, 삿포로, 도쿄	이 시간대는 일광 절약 시간을 준수하지 않습니다.
톰스크 표준시	(UTC+07:00)	톰스크	
통가 표준시	(UTC+13:00)	누쿠알로파	이 시간대는 일광 절약 시간을 준수하지 않습니다.
트란스바이칼 표준시	(UTC+09:00)	치타	
터키 표준시	(UTC+03:00)	이스탄불	
터크스 케이커스 표준시	(UTC–05:00)	터크스 케이커스	
울란바토르 표준시	(UTC+08:00)	울란바토르	이 시간대는 일광 절약 시간을 준수하지 않습니다.
미국 동부 표준시	(UTC–05:00)	인디애나(동부)	
미국 산지 표준시	(UTC–07:00)	애리조나	이 시간대는 일광 절약 시간을 준수하지 않습니다.
UTC	UTC	협정 세계시	이 시간대는 일광 절약 시간을 준수하지 않습니다.

시간대	표준 시간 오프셋	설명	참고
UTC–02	(UTC–02:00)	협정 세계시–02	이 시간대는 일광 절약 시간을 준수하지 않습니다.
UTC–08	(UTC–08:00)	협정 세계시–08	
UTC–09	(UTC–09:00)	협정 세계시–09	
UTC–11	(UTC–11:00)	협정 세계시–11	이 시간대는 일광 절약 시간을 준수하지 않습니다.
UTC+12	(UTC+12:00)	협정 세계시+12	이 시간대는 일광 절약 시간을 준수하지 않습니다.
UTC+13	(UTC+13:00)	협정 세계시+13	
베네수엘라 표준시	(UTC–04:00)	카라카스	이 시간대는 일광 절약 시간을 준수하지 않습니다.
블라디보스토크 표준시	(UTC+10:00)	블라디보스토크	
볼고그라드 표준시	(UTC+04:00)	볼고그라드	
W. 오스트레일리아 표준시	(UTC+08:00)	퍼스	이 시간대는 일광 절약 시간을 준수하지 않습니다.
W. 중앙 아프리카 표준시	(UTC+01:00)	서중앙 아프리카	이 시간대는 일광 절약 시간을 준수하지 않습니다.
W. 유럽 표준시	(UTC+01:00)	암스테르담, 베를린, 베른, 로마, 스톡홀름, 비엔나	

시간대	표준 시간 오프셋	설명	참고
W. 몽골 표준시	(UTC+07:00)	호브드	
서아시아 표준시	(UTC+05:00)	아슈하바트, 타슈켄트	이 시간대는 일광 절약 시간을 준수하지 않습니다.
웨스트 뱅크 표준시	(UTC+02:00)	가자, 헤브론	
서태평양 표준시	(UTC+10:00)	괌, 포트모르즈비	이 시간대는 일광 절약 시간을 준수하지 않습니다.
야쿠츠크 표준시	(UTC+09:00)	야쿠츠크	

Amazon RDS의 Microsoft SQL Server 라이선싱

Microsoft SQL Server용으로 Amazon RDS DB 인스턴스를 설정하는 경우 소프트웨어 라이선스가 포함됩니다.

따라서 SQL Server 라이선스를 별도로 구매할 필요가 없습니다. AWS는 SQL Server 데이터베이스 소프트웨어에 대한 라이선스를 보유합니다. Amazon RDS 요금에는 소프트웨어 라이선스, 기본 하드웨어 리소스 및 Amazon RDS 관리 기능이 포함됩니다.

Amazon RDS는 다음 Microsoft SQL Server 에디션을 지원합니다.

- 엔터프라이즈
- 표준
- 웹
- Express

> **Note**
>
> SQL Server Web Edition 라이선스를 사용해야만 퍼블릭 및 인터넷으로 액세스 가능한 웹 페이지, 웹 사이트, 웹 애플리케이션 및 웹 서비스를 지원할 수 있습니다. 이 지원 수준은 Microsoft의 사용 권한을 준수하는 데 필요합니다. 자세한 내용은 AWS 서비스 약관을 참조하세요.

Amazon RDS는 SQL Server 데이터베이스 미러링(DBM) 또는 상시 가동 가용성 그룹(AG)을 사용하여 Microsoft SQL Server 기반 DB 인스턴스의 다중 AZ 배포를 지원합니다. 다중 AZ 배포에 대한 추가 라이선스 요구 사항은 없습니다. 자세한 내용은 Amazon RDS for Microsoft SQL Server의 다중 AZ 배포 섹션을 참조하세요.

라이선스가 종료된 DB 인스턴스 복원

Amazon RDS는 라이선스 종료된 DB 인스턴스의 스냅샷을 생성합니다. 라이선스 문제로 인스턴스가 종료되는 경우 스냅샷에서 새 DB 인스턴스로 복원할 수 있습니다. 새 DB 인스턴스에는 라이선스가 포함되어 있습니다.

자세한 내용은 Amazon RDS for SQL Server에 대한 라이선스 종료 DB 인스턴스 복원 섹션을 참조하세요.

개발 및 테스트

라이선스 요구 사항으로 인해 Amazon RDS에서는 SQL Server Developer 버전을 제공할 수 없습니다. 여러 개발, 테스트 및 기타 비 프로덕션 요구에 대해 Express 버전을 사용할 수 있습니다. 그러나 개발용 SQL Server의 엔터프라이즈급 설치의 전체 기능이 필요한 경우 BYOM이 있는 CEV를 사용하여 RDS Custom for SQL Server에 SQL Server Developer 버전을 다운로드하여 설치할 수 있습니다. 자세한 내용은 기존 보유 미디어를 사용(BYOM)하여 CEV 준비 단원을 참조하십시오. Developer 버전에는 전용 인프라가 필요하지 않습니다. 고유 호스트를 사용하여 Amazon RDS에서 액세스할 수 없는 다른 프로그래밍 기능에 대한 액세스 권한을 부여할 수도 있습니다. SQL Server 버전 간의 차이에 대한 자세한 정보는 Microsoft 설명서의 SQL Server 2019의 버전과 지원하는 기능을 참조하세요.

Microsoft SQL Server DB 인스턴스에 연결

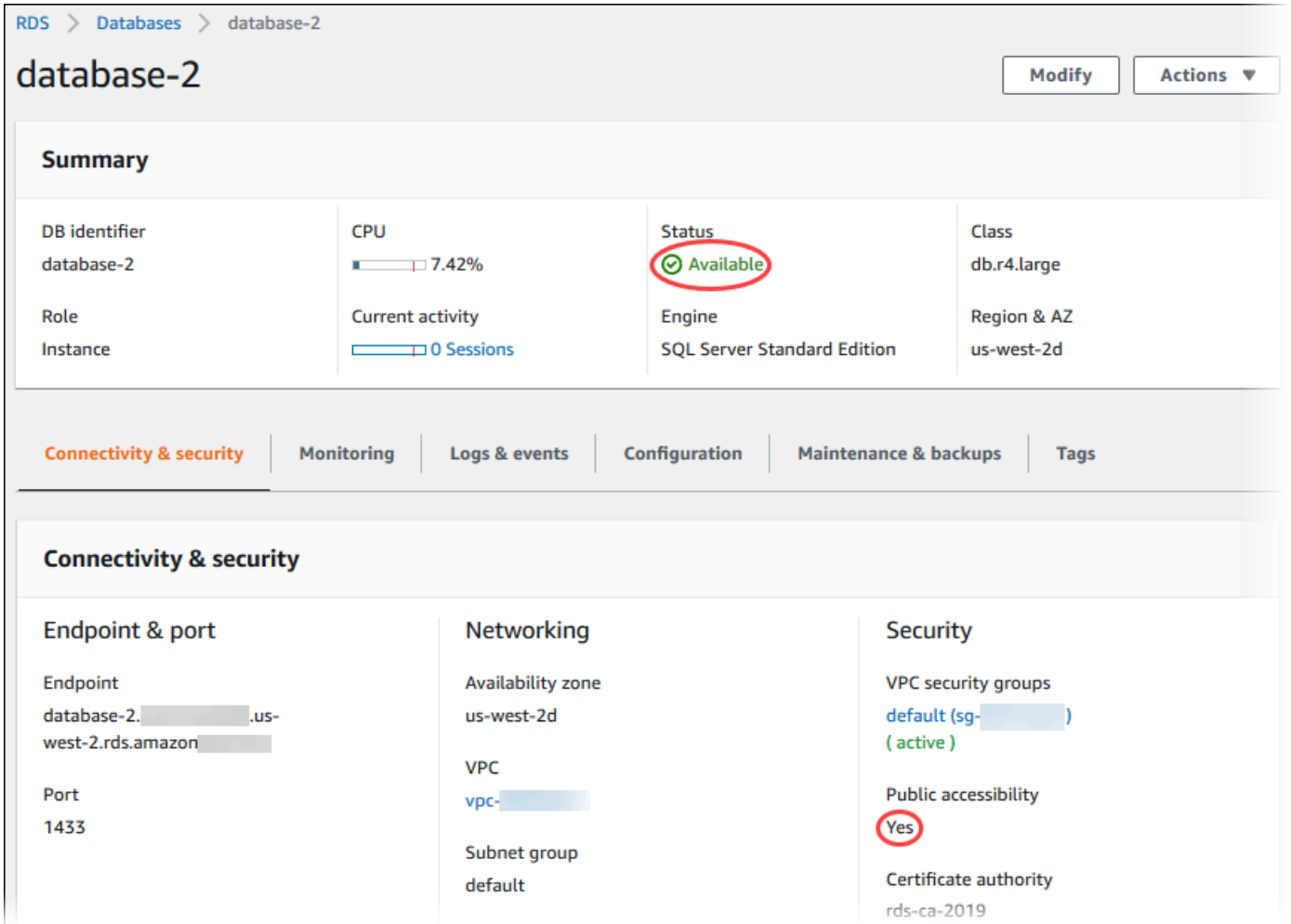
Amazon RDS가 DB 인스턴스를 프로비저닝한 후에는 표준 SQL 클라이언트 애플리케이션을 사용해 DB 인스턴스에 연결할 수 있습니다. 이 주제에서는 Microsoft SQL Server Management Studio(SSMS) 또는 SQL Workbench/J를 사용하여 DB 인스턴스에 연결합니다.

사용자가 샘플 DB 인스턴스를 만들어 연결하는 절차를 실습하는 예제는 Microsoft SQL Server DB 인스턴스 생성 및 해당 인스턴스에 연결 단원을 참조하십시오.

연결하기 전에

DB 인스턴스에 연결하려면 먼저 인스턴스를 사용할 수 있고 액세스할 수 있어야 합니다.

1. 상태가 `available`인지 확인합니다. AWS Management Console의 인스턴스 세부 정보 페이지에서 확인하거나 describe-db-instances AWS CLI 명령을 사용하여 확인할 수 있습니다.



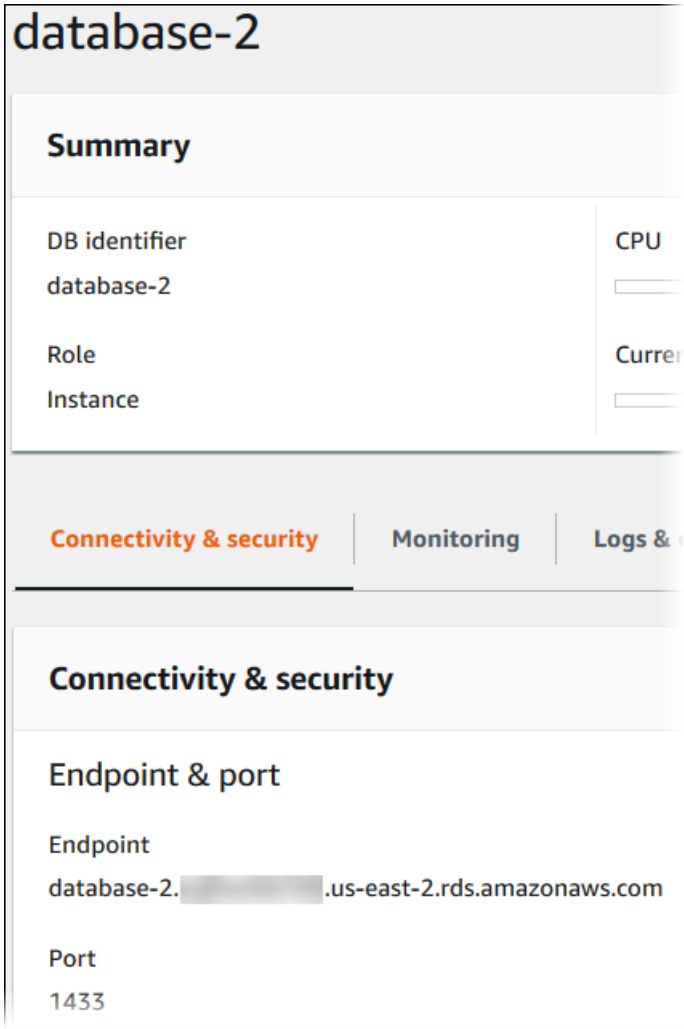
2. 소스에서 액세스할 수 있는지 확인하세요. 시나리오에 따라 공개적으로 액세스할 필요가 없을 수도 있습니다. 자세한 내용은 Amazon VPC 및 Amazon RDS 단원을 참조하십시오.
3. VPC 보안 그룹의 인바운드 규칙이 DB 인스턴스에 대한 액세스를 허용하는지 확인합니다. 자세한 내용은 Amazon RDS DB 인스턴스에 연결할 수 없음 섹션을 참조하세요.

DB 인스턴스 엔드포인트 및 포트 번호 찾기

DB 인스턴스에 연결하려면 엔드포인트와 포트 번호가 모두 필요합니다.

엔드포인트 및 포트를 찾으려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 콘솔의 오른쪽 상단에서 DB 인스턴스의 AWS 리전을 선택합니다.
3. DB 인스턴스의 도메인 이름 시스템(DNS) 이름(엔드포인트) 및 포트 번호를 찾습니다.
    a. RDS 콘솔을 연 다음 데이터베이스를 선택하여 DB 인스턴스의 목록을 표시합니다.
    b. 세부 정보를 표시하고자 하는 SQL Server DB 인스턴스 이름을 선택합니다.
    c. Connectivity & security(연결 및 보안) 탭에서 엔드포인트를 복사합니다.



d. 포트 번호를 적어 둡니다.

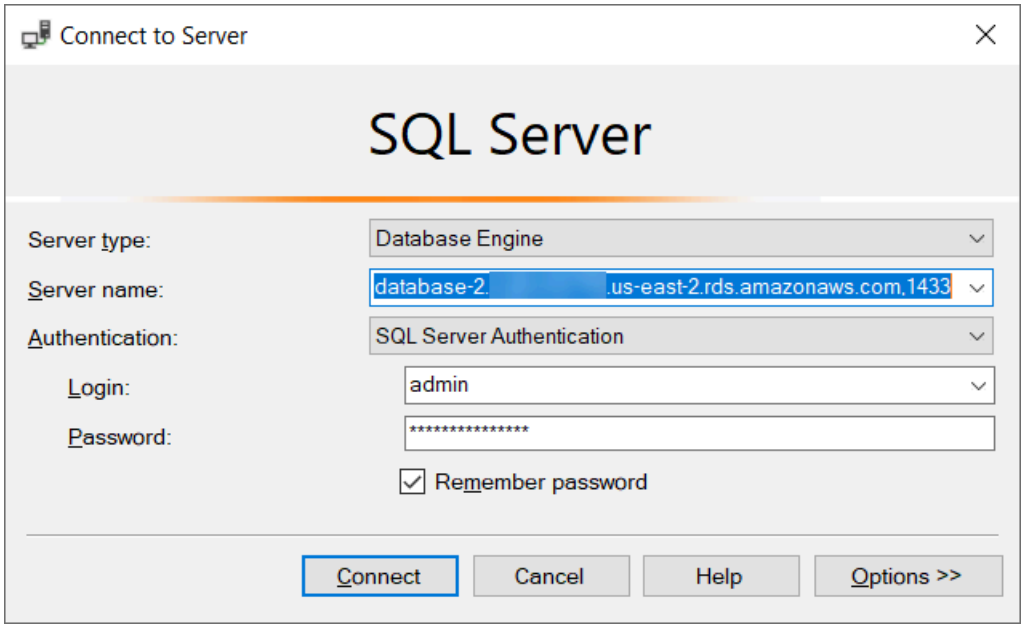
Microsoft SQL Server Management Studio로 DB 인스턴스에 연결

이 절차에서는 Microsoft SQL Server Management Studio(SSMS)를 사용하여 샘플 DB 인스턴스에 연결합니다. 이 유틸리티의 독립 실행형 버전을 다운로드하려면 Microsoft 설명서의 SQL Server Management Studio(SSMS) 다운로드를 참조하십시오.

SSMS를 사용하여 DB 인스턴스에 연결하려면

1. SQL Server Management Studio를 시작합니다.

   Connect to Server 대화 상자가 나타납니다.



2. DB 인스턴스에 대한 정보를 제공합니다.

   a. [Server type]에서 [Database Engine]을 선택합니다.

   b. [서버 이름(Server name)]에 DB 인스턴스의 DNS 이름(엔드포인트) 및 포트 번호를 쉼표로 구분하여 입력합니다.

   > **Important**
   >
   > 엔드포인트와 포트 번호 사이의 콜론을 쉼표로 바꿉니다.

   서버 이름은 다음 예제와 같은 형식이어야 합니다.

   ```
   database-2.cg034itsfake.us-east-1.rds.amazonaws.com,1433
   ```

   c. [Authentication]의 경우 [SQL Server Authentication]을 선택합니다.

   d. 로그인에는 DB 인스턴스의 마스터 사용자 이름을 입력합니다.

   e. 암호에는 DB 인스턴스의 암호를 입력합니다.

3. [Connect]를 선택합니다.

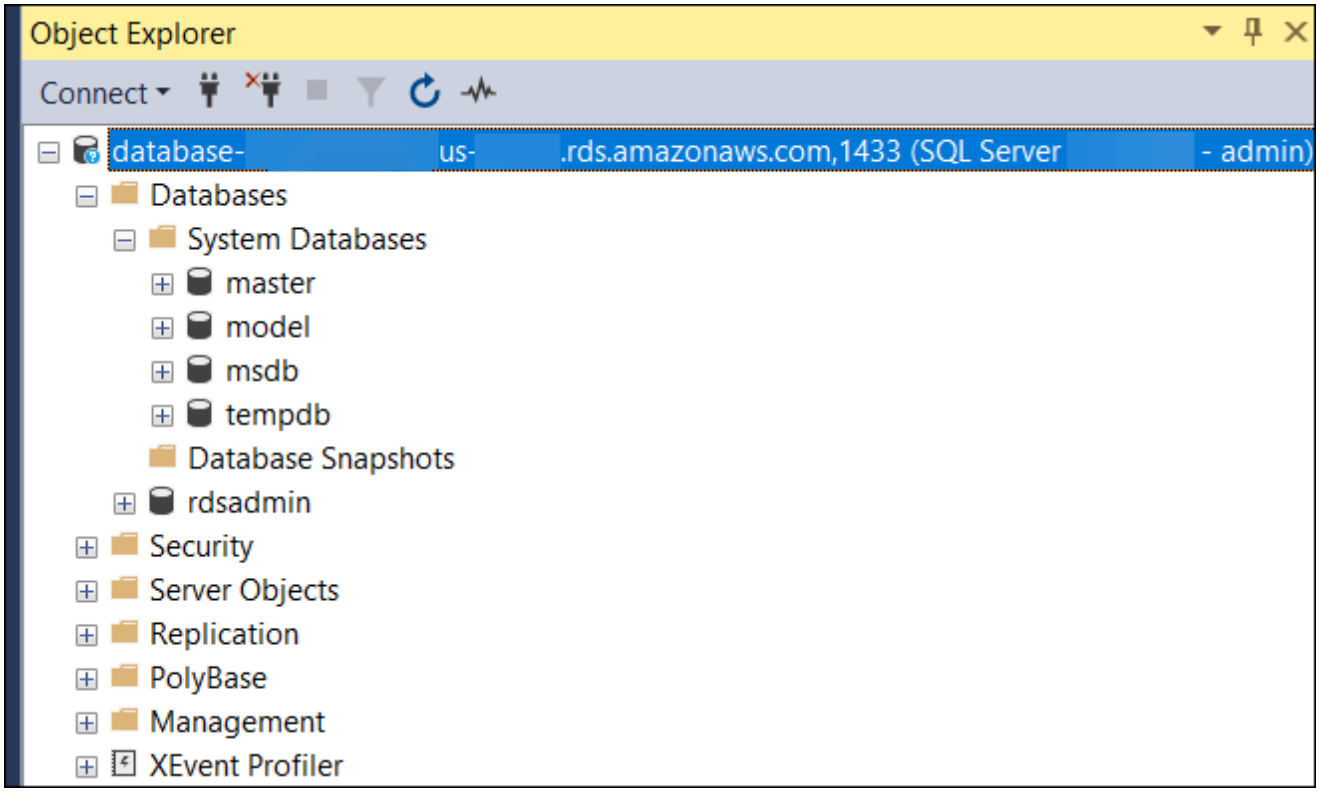
   몇 분 정도 지나면 SSMS가 DB 인스턴스에 연결됩니다.

   DB 인스턴스에 연결할 수 없는 경우 보안 그룹 고려 사항 및 SQL Server DB 인스턴스에 대한 연결 문제 해결 단원을 참조하십시오.

4. SQL Server DB 인스턴스는 SQL Server의 표준 기본 제공 시스템 데이터베이스(`master`, `model`, `msdb` 및 `tempdb`)와 함께 제공됩니다. 시스템 데이터베이스를 탐색하려면 다음을 수행하십시오.

   a. SSMS의 [View] 메뉴에서 [Object Explorer]를 선택합니다.

   b. DB 인스턴스와 데이터베이스를 확장하고 다음과 같이 시스템 데이터베이스를 확장합니다.



5. SQL Server DB 인스턴스는 `rdsadmin`이라는 이름의 데이터베이스와 함께 제공됩니다. Amazon RDS는 이 데이터베이스를 사용하여 데이터베이스를 관리하는 데 사용하는 객체를 저장합니다. `rdsadmin` 데이터베이스에도 고급 작업 수행을 위해 실행할 수 있는 저장 절차가 포함됩니다. 자세한 내용은 Amazon RDS for Microsoft SQL Server에 대한 일반 DBA 작업 섹션을 참조하세요.

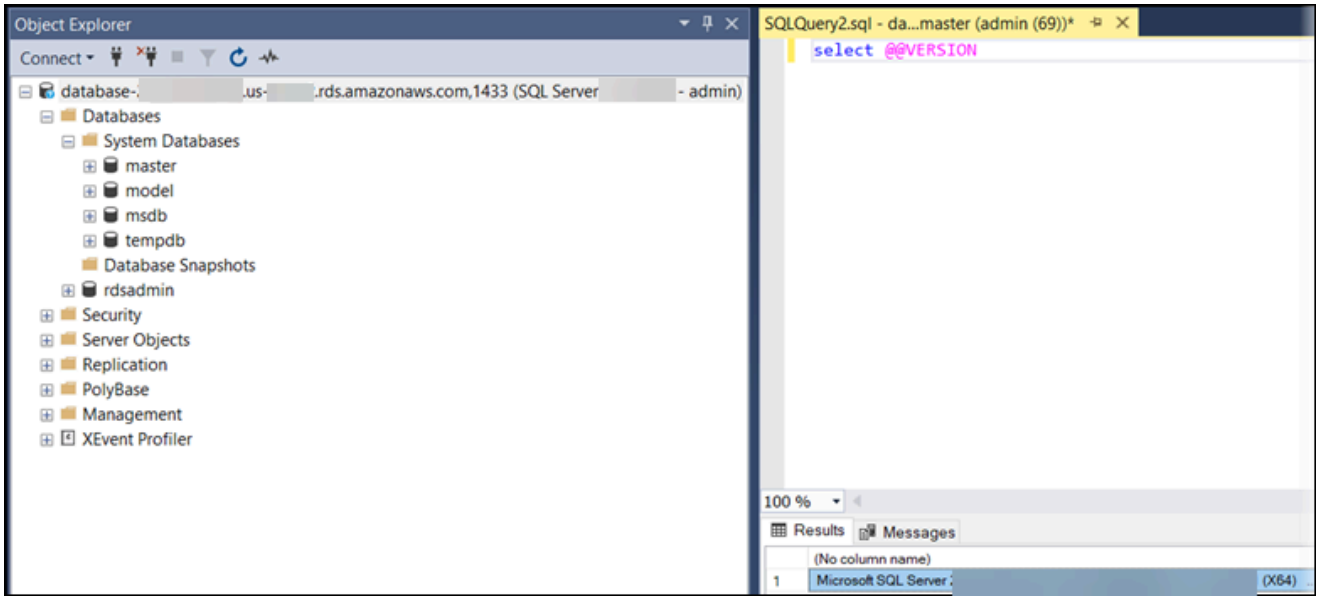
6. 이제 자체 데이터베이스 생성을 시작하고 평소대로 DB 인스턴스와 데이터베이스에 대한 쿼리 실행을 시작할 수 있습니다. DB 인스턴스에 대한 테스트 쿼리를 실행하려면 다음 중 하나를 수행합니다.

   a. SSMS의 [File] 메뉴에서 [New]를 가리킨 후 [Query with Current Connection]을 선택합니다.

   b. 다음 SQL 쿼리를 입력합니다.

```
select @@VERSION
```

   c. 쿼리를 실행합니다. SSMS가 Amazon RDS DB 인스턴스의 SQL Server 버전을 반환합니다.



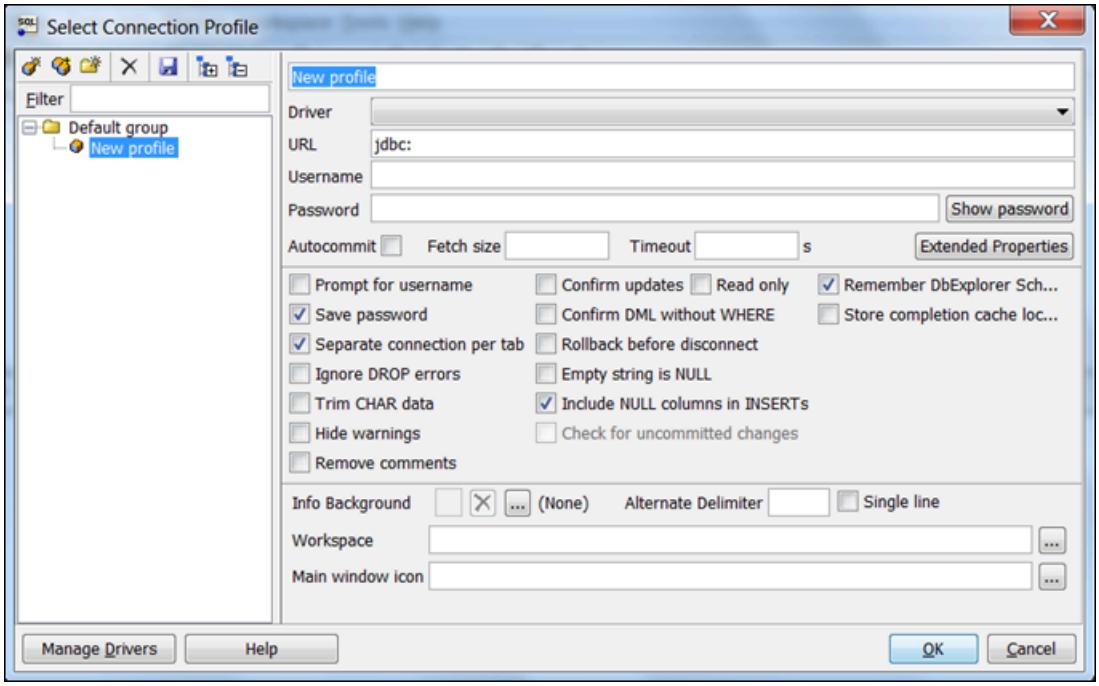
SQL Workbench/J로 DB 인스턴스에 연결

이번 예에서는 SQL Workbench/J 데이터베이스 도구를 사용하여 Microsoft SQL Server 데이터베이스 엔진 기반 DB 인스턴스에 연결하는 방법을 나타냅니다. SQL Workbench/J를 다운로드하려면 SQL Workbench/J를 참조하십시오.

SQL Workbench/J가 JDBC를 이용해 DB 인스턴스에 연결합니다. 그 밖에 SQL Server용 JDBC 드라이버도 필요합니다. 이 드라이버를 다운로드하려면 SQL Server용 Microsoft JDBC 드라이버 다운로드를 참조하세요.

SQL Workbench/J를 사용하여 DB 인스턴스에 연결하려면

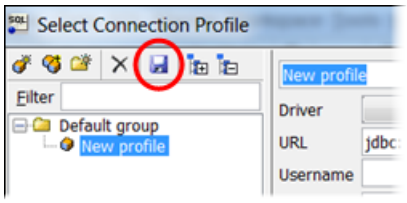
1. SQL Workbench/J를 엽니다. 아래와 같이 [연결 프로파일 선택(Select Connection Profile)] 대화 상자가 나타납니다.



2. 대화 상자 상단의 첫 번째 상자에 프로파일 이름을 입력합니다.
3. 드라이버에서 **SQL JDBC 4.0**을 선택합니다.
4. URL에 **jdbc:sqlserver://**를 입력한 후, DB 인스턴스의 엔드포인트를 입력합니다. 예를 들면 URL 값은 다음과 같습니다.

```
jdbc:sqlserver://sqlsvr-pdz.abcd12340.us-west-2.rds.amazonaws.com:1433
```

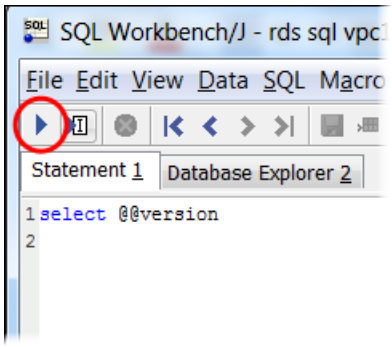
5. 사용자 이름에 DB 인스턴스의 마스터 사용자 이름을 입력하거나 붙여 넣습니다.
6. 암호에 마스터 사용자 암호를 입력합니다.
7. 아래 그림과 같이 대화 상자 도구 모음에서 저장 아이콘을 선택합니다.



8. 확인을 선택합니다. 몇 분 정도 지나면 SQL Workbench/J가 DB 인스턴스에 연결됩니다. DB 인스턴스에 연결할 수 없는 경우 보안 그룹 고려 사항 및 SQL Server DB 인스턴스에 대한 연결 문제 해결 단원을 참조하십시오.
9. 쿼리 창에 다음과 같이 SQL 쿼리를 입력합니다.

```
select @@VERSION
```

10. 아래 그림과 같이 도구 모음에서 Execute 아이콘을 선택합니다.



쿼리가 다음과 같이 DB 인스턴스의 버전 정보를 반환합니다.

```
Microsoft SQL Server 2017 (RTM-CU22) (KB4577467) - 14.0.3356.20 (X64)
```

보안 그룹 고려 사항

DB 인스턴스에 연결하려면 DB 인스턴스가 보안 그룹에 연결되어 있어야 합니다. 이 보안 그룹에는 DB 인스턴스에 액세스하는 데 사용하는 IP 주소와 네트워크 구성이 포함되어 있습니다. DB 인스턴스를 생성할 때 DB 인스턴스를 적합한 보안 그룹에 연결했을 수도 있습니다. DB 인스턴스를 생성할 때 따로 설정할 필요가 없는 기본 보안 그룹을 할당한 경우 DB 인스턴스 방화벽이 연결을 차단합니다.

경우에 따라 액세스를 활성화하기 위해 새 보안 그룹을 생성해야 할 수도 있습니다. 새 보안 그룹 생성에 대한 자세한 내용은 보안 그룹을 통한 액세스 제어 단원을 참조하십시오. VPC 보안 그룹의 규칙 설정 절차를 안내하는 주제는 자습서: DB 인스턴스에 사용할 Amazon VPC 생성(IPv4 전용) 단원을 참조하십시오.

새 보안 그룹을 생성하였으면 보안 그룹과 연결되도록 DB 인스턴스를 수정합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

SSL을 사용하여 DB 인스턴스 연결을 암호화함으로써 보안을 강화할 수 있습니다. 자세한 내용은 ??? 단원을 참조하십시오.

SQL Server DB 인스턴스에 대한 연결 문제 해결

다음 표에는 SQL Server DB 인스턴스에 연결을 시도할 때 발생할 수 있는 오류 메시지가 나와 있습니다.

문제	문제 해결 제안
Could not open a connection to SQL Server – Microsoft SQL Server, Error: 53(SQL Server에 대한 연결을 열 수 없습니다. - Microsoft SQL Server, 오류: 53)	서버 이름을 정확하게 지정했는지 확인하십시오. 서버 이름에서 쉼표로 구분한 샘플 DB 인스턴스의 DNS 이름과 포트 번호를 입력합니다. **Important** DNS 이름과 포트 번호 사이에 콜론이 있는 경우 콜론을 쉼표로 변경합니다. 서버 이름은 다음 예제와 같은 형식이어야 합니다. `sample-instance.cg034itsfake.us-east-1.rds.amazonaws.com,1433`
No connection could be made because the target machine actively refused it – Microsoft SQL Server, Error: 10061(대상 컴퓨터에서 연결을 거부했으므로 연결하지 못했습니다. - Microsoft SQL Server, 오류: 10061)	DB 인스턴스에 연결할 수 있지만 연결이 거부되었습니다. 이 문제는 주로 사용자 이름이나 암호를 잘못 지정하면 발생합니다. 사용자 이름과 암호를 확인하고 다시 시도하십시오.
A network-related or instance-specific error occurred while establishing a connection to SQL Server. The server was not found or was not accessible... The wait operation timed out – Microsoft SQL Server, Error: 258(SQL Server에 대한 연결 설정 중	로컬 방화벽에서 적용되는 액세스 규칙과 DB 인스턴스에 액세스할 수 있는 권한이 부여된 IP 주소가 일치하지 않을 수 있습니다. 보안 그룹의 인바운드 규칙에 문제가 있을 가능성이 매우 높습니다. 자세한 내용은 Amazon RDS의 보안 단원을 참조하십시오. 데이터베이스 인스턴스에 공개적으로 액세스할 수 있어야 합니다. VPC 외부에서 인스턴스에 연결하려면 인스턴스에 퍼블릭 IP 주소가 할당되어 있어야 합니다.

문제	문제 해결 제안
네트워크 관련 또는 인스턴스 관련 오류가 발생했습니다. 서버를 찾을 수 없거나 서버에 액세스할 수 없습니다. 대기 작업이 시간 초과되었습니다. - Microsoft SQL Server, 오류: 258)	

> **Note**
>
> 연결 문제에 대한 자세한 내용은 Amazon RDS DB 인스턴스에 연결할 수 없음 단원을 참조하십시오.

RDS for SQL Server를 사용하여 Active Directory 작업

RDS for SQL Server DB 인스턴스를 Microsoft Active Directory(AD) 도메인에 가입시킬 수 있습니다. AD 도메인은 AWS 내의 AWS 관리형 AD, 선택한 위치의 자체 관리형 AD(기업 데이터 센터 포함) 또는 AWS EC2에서 호스팅하거나 다른 클라우드 공급자를 통해 호스팅할 수 있습니다.

자체 관리형 Active Directory의 NTLM 인증을 사용하여 도메인 사용자를 인증할 수 있습니다. AWS 관리형 Active Directory에서 Kerberos 및 NTLM 인증을 사용할 수 있습니다.

다음 섹션에서는 Amazon RDS에서 Microsoft SQL Server용 자체 관리형 Active Directory 및 AWS 관리형 Active Directory를 사용하여 작업하는 정보를 찾을 수 있습니다.

주제

- Amazon RDS for SQL Server DB 인스턴스를 사용하여 자체 관리형 Active Directory 작업
- RDS for SQL Server를 사용하여 AWS 관리형 Active Directory 작업

Amazon RDS for SQL Server DB 인스턴스를 사용하여 자체 관리형 Active Directory 작업

AD가 호스팅되는 위치(기업 데이터 센터, AWS EC2 또는 기타 클라우드 공급자)에 관계없이 RDS for SQL Server DB 인스턴스를 자체 관리형 Active Directory(AD) 도메인에 직접 가입시킬 수 있습니다. 자체 관리형 AD를 사용하면 중간 도메인 및 포리스트 신뢰를 사용하지 않고도 NTLM 인증을 사용하여 RDS for SQL Server DB 인스턴스의 사용자 및 서비스 인증을 직접 제어할 수 있습니다. 사용자가 자체 관리형 AD 도메인에 가입된 RDS for SQL Server DB 인스턴스를 사용하여 인증하면 인증 요청이 지정한 자체 관리형 AD 도메인으로 전달됩니다.

주제

- 리전 및 버전 사용 가능 여부
- 요구 사항
- 제한 사항
- 자체 관리형 Active Directory 설정 개요
- 자체 관리형 Active Directory 설정
- 자체 관리형 Active Directory 도메인에서 DB 인스턴스 관리
- 자체 관리형 Active Directory 도메인 멤버십에 대한 이해
- 자체 관리형 Active Directory 문제 해결
- SQL Server DB 인스턴스를 복원한 후 자체 관리형 Active Directory 도메인에 추가

리전 및 버전 사용 가능 여부

Amazon RDS는 모든 AWS 리전에서 NTLM을 사용하는 SQL Server용 자체 관리형 AD를 지원합니다.

요구 사항

RDS for SQL Server DB 인스턴스를 자체 관리형 AD 도메인에 가입시키기 전에 다음 요구 사항을 충족해야 합니다.

주제

- 온프레미스 AD 구성
- 네트워크 연결 구성
- AD 도메인 서비스 계정 구성

온프레미스 AD 구성

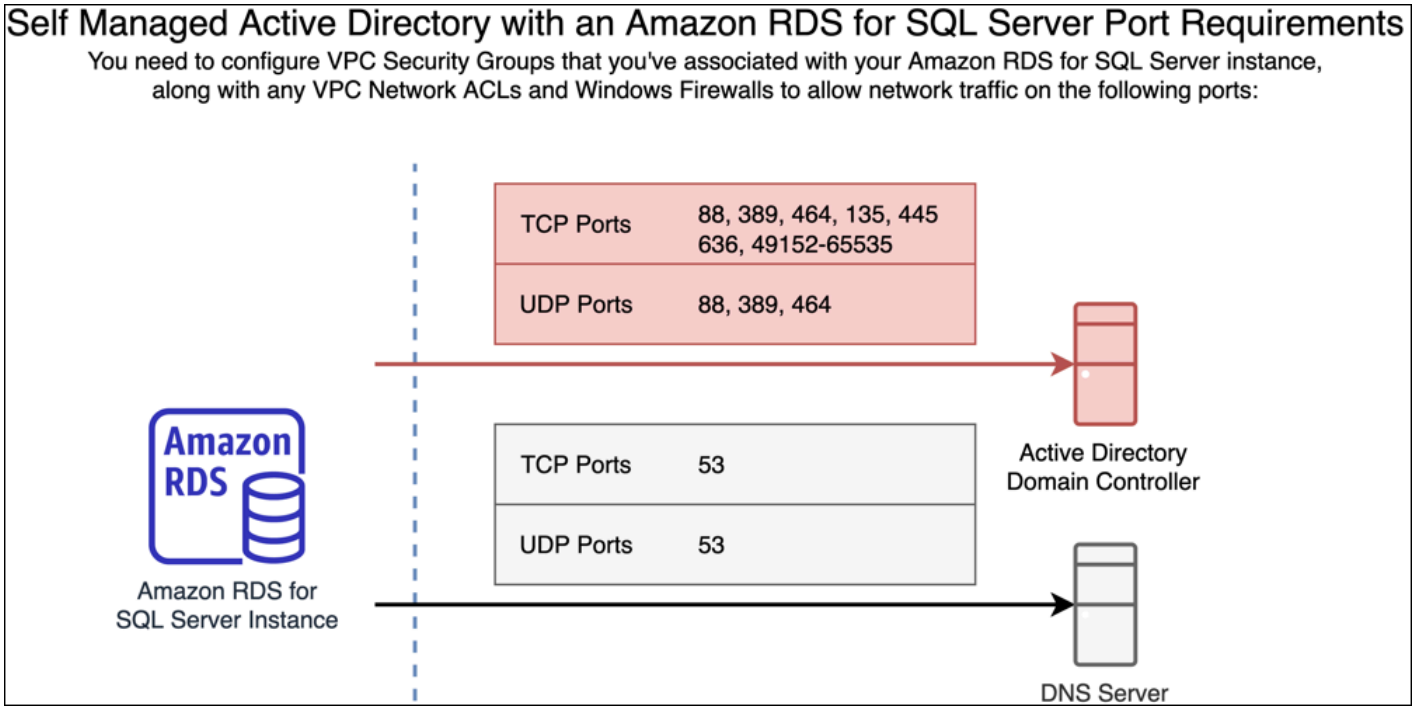
Amazon RDS for SQL Server 인스턴스에 가입시킬 수 있는 온프레미스 또는 기타 자체 관리형 Microsoft AD가 있어야 합니다. 온프레미스 AD는 다음과 같은 구성을 가져야 합니다.

- Active Directory 사이트가 정의되어 있는 경우 RDS for SQL Server DB 인스턴스와 연결된 VPC의 서브넷이 Active Directory 사이트에 정의되어 있는지 확인합니다. VPC의 서브넷과 다른 AD 사이트의 서브넷 간에 충돌이 없는지 확인합니다.
- AD 도메인 컨트롤러의 도메인 기능 수준은 Windows Server 2008 R2 이상입니다.
- AD 도메인 이름은 단일 레이블 도메인(SLD) 형식일 수 없습니다. RDS for SQL Server는 SLD 도메인을 지원하지 않습니다.
- AD의 정규화된 도메인 이름(FQDN)이 47자를 초과하면 안 됩니다.

네트워크 연결 구성

다음 네트워크 구성을 충족하는지 확인합니다.

- RDS for SQL Server DB 인스턴스를 생성하려는 Amazon VPC와 자체 관리형 Active Directory 간에 연결이 구성되었습니다. AWS Direct Connect, AWS VPN, VPC 피어링 또는 AWS 전송 게이트웨이를 사용하여 연결을 설정할 수 있습니다.
- VPC 보안 그룹의 경우 기본 Amazon VPC의 기본 보안 그룹이 콘솔의 RDS for SQL Server DB 인스턴스에 이미 추가되었습니다. RDS for SQL Server DB 인스턴스를 만드는 서브넷의 보안 그룹과 VPC 네트워크 ACL이 다음 다이어그램에 표시된 방향으로 포트를 통한 트래픽을 허용하는지 확인합니다.



다음 테이블에는 각 포트의 역할이 나와 있습니다.

프로토콜	포트	역할
TCP/UDP	53	도메인 이름 시스템(DNS)
TCP/UDP	88	Kerberos 인증
TCP/UDP	464	암호 변경/설정
TCP/UDP	389	Lightweight Directory Access Protocol(LDAP)
TCP	135	분산 컴퓨팅 환경/엔드포인트 매퍼(DCE/EPMAP)
TCP	445	디렉터리 서비스 SMB 파일 공유
TCP	636	Lightweight Directory Access Protocol over TLS/SSL(LDAPS)

프로토콜	포트	역할
TCP	49,152~65,535	RPC용 임시 포트

- 일반적으로 도메인 DNS 서버는 AD 도메인 컨트롤러에 있습니다. 이 기능을 사용하기 위해 VPC DHCP 옵션 세트를 구성할 필요는 없습니다. 자세한 정보는 Amazon VPC 사용 설명서의 DHCP 옵션 세트를 참조하세요.

> **Important**
>
> VPC 네트워크 ACL을 사용하는 경우 RDS for SQL Server DB 인스턴스의 동적 포트(49152-65535)를 통한 아웃바운드 트래픽도 허용해야 합니다. 이러한 트래픽 규칙이 각 AD 도메인 컨트롤러, DNS 서버 및 RDS for SQL Server DB 인스턴스에 적용되는 방화벽에도 반영되는지 확인하세요.
> VPC 보안 그룹에서는 네트워크 트래픽이 시작되는 방향으로만 포트를 열어야 하지만, 대부분의 Windows 방화벽과 VPC 네트워크 ACL에서는 포트가 양방향으로 열려 있어야 합니다.

AD 도메인 서비스 계정 구성

AD 도메인 서비스 계정에 대한 다음 요구 사항을 충족했는지 확인합니다.

- 자체 관리형 AD 도메인에는 컴퓨터를 도메인에 연결할 수 있는 권한이 위임된 서비스 계정이 있어야 합니다. 도메인 서비스 계정은 특정 작업을 수행할 권한이 위임된 자체 관리형 AD의 사용자 계정입니다.
- 도메인 서비스 계정은 RDS for SQL Server DB 인스턴스에 가입시키려는 조직 단위(OU)에서 다음 권한을 위임받아야 합니다.
  - DNS 호스트 이름에 쓸 수 있는 검증된 기능
  - 서비스 보안 주체 이름에 쓸 수 있는 검증된 기능
  - 컴퓨터 객체 생성 및 삭제

  이러한 권한은 컴퓨터 객체를 자체 관리형 Active Directory에 가입시키는 데 필요한 최소 권한 집합을 나타냅니다. 자세한 내용은 Microsoft Windows Server 설명서에서 컴퓨터를 도메인에 연결하려고 할 때 발생하는 오류를 참조하세요.

> **Important**
>
> DB 인스턴스를 만든 후에는 RDS for SQL Server가 조직 단위에서 생성한 컴퓨터 객체를 옮기지 마세요. 연결된 객체를 이동하면 RDS for SQL Server DB 인스턴스가 잘못 구성될 수 있습니다. Amazon RDS에서 생성한 컴퓨터 객체를 이동해야 하는 경우 ModifyDBInstance RDS API 작업을 사용하여 컴퓨터 객체의 원하는 위치로 도메인 파라미터를 수정하세요.

제한 사항

SQL Server용 자체 관리형 AD에는 다음과 같은 제한 사항이 적용됩니다.

- NTLM이 지원되는 유일한 인증 유형입니다. Kerberos 인증은 지원되지 않습니다. Kerberos 인증을 사용해야 하는 경우 자체 관리형 AD 대신 AWS 관리형 AD를 사용할 수 있습니다.
- Microsoft Distributed Transaction Coordinator(MSDTC) 서비스는 Kerberos 인증이 필요하므로 지원되지 않습니다.
- RDS for SQL Server DB 인스턴스에서는 자체 관리형 AD 도메인의 Network Time Protocol(NTP) 서버를 사용하지 않습니다. 대신 AWS NTP 서비스를 사용합니다.
- SQL Server 연결 서버는 자체 관리형 AD 도메인에 가입된 다른 RDS for SQL Server DB 인스턴스에 연결하는 데 SQL 인증을 사용해야 합니다.
- 자체 관리형 AD 도메인의 Microsoft Group Policy Object(GPO) 설정은 RDS for SQL Server DB 인스턴스에 적용되지 않습니다.

자체 관리형 Active Directory 설정 개요

RDS for SQL Server DB 인스턴스용 자체 관리형 AD를 설정하려면 다음 단계를 수행하세요. 자세한 내용은 자체 관리형 Active Directory 설정에서 설명합니다.

AD 도메인에서:

- 조직 단위(OU)를 생성합니다.
- AD 도메인 사용자를 생성합니다.
- AD 도메인 사용자에게 제어를 위임합니다.

AWS Management Console 또는 API에서:

- AWS KMS 키를 생성합니다.

- AWS Secrets Manager를 사용하여 보안 암호를 생성합니다.
- RDS for SQL Server DB 인스턴스를 만들거나 수정하여 자체 관리형 AD 도메인에 가입시킵니다.

자체 관리형 Active Directory 설정

자체 관리형 AD를 설정하려면 다음 단계를 따르세요.

주제

- 1단계: AD에서 조직 단위 생성
- 2단계: AD에서 AD 도메인 사용자 생성
- 3단계: AD 사용자에게 제어 권한 위임
- 4단계: AWS KMS 키 생성
- 5단계: AWS 보안 암호 생성
- 6단계: SQL Server DB 인스턴스 생성 또는 수정
- 7단계: Windows 인증 SQL Server 로그인 생성

1단계: AD에서 조직 단위 생성

> **Important**
>
> 자체 관리형 AD 도메인에 가입된 RDS for SQL Server DB 인스턴스를 소유한 AWS 계정에 대해 전용 OU 및 해당 OU 범위의 서비스 보안 인증을 만드는 것이 좋습니다. 전용 OU 및 서비스 보안 인증을 지정하면 권한 충돌을 피하고 최소 권한 원칙을 따를 수 있습니다.

AD에서 OU를 생성하려면

1. 도메인 관리자로 AD 도메인에 연결합니다.
2. Active Directory 사용자 및 컴퓨터를 열고 OU를 생성할 도메인을 선택합니다.
3. 도메인을 마우스 오른쪽 버튼으로 클릭하고 새로 만들기를 선택한 다음 조직 단위를 선택합니다.
4. OU 이름을 입력합니다.
5. 컨테이너가 실수로 삭제되지 않도록 보호 확인란을 선택한 상태로 유지합니다.
6. 확인을 클릭합니다. 새 OU는 도메인 아래에 표시됩니다.

2단계: AD에서 AD 도메인 사용자 생성

도메인 사용자 보안 인증 정보는 AWS Secrets Manager에서 보안 암호로 사용됩니다.

AD에서 AD 도메인 사용자를 생성하려면

1. Active Directory 사용자 및 컴퓨터를 열고 사용자를 생성할 도메인과 OU를 선택합니다.
2. 사용자 객체를 마우스 오른쪽 버튼으로 클릭하고 새로 만들기, 사용자 순으로 선택합니다.
3. 사용자의 이름, 성 및 로그온 이름을 입력합니다. 다음을 클릭합니다.
4. 사용자의 암호를 입력합니다. “다음 로그인 시 사용자가 암호를 변경해야 함”을 선택하지 마세요. “계정이 비활성화됨”을 선택하지 마세요. 다음을 클릭합니다.
5. 확인을 클릭합니다. 새 사용자는 도메인 아래에 표시됩니다.

3단계: AD 사용자에게 제어 권한 위임

도메인의 AD 도메인 사용자에게 제어 권한을 위임하려면

1. Active Directory 사용자 및 컴퓨터 MMC 스냅인을 열고 사용자를 생성할 도메인을 선택합니다.
2. 이전에 만든 OU를 마우스 오른쪽 버튼으로 클릭하고 제어 위임을 선택합니다.
3. 제어 위임 마법사에서 다음을 클릭합니다.
4. 사용자 또는 그룹 섹션에서 추가를 클릭합니다.
5. 사용자, 컴퓨터 또는 그룹 선택 섹션에서 생성한 AD 사용자를 입력하고 이름 확인을 클릭합니다. AD 사용자 확인에 성공하면 확인을 클릭합니다.
6. 사용자 또는 그룹 섹션에서 AD 사용자가 추가되었는지 확인하고 다음을 클릭합니다.
7. 위임할 작업 섹션에서 위임할 사용자 지정 작업 생성을 선택하고 다음을 클릭합니다.
8. Active Directory 객체 유형 섹션에서:
    a. 폴더의 다음 객체만을 선택합니다.
    b. 컴퓨터 객체를 선택합니다.
    c. 이 폴더에서 선택한 객체 생성을 선택합니다.
    d. 이 폴더에서 선택한 객체 삭제를 선택하고 다음을 클릭합니다.
9. 권한 섹션에서:
    a. 일반을 선택한 상태로 유지합니다.
    b. DNS 호스트 이름에 대한 검증된 쓰기를 선택합니다.

c. 서비스 보안 주체 이름에 대한 검증된 쓰기를 선택하고 다음을 클릭합니다.

10. 제어 위임 마법사를 완료하려면 설정을 검토 및 확인한 다음 마침을 클릭합니다.

4단계: AWS KMS 키 생성

KMS 키는 AWS 보안 암호를 암호화하는 데 사용됩니다.

AWS KMS 키를 생성하려면

> **Note**
>
> 암호화 키의 경우 AWS 기본 KMS 키를 사용하지 마세요. 자체 관리형 AD에 가입시키려는 RDS for SQL Server DB 인스턴스가 포함된 동일한 AWS 계정에 AWS KMS 키를 만들어야 합니다.

1. AWS KMS 콘솔에서 키 생성을 선택합니다.
2. 키 유형에 대해 대칭을 선택합니다.
3. 키 사용에서 암호화 및 암호 해독을 선택합니다.
4. 고급 옵션에서 다음을 수행합니다.
   a. 키 구성 요소 오리진에서 KMS를 선택합니다.
   b. 리전 구분에서 단일 리전 키를 선택하고 다음을 클릭합니다.
5. 별칭의 경우 KMS 키의 이름을 입력합니다.
6. (선택 사항) 설명에 KMS 키에 대한 설명을 입력합니다.
7. (선택 사항) 태그의 경우 KMS 키에 태그를 지정하고 다음을 클릭합니다.
8. 키 관리자의 경우 IAM 사용자의 이름을 입력하고 선택합니다.
9. 키 삭제의 경우 키 관리자가 이 키를 삭제하도록 허용 확인란을 선택한 상태로 유지하고 다음을 클릭합니다.
10. 키 사용자의 경우 이전 단계와 동일한 IAM 사용자를 입력하고 해당 사용자를 선택합니다. 다음을 클릭합니다.
11. 구성을 검토합니다.
12. 주요 정책의 경우 정책 문에 다음을 포함하세요.

```
{
```

```
    "Sid": "Allow use of the KMS key on behalf of RDS",
    "Effect": "Allow",
    "Principal": {
        "Service": [
            "rds.amazonaws.com"
        ]
    },
    "Action": "kms:Decrypt",
    "Resource": "*"
}
```

13. 완료를 클릭합니다.

5단계: AWS 보안 암호 생성

보안 암호 생성

> **Note**
>
> 자체 관리형 AD에 가입시키려는 RDS for SQL Server DB 인스턴스가 포함된 동일한 AWS 계정에 보안 암호를 만들어야 합니다.

1. AWS Secrets Manager에서 새 보안 암호 저장을 선택합니다.
2. 보안 암호 유형에서 다른 유형의 보안 암호를 선택합니다.
3. 키/값 쌍의 경우 2개의 키를 추가합니다.
    a. 첫 번째 키에는 CUSTOMER_MANAGED_ACTIVE_DIRECTORY_USERNAME를 입력합니다.
    b. 첫 번째 키 값에는 AD 사용자의 사용자 이름(도메인 접두사 제외)만 입력합니다. 도메인 이름을 포함하면 인스턴스 생성이 실패하므로 포함하지 마세요.
    c. 두 번째 키에는 CUSTOMER_MANAGED_ACTIVE_DIRECTORY_PASSWORD를 입력합니다.
    d. 두 번째 키의 값으로 도메인의 AD 사용자에 대해 생성한 암호를 입력합니다.
4. 암호화 키에는 이전 단계에서 생성한 KMS 키를 입력하고 다음을 클릭합니다.
5. 보안 암호 이름에는 나중에 암호를 찾는 데 도움이 되는 설명이 포함된 이름을 입력합니다.
6. (선택 사항) 설명에 보안 암호 이름에 대한 설명을 입력합니다.
7. 리소스 권한의 경우 편집을 클릭합니다.
8. 권한 정책에 다음 정책을 추가합니다.

> **Note**
>
> 혼란스러운 대리인 문제를 피하기 위해 정책의 `aws:sourceAccount` 및 `aws:sourceArn` 조건을 사용하는 것이 좋습니다. `aws:sourceAccount`에는 AWS 계정를 사용하고 `aws:sourceArn`에는 RDS for SQL Server DB 인스턴스를 사용합니다. 자세한 내용은 교차 서비스 혼동된 대리자 문제 방지 섹션을 참조하세요.

JSON

```
{
    "Version": "2012-10-17",
    "Statement":
    [
        {
            "Effect": "Allow",
            "Principal":
            {
                "Service": "rds.amazonaws.com"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "*",
            "Condition":
            {
                "StringEquals":
                {
                    "aws:sourceAccount": "123456789012"
                },
                "ArnLike":
                {
                    "aws:sourceArn": "arn:aws:rds:us-
west-2:123456789012:db:*"
                }
            }
        }
    ]
}
```

9. 저장을 클릭한 후 다음을 클릭합니다.
10. 교체 설정 구성에서 기본값을 유지하고 다음을 선택합니다.

11. 보안 암호 설정을 검토하고 저장을 클릭합니다.
12. 생성한 보안 암호를 선택하고 보안 암호 ARN의 값을 복사합니다. 이는 다음 단계에서 자체 관리형 Active Directory를 설정하는 데 사용됩니다.

6단계: SQL Server DB 인스턴스 생성 또는 수정

콘솔, CLI 또는 RDS API를 사용하여 RDS for SQL Server DB 인스턴스를 자체 관리형 AD 도메인에 연결할 수 있습니다. 이 작업을 다음 중 한 가지 방법으로 수행할 수 있습니다.

- 콘솔, create-db-instance CLI 명령 또는 CreateDBInstance RDS API 작업을 사용하여 새 SQL Server DB 인스턴스를 생성합니다.

  지침은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
- 콘솔, modify-db-instance CLI 명령 또는 ModifyDBInstance RDS API 작업을 사용하여 기존 SQL Server DB 인스턴스를 수정합니다.

  지침은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.
- 콘솔, restore-db-instance-from-db-snapshot CLI 명령 또는 RestoreDBInstanceFromDBSnapshot RDS API 작업을 사용하여 DB 스냅샷에서 SQL Server DB 인스턴스를 복원합니다.

  지침은 DB 인스턴스 복원 섹션을 참조하세요.
- 콘솔, restore-db-instance-to-point-in-time CLI 명령 또는 RestoreDBInstanceToPointInTime RDS API 작업을 사용하여 SQL Server DB 인스턴스를 특정 시점으로 복구합니다.

  지침은 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 섹션을 참조하세요.

AWS CLI를 사용하는 경우 DB 인스턴스가 생성한 자체 관리형 Active Directory 도메인을 사용하려면 다음과 같은 파라미터가 필요합니다.

- `--domain-fqdn` 파라미터에는 자체 관리형 Active Directory의 정규화된 도메인 이름(FQDN)을 사용합니다.
- `--domain-ou` 파라미터에는 자체 관리형 AD에서 만든 OU를 사용합니다.
- `--domain-auth-secret-arn` 파라미터에는 이전 단계에서 생성한 보안 암호 ARN의 값을 사용합니다.
- `--domain-dns-ips` 파라미터에는 자체 관리형 AD용 DNS 서버의 프라이머리 및 보조 IPv4 주소를 사용합니다. 보조 DNS 서버 IP 주소가 없는 경우 프라이머리 IP 주소를 2번 입력합니다.

다음 예제 CLI 명령은 자체 관리형 AD 도메인이 있는 RDS for SQL Server DB 인스턴스를 생성, 수정 및 제거하는 방법을 보여줍니다.

> **Important**
>
> 자체 관리형 AD 도메인에 가입하거나 자체 관리형 AD 도메인에서 제거하도록 DB 인스턴스를 수정한 경우 변경 사항이 적용되려면 DB 인스턴스를 재부팅해야 합니다. 변경 사항을 즉시 적용하거나 다음 유지 관리 기간까지 기다릴 수 있습니다. 즉시 적용 옵션을 선택하면 단일 AZ DB 인스턴스에 가동 중단이 발생합니다. 다중 AZ DB 인스턴스는 재부팅을 완료하기 전에 장애 조치를 수행합니다. 자세한 내용은 수정 예약 설정 사용 섹션을 참조하세요.

다음 CLI 명령은 새 RDS for SQL Server DB 인스턴스를 만들어 자체 관리형 AD 도메인에 가입시킵니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance \
    --db-instance-identifier my-DB-instance \
    --db-instance-class db.m5.xlarge \
    --allocated-storage 50 \
    --engine sqlserver-se \
    --engine-version 15.00.4043.16.v1 \
    --license-model license-included \
    --master-username my-master-username \
    --master-user-password my-master-password \
    --domain-fqdn my_AD_domain.my_AD.my_domain \
    --domain-ou OU=my-AD-test-OU,DC=my-AD-test,DC=my-AD,DC=my-domain \
    --domain-auth-secret-arn "arn:aws:secretsmanager:region:account-number:secret:my-
AD-test-secret-123456" \
    --domain-dns-ips "10.11.12.13" "10.11.12.14"
```

Windows의 경우:

```
aws rds create-db-instance ^
    --db-instance-identifier my-DB-instance ^
    --db-instance-class db.m5.xlarge ^
    --allocated-storage 50 ^
    --engine sqlserver-se ^
    --engine-version 15.00.4043.16.v1 ^
    --license-model license-included ^
```

```
    --master-username my-master-username ^
    --master-user-password my-master-password ^
    --domain-fqdn my-AD-test.my-AD.mydomain ^
    --domain-ou OU=my-AD-test-OU,DC=my-AD-test,DC=my-AD,DC=my-domain ^
    --domain-auth-secret-arn "arn:aws:secretsmanager:region:account-number:secret:my-
AD-test-secret-123456" \ ^
    --domain-dns-ips "10.11.12.13" "10.11.12.14"
```

다음 CLI 명령은 자체 관리형 Active Directory 도메인을 사용하도록 기존 RDS for SQL Server DB 인스턴스를 수정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier my-DB-instance \
    --domain-fqdn my_AD_domain.my_AD.my_domain \
    --domain-ou OU=my-AD-test-OU,DC=my-AD-test,DC=my-AD,DC=my-domain \
    --domain-auth-secret-arn "arn:aws:secretsmanager:region:account-number:secret:my-
AD-test-secret-123456" \
    --domain-dns-ips "10.11.12.13" "10.11.12.14"
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-DBinstance ^
    --domain-fqdn my_AD_domain.my_AD.my_domain ^
    --domain-ou OU=my-AD-test-OU,DC=my-AD-test,DC=my-AD,DC=my-domain ^
    --domain-auth-secret-arn "arn:aws:secretsmanager:region:account-number:secret:my-
AD-test-secret-123456" ^
    --domain-dns-ips "10.11.12.13" "10.11.12.14"
```

다음 CLI 명령은 자체 관리형 Active Directory 도메인에서 RDS for SQL Server DB 인스턴스를 제거합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier my-DB-instance \
    --disable-domain
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-DB-instance ^
    --disable-domain
```

7단계: Windows 인증 SQL Server 로그인 생성

다른 DB 인스턴스의 경우와 같은 방법으로 Amazon RDS 마스터 사용자 보안 인증 정보를 사용하여 SQL Server DB 인스턴스에 연결합니다. DB 인스턴스는 자체 관리형 AD 도메인에 가입되므로 SQL Server 로그인 및 사용자를 프로비저닝할 수 있습니다. 이 작업은 자체 관리형 AD 도메인의 AD 사용자 및 그룹 유틸리티에서 수행합니다. 데이터베이스 권한은 이러한 Windows 로그인에 부여되거나 취소되는 표준 SQL Server 권한을 통해 관리됩니다.

자체 관리형 AD 사용자가 SQL 서버를 사용하여 인증하려면 자체 관리형 AD 사용자 또는 사용자가 구성원인 자체 관리형 Active Directory 그룹에 대한 SQL Server Windows 로그인이 있어야 합니다. 세분화된 액세스 제어는 이러한 SQL Server 로그인에 대한 권한을 부여하거나 취소하여 처리합니다. SQL Server 로그인이 없거나 이러한 로그인이 있는 자체 관리형 AD 그룹에 속하지 않은 자체 관리형 AD 사용자는 SQL Server DB 인스턴스에 액세스할 수 없습니다.

자체 관리형 AD SQL Server 로그인을 생성하려면 ALTER ANY LOGIN 권한이 필요합니다. 이 권한으로 로그인을 생성하지 않은 경우 SQL Server 인증을 사용하여 DB 인스턴스의 마스터 사용자로 연결하고 마스터 사용자의 맥락에서 자체 관리형 AD SQL Server 로그인을 생성합니다.

다음과 같은 데이터 정의 언어(DDL) 명령을 실행하여 자체 관리형 AD 사용자 또는 그룹에 대한 SQL Server 로그인을 생성할 수 있습니다.

> **Note**
>
> Windows 2000 이전 로그인 이름을 사용하여 사용자 및 그룹을 *my_AD_domain**my_AD_domain_user* 형식으로 지정합니다. UPN(User Principle Name)을 *my_AD_domain_user*@*my_AD_domain* 형식으로 사용할 수 없습니다.

```
USE [master]
GO
CREATE LOGIN [my_AD_domain\my_AD_domain_user] FROM WINDOWS WITH DEFAULT_DATABASE =
 [master], DEFAULT_LANGUAGE = [us_english];
GO
```

자세한 내용은 Microsoft Developer Network 설명서에서 CREATE LOGIN(Transact-SQL)을 참조하세요.

도메인의 사용자(사람 및 애플리케이션)는 이제 Windows 인증을 사용하여 자체 관리형 AD 도메인이 연결된 클라이언트 컴퓨터의 RDS for SQL Server 인스턴스에 연결할 수 있습니다.

자체 관리형 Active Directory 도메인에서 DB 인스턴스 관리

콘솔, AWS CLI 또는 Amazon RDS API를 사용하여 DB 인스턴스 및 DB 인스턴스와 자체 관리형 AD 도메인과의 관계를 관리할 수 있습니다. 예를 들어, DB 인스턴스를 도메인 내로, 도메인 외부로 또는 도메인 간에 이동할 수 있습니다.

예를 들어 Amazon RDS API를 사용하여 다음을 수행할 수 있습니다.

- 실패한 멤버십에 대해 자체 관리형 도메인 가입을 다시 시도하려면 ModifyDBInstance API 작업을 사용하고 동일한 파라미터 세트를 지정하세요.
  - `--domain-fqdn`
  - `--domain-dns-ips`
  - `--domain-ou`
  - `--domain-auth-secret-arn`
- 자체 관리형 도메인에서 DB 인스턴스를 제거하려면 `ModifyDBInstance` API 작업을 사용하고 `--disable-domain`을 도메인 파라미터로 지정합니다.
- DB 인스턴스를 한 자체 관리형 도메인에서 다른 도메인으로 이동하려면 `ModifyDBInstance` API 작업을 사용하고 새 도메인에 대한 도메인 파라미터를 지정합니다.
  - `--domain-fqdn`
  - `--domain-dns-ips`
  - `--domain-ou`
  - `--domain-auth-secret-arn`
- 각 DB 인스턴스에 대한 자체 관리형 AD 도메인 멤버십을 나열하려면 DescribeDBInstances API 작업을 사용합니다.

자체 관리형 Active Directory 도메인 멤버십에 대한 이해

DB 인스턴스를 생성하거나 수정한 경우 해당 인스턴스는 자체 관리형 AD 도메인의 구성원이 됩니다. AWS 콘솔은 DB 인스턴스에 대한 자체 관리형 Active Directory 도메인 멤버십의 상태를 나타냅니다. DB 인스턴스의 상태는 다음 중 한 가지가 될 수 있습니다.

- joined – 인스턴스가 AD 도메인의 구성원입니다.
- joining – 인스턴스가 AD 도메인 구성원이 되기 위한 과정을 진행하고 있습니다.
- pending-join – 인스턴스 멤버십이 보류 중입니다.
- pending-maintenance-join - AWS에서 다음 예약된 유지 관리 기간 동안 인스턴스를 AD 도메인의 구성원으로 만들려고 시도합니다.
- pending-removal – AD 도메인에서 인스턴스 제거 작업이 보류 중입니다.
- pending-maintenance-removal - AWS에서 다음 예약된 유지 관리 기간 동안 AD 도메인에서 인스턴스를 제거하려고 시도합니다.
- failed – 구성 문제가 발생하여 인스턴스가 AD 도메인에 가입되지 않았습니다. 인스턴스 수정 명령을 다시 실행하기 전에 구성을 확인하고 수정합니다.
- removing – 인스턴스를 자체 관리형 AD 도메인에서 제거하고 있습니다.

네트워크 연결 문제로 인해 자체 관리형 AD 도메인 구성원 되기 요청이 실패할 수 있습니다. 예를 들어 DB 인스턴스를 생성하거나 기존 인스턴스를 수정하여 DB 인스턴스가 자체 관리형 AD 도메인의 구성원이 되려는 시도를 못하게 할 수 있습니다. 이 경우 명령을 다시 실행하여 DB 인스턴스를 생성 또는 수정하거나 새로 생성된 인스턴스를 수정하여 자체 관리형 AD 도메인에 가입할 수 있습니다.

자체 관리형 Active Directory 문제 해결

다음은 자체 관리형 AD를 설정하거나 수정할 때 발생할 수 있는 문제입니다.

오류 코드	설명	일반적인 원인	문제 해결 제안
오류 2/0x2	시스템이 지정된 파일을 찾을 수 없습니다.	`–domain-ou` 파라미터로 지정된 조직 단위(OU)의 형식 또는 위치가 잘못되었습니다. AWS Secrets Manager를 통해 지정된 도메인 서비스 계정에는 OU에 가입하는 데 필요한 권한이 없습니다.	`–domain-ou` 파라미터를 검토합니다. 도메인 서비스 계정에 OU에 대한 올바른 권한이 있는지 확인합니다. 자세한 내용은 AD 도메인 서비스 계정 구성 단원을 참조하십시오.
오류 5/0x5	액세스가 거부되었습니다.	도메인 서비스 계정에 대한 권한이 잘못 구성되었거나 컴퓨터 계정이 이미 도메인에 있습니다.	도메인의 도메인 서비스 계정 권한을 검토하고 RDS 컴퓨터 계정이 도메인에 중복되지 않았는지 확인합니

오류 코드	설명	일반적인 원인	문제 해결 제안
			다. RDS for SQL Server DB 인스턴스에서 `SELECT @@SERVERNAME` 을 실행하여 RDS 컴퓨터 계정의 이름을 확인할 수 있습니다. 다중 AZ를 사용하는 경우 장애 조치를 사용하여 재부팅한 다음 RDS 컴퓨터 계정을 다시 확인합니다. 자세한 내용은 DB 인스턴스 재부팅 단원을 참조하십시오.
오류 87/0x57	파라미터가 올바르지 않습니다.	AWS Secrets Manager를 통해 지정된 도메인 서비스 계정에 올바른 권한이 없습니다. 사용자 프로필도 손상되었을 수 있습니다.	도메인 서비스 계정의 요구 사항을 검토합니다. 자세한 내용은 AD 도메인 서비스 계정 구성 단원을 참조하십시오.
오류 234/0xEA	지정된 조직 단위(OU)가 없습니다.	`–domain-ou` 파라미터로 지정된 OU가 자체 관리형 AD에 존재하지 않습니다.	`–domain-ou` 파라미터를 검토하고 지정된 OU가 자체 관리형 AD에 있는지 확인합니다.
오류 1326/0x52E	사용자 이름 또는 암호가 잘못되었습니다.	AWS Secrets Manager에 제공된 도메인 서비스 계정 보안 인증 정보에 알 수 없는 사용자 이름이나 잘못된 암호가 있습니다. 자체 관리형 AD에서 도메인 계정을 사용하지 않도록 설정할 수도 있습니다.	AWS Secrets Manager에 제공된 보안 인증 정보가 올바르고 자체 관리형 Active Directory에서 도메인 계정이 활성화되어 있는지 확인합니다.

오류 코드	설명	일반적인 원인	문제 해결 제안
오류 1355/0x54B	지정된 도메인이 존재하지 않거나 해당 주소를 찾을 수 없습니다.	도메인이 중지되었거나, 지정된 DNS IP 집합에 연결할 수 없거나, 지정된 FQDN에 연결할 수 없습니다.	`–domain-dns-ips` 및 `–domain-fqdn` 파라미터를 검토하여 올바른지 확인합니다. RDS for SQL Server DB 인스턴스의 네트워킹 구성을 검토하고 자체 관리형 AD에 연결할 수 있는지 확인합니다. 자세한 내용은 네트워크 연결 구성 단원을 참조하십시오.
오류 1722/0x6BA	RPC 서버를 사용할 수 없습니다.	AD 도메인의 RPC 서비스에 연결하는 중 문제가 발생했습니다. 서비스 또는 네트워크 문제일 수 있습니다.	RPC 서비스가 도메인 컨트롤러에서 실행되고 있고 TCP 포트 135 및 `49152-65535` 에서 RDS for SQL Server DB 인스턴스의 도메인에 연결할 수 있는지 확인합니다.
오류 2224/0x8B0	계정이 이미 있습니다.	자체 관리형 AD에 추가하려는 컴퓨터 계정이 이미 있습니다.	RDS for SQL Server DB 인스턴스에서 `SELECT @@SERVERNAME` 을 실행하여 컴퓨터 계정을 식별한 다음 자체 관리형 AD에서 해당 계정을 신중히 제거합니다.
오류 2242/0x8c2	이 사용자의 암호가 만료되었습니다.	AWS Secrets Manager를 통해 지정한 도메인 서비스 계정의 암호가 만료되었습니다.	RDS for SQL Server DB 인스턴스를 자체 관리형 AD에 가입시키는 데 사용되는 도메인 서비스 계정의 암호를 업데이트합니다.

SQL Server DB 인스턴스를 복원한 후 자체 관리형 Active Directory 도메인에 추가

DB 스냅샷을 복원하거나 SQL Server DB 인스턴스에 대한 특정 시점 복구(PITR)를 수행한 후 자체 관리형 Active Directory 도메인에 추가할 수 있습니다. DB 인스턴스가 복원된 후 6단계: SQL Server DB 인스턴스 생성 또는 수정에 설명된 프로세스를 사용하여 DB 인스턴스를 자체 관리형 AD 도메인에 추가하도록 인스턴스를 수정합니다.

RDS for SQL Server를 사용하여 AWS 관리형 Active Directory 작업

사용자가 RDS for SQL Server DB 인스턴스에 연결할 때 AWS Managed Microsoft AD를 통해 Windows 인증으로 사용자를 인증할 수 있습니다. DB 인스턴스는 AWS Directory Service for Microsoft Active Directory라고도 하는 AWS Managed Microsoft AD과 함께 작동하여 Windows 인증을 활성화합니다. 사용자가 신뢰할 수 있는 도메인에 가입된 SQL Server DB 인스턴스를 사용하여 인증할 경우 AWS Directory Service를 사용하여 만든 도메인 디렉터리에 인증 요청이 전달됩니다.

리전 및 버전 사용 가능 여부

Amazon RDS는 AWS Managed Microsoft AD를 Windows 인증에만 사용할 수 있도록 지원합니다. RDS는 AD Connector 사용을 지원하지 않습니다. 자세한 내용은 다음 자료를 참조하세요.

- AWS Managed Microsoft AD의 애플리케이션 호환성 정책
- AD Connector의 애플리케이션 호환성 정책

버전 및 리전 가용성에 관한 자세한 내용은 Kerberos 인증을 사용하는 RDS for SQL Server를 참조하세요.

Windows 인증 설정 개요

Amazon RDS는 Windows 인증에 혼합 모드를 사용합니다. 이 접근 방식에서 마스터 사용자(SQL Server DB 인터페이스를 생성하는 데 사용된 이름과 암호)는 SQL 인증을 사용합니다. 마스터 사용자 계정은 권한 있는 자격 증명이므로 이 계정에 대한 액세스를 제한해야 합니다.

온프레미스 또는 자체 호스팅된 Microsoft Active Directory를 사용하여 Windows 인증을 얻으려면 포리스트 신뢰를 생성합니다. 단방향 또는 양방향 트러스트가 가능합니다. AWS Directory Service를 사용하여 포리스트 신뢰를 설정하는 방법에 대한 자세한 내용은 AWS Directory Service 관리 안내서의 신뢰 관계를 생성해야 하는 경우를 참조하십시오.

SQL Server DB 인스턴스에 대한 Windows 인증을 설정하려면 SQL Server DB 인스턴스에 대한 Windows 인증 설정에 자세히 설명된 다음 단계를 수행합니다.

1. AWS Managed Microsoft AD 또는 AWS Management Console API에서 AWS Directory Service를 사용하여 AWS Managed Microsoft AD 디렉터리를 생성합니다.
2. AWS CLI 또는 Amazon RDS API를 사용하여 SQL Server DB 인스턴스를 생성하는 경우 AWS Identity and Access Management(IAM) 역할을 생성합니다. 이 역할은 관리형 IAM 정책 `AmazonRDSDirectoryServiceAccess`를 사용하며 이 역할을 통해 Amazon RDS에서 디렉터리

를 호출할 수 있습니다. 콘솔을 사용하여 SQL Server DB 인스턴스를 생성하는 경우 AWS에서 IAM 역할을 자동으로 생성합니다.

역할이 액세스를 허용하려면 AWS Security Token Service 리전에서 AWS STS 계정의 AWS(AWS) 엔드포인트를 활성화해야 합니다. AWS STS 엔드포인트는 기본적으로 모든 AWS 리전에서 활성화되어 있으므로 더 이상의 조치 없이 사용할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 리전에서 AWS STS 관리를 참조하세요.

3. Microsoft Active Directory 도구를 사용하여 AWS Managed Microsoft AD 디렉터리에서 사용자 및 그룹을 만들고 구성합니다. Active Directory에서 사용자 및 그룹을 생성하는 방법에 대한 자세한 내용은 AWS Directory Service 관리 안내서의 에서 AWS Managed Microsoft AD에서 사용자 및 그룹 관리를 참조하세요.
4. 디렉터리와 DB 인스턴스를 다른 VPC에 배치하려면 VPC 간 트래픽을 활성화하십시오.
5. Amazon RDS를 사용하여 콘솔, AWS CLI 또는 Amazon RDS API에서 새 SQL Server DB 인스턴스를 생성합니다. 생성 요청에서 디렉터리를 만들 때 생성된 도메인 식별자("d-*" 식별자)와 생성된 역할의 이름을 제공합니다. DB 인스턴스에 대한 도메인 및 IAM 역할 파라미터를 설정하여 Windows 인증을 사용하도록 기존 SQL Server DB 인스턴스를 수정할 수도 있습니다.
6. 다른 DB 인스턴스의 경우와 같은 방법으로 Amazon RDS 마스터 사용자 자격 증명을 사용하여 SQL Server DB 인스턴스에 연결합니다. DB 인스턴스는 AWS Managed Microsoft AD 도메인에 조인되므로 도메인에 있는 Active Directory 사용자 및 그룹의 SQL Server 로그인과 사용자를 프로비저닝할 수 있습니다. (이를 SQL Server "Windows" 로그인이라고 합니다.) 데이터베이스 권한은 이러한 Windows 로그인에 부여되거나 취소되는 표준 SQL Server 권한을 통해 관리됩니다.

Amazon RDS 콘솔을 사용하여 도메인 연결 RDS for SQL Server DB 인스턴스를 생성하면 AWS가 `rds-directoryservice-access-role` IAM 역할을 자동으로 생성합니다. 이 역할은 도메인 연결 인스턴스를 관리하는 데 필수적이며 다음 작업에 필요합니다.

- 도메인 연결 SQL Server 인스턴스에 대한 구성 변경
- Active Directory 통합 설정 관리
- 도메인 조인 인스턴스에 대한 유지 관리 작업 수행

> **Important**
>
> `rds-directoryservice-access-role` IAM 역할을 삭제하면 Amazon RDS 콘솔 또는 API를 통해 도메인 연결 SQL Server 인스턴스를 변경할 수 없습니다. 이 인스턴스를 수정하려

고 하면 iam:CreateRole에 대한 권한이 없다는 오류 메시지가 표시됩니다. 액세스를 요청하려면 다음 텍스트를 복사하여 AWS 관리자에게 전송합니다.
이 오류는 Amazon RDS가 도메인 연결을 관리하기 위해 역할을 다시 생성해야 하지만 필요한 권한이 없기 때문에 발생합니다. 또한 이 오류는 CloudTrail에 로깅되지 않으므로 문제 해결이 더 어려울 수 있습니다.

실수로 `rds-directoryservice-access-role`을 삭제한 경우 다시 생성할 수 있는 `iam:CreateRole` 권한이 있어야 도메인 연결 SQL Server 인스턴스를 변경할 수 있습니다. 역할을 수동으로 다시 생성하려면 `AmazonRDSDirectoryServiceAccess` 관리형 정책이 연결되어 있고 RDS 서비스가 해당 역할을 수임하도록 허용하는 적절한 신뢰 관계가 있는지 확인합니다.

Kerberos 인증에 대한 엔드포인트 만들기

Kerberos 기반 인증을 사용하려면 엔드포인트가 고객이 지정한 호스트 이름과 마침표(.) 및 정규화된 도메인 이름(FQDN) 순으로 구성되어야 합니다. 예를 들어, 다음은 Kerberos 기반 인증과 함께 사용할 수 있는 엔드포인트의 예입니다. 이 예에서 SQL Server DB 인스턴스 호스트 이름은 `ad-test`이고 도메인 이름은 `corp-ad.company.com`입니다.

```
ad-test.corp-ad.company.com
```

현재 연결에서 Kerberos를 사용 중인지 확인하려면 다음 쿼리를 실행합니다.

```
SELECT net_transport, auth_scheme
  FROM sys.dm_exec_connections
 WHERE session_id = @@SPID;
```

SQL Server DB 인스턴스에 대한 Windows 인증 설정

AWS Directory Service for Microsoft Active Directory라고도 하는 AWS Managed Microsoft AD를 사용하여 SQL Server DB 인스턴스의 Windows 인증을 설정합니다. Windows 인증을 설정하려면 다음 단계를 수행합니다.

1단계: AWS Directory Service for Microsoft Active Directory를 사용하여 디렉터리 만들기

AWS Directory Service는 AWS 클라우드에 완전관리형 Microsoft Active Directory를 만듭니다. AWS Managed Microsoft AD 디렉터리를 생성할 때 AWS Directory Service에서 두 개의 도메인 컨트롤러 및 Domain Name System(DNS) 서버가 자동으로 생성됩니다. 디렉터리 서버는 VPC 내 가용 영역 두 곳

에 있는 두 개의 서브넷에서 생성됩니다. 이러한 중복은 장애가 발생해도 디렉터리에 액세스할 수 있도록 보장하는 데 도움이 됩니다.

AWS Managed Microsoft AD 디렉터리를 생성하는 경우 AWS Directory Service에서 다음 작업이 자동으로 수행됩니다.

- VPC 내에서 Microsoft Active Directory를 설정합니다.
- 사용자 이름 Admin과 지정된 암호를 사용하여 디렉터리 관리자 계정을 생성합니다. 이 계정을 사용하여 디렉터리를 관리할 수 있습니다.
- 디렉터리 컨트롤러에 대한 보안 그룹을 만듭니다.

AWS Directory Service for Microsoft Active Directory를 시작하면 AWS에서 모든 디렉터리의 객체를 포함하는 OU(조직 단위)를 생성합니다. 디렉터리를 만들 때 입력한 NetBIOS 이름을 가진 이 OU는 도메인 루트에 있습니다. 도메인 루트는 AWS에서 소유하고 관리합니다.

AWS Managed Microsoft AD 디렉터리를 사용하여 만든 admin 계정은 OU의 가장 일반적인 관리 활동에 대한 권한을 갖습니다.

- 사용자, 그룹 및 컴퓨터를 생성, 업데이트 또는 삭제합니다.
- 도메인(예: 파일 또는 인쇄 서버)에 리소스를 추가한 다음 OU 내의 사용자 및 그룹에 해당 리소스에 대한 권한 할당.
- 추가 OU 및 컨테이너 생성.
- 권한 위임.
- 그룹 정책 생성 및 연결.
- Active Directory 휴지통에서 삭제된 객체 복원.
- Active Directory 웹 서비스에서 AD 및 DNS Windows PowerShell 모듈 실행.

또한 admin 계정은 다음 도메인 차원 활동을 수행할 권한이 있습니다.

- DNS 구성 관리(레코드, 영역 및 전달자 추가, 제거 또는 업데이트)
- DNS 이벤트 로그 보기
- 보안 이벤트 로그 보기

AWS Managed Microsoft AD으로 디렉터리를 생성하려면

1. AWS Directory Service 콘솔 탐색 창에서 디렉터리를 선택한 후 디렉터리 설정을 선택합니다.

2. 를 선택합니다..AWS Managed Microsoft AD 이것은 현재 Amazon RDS에서 사용하도록 지원되는 유일한 옵션입니다.
3. 다음을 선택합니다.
4. 디렉터리 정보 입력 페이지에서 다음 정보를 제공합니다.

   Edition

   요구 사항에 맞는 에디션을 선택합니다.

   디렉터리 DNS 이름

   디렉터리를 위한 정규화된 이름(예: `corp.example.com`)입니다. 47자를 초과하는 이름은 SQL Server에서 지원되지 않습니다.

   디렉터리 NetBIOS 이름

   디렉터리의 선택적 짧은 이름(예: `CORP`)입니다.

   디렉터리 설명

   디렉터리에 대한 선택적 설명을 입력합니다.

   관리자 암호

   디렉터리 관리자의 암호입니다. 디렉터리 생성 프로세스에서는 사용자 이름 Admin과 이 암호를 사용하여 관리자 계정을 생성합니다.

   디렉터리 관리자 암호는 `admin`이라는 단어를 포함할 수 없습니다. 암호는 대소문자를 구분하며 길이가 8~64자 사이여야 합니다. 또한 다음 네 범주 중 세 개에 해당하는 문자를 1자 이상 포함해야 합니다.

   - 소문자(a-z)
   - 대문자(A-Z)
   - 숫자(0-9)
   - 영숫자 외의 특수 문자(~!@#$%^&*_-+=`|\(){}[]:;'"<>,.?/)

   [Confirm Password]

   관리자 암호를 다시 입력합니다.
5. 다음을 선택합니다.
6. Choose VPC and subnets(VPC 및 서브넷 선택) 페이지에 다음 정보를 입력합니다.

VPC

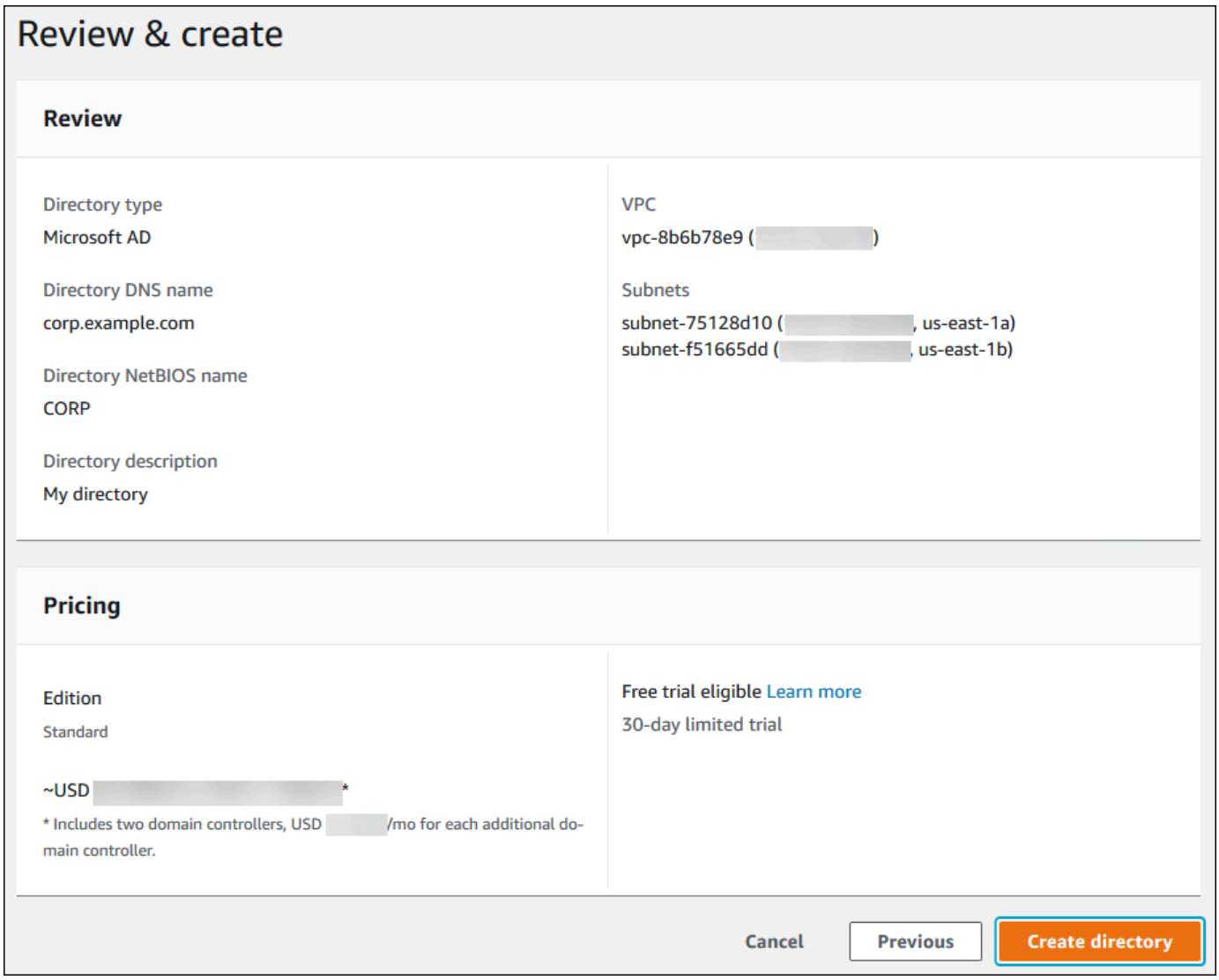
디렉터리에 대한 VPC를 선택합니다.

> **Note**
>
> 디렉터리와 DB 인스턴스를 서로 다른 VPC에서 찾을 수 있지만, 그렇게 할 경우 교차 VPC 트래픽을 활성화해야 합니다. 자세한 내용은 4단계: 디렉터리와 DB 인스턴스 사이에 VPC 간 트래픽 활성화 섹션을 참조하세요.

서브넷

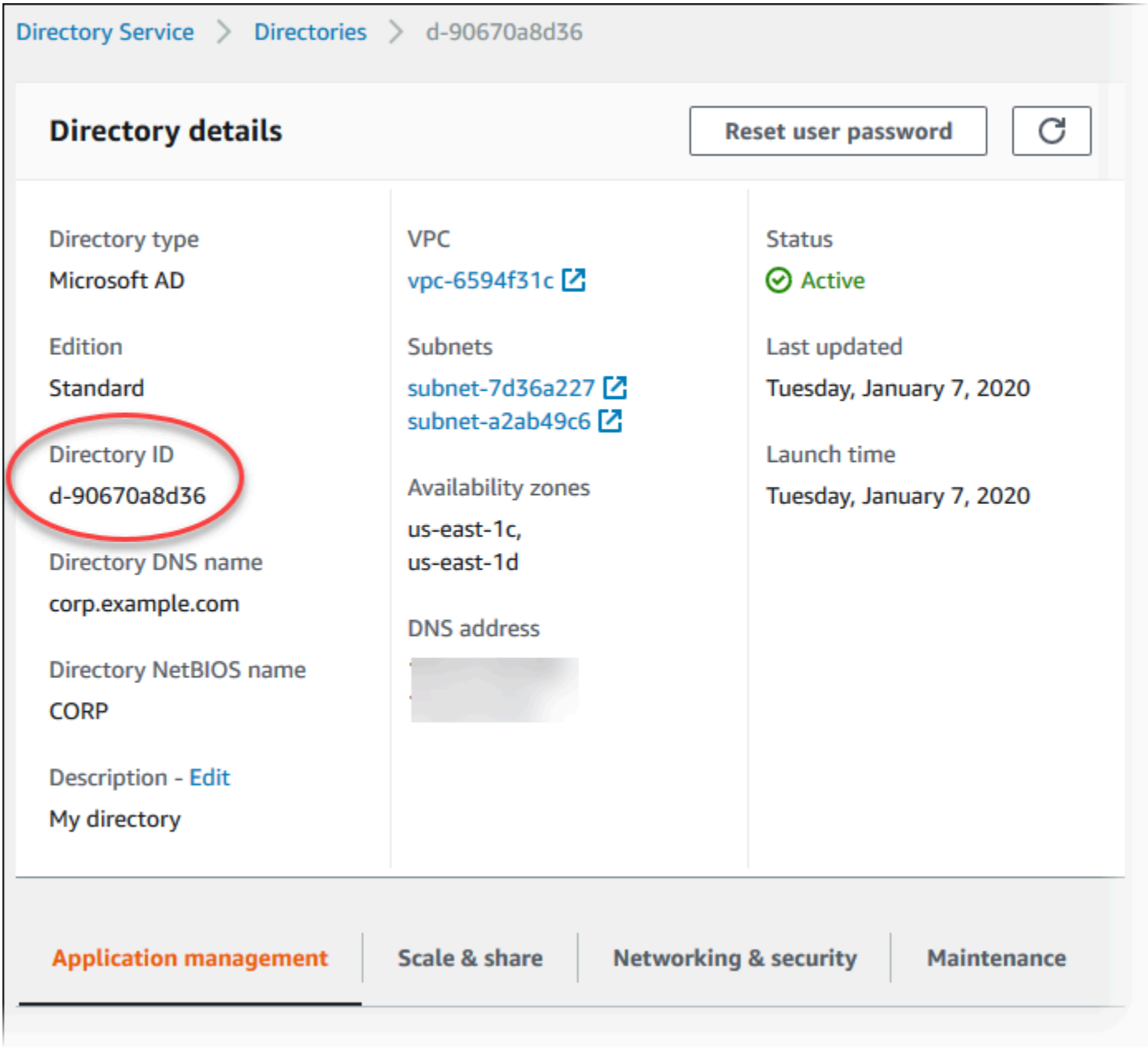
디렉터리 서버에 대한 서브넷을 선택합니다. 두 서브넷이 서로 다른 가용 영역에 있어야 합니다.

7. Next(다음)를 선택합니다.
8. 디렉터리 정보를 검토합니다. 변경이 필요하면 이전을 선택합니다. 정보가 올바르면 Create directory(디렉터리 생성)을 선택합니다.



디렉터리를 생성하는 데 몇 분 정도 걸립니다. 디렉터리가 성공적으로 생성되면 상태 값이 활성으로 변경됩니다.

디렉터리에 대한 정보를 보려면 디렉터리 목록에서 해당 디렉터리 ID를 선택합니다. 디렉터리 ID를 적어 두십시오. SQL Server DB 인스턴스를 생성하거나 수정할 때 이 값이 필요합니다.



2단계: Amazon RDS에 사용할 IAM 역할 생성

콘솔을 사용하여 SQL Server DB 인스턴스를 생성하는 경우 이 단계를 건너뛸 수 있습니다. CLI 또는 RDS API를 사용하여 SQL Server DB 인스턴스를 생성하는 경우 `AmazonRDSDirectoryServiceAccess` 관리형 IAM 정책을 사용하는 IAM 역할을 생성해야 합니다. 이 역할을 사용하여 Amazon RDS에서 AWS Directory Service를 자동으로 호출할 수 있습니다.

AWS 관리형 `AmazonRDSDirectoryServiceAccess` 정책을 사용하는 대신 도메인 가입에 사용자 지정 정책을 사용하는 경우 `ds:GetAuthorizedApplicationDetails` 작업을 허용해야 합니다. 이 요구 사항은 AWS Directory Service API 변경으로 인해 2019년 7월부터 유효합니다.

다음 IAM 정책 AmazonRDSDirectoryServiceAccess는 AWS Directory Service에 대한 액세스를 제공합니다.

Example AWS Directory Service에 대한 액세스 제공을 위한 IAM 정책

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
            "ds:DescribeDirectories",
            "ds:AuthorizeApplication",
            "ds:UnauthorizeApplication",
            "ds:GetAuthorizedApplicationDetails"
        ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

서비스 권한을 특정 리소스로 제한하는 리소스 기반 신뢰 관계의 aws:SourceArn 및 aws:SourceAccount 전역 조건 컨텍스트 키를 사용하는 것이 좋습니다. 이는 혼동된 대리자 문제를 방지하는 가장 효과적인 방법입니다.

전역 조건 컨텍스트 키를 모두 사용하고 aws:SourceArn 값에 계정 ID가 포함되도록 할 수 있습니다. 이 경우 aws:SourceAccount 값과 aws:SourceArn 값의 계정이 동일한 문에서 사용될 때 동일한 계정 ID를 사용해야 합니다.

- 단일 리소스에 대한 교차 서비스 액세스를 원하는 경우 aws:SourceArn을 사용하세요.
- 해당 계정의 모든 리소스가 교차 서비스 사용과 연결되도록 허용하려는 경우 aws:SourceAccount를 사용하세요.

신뢰 정책에서는 역할에 액세스하는 리소스의 전체 Amazon 리소스 이름(ARN)이 포함된 aws:SourceArn 전역 조건 컨텍스트 키를 사용해야 합니다. Windows 인증의 경우 다음 예와 같이 DB 인스턴스를 포함해야 합니다.

Example Windows 인증을 위한 전역 조건 컨텍스트 키와의 신뢰 관계

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rds.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceArn": [

 "arn:aws:rds:Region:my_account_ID:db:db_instance_identifier"
                    ]
                }
            }
        }
    ]
}
```

이 IAM 정책 및 신뢰 관계를 사용하여 IAM 역할을 생성합니다. IAM 역할 생성에 대한 자세한 내용은 IAM 사용 설명서의 고객 관리형 정책 생성을 참조하세요.

3단계: 사용자 및 그룹 생성 및 구성

Active Directory 사용자 및 컴퓨터 도구를 사용하여 사용자 및 그룹을 생성할 수 있습니다. 이 도구는 Active Directory 도메인 서비스 및 Active Directory Lightweight Directory Services 도구 중 하나입니다. 사용자는 디렉터리에 액세스할 수 있는 개별 사용자 또는 개체를 나타냅니다. 그룹은 개별 사용자에게 권한을 적용할 필요 없이 사용자 그룹에 권한을 부여하거나 거부하는 데 매우 유용합니다.

AWS Directory Service 디렉터리에 사용자 및 그룹을 생성하려면 AWS Directory Service 디렉터리의 멤버인 Windows EC2 인스턴스에 연결해야 합니다. 또한 사용자 및 그룹을 생성할 수 있는 권한을 가진 사용자로 로그인해야 합니다. 자세한 내용은 AWS Directory Service관리 안내서에서 사용자 및 그룹 추가(Simple AD 및 AWS Managed Microsoft AD)를 참조하세요.

4단계: 디렉터리와 DB 인스턴스 사이에 VPC 간 트래픽 활성화

디렉터리와 DB 인스턴스를 동일한 VPC에 배치하려면 이 단계를 건너뛰고 5단계: SQL Server DB 인스턴스 생성 또는 수정 섹션으로 이동하세요.

디렉터리와 DB 인스턴스를 서로 다른 VPC에 배치하려면 VPC 피어링 또는 AWS Transit Gateway를 사용하여 VPC 간 트래픽을 구성하세요.

다음 절차는 VPC 피어링을 사용하여 VPC 간 트래픽을 활성화합니다. Amazon Virtual Private Cloud 피어링 안내서의 VPC 피어링이란? 지침을 따르십시오.

VPC 피어링을 사용하여 VPC 간 트래픽을 활성화하려면

1. 네트워크 트래픽이 양방향으로 흐를 수 있도록 적절한 VPC 라우팅 규칙을 설정합니다.
2. DB 인스턴스의 보안 그룹이 디렉터리의 보안 그룹에서 인바운드 트래픽을 수신할 수 있는지 확인합니다.
3. 트래픽을 차단하는 네트워크 ACL(액세스 제어 목록) 규칙이 없어야 합니다.

다른 AWS 계정이 디렉터리를 소유하는 경우 디렉터리를 공유해야 합니다.

AWS 계정 간에 디렉터리를 공유하려면

1. AWS 관리 안내서의 자습서: 원활한 EC2 도메인 조인을 위해 AWS Managed Microsoft AD 디렉터리 공유에 있는 지침에 따라 DB 인스턴스가 생성될 AWS Directory Service 계정과 디렉터리를 공유하는 작업을 시작합니다.
2. DB 인스턴스용 계정을 사용하여 AWS Directory Service 콘솔에 로그인하고 계속하기 전에 도메인이 SHARED 상태가 되었는지 확인합니다.
3. DB 인스턴스용 계정을 사용하여 AWS Directory Service 콘솔에 로그인하는 동안 디렉터리 ID 값을 기록해 둡니다. 이 디렉터리 ID를 사용하여 DB 인스턴스를 도메인에 조인합니다.

5단계: SQL Server DB 인스턴스 생성 또는 수정

디렉터리에서 사용할 SQL Server DB 인스턴스를 생성하거나 수정합니다. 콘솔, CLI 또는 RDS API를 사용하여 DB 인스턴스를 디렉터리에 연결할 수 있습니다. 이 작업을 다음 중 한 가지 방법으로 수행할 수 있습니다.

- 콘솔, create-db-instance CLI 명령 또는 CreateDBInstance RDS API 작업을 사용하여 새 SQL Server DB 인스턴스를 생성합니다.

지침은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.

- 콘솔, modify-db-instance CLI 명령 또는 ModifyDBInstance RDS API 작업을 사용하여 기존 SQL Server DB 인스턴스를 수정합니다.

  지침은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

- 콘솔, restore-db-instance-from-db-snapshot CLI 명령 또는 RestoreDBInstanceFromDBSnapshot RDS API 작업을 사용하여 DB 스냅샷에서 SQL Server DB 인스턴스를 복원합니다.

  지침은 DB 인스턴스 복원 섹션을 참조하세요.

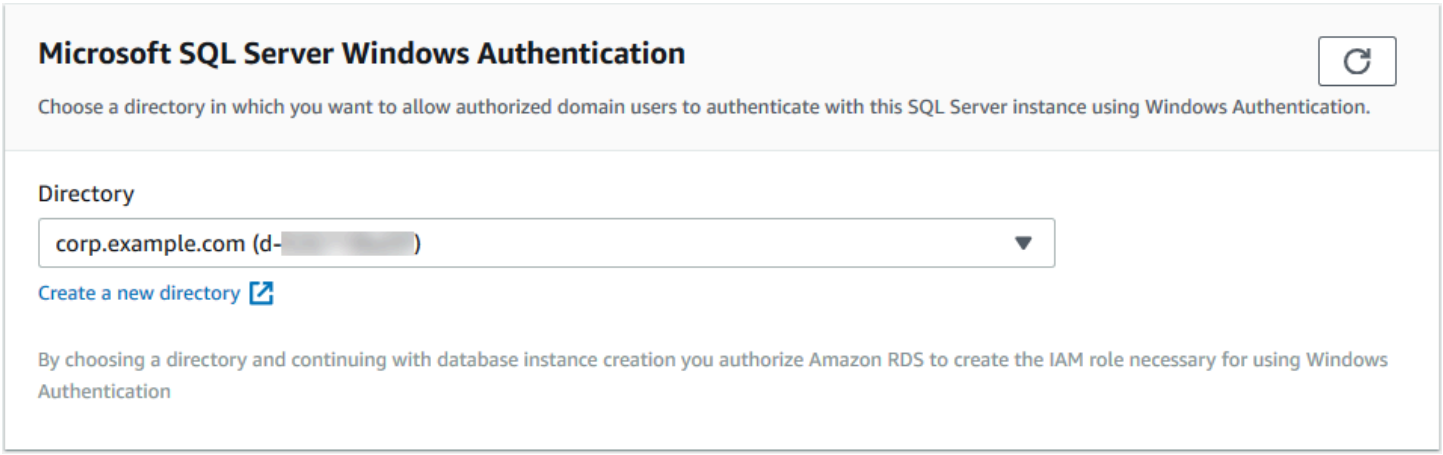
- 콘솔, restore-db-instance-to-point-in-time CLI 명령 또는 RestoreDBInstanceToPointInTime RDS API 작업을 사용하여 SQL Server DB 인스턴스를 특정 시점으로 복구합니다.

  지침은 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 섹션을 참조하세요.

Windows 인증은 VPC의 SQL Server DB 인스턴스에 대해서만 지원됩니다.

DB 인스턴스에서 생성한 도메인 디렉터리를 사용할 수 있으려면 다음이 필요합니다.

- 디렉터리의 경우 디렉터리를 만들 때 생성된 도메인 식별자(d-*ID*)를 선택해야 합니다.
- VPC 보안 그룹에 DB 인스턴스가 디렉터리와 통신할 수 있도록 하는 아웃바운드 규칙이 있는지 확인합니다.



AWS CLI를 사용하는 경우 생성한 도메인 디렉터리를 DB 인스턴스에서 사용하려면 다음과 같은 파라미터가 필요합니다.

- `--domain` 파라미터의 경우 디렉터리를 만들 때 생성된 도메인 식별자(d-*ID*)를 사용합니다.

- `--domain-iam-role-name` 파라미터의 경우 관리형 IAM 정책 `AmazonRDSDirectoryServiceAccess`를 사용하여 생성한 역할을 사용합니다.

예를 들어 다음 CLI 명령에서는 디렉터리를 사용하도록 DB 인스턴스를 수정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --domain d-ID \
    --domain-iam-role-name role-name
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --domain d-ID ^
    --domain-iam-role-name role-name
```

> **Important**
>
> DB 인스턴스를 수정하여 Kerberos 인증을 활성화하는 경우에는 변경 후 DB 인스턴스를 재부팅하세요.

6단계: Windows 인증 SQL Server 로그인 생성

다른 DB 인스턴스의 경우와 같은 방법으로 Amazon RDS 마스터 사용자 자격 증명을 사용하여 SQL Server DB 인스턴스에 연결합니다. DB 인스턴스는 AWS Managed Microsoft AD 도메인에 조인되므로 SQL Server 로그인 및 사용자를 프로비저닝할 수 있습니다. 도메인의 Active Directory 사용자 및 그룹에서 이 작업을 수행합니다. 데이터베이스 권한은 이러한 Windows 로그인에 부여되거나 취소되는 표준 SQL Server 권한을 통해 관리됩니다.

Active Directory 사용자가 SQL Server로 인증하려면 사용자 또는 사용자가 속한 그룹에 대한 SQL Server Windows 로그인이 있어야 합니다. 세분화된 액세스 제어는 이러한 SQL Server 로그인에 대한 권한을 부여하거나 취소하여 처리합니다. SQL Server 로그인이 없거나 이러한 로그인이 있는 그룹에 속하지 않은 사용자는 SQL Server DB 인스턴스에 액세스할 수 없습니다.

Active Directory SQL Server 로그인을 생성하려면 ALTER ANY LOGIN 권한이 필요합니다. 이 권한을 가진 로그인을 아직 생성하지 않은 경우 SQL Server 인증을 사용하여 DB 인스턴스의 마스터 사용자로 연결합니다.

다음 예와 같은 DDL(데이터 정의 언어) 명령을 실행하여 Active Directory 사용자 또는 그룹에 대한 SQL Server 로그인을 생성합니다.

> **Note**
>
> Windows 2000 이전 로그인 이름을 사용하여 사용자 및 그룹을 *domainName**login_name* 형식으로 지정합니다. UPN(User Principle Name)을 *login_name*@*DomainName* 형식으로 사용할 수 없습니다.
> T-SQL 문을 사용해야만 RDS for SQL Server 인스턴스에서 Windows 인증 로그인을 만들 수 있습니다. SQL Server 관리 스튜디오를 사용하여 Windows 인증 로그인을 만들 수 없습니다.

```
USE [master]
GO
CREATE LOGIN [mydomain\myuser] FROM WINDOWS WITH DEFAULT_DATABASE = [master],
 DEFAULT_LANGUAGE = [us_english];
GO
```

자세한 내용은 Microsoft Developer Network 설명서에서 CREATE LOGIN(Transact-SQL)을 참조하세요.

도메인의 사용자(사람 및 애플리케이션)는 이제 Windows 인증을 사용하여 도메인이 조인된 클라이언트 컴퓨터의 RDS for SQL Server 인스턴스에 연결할 수 있습니다.

도메인에서 DB 인스턴스 관리

콘솔, AWS CLI 또는 Amazon RDS API를 사용하여 DB 인스턴스 및 DB 인스턴스와 도메인과의 관계를 관리할 수 있습니다. 예를 들어, DB 인스턴스를 도메인 내로, 도메인 외부로 또는 도메인 간에 이동할 수 있습니다.

예를 들어 Amazon RDS API를 사용하여 다음을 수행할 수 있습니다.

- 실패한 멤버십에 대한 도메인 조인을 다시 시도하려면 ModifyDBInstance API 작업을 사용하고 현재 멤버십의 디렉터리 ID를 지정합니다.
- 멤버십에 대한 IAM 역할 이름을 업데이트하려면 `ModifyDBInstance` API 작업을 사용하고 현재 멤버십의 디렉터리 ID 및 새 IAM 역할을 지정합니다.

- 도메인에서 DB 인스턴스를 제거하려면 `ModifyDBInstance` API 작업을 사용하고 `none`을 도메인 파라미터로 지정합니다.
- 한 도메인에서 다른 도메인으로 DB 인스턴스를 이동하려면 `ModifyDBInstance` API 작업을 사용하여 새 도메인의 도메인 식별자를 도메인 파라미터로 지정합니다.
- 각 DB 인스턴스에 대한 멤버십을 나열하려면 DescribeDBInstances API 작업을 사용합니다.

도메인 멤버십 이해

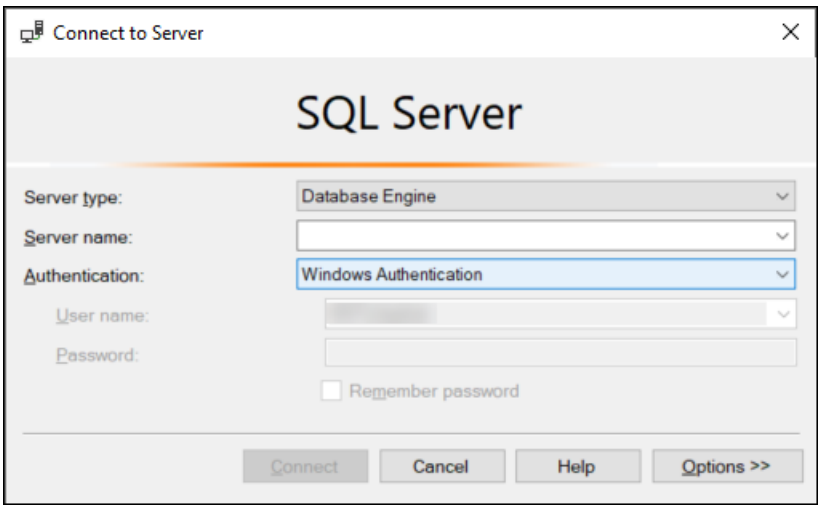
DB 인스턴스를 생성하거나 수정한 경우 해당 인스턴스는 도메인의 구성원이 됩니다. AWS 콘솔은 DB 인스턴스에 대한 도메인 멤버십의 상태를 나타냅니다. DB 인스턴스의 상태는 다음 중 한 가지가 될 수 있습니다.

- joined – 인스턴스가 도메인의 구성원입니다.
- joining – 인스턴스가 도메인 구성원이 되기 위한 과정을 진행하고 있습니다.
- pending-join – 인스턴스 멤버십이 보류 중입니다.
- pending-maintenance-join - AWS에서 다음 예약된 유지 관리 기간 동안 인스턴스를 도메인의 멤버로 만들려고 시도합니다.
- pending-removal – 도메인에서 인스턴스 제거 작업이 보류 중입니다.
- pending-maintenance-removal - AWS에서 다음 예약된 유지 관리 기간 동안 도메인에서 인스턴스를 제거하려고 시도합니다.
- failed – 구성 문제가 발생하여 인스턴스가 도메인에 조인되지 않았습니다. 인스턴스 수정 명령을 다시 실행하기 전에 구성을 확인하고 수정합니다.
- removing – 인스턴스를 도메인에서 제거하고 있습니다.

네트워크 연결 문제 또는 잘못된 IAM 역할로 인해 도메인 구성원 되기 요청이 실패할 수 있습니다. 예를 들어, DB 인스턴스를 생성하거나 기존 인스턴스를 수정하여 DB 인스턴스가 도메인의 멤버가 되려는 시도를 못하게 할 수 있습니다. 이 경우 명령을 다시 실행하여 DB 인스턴스를 생성 또는 수정하거나 새로 생성된 인스턴스를 수정하여 도메인에 조인할 수 있습니다.

Windows 인증을 사용하여 SQL Server에 연결

Windows 인증을 사용하여 SQL Server에 연결하려면 도메인에 조인된 컴퓨터에 도메인 사용자로 로그인해야 합니다. 다음과 같이 SQL Server Management Studio를 시작한 후 Windows 인증을 인증 유형으로 선택합니다.



SQL Server DB 인스턴스를 복원한 후 도메인에 추가

DB 스냅샷을 복원하거나 SQL Server DB 인스턴스에 대한 특정 시점 복구(PITR)를 수행한 후 도메인에 추가할 수 있습니다. DB 인스턴스가 복원된 후 5단계: SQL Server DB 인스턴스 생성 또는 수정에 설명된 프로세스를 사용하여 DB 인스턴스를 도메인에 추가하도록 인스턴스를 수정합니다.

새 SSL/TLS 인증서를 사용해 Microsoft SQL Server DB 인스턴스에 연결할 애플리케이션을 업데이트

2023년 1월 13일부터 Amazon RDS는 보안 소켓 계층(SSL) 또는 전송 계층 보안(TLS)을 사용해 RDS DB 인스턴스에 연결하기 위한 용도의 새 인증 기관(CA) 인증서를 게시하였습니다. 아래에서 새 인증서를 사용하기 위해 애플리케이션을 업데이트하는 방법에 관한 정보를 찾으실 수 있습니다.

이 주제는 클라이언트 애플리케이션에서 SSL/TLS를 사용해 DB 인스턴스에 연결하는지 여부를 판단하는 데 도움이 됩니다. SSL/TLS를 사용해 연결한다면 이 애플리케이션에서 연결 시 인증서 확인이 필요한지 여부를 추가로 확인할 수 있습니다.

> **Note**
>
> 어떤 애플리케이션은 서버에서 인증서를 성공적으로 확인할 수 있는 경우에만 SQL Server DB 인스턴스에 연결하도록 구성되어 있습니다.
> 이러한 애플리케이션의 경우 클라이언트 애플리케이션 트러스트 스토어를 업데이트하여 새 CA 인증서를 포함해야 합니다.

클라이언트 애플리케이션 트러스트 스토어에서 CA 인증서를 업데이트한 후에는 DB 인스턴스에서 인증서를 교환할 수 있습니다. 이 절차를 프로덕션 환경에서 구현하기 전에 개발 또는 스테이징 환경에서 테스트해볼 것을 적극 권장합니다.

인증서 교환에 대한 자세한 내용은 SSL/TLS 인증서 교체 단원을 참조하십시오. 인증서 다운로드에 대한 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 단원을 참조하십시오. Microsoft SQL Server DB 인스턴스에서 SSL/TLS를 사용하는 방법에 관한 자세한 내용은 ??? 단원을 참조하십시오.

주제

- 애플리케이션에서 SSL을 사용해 Microsoft SQL Server DB 인스턴스에 연결하는지 여부 확인
- 클라이언트에서 연결을 위해 인증서 확인이 필요한지 여부 확인
- 애플리케이션 트러스트 스토어 업데이트

애플리케이션에서 SSL을 사용해 Microsoft SQL Server DB 인스턴스에 연결하는지 여부 확인

`rds.force_ssl` 파라미터의 값에 대한 DB 인스턴스 구성을 확인하십시오. 기본적으로 `rds.force_ssl` 파라미터는 0(해제)으로 설정됩니다. `rds.force_ssl` 파라미터가 1(켜짐)로 설정된 경우 클라이언트는 연결 시 SSL/TLS를 사용해야 합니다. 파라미터 그룹에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하세요.

다음 쿼리를 실행하여 DB 인스턴스에 대한 모든 열린 연결에 대한 현재 암호화 옵션을 가져옵니다. 연결이 암호화되면 `ENCRYPT_OPTION` 열이 `TRUE`를 반환합니다.

```
select SESSION_ID,
    ENCRYPT_OPTION,
    NET_TRANSPORT,
    AUTH_SCHEME
    from SYS.DM_EXEC_CONNECTIONS
```

이 쿼리는 현재 연결만 보여줍니다. 과거에 연결/연결 해제된 애플리케이션이 SSL을 사용했는지 여부는 표시되지 않습니다.

클라이언트에서 연결을 위해 인증서 확인이 필요한지 여부 확인

다양한 유형의 클라이언트에서 연결 시 인증서 확인이 필요한지 여부를 확인할 수 있습니다.

> **Note**
>
> 나열된 것 이외의 커넥터를 사용하는 경우 암호화된 연결을 적용하는 방법에 대한 정보는 특정 커넥터 설명서를 참조하십시오. 자세한 정보는 Microsoft SQL Server 설명서의 Microsoft SQL 데이터베이스용 연결 모듈을 참조하십시오.

SQL Server Management Studio

SQL Server Management Studio 연결에 암호화가 적용되는지 확인하십시오.

1. SQL Server Management Studio를 시작합니다.
2. 서버에 연결에 서버 정보, 로그인 사용자 이름 및 암호를 입력합니다.

3. 옵션(Options)을 선택합니다.
4. 연결 페이지에서 연결 암호화가 선택되어 있는지 확인하십시오.

SQL Server Management Studio에 대한 자세한 내용은 SQL Server Management Studio 사용을 참조하십시오.

sqlcmd

sqlcmd 클라이언트를 사용하는 다음 예제에서는 스크립트의 SQL Server 연결을 확인하여 성공적인 연결을 위해 유효한 인증서가 필요한지 여부를 판단하는 방법을 보여줍니다. 자세한 내용은 Microsoft SQL Server 설명서의 sqlcmd로 연결을 참조하십시오.

sqlcmd를 사용하는 경우 다음 예제와 같이 -N 명령 인수를 사용하여 연결을 암호화한다면 SSL 연결 시 서버 인증서에 대한 확인이 필요합니다.

```
$ sqlcmd -N -S dbinstance.rds.amazon.com -d ExampleDB
```

> **Note**
>
> sqlcmd 옵션을 사용하여 -C를 호출하면 클라이언트 측 트러스트 스토어와 일치하지 않더라도 서버 인증서를 신뢰합니다.

ADO.NET

다음 예에서 애플리케이션은 SSL을 사용하여 연결하고, 서버 인증서를 확인해야 합니다.

```
using SQLC = Microsoft.Data.SqlClient;

...

    static public void Main()
    {
        using (var connection = new SQLC.SqlConnection(
            "Server=tcp:dbinstance.rds.amazon.com;" +
            "Database=ExampleDB;User ID=LOGIN_NAME;" +
            "Password=YOUR_PASSWORD;" +
```

```
            "Encrypt=True;TrustServerCertificate=False;"
            ))
        {
            connection.Open();
            ...
        }
```

Java

다음 예에서 애플리케이션은 SSL을 사용하여 연결하고, 서버 인증서를 확인해야 합니다.

```
String connectionUrl =
    "jdbc:sqlserver://dbinstance.rds.amazon.com;" +
    "databaseName=ExampleDB;integratedSecurity=true;" +
    "encrypt=true;trustServerCertificate=false";
```

JDBC를 사용하여 연결하는 클라이언트에 대해 SSL 암호화를 활성화하려면 Java CA 인증서 저장소에 Amazon RDS 인증서를 추가해야 할 수도 있습니다. 지침은 Microsoft SQL Server 설명서의 암호화를 위한 클라이언트 구성을 참조하십시오. 연결 문자열에 `trustStore=`*`path-to-certificate-trust-store-file`*을 추가하여 신뢰할 수 있는 CA 인증서 파일 이름을 직접 제공할 수도 있습니다.

> **Note**
>
> 연결 문자열에 `TrustServerCertificate=true`(또는 이와 동등한)를 사용하면 연결 프로세스가 트러스트 체인 검증을 건너뜁니다. 이 경우 인증서를 검증할 수 없는 경우에도 애플리케이션이 연결됩니다. `TrustServerCertificate=false` 사용은 인증서 검증을 강제하며, 이것이 모범 사례입니다.

애플리케이션 트러스트 스토어 업데이트

Microsoft SQL Server를 사용하는 애플리케이션의 트러스트 스토어를 업데이트할 수 있습니다. 지침은 ??? 단원을 참조하십시오. 또한 Microsoft SQL Server 설명서의 암호화를 위한 클라이언트 구성을 참조하십시오.

Microsoft Windows 이외의 운영 체제를 사용하는 경우 새 루트 CA 인증서 추가에 대한 정보는 SSL/TLS 구현을 위한 소프트웨어 배포 설명서를 참조하십시오. 예를 들어 OpenSSL 및 GnuTLS가 널리 사

용되는 옵션입니다. 이 구현 방법을 사용하여 RDS 루트 CA 인증서에 신뢰를 추가하십시오. Microsoft는 일부 시스템에 대한 인증서 구성 지침을 제공합니다.

루트 인증서 다운로드에 대한 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 단원을 참조하십시오.

인증서를 가져오는 샘플 스크립트는 트러스트 스토어로 인증서를 가져오기 위한 샘플 스크립트 섹션을 참조하세요.

> **Note**
>
> 트러스트 스토어를 업데이트할 때 새 인증서를 추가할 뿐 아니라 이전 인증서를 유지할 수도 있습니다.

Microsoft SQL Server DB 엔진 업그레이드

Amazon RDS에서 새 데이터베이스 엔진 버전을 지원하는 경우, DB 인스턴스를 새 버전으로 업그레이드할 수 있습니다. SQL Server DB 인스턴스의 업그레이드에는 메이저 버전 업그레이드와 마이너 버전 업그레이드라는 두 가지 업그레이드가 있습니다.

메이저 버전 업그레이드에는 기존 애플리케이션과 호환되지 않는 데이터베이스 변경 사항이 포함될 수 있습니다. 따라서 DB 인스턴스의 메이저 버전 업그레이드를 수동으로 수행해야 합니다. DB 인스턴스를 수정하여 메이저 버전 업그레이드를 시작할 수 있습니다. 그러나 메이저 버전 업그레이드를 수행하기 전에 RDS for SQL Server 업그레이드 테스트에 설명된 단계에 따라 업그레이드를 테스트하는 것이 좋습니다.

마이너 버전 업그레이드에는 기존 애플리케이션과 호환되는 변경 사항만 포함됩니다. 다음 두 가지 방법으로 DB 인스턴스의 마이너 버전을 업그레이드할 수 있습니다.

- 수동 - 업그레이드를 시작하도록 DB 인스턴스 수정
- 자동 - DB 인스턴스에 대해 자동 마이너 버전 업그레이드 활성화

자동 마이너 버전 업그레이드를 활성화하면 RDS for SQL Server는 새로운 마이너 버전에서 중요한 보안 업데이트를 사용할 수 있을 때 예약된 유지 관리 기간 동안 데이터베이스 인스턴스를 자동으로 업그레이드합니다.

`16.00.4120.1`, `15.00.4365.2`, `14.00.3465.1`, `13.00.6435.1` 이후 마이너 엔진 버전의 경우 다음 보안 프로토콜이 기본적으로 비활성화됩니다.

- `rds.tls10`(TLS 1.0 프로토콜)
- `rds.tls11`(TLS 1.1 프로토콜)
- `rds.rc4`(RC4 암호)
- `rds.curve25519`(Curve25519 암호화)
- `rds.3des168`(트리플 DES 암호화)

이전 엔진 버전의 경우 Amazon RDS는 기본적으로 이러한 보안 프로토콜을 활성화합니다.

```
...

"ValidUpgradeTarget": [
```

```
    {
        "Engine": "sqlserver-se",
        "EngineVersion": "14.00.3281.6.v1",
        "Description": "SQL Server 2017 14.00.3281.6.v1",
        "AutoUpgrade": false,
        "IsMajorVersionUpgrade": false
    }
...
```

업그레이드 수행에 대한 자세한 내용은 SQL Server DB 인스턴스 업그레이드 단원을 참조하세요. Amazon RDS에서 사용할 수 있는 SQL Server 버전에 대한 자세한 내용은 Amazon RDS for Microsoft SQL Server 단원을 참조하십시오.

주제

- RDS for SQL Server의 메이저 버전 업그레이드
- SQL Server 업그레이드 고려 사항
- RDS for SQL Server 업그레이드 테스트
- SQL Server DB 인스턴스 업그레이드
- 지원 종료 전에 사용되지 않는 DB 인스턴스 업그레이드

RDS for SQL Server의 메이저 버전 업그레이드

Amazon RDS는 현재 Microsoft SQL Server DB 인스턴스에 대해 다음 메이저 버전의 업그레이드를 지원합니다.

SQL Server 2008을 제외한 어떤 버전에서든 SQL Server 2017 또는 2019로 기존 DB 인스턴스를 업그레이드할 수 있습니다. SQL Server 2008에서 업그레이드하려면 다음 버전 중 하나로 업그레이드하십시오.

현재 버전	지원하는 업그레이드 버전
SQL Server 2019	SQL Server 2022
SQL Server 2017	SQL Server 2022 SQL Server 2019
SQL Server 2016	SQL Server 2022

현재 버전	지원하는 업그레이드 버전
	SQL Server 2019 SQL Server 2017

다음 예와 같은 AWS CLI 쿼리를 사용하여 특정 데이터베이스 엔진 버전에 사용 가능한 업그레이드를 찾을 수 있습니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-engine-versions \
    --engine sqlserver-se \
    --engine-version 14.00.3281.6.v1 \
    --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" \
    --output table
```

Windows의 경우:

```
aws rds describe-db-engine-versions ^
    --engine sqlserver-se ^
    --engine-version 14.00.3281.6.v1 ^
    --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" ^
    --output table
```

출력은 버전 14.00.3281.6을 사용 가능한 최신 SQL Server 2017 또는 2019 버전으로 업그레이드할 수 있음을 보여줍니다.

```
--------------------------
|DescribeDBEngineVersions|
+------------------------+
|      EngineVersion     |
+------------------------+
|  14.00.3294.2.v1       |
|  14.00.3356.20.v1      |
|  14.00.3381.3.v1       |
|  14.00.3401.7.v1       |
|  14.00.3421.10.v1      |
```

```
|  14.00.3451.2.v1       |
|  15.00.4043.16.v1      |
|  15.00.4073.23.v1      |
|  15.00.4153.1.v1       |
|  15.00.4198.2.v1       |
|  15.00.4236.7.v1       |
+------------------------+
```

데이터베이스 호환성 수준

Microsoft SQL Server 데이터베이스 호환성 수준을 이용해 일부 데이터베이스 동작이 이전 버전의 SQL Server를 모방하도록 조정할 수 있습니다. 자세한 내용은 Microsoft 설명서의 호환성 수준을 참조하십시오. DB 인스턴스를 업그레이드할 때 기존의 모든 데이터베이스는 원래 호환성 수준으로 유지됩니다.

ALTER DATABASE 명령을 사용하여 데이터베이스의 호환성 수준을 변경할 수 있습니다. 예를 들어, `customeracct`라는 이름의 데이터베이스를 SQL Server 2016과 호환되도록 변경하려면 다음 명령을 실행합니다.

```
ALTER DATABASE customeracct SET COMPATIBILITY_LEVEL = 130
```

SQL Server 업그레이드 고려 사항

Amazon RDS는 업그레이드 프로세스 중에 DB 스냅샷을 2개 캡처합니다. 첫 번째 DB 스냅샷은 업그레이드 변경 이전 DB 인스턴스의 스냅샷입니다. 두 번째 DB 스냅샷은 업그레이드 완료 이후에 캡처됩니다.

> **Note**
>
> DB 인스턴스에 대한 백업 보존 기간을 0보다 큰 수로 설정하면 Amazon RDS는 DB 스냅샷만 캡처합니다. 백업 보존 기간을 변경하려면 Amazon RDS DB 인스턴스 수정 단원을 참조하세요.

업그레이드가 완료되고 나면 이전 버전의 데이터베이스 엔진으로 되돌릴 수 없습니다. 이때 이전 버전으로 되돌리려면 업그레이드 전에 캡처한 DB 스냅샷으로 복구하여 새로운 DB 인스턴스를 생성해야 합니다.

SQL Server의 마이너 버전 또는 메이저 버전 업그레이드 중에는 여유 스토리지 공간 및 디스크 대기열 깊이 측정치가 -1로 표시됩니다. 이후 업그레이드가 완료되면 두 측정치 모두 정상적으로 돌아옵니다.

SQL Server 인스턴스를 업그레이드하기 전에 다음 정보를 검토합니다.

주제

- 업그레이드를 시작하기 전 모범 사례
- 다중 AZ 고려 사항
- 읽기 전용 복제본 고려 사항
- 옵션 그룹 고려 사항
- 파라미터 그룹 고려 사항

업그레이드를 시작하기 전 모범 사례

업그레이드 프로세스를 시작하기 전에 다음과 같은 준비 단계를 구현하여 최적의 업그레이드 성능을 허용하고 잠재적 문제를 최소화합니다.

타이밍 및 워크로드 관리

- 트랜잭션 볼륨이 적은 기간 동안 업그레이드를 예약합니다.
- 업그레이드 기간 동안 쓰기 작업을 최소화합니다.

  이렇게 하면 Amazon RDS가 보조-기본 페어링 중에 RDS가 복원해야 하는 트랜잭션 로그 백업 파일의 수를 줄임으로써 업그레이드를 더 빠르게 완료할 수 있습니다.

트랜잭션 관리

- 장기 실행 트랜잭션 식별 및 모니터링
- 업그레이드를 시작하기 전에 모든 중요한 트랜잭션이 커밋되었는지 확인합니다.
- 업그레이드 기간 동안 장기 실행 트랜잭션을 방지합니다.

로그 파일 최적화

다음과 같이 트랜잭션 로그 파일을 검토하고 최적화합니다.

- 과대 로그 파일을 줄입니다.
- 높은 로그 소비 패턴을 줄입니다.
- 가상 로그 파일(VLF)을 관리합니다.
- 정상적인 작업을 위해 충분한 여유 공간을 유지합니다.

다중 AZ 고려 사항

Amazon RDS는 SQL Server 데이터베이스 미러링(DBM) 또는 상시 가동 가용성 그룹(AG)을 사용하여 Microsoft SQL Server 기반 DB 인스턴스의 다중 AZ 배포를 지원합니다. 자세한 내용은 Amazon RDS for Microsoft SQL Server의 다중 AZ 배포 섹션을 참조하세요.

다중 AZ 배포(미러링/AlwaysOn)에서 업그레이드가 요청되면 RDS는 기본 및 보조 인스턴스에 대한 롤링 업그레이드 전략을 따릅니다. 롤링 업그레이드를 사용하면 보조 인스턴스가 업그레이드되는 동안 트랜잭션에 최소 하나의 인스턴스를 사용할 수 있습니다. 중단은 장애 조치 기간 동안만 지속될 것으로 예상됩니다.

업그레이드 중에 RDS는 다중 AZ 구성에서 보조 인스턴스를 제거하고, 보조 인스턴스의 업그레이드를 수행하고, 연결이 해제된 시간 동안 가져온 기본 인스턴스에서 트랜잭션 로그 백업을 복원합니다. 모든 로그 백업이 복원되면 RDS는 업그레이드된 보조 인스턴스를 기본 인스턴스에 조인합니다. 모든 데이터베이스가 동기화된 상태이면 RDS는 업그레이드된 보조 인스턴스로의 장애 조치를 수행합니다. 장애 조치가 완료되면 RDS는 이전 기본 인스턴스 업그레이드를 진행하고 트랜잭션 로그 백업을 복원한 다음, 새 기본 인스턴스와 페어링합니다.

이 장애 조치 기간을 최소화하려면 연결 문자열에서 `MultiSubnetFailover` 연결 옵션을 지원하는 클라이언트 라이브러리를 사용할 때 AlwaysOn AGs 가용성 그룹 리스너 엔드포인트를 사용하는 것이 좋습니다. 가용성 그룹 리스너 엔드포인트를 사용할 때 장애 조치 시간은 일반적으로 10초 미만이지만, 이 기간에는 추가 충돌 복구 시간이 포함되지 않습니다.

읽기 전용 복제본 고려 사항

데이터베이스 버전 업그레이드 중에 Amazon RDS는 기본 DB 인스턴스와 함께 읽기 전용 복제본도 모두 업그레이드합니다. Amazon RDS는 읽기 전용 복제본에 대한 데이터베이스 버전 업그레이드를 별도로 지원하지 않습니다. 읽기 전용 복제본에 대한 자세한 내용은 Amazon RDS에서 Microsoft SQL Server용 읽기 전용 복제본 작업 섹션을 참조하세요.

기본 DB 인스턴스의 데이터베이스 버전 업그레이드를 수행하면 읽기 전용 복제본도 모두 자동으로 업그레이드됩니다. Amazon RDS는 기본 DB 인스턴스를 업그레이드하기 전에 읽기 전용 복제본을 모두 동시에 업그레이드합니다. 기본 DB 인스턴스의 데이터베이스 버전 업그레이드가 완료되기 전까지는 읽기 전용 복제본을 사용하지 못할 수 있습니다.

옵션 그룹 고려 사항

DB 인스턴스에서 사용자 지정 DB 옵션 그룹을 사용할 경우 Amazon RDS에서 DB 인스턴스에 새 옵션 그룹을 할당할 수 없는 경우도 있습니다. 예를 들어 새로운 메이저 버전으로 업그레이드할 경우 새로운

옵션 그룹을 지정해야 합니다. 새 옵션 그룹을 생성하고 동일한 옵션을 기존 사용자 지정 옵션 그룹에 추가하는 것이 좋습니다.

자세한 내용은 옵션 그룹 생성 또는 옵션 그룹 생성 섹션을 참조하세요.

파라미터 그룹 고려 사항

DB 인스턴스에서 사용자 지정 DB 파라미터 그룹을 사용하는 경우:

- 업그레이드 후 Amazon RDS가 DB 인스턴스를 자동으로 재부팅합니다.
- 경우에 따라 RDS가 DB 인스턴스에 새 파라미터 그룹을 자동으로 할당하지 못할 수 있습니다.

  예를 들어 새로운 메이저 버전으로 업그레이드할 경우 새로운 파라미터 그룹을 지정해야 합니다. 새 파라미터 그룹을 생성하고 기존 사용자 지정 파라미터 그룹에서와 같은 방법으로 파라미터를 구성하는 것이 좋습니다.

자세한 내용은 Amazon RDS에서 DB 파라미터 그룹 생성 또는 Amazon RDS에서 DB 파라미터 그룹 복사 단원을 참조하세요.

RDS for SQL Server 업그레이드 테스트

DB 인스턴스에 대한 메이저 버전 업그레이드를 수행하기 전에 데이터베이스 및 해당 데이터베이스에 액세스하는 모든 애플리케이션이 새 버전과 호환되는지 여부를 철저하게 테스트해야 합니다. 다음 절차를 참조하는 것이 좋습니다.

메이저 버전 업그레이드를 테스트하려면

1. 새 버전의 데이터베이스 엔진에 대한 Microsoft 설명서에서 SQL Server 업그레이드를 검토하여 데이터베이스나 애플리케이션에 영향을 끼칠 수도 있는 호환성 문제가 있는지 살펴봅니다.
2. DB 인스턴스에서 사용자 지정 옵션 그룹을 사용할 경우 업그레이드하려는 새 버전과 호환되는 새 옵션 그룹을 생성합니다. 자세한 내용은 옵션 그룹 고려 사항 섹션을 참조하세요.
3. DB 인스턴스에서 사용자 지정 파라미터 그룹을 사용할 경우 업그레이드하려는 새 버전과 호환되는 새 파라미터 그룹을 생성합니다. 자세한 내용은 파라미터 그룹 고려 사항 섹션을 참조하세요.
4. 업그레이드할 DB 인스턴스의 DB 스냅샷을 생성합니다. 자세한 내용은 Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성 섹션을 참조하세요.
5. DB 스냅샷을 복구하여 새로운 테스트 DB 인스턴스를 생성합니다. 자세한 내용은 DB 인스턴스 복원 섹션을 참조하세요.

6. 다음 방법 중 한 가지를 사용하여 이 새로운 테스트 DB 인스턴스를 변경하고 새로운 버전으로 업그레이드합니다.

   - 콘솔
   - AWS CLI
   - RDS API

7. 업그레이드한 인스턴스에서 사용할 스토리지를 평가하여 업그레이드 시 추가 스토리지의 필요 여부를 결정합니다.
8. 업그레이드한 DB 인스턴스와 관련하여 데이터베이스 및 애플리케이션과 새로운 버전의 호환성을 보장하는 데 필요하다면 최대한 많은 수의 품질 보증 테스트를 실행합니다. 또한 1단계에서 발견된 호환성 문제의 영향을 평가하는 데 필요한 새로운 테스트도 모두 실행합니다. 저장된 프로시저와 함수를 모두 테스트합니다. 업그레이드한 DB 인스턴스에 대해 애플리케이션의 테스트 버전을 실행합니다.
9. 모든 테스트가 통과되면 프로덕션 환경의 DB 인스턴스에도 업그레이드를 실행합니다. 단, 모든 기능이 정상 작동하는 것을 확인할 때까지 쓰기 연산은 DB 인스턴스에 실행하지 않는 것이 좋습니다.

SQL Server DB 인스턴스 업그레이드

SQL Server DB 인스턴스의 수동 또는 자동 업그레이드에 대한 자세한 내용은 다음을 참조하십시오.

- DB 인스턴스 엔진 버전 업그레이드
- Amazon RDS for SQL Server의 SQL Server 2008 R2를 SQL Server 2016으로 업그레이드하는 모범 사례

> **Important**
>
> AWS KMS를 사용하여 암호화한 스냅샷이 있는 경우, 지원이 끝나기 전에 업그레이드를 시작하는 것이 좋습니다.

지원 종료 전에 사용되지 않는 DB 인스턴스 업그레이드

메이저 버전이 사용되지 않으면 새 DB 인스턴스에 설치할 수 없습니다. RDS는 기존의 모든 DB 인스턴스를 자동으로 업그레이드하려고 시도합니다.

사용되지 않는 DB 인스턴스를 복원해야 하는 경우 특정 시점으로 복구(PITR)하거나 스냅샷을 복원할 수 있습니다. 이렇게 하면 사용되지 않는 버전을 사용하는 DB 인스턴스에 임시 액세스할 수 있습니다. 그러나 메이저 버전이 완전히 사용되지 않게 되면 이 DB 인스턴스도 지원되는 버전으로 자동 업그레이드됩니다.

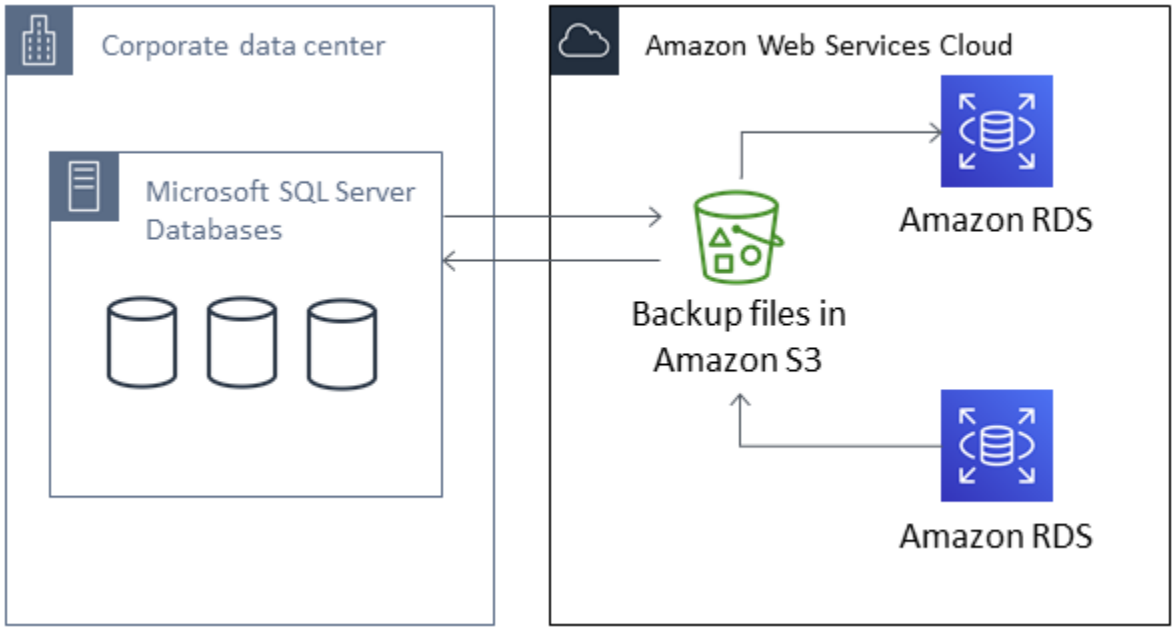
기본 백업 및 복원 기능을 사용하여 SQL Server 데이터베이스 가져오기 및 내보내기

Amazon RDS는 전체 백업 파일(.bak 파일)을 사용하여 Microsoft SQL Server 데이터베이스에 기본 백업 및 복원을 할 수 있도록 지원합니다. RDS를 사용할 때 데이터베이스 서버의 로컬 파일 시스템을 사용하는 대신 Amazon S3에 저장된 파일에 액세스합니다.

예를 들어 로컬 서버에서 전체 백업을 생성하고, 이를 S3에 저장한 후, 기존 Amazon RDS DB 인스턴스에서 복원할 수 있습니다. 또한 RDS에서 백업을 만들고, 이를 S3에 저장한 후, 어디든 원하는 곳에서 복원할 수 있습니다.

읽기 전용 복제본이 있는 다중 AZ DB 인스턴스를 포함하여 모든 AWS 리전에서 단일 AZ 및 다중 AZ DB 인스턴스에 대해 기본 백업 및 복원을 사용할 수 있습니다. Amazon RDS에서 지원되는 Microsoft SQL Server의 모든 버전에 기본 백업 및 복원이 제공됩니다.

다음 다이어그램은 지원되는 시나리오를 보여 줍니다.



기본 .bak 파일을 사용하여 데이터베이스를 백업 및 복원하는 과정은 대개의 경우 데이터베이스를 가장 빨리 백업하고 복원할 수 있는 방법입니다. 또한 기본 백업 및 복원을 이용하는 것보다 장점이 더 많습니다. 예를 들면,

- Amazon RDS로/에서 데이터베이스 마이그레이션
- RDS for SQL Server DB 인스턴스 간에 데이터베이스를 이동합니다.

- .bak 파일 내부의 데이터, 스키마, 저장 프로시저, 트리거 및 기타 데이터베이스 코드를 마이그레이션합니다.
- DB 인스턴스 전체가 아닌 데이터베이스 하나를 백업 및 복원합니다.
- 개발, 테스트, 교육, 데모를 위해 데이터베이스 사본을 만듭니다.
- 재해 복구를 위한 추가 보호 계층을 위해 Amazon S3를 통해 백업 파일을 저장 및 전송합니다.
- 투명한 데이터 암호화(TDE)가 설정된 데이터베이스의 기본 백업을 생성하고, 이러한 백업을 온프레미스 데이터베이스에 복원합니다. 자세한 내용은 SQL Server에서 TDE(투명한 데이터 암호화) 지원 섹션을 참조하세요.
- TDE가 설정된 온프레미스 데이터베이스의 기본 백업을 RDS for SQL Server DB 인스턴스로 복원합니다. 자세한 내용은 SQL Server에서 TDE(투명한 데이터 암호화) 지원 섹션을 참조하세요.

목차

- 제한 및 권장 사항
- 기본 백업 및 복원 설정
  - 기본 백업 및 복원을 위한 IAM 역할 수동으로 만들기
- 기본 백업 및 복원 사용
  - 데이터베이스 백업
    - 사용법
    - 예시
  - 데이터베이스 복원
    - 사용법
    - 예시
  - 로그 복원
    - 사용법
    - 예시
  - 데이터베이스 복원 마무리
    - 사용법
  - 부분적으로 복원된 데이터베이스 작업
    - 부분적으로 복원된 데이터베이스 삭제
    - 부분적으로 복원된 데이터베이스에 대한 스냅샷 복원 및 특정 시점으로 복구 동작
  - 작업 취소

        - 사용법
    - 작업 상태 추적
        - 사용법
        - 예시
        - 응답
- 백업 파일 압축
- 문제 해결
- 다른 방법으로 SQL Server 데이터 가져오기 및 내보내기
    - 스냅샷을 사용하여 RDS for SQL Server로 데이터 가져오기
        - 데이터 가져오기
            - 스크립트 생성 및 게시 마법사
            - 가져오기 및 내보내기 마법사
            - 대량 복사
    - RDS for SQL Server에서 데이터 내보내기
        - SQL Server 가져오기 및 내보내기 마법사
        - SQL Server 스크립트 생성 및 게시 마법사와 bcp 유틸리티
- Linux의 BCP 유틸리티를 사용하여 데이터 가져오기 및 내보내기
    - 사전 조건
    - Linux에 SQL Server 명령줄 도구 설치
    - RDS for SQL Server에서 데이터 내보내기
        - 기본 내보내기 구문
        - 내보내기 예시
    - RDS for SQL Server로 데이터 가져오기
        - 기본 가져오기 구문
        - 가져오기 예시
    - 일반적인 BCP 옵션
    - 모범 사례 및 고려 사항
    - 일반적인 문제 해결

제한 및 권장 사항

다음은 기본 백업 및 복원을 사용할 때 적용되는 몇 가지 제한 사항입니다.

- Amazon RDS DB 인스턴스와 다른 AWS 리전에서는 Amazon S3 버킷으로 백업하거나 복원할 수 없습니다.
- 기존 데이터베이스와 이름이 같은 데이터베이스는 복원할 수 없습니다. 데이터베이스 이름은 고유합니다.
- 한 표준 시간대의 백업 파일을 다른 표준 시간대로 복원하지 않는 것이 좋습니다. 한 표준 시간대의 백업 파일을 다른 표준 시간대로 복원하는 경우 쿼리와 애플리케이션을 감사하여 표준 시간대 변경의 영향을 확인해야 합니다.
- Amazon S3에는 파일당 5TB의 크기 제한이 있습니다. 큰 데이터베이스의 기본 백업의 경우 다중 파일 백업을 사용할 수 있습니다.
- S3에 백업할 수 있는 최대 데이터베이스 크기는 DB 인스턴스에서 사용 가능한 메모리, CPU, I/O 및 네트워크 리소스에 따라 다릅니다. 데이터베이스가 클수록 백업 에이전트에서 더 많은 메모리를 사용합니다.
- 10개 이상의 백업 파일에서 동시에 백업 또는 복원할 수 없습니다.
- 차등 백업은 마지막 전체 백업을 기반을 합니다. 차등 백업이 작동할 수 있도록 마지막 전체 백업과 차등 백업 간에는 스냅샷을 만들 수 없습니다. 차등 백업을 만들려고 하는데 수동 또는 자동 스냅샷이 이미 있으면 차등 백업을 진행하기 전에 다른 전체 백업을 만드십시오.
- file_guid(고유 식별자)가 NULL로 설정된 파일이 있는 데이터베이스에서는 차등 복원 및 로그 복원이 지원되지 않습니다.
- 백업 또는 복원 작업은 최대 2개까지 동시에 실행할 수 있습니다.
- Amazon RDS의 SQL Server에서 기본 로그 백업을 수행할 수 없습니다.
- RDS는 데이터베이스의 기본 복원을 최대 64TiB까지 지원합니다. SQL Server Express Edition의 데이터베이스 기본 복원은 10GB로 제한됩니다.
- 유지 관리 기간 또는 Amazon RDS에서 데이터베이스 스냅샷을 만드는 동안에는 기본 백업을 수행할 수 없습니다. 기본 백업 작업이 RDS 일일 백업 기간과 겹치는 경우 기본 백업 작업이 취소됩니다.
- 다중 AZ DB 인스턴스에서는 전체 복구 모델로 백업된 데이터베이스만 기본적으로 복원할 수 있습니다.
- 다중 AZ 인스턴스의 차등 백업에서 복원하는 것은 지원되지 않습니다.
- 트랜잭션 내에서 기본 백업 및 복원을 위한 RDS 프로시저 호출은 지원되지 않습니다.

- 대칭 암호화 AWS KMS key을(를) 사용하여 백업을 암호화합니다. Amazon RDS에서는 비대칭 KMS 키가 지원되지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 대칭 암호화 KMS 키 생성을 참조하세요.
- 기본 백업 파일은 "암호화 전용" 암호 모드를 사용하여 지정된 KMS 키로 암호화됩니다. 암호화된 백업 파일을 복원할 때 파일이 "암호화 전용" 암호 모드로 암호화되었음을 알고 있어야 합니다.
- FILESTREAM 파일 그룹이 포함된 데이터베이스는 복원할 수 없습니다.
- AWS KMS를 사용한 Amazon S3 서버 측 암호화(SSE-KMS)는 현재 지원되지 않습니다. 저장 프로시저에 KMS 키를 제공하면 기본 백업 및 복원이 KMS 키를 사용하여 클라이언트 측에서 암호화되고 해독됩니다. AWS는 SSE-S3를 사용하여 S3 버킷에 백업을 저장합니다.

백업 파일을 만들고, 복사하고, 복원하는 동안 데이터베이스를 오프라인 상태로 둘 수 있다면 RDS로 마이그레이션할 때 기본 백업 및 복원을 사용하는 것이 좋습니다. 온프레미스 데이터베이스를 오프라인으로 전환할 수 없다면 AWS Database Migration Service를 사용하여 데이터베이스를 Amazon RDS로 마이그레이션하는 것이 좋습니다. 자세한 내용은 AWS Database Migration Service란 무엇입니까?를 참조하십시오.

기본 백업 및 복원은 리전 간 스냅샷 복사 기능의 데이터 복구 대신 사용하기 위한 기능이 아닙니다. Amazon RDS에서 교차 리전 재해 복구를 위해 데이터베이스 스냅샷을 다른 AWS 리전으로 복사할 때는 스냅샷 복사를 이용하는 것이 좋습니다. 자세한 내용은 Amazon RDS용 DB 스냅샷 복사 섹션을 참조하세요.

기본 백업 및 복원 설정

기본 백업 및 복원을 설정하려면 다음 세 가지 구성 요소가 필요합니다.

1. 백업 파일을 저장할 Amazon S3 버킷.

   백업 파일에 사용한 후 RDS로 마이그레이션하려는 백업을 업로드할 S3 버킷이 있어야 합니다. 이미 Amazon S3 버킷이 있으면 그 버킷을 사용하면 됩니다. 없는 경우 버킷을 생성할 수 있습니다. 또는 SQLSERVER_BACKUP_RESTORE을 사용하여 AWS Management Console 옵션을 추가할 때 새 버킷이 생성되도록 선택할 수도 있습니다.

   S3 사용에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서를 참조하세요.
2. 버킷에 액세스하기 위한 AWS Identity and Access Management(IAM) 역할.

   이미 IAM 역할이 있으면 그 역할을 사용하면 됩니다. AWS Management Console을 사용하여 SQLSERVER_BACKUP_RESTORE 옵션을 추가할 때 새 IAM 역할이 생성되도록 선택할 수도 있습니다. 또는 수동으로 역할을 새로 만들 수 있습니다.

수동으로 새 IAM 역할을 만들려면 다음 단원에서 설명하는 방법을 사용하십시오. 기존 IAM 역할에 신뢰 관계와 권한 정책을 연결하려는 경우에도 동일한 작업을 수행합니다.

3. DB 인스턴스의 옵션 그룹에 SQLSERVER_BACKUP_RESTORE 옵션 추가.

   DB 인스턴스에서 기본 백업 및 복원을 활성화하려면 DB 인스턴스의 옵션 그룹에 SQLSERVER_BACKUP_RESTORE 옵션을 추가합니다. 자세한 정보와 지침은 SQL Server에서 기본 백업 및 복원 지원 단원을 참조하십시오.

기본 백업 및 복원을 위한 IAM 역할 수동으로 만들기

기본 백업 및 복원에 사용할 새 IAM 역할을 수동으로 생성할 수 있습니다. 이 경우 Amazon RDS 서비스에서 Amazon S3 버킷으로 권한을 위임하는 역할을 생성합니다. IAM 역할을 만들 때 신뢰 관계와 권한 정책을 연결합니다. 신뢰 관계를 통해 RDS가 이 역할을 맡도록 허용합니다. 권한 정책은 이 역할이 수행할 수 있는 작업을 정의합니다. 역할을 만드는 방법에 대한 자세한 내용은 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.

기본 백업 및 복원 기능의 경우, 이 단원의 예제와 비슷한 신뢰 관계 및 권한 정책을 사용합니다. 다음 예제에서는 서비스 보안 주체 이름 rds.amazonaws.com을 모든 서비스 계정의 별칭으로 사용합니다. 다른 예에서는 신뢰 정책에서 액세스 권한을 부여할 다른 계정, 사용자, 역할을 식별하기 위해 Amazon 리소스 이름(ARN)을 지정합니다.

서비스 권한을 특정 리소스로 제한하는 리소스 기반 신뢰 관계의 aws:SourceArn 및 aws:SourceAccount 전역 조건 컨텍스트 키를 사용하는 것이 좋습니다. 이는 혼동된 대리자 문제를 방지하는 가장 효과적인 방법입니다.

전역 조건 컨텍스트 키를 모두 사용하고 aws:SourceArn 값에 계정 ID가 포함되도록 할 수 있습니다. 이 경우 aws:SourceAccount 값과 aws:SourceArn 값의 계정이 동일한 문에서 사용될 때 동일한 계정 ID를 사용해야 합니다.

- 단일 리소스에 대한 교차 서비스 액세스를 원하는 경우 aws:SourceArn을 사용하세요.
- 해당 계정의 모든 리소스가 교차 서비스 사용과 연결되도록 허용하려는 경우 aws:SourceAccount를 사용하세요.

신뢰 관계에서는 역할에 액세스하는 리소스의 전체 ARN이 포함된 aws:SourceArn 전역 조건 컨텍스트 키를 사용해야 합니다. 기본 백업 및 복원의 경우 다음 예와 같이 DB 옵션 그룹과 DB 인스턴스를 모두 포함해야 합니다.

Example 기본 백업 및 복원을 위한 전역 조건 컨텍스트 키와의 신뢰 관계

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rds.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceArn": [

 "arn:aws:rds:Region:0123456789:db:db_instance_identifier",
                        "arn:aws:rds:Region:0123456789:og:option_group_name",
                  "arn:aws:s3:::amzn-s3-demo-bucket"
                    ],
                    "aws:SourceAccount": "0123456789"
                }
            }
        }
    ]
}
```

다음 예에서는 ARN을 사용하여 리소스를 지정합니다. ARN 사용에 대한 자세한 내용은 Amazon 리소스 이름(ARN)을 참조하십시오.

Example 암호화 지원 없는 기본 백업 및 복원을 위한 권한 정책

JSON

```
{
    "Version": "2012-10-17",
    "Statement":
    [
        {
        "Effect": "Allow",
```

```
        "Action":
            [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
        },
        {
        "Effect": "Allow",
        "Action":
            [
                "s3:GetObjectAttributes",
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

Example 암호화 지원 있는 기본 백업 및 복원을 위한 권한 정책

백업 파일을 암호화하려면 권한 정책에 암호화 키를 포함시켜야 합니다. 암호화 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서에서 시작하기를 참조하세요.

> **Note**
>
> 대칭 암호화 KMS 키를 사용하여 백업을 암호화해야 합니다. Amazon RDS에서는 비대칭 KMS 키가 지원되지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 대칭 암호화 KMS 키 생성을 참조하세요.
> IAM 역할은 KMS 키의 키 사용자 및 키 관리자여야 합니다. 즉, 키 정책에서 지정해야 합니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 대칭 암호화 KMS 키 생성을 참조하세요.

JSON

```
{
```

```
    "Version": "2012-10-17",
    "Statement":
    [
        {
        "Effect": "Allow",
        "Action":
            [
                "kms:DescribeKey",
                "kms:GenerateDataKey",
                "kms:Encrypt",
                "kms:Decrypt"
            ],
        "Resource": "arn:aws:kms:region:account-id:key/key-id"
        },
        {
        "Effect": "Allow",
        "Action":
            [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
        },
        {
        "Effect": "Allow",
        "Action":
            [
                "s3:GetObjectAttributes",
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

기본 백업 및 복원 사용

기본 백업 및 복원을 활성화하고 구성한 다음에는 사용하기 시작할 수 있습니다. 먼저 Microsoft SQL Server 데이터베이스에 연결한 다음 해당 작업을 위한 Amazon RDS 저장 프로시저를 호출합니다. 데이터베이스 연결 방법에 대한 자세한 내용은 Microsoft SQL Server DB 인스턴스에 연결 단원을 참조하십시오.

저장 프로시저 중에는 Amazon S3 버킷과 파일에 ARN(Amazon 리소스 이름)을 제공해야 하는 것도 있습니다. ARN의 형식은 `arn:aws:s3:::bucket_name/file_name.extension`입니다. Amazon S3에서는 ARN에 계정 번호나 AWS 리전을 요구하지 않습니다.

KMS 키(선택 사항)도 제공하는 경우 키의 ARN 형식은 `arn:aws:kms:region:account-id:key/key-id`입니다. 자세한 내용은 Amazon 리소스 이름(ARN) 및 AWS 서비스 네임스페이스를 참조하세요. 대칭 암호화 KMS 키를 사용하여 백업을 암호화해야 합니다. Amazon RDS에서는 비대칭 KMS 키가 지원되지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 대칭 암호화 KMS 키 생성을 참조하세요.

> **Note**
>
> KMS 키를 사용하는지 여부에 관계없이 기본 백업 및 복원 작업을 통해 S3에 업로드된 파일에 대해 기본값으로 서버 측 고급 암호화 표준(AES) 256비트 암호화를 사용할 수 있습니다.

각각의 저장 프로시저를 호출하는 방법에 대한 지침은 다음 주제를 참조하십시오.

- 데이터베이스 백업
- 데이터베이스 복원
- 로그 복원
- 데이터베이스 복원 마무리
- 부분적으로 복원된 데이터베이스 작업
- 작업 취소
- 작업 상태 추적

데이터베이스 백업

데이터베이스를 백업하려면 `rds_backup_database` 저장 프로시저를 사용하십시오.

> **Note**
>
> 유지 관리 기간 중 또는 Amazon RDS에서 스냅샷을 만드는 동안에는 데이터베이스를 백업할 수 없습니다.

사용법

```
exec msdb.dbo.rds_backup_database
 @source_db_name='database_name',
 @s3_arn_to_backup_to='arn:aws:s3:::bucket_name/file_name.extension',
 [@kms_master_key_arn='arn:aws:kms:region:account-id:key/key-id'],
 [@overwrite_s3_backup_file=0|1],
 [@block_size=512|1024|2048|4096|8192|16384|32768|65536],
        [@max_transfer_size=n],
        [@buffer_count=n],
 [@type='DIFFERENTIAL|FULL'],
 [@number_of_files=n];
```

다음 파라미터는 필수 파라미터입니다.

- `@source_db_name` – 백업할 데이터베이스의 이름입니다.
- `@s3_arn_to_backup_to` – 백업에 사용할 Amazon S3 버킷과 백업 파일 이름을 나타내는 ARN입니다.

  파일에는 어떤 확장명이든 있을 수 있지만 `.bak`이 주로 사용됩니다.

다음 파라미터는 선택적입니다.

- `@kms_master_key_arn` – 항목을 암호화하는 데 사용할 대칭 암호화 KMS 키의 ARN입니다.
  - 기본 암호화 키는 사용할 수 없습니다. 기본 키를 사용하면 데이터베이스가 백업되지 않습니다.
  - KMS 키 식별자를 지정하지 않으면 백업 파일이 암호화되지 않습니다. 자세한 내용은 Amazon RDS 리소스 암호화 단원을 참조하십시오.
  - KMS 키를 지정하면 클라이언트 측 암호화가 사용됩니다.
  - Amazon RDS에서는 비대칭 KMS 키가 지원되지 않습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 대칭 암호화 KMS 키 생성을 참조하세요.
- `@overwrite_s3_backup_file` – 기존 백업 파일을 덮어쓸지 여부를 나타내는 값입니다.
  - `0` – 기존 파일을 덮어쓰지 않습니다. 이 값이 기본값입니다.

@overwrite_s3_backup_file을 0으로 설정하면 파일이 이미 존재할 경우 오류를 반환합니다.

  - 1 – 백업 파일이 아니더라도 지정된 이름이 있는 기존 파일을 덮어씁니다.
- @type – 백업 유형입니다.
  - DIFFERENTIAL – 차등 백업을 수행합니다.
  - FULL – 전체 백업을 수행합니다. 이 값이 기본값입니다.

  차등 백업은 마지막 전체 백업을 기반을 합니다. 차등 백업이 작동할 수 있도록 마지막 전체 백업과 차등 백업 간에는 스냅샷을 만들 수 없습니다. 차등 백업을 만들려고 하는데 스냅샷이 이미 있으면 차등 백업을 진행하기 전에 다른 전체 백업을 만드십시오.

  다음 예제 SQL 쿼리를 사용하여 마지막 전체 백업 또는 스냅샷을 찾을 수 있습니다.

```
select top 1
database_name
,  backup_start_date
,  backup_finish_date
from    msdb.dbo.backupset
where   database_name='mydatabase'
and     type = 'D'
order by backup_start_date desc;
```

- @number_of_files – 백업을 분할(청크)할 파일 수입니다. 최대 개수는 10입니다.
  - 다중 파일 백업은 전체 백업과 차등 백업에 모두 지원됩니다.
  - 값 1을 입력하거나 파라미터를 생략하면 단일 백업 파일이 생성됩니다.

  파일에 공통으로 사용되는 접두사를 입력한 다음 접미사로 별표(*)를 붙입니다. 별표는 S3 ARN의 *file_name* 부분 아무 곳이나 지정할 수 있습니다. 별표는 생성된 파일에서 일련의 영숫자 문자열로 대체되며 1-of-*number_of_files*부터 시작합니다.

  예를 들어, S3 ARN의 파일 이름이 backup*.bak이고 @number_of_files=4를 설정한 경우 생성되는 백업 파일은 backup1-of-4.bak, backup2-of-4.bak, backup3-of-4.bak 및 backup4-of-4.bak입니다.

  - 파일 이름이 이미 존재하고 @overwrite_s3_backup_file이 0이면 오류가 반환됩니다.
  - 다중 파일 백업에는 S3 ARN의 *file_name* 부분에 별표를 하나만 지정할 수 있습니다.
  - 단일 파일 백업에는 S3 ARN의 *file_name* 부분에 별표를 여러 개 지정할 수 있습니다. 별표는 생성된 파일 이름에서 제거되지 않습니다.

- `@block_size` - 백업 작업의 물리적 블록 크기를 지정하는 블록 크기(바이트)입니다. 유효한 값은 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536입니다.
- `@max_transfer_size` – 최대 전송 크기는 백업 프로세스 중에 I/O 작업당 전송되는 데이터 볼륨의 상한(바이트)을 나타냅니다. 유효한 값은 65,536바이트(64KB)의 배수로, 최대 4,194,304바이트(4MB)입니다.
- `@buffer_count` - 백업 프로세스에 사용할 I/O 버퍼의 총 수입니다.

예시

Example 차등 백업

```
exec msdb.dbo.rds_backup_database
@source_db_name='mydatabase',
@s3_arn_to_backup_to='arn:aws:s3:::mybucket/backup1.bak',
@overwrite_s3_backup_file=1,
@type='DIFFERENTIAL';
```

Example 암호화를 사용하는 전체 백업

```
exec msdb.dbo.rds_backup_database
@source_db_name='mydatabase',
@s3_arn_to_backup_to='arn:aws:s3:::mybucket/backup1.bak',
@kms_master_key_arn='arn:aws:kms:us-east-1:123456789012:key/AKIAIOSFODNN7EXAMPLE',
@overwrite_s3_backup_file=1,
@type='FULL';
```

Example 다중 파일 백업

```
exec msdb.dbo.rds_backup_database
@source_db_name='mydatabase',
@s3_arn_to_backup_to='arn:aws:s3:::mybucket/backup*.bak',
@number_of_files=4;
```

Example 다중 파일 차등 백업

```
exec msdb.dbo.rds_backup_database
@source_db_name='mydatabase',
@s3_arn_to_backup_to='arn:aws:s3:::mybucket/backup*.bak',
@type='DIFFERENTIAL',
```

```
@number_of_files=4;
```

Example 암호화를 사용한 다중 파일 백업

```
exec msdb.dbo.rds_backup_database
@source_db_name='mydatabase',
@s3_arn_to_backup_to='arn:aws:s3:::mybucket/backup*.bak',
@kms_master_key_arn='arn:aws:kms:us-east-1:123456789012:key/AKIAIOSFODNN7EXAMPLE',
@number_of_files=4;
```

Example S3 덮어쓰기를 사용한 다중 파일 백업

```
exec msdb.dbo.rds_backup_database
@source_db_name='mydatabase',
@s3_arn_to_backup_to='arn:aws:s3:::mybucket/backup*.bak',
@overwrite_s3_backup_file=1,
@number_of_files=4;
```

Example 블록 크기를 사용한 백업

```
exec msdb.dbo.rds_backup_database
@source_db_name='mydatabase',
@s3_arn_to_backup_to='arn:aws:s3:::mybucket/backup*.bak',
@block_size=512;
```

Example **@max_transfer_size** 및 **@buffer_count**를 사용한 멀티파일 백업

```
exec msdb.dbo.rds_backup_database
@source_db_name='mydatabase',
@s3_arn_to_backup_to='arn:aws:s3:::mybucket/backup*.bak',
@number_of_files=4,
@max_transfer_size=4194304,
@buffer_count=10;
```

Example @number_of_files 파라미터를 사용하는 단일 파일 백업

이 예제에서는 backup*.bak 백업 파일을 생성합니다.

```
exec msdb.dbo.rds_backup_database
@source_db_name='mydatabase',
@s3_arn_to_backup_to='arn:aws:s3:::mybucket/backup*.bak',
```

```
@number_of_files=1;
```

데이터베이스 복원

데이터베이스를 복원하려면 rds_restore_database 저장 프로시저를 호출합니다. 복원 태스크가 완료되고 데이터베이스가 열린 후 Amazon RDS가 데이터베이스의 초기 스냅샷을 생성합니다.

사용법

```
exec msdb.dbo.rds_restore_database
 @restore_db_name='database_name',
 @s3_arn_to_restore_from='arn:aws:s3:::bucket_name/file_name.extension',
 @with_norecovery=0|1,
 [@kms_master_key_arn='arn:aws:kms:region:account-id:key/key-id'],
        [@block_size=512|1024|2048|4096|8192|16384|32768|65536],
        [@max_transfer_size=n],
        [@buffer_count=n],
 [@type='DIFFERENTIAL|FULL'];
```

다음 파라미터는 필수 파라미터입니다.

- @restore_db_name – 복원할 데이터베이스의 이름입니다. 데이터베이스 이름은 고유합니다. 기존 데이터베이스와 이름이 같은 데이터베이스는 복원할 수 없습니다.
- @s3_arn_to_restore_from – 데이터베이스를 복원하는 데 사용되는 백업 파일의 Amazon S3 접두사 및 이름을 나타내는 ARN입니다.
  - 단일 파일 백업의 경우 전체 파일 이름을 입력하십시오.
  - 다중 파일 백업의 경우 파일에 공통으로 사용되는 접두사를 제공한 다음 접미사로 별표(*)를 붙입니다.
  - @s3_arn_to_restore_from이 비어 있으면 다음 오류가 반환됩니다. S3 ARN prefix cannot be empty(S3 ARN 접두사는 비워 둘 수 없습니다).

차등 복원에는 다음 파라미터가 필수적이지만 전체 복원에는 선택적입니다.

- @with_norecovery – 복원 작업에 사용할 복구 절입니다.
  - RECOVERY로 복원하려면 0으로 설정합니다. 이 경우 데이터베이스는 복원 후 온라인 상태입니다.
  - NORECOVERY로 복원하려면 1로 설정합니다. 이 경우 데이터베이스는 복원 작업 완료 후 RESTORING 상태로 유지됩니다. 이 방법을 사용하면 나중에 차등 복원을 수행할 수 있습니다.

- DIFFERENTIAL 복원의 경우 0 또는 1을 지정하십시오.
- FULL 복원의 경우, 이 값은 기본적으로 0로 설정됩니다.

다음 파라미터는 선택적입니다.

- @kms_master_key_arn – 백업 파일을 암호화한 경우 파일 복호화에 사용할 KMS 키입니다.

  KMS 키를 지정하면 클라이언트 측 암호화가 사용됩니다.
- @type – 복원의 유형입니다. 유효한 형식은 DIFFERENTIAL 및 FULL입니다. 기본 값은 FULL입니다.
- @block_size - 백업 작업의 물리적 블록 크기를 지정하는 블록 크기(바이트)입니다. 유효한 값은 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536입니다.
- @max_transfer_size – 최대 전송 크기는 백업 프로세스 중에 I/O 작업당 전송되는 데이터 볼륨의 상한(바이트)을 나타냅니다. 유효한 값은 65,536바이트(64KB)의 배수로, 최대 4,194,304바이트(4MB)입니다.
- @buffer_count - 백업 프로세스에 사용할 I/O 버퍼의 총 수입니다.

> **Note**
>
> 차등 복원의 경우 데이터베이스가 RESTORING 상태이거나 NORECOVERY로 복원하는 작업이 이미 존재해야 합니다.
> 데이터베이스가 온라인 상태인 동안 나중에 차등 백업을 복원할 수 없습니다.
> 이미 RECOVERY로 복원 작업을 보류 중인 데이터베이스에 대해서는 복원 작업을 제출할 수 없습니다.
> NORECOVERY로 전체 복원 및 차등 복원은 다중 AZ 인스턴스에서 지원되지 않습니다.
> 읽기 전용 복제본을 사용하여 다중 AZ 인스턴스에서 데이터베이스를 복원하는 것은 다중 AZ 인스턴스에서 데이터베이스를 복원하는 것과 유사합니다. 복제본에서 데이터베이스를 복원하기 위해 추가 작업을 수행할 필요가 없습니다.

예시

Example 단일 파일 복원

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
```

```
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup1.bak';
```

Example 다중 파일 복원

여러 파일을 복원하는 동안 오류를 방지하려면 모든 백업 파일의 접두사가 같고 다른 파일은 이 접두사를 사용하지 않아야 합니다.

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup*';
```

Example RECOVERY로 전체 데이터베이스 복원

다음 세 가지 예는 동일한 작업, RECOVERY를 사용한 전체 복원을 수행합니다.

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup1.bak';
```

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup1.bak',
[@type='DIFFERENTIAL|FULL'];
```

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup1.bak',
@type='FULL',
@with_norecovery=0;
```

Example 암호화로 전체 데이터베이스 복원

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup1.bak',
@kms_master_key_arn='arn:aws:kms:us-east-1:123456789012:key/AKIAIOSFODNN7EXAMPLE';
```

Example 블록 크기를 사용한 복원

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
```

```
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup1.bak',
@block_size=512;
```

Example @max_transfer_size 및 @buffer_count를 사용한 멀티파일 복원

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup*',
@max_transfer_size=4194304,
@buffer_count=10;
```

Example NORECOVERY로 전체 데이터베이스 복원

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup1.bak',
@type='FULL',
@with_norecovery=1;
```

Example NORECOVERY로 차등 복원

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup1.bak',
@type='DIFFERENTIAL',
@with_norecovery=1;
```

Example RECOVERY로 차등 복원

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup1.bak',
@type='DIFFERENTIAL',
@with_norecovery=0;
```

로그 복원

로그를 복원하려면 `rds_restore_log` 저장 프로시저를 호출합니다.

사용법

```
exec msdb.dbo.rds_restore_log
```

```
@restore_db_name='database_name',
@s3_arn_to_restore_from='arn:aws:s3:::bucket_name/log_file_name.extension',
[@kms_master_key_arn='arn:aws:kms:region:account-id:key/key-id'],
[@with_norecovery=0|1],
[@stopat='datetime'],
[@block_size=512|1024|2048|4096|8192|16384|32768|65536],
       [@max_transfer_size=n],
       [@buffer_count=n];
```

다음 파라미터는 필수 파라미터입니다.

- `@restore_db_name` – 로그를 복원할 데이터베이스의 이름입니다.
- `@s3_arn_to_restore_from` – 로그를 복원하는 데 사용되는 로그 파일의 Amazon S3 접두사 및 이름을 나타내는 ARN입니다. 파일에는 어떤 확장명이든 있을 수 있지만 `.trn`이 주로 사용됩니다.

  `@s3_arn_to_restore_from`이 비어 있으면 다음 오류가 반환됩니다. S3 ARN prefix cannot be empty(S3 ARN 접두사는 비워 둘 수 없습니다).

다음 파라미터는 선택적입니다.

- `@kms_master_key_arn` – 로그를 암호화한 경우, 로그 복호화에 사용할 KMS 키입니다.
- `@with_norecovery` – 복원 작업에 사용할 복구 절입니다. 기본값은 1입니다.
  - RECOVERY로 복원하려면 0으로 설정합니다. 이 경우 데이터베이스는 복원 후 온라인 상태입니다. 데이터베이스가 온라인 상태인 동안 더 이상 로그 백업을 복원할 수 없습니다.
  - NORECOVERY로 복원하려면 1로 설정합니다. 이 경우 데이터베이스는 복원 작업 완료 후 RESTORING 상태로 유지됩니다. 이 방법을 사용하면 나중에 로그 복원을 수행할 수 있습니다.
- `@stopat` – 데이터베이스가 지정된 날짜 및 시간(날짜 시간 형식)에서 해당 상태로 복원되도록 지정하는 값입니다. 지정된 날짜 및 시간 이전에 작성된 트랜잭션 로그 레코드만 데이터베이스에 적용됩니다.

  이 파라미터를 지정하지 않으면(NULL 인 경우) 전체 로그가 복원됩니다.
- `@block_size` - 백업 작업의 물리적 블록 크기를 지정하는 블록 크기(바이트)입니다. 유효한 값은 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536입니다.
- `@max_transfer_size` – 최대 전송 크기는 백업 프로세스 중에 I/O 작업당 전송되는 데이터 볼륨의 상한(바이트)을 나타냅니다. 유효한 값은 65,536바이트(64KB)의 배수로, 최대 4,194,304바이트(4MB)입니다.
- `@buffer_count` - 백업 프로세스에 사용할 I/O 버퍼의 총 수입니다.

> **Note**
>
> 로그 복원의 경우 데이터베이스가 복원 상태이거나 NORECOVERY로 복원하는 작업이 이미 존재해야 합니다.
> 데이터베이스가 온라인 상태인 동안 로그 백업을 복원할 수 없습니다.
> 이미 RECOVERY로 복원 작업을 보류 중인 데이터베이스에서는 로그 복원 작업을 제출할 수 없습니다.
> 다중 AZ 인스턴스에는 로그 복원이 지원되지 않습니다.

예시

Example 로그 복원

```
exec msdb.dbo.rds_restore_log
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/mylog.trn';
```

Example 암호화로 로그 복원

```
exec msdb.dbo.rds_restore_log
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/mylog.trn',
@kms_master_key_arn='arn:aws:kms:us-east-1:123456789012:key/AKIAIOSFODNN7EXAMPLE';
```

Example NORECOVERY로 로그 복원

다음 두 가지 예는 동일한 작업, NORECOVERY를 사용한 로그 복원을 수행합니다.

```
exec msdb.dbo.rds_restore_log
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/mylog.trn',
@with_norecovery=1;
```

```
exec msdb.dbo.rds_restore_log
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/mylog.trn';
```

Example 블록 크기를 사용한 복원

```
exec msdb.dbo.rds_restore_log
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/mylog.trn',
@block_size=512;
```

Example RECOVERY로 로그 복원

```
exec msdb.dbo.rds_restore_log
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/mylog.trn',
@with_norecovery=0;
```

Example STOPAT 절로 로그 복원

```
exec msdb.dbo.rds_restore_log
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/mylog.trn',
@with_norecovery=0,
@stopat='2019-12-01 03:57:09';
```

데이터베이스 복원 마무리

`@with_norecovery=1`을 사용하여 데이터베이스에서 마지막 복원 작업을 수행한 경우, 데이터베이스는 이제 RESTORING 상태입니다. `rds_finish_restore` 저장 프로시저를 사용하여 정상적으로 작동할 이 데이터베이스를 엽니다.

사용법

```
exec msdb.dbo.rds_finish_restore @db_name='database_name';
```

> **Note**
>
> 이 방법을 사용하려면 데이터베이스가 보류 중인 복원 작업 없이 RESTORING 상태여야 합니다.
> 다중 AZ 인스턴스에는 `rds_finish_restore` 프로시저가 지원되지 않습니다.
> 데이터베이스 복원을 마치려면 마스터 로그인을 사용하십시오. 또는 가장 최근에 NORECOVERY로 데이터베이스나 로그를 복원한 사용자 로그인 정보를 사용하십시오.

부분적으로 복원된 데이터베이스 작업

부분적으로 복원된 데이터베이스 삭제

부분적으로 복원된 데이터베이스를 삭제하려면(RESTORING 상태로 남음), `rds_drop_database` 저장 프로시저를 사용하십시오.

```
exec msdb.dbo.rds_drop_database @db_name='database_name';
```

> **Note**
>
> 이미 보류 중인 복원 작업이 있거나 복원 작업을 마친 데이터베이스에 대해서는 DROP 데이터베이스 요청을 제출할 수 없습니다.
> 데이터베이스를 삭제하려면 마스터 로그인을 사용합니다. 또는 가장 최근에 NORECOVERY로 데이터베이스나 로그를 복원한 사용자 로그인 정보를 사용하십시오.

부분적으로 복원된 데이터베이스에 대한 스냅샷 복원 및 특정 시점으로 복구 동작

원본 인스턴스에서 부분적으로 복원된 데이터베이스(RESTORING 상태로 남음)는 스냅샷 복원 및 특정 시점으로 복구 도중에 대상 인스턴스에서 삭제됩니다.

작업 취소

백업 또는 복원 작업을 취소하려면 `rds_cancel_task` 저장 프로시저를 호출합니다.

> **Note**
>
> FINISH_RESTORE 작업을 취소할 수 없습니다.

사용법

```
exec msdb.dbo.rds_cancel_task @task_id=ID_number;
```

다음 파라미터는 필수입니다.

- `@task_id` – 취소할 작업의 ID입니다. 작업 ID는 `rds_task_status`를 호출하여 확인할 수 있습니다.

작업 상태 추적

백업 및 복원 작업의 상태를 추적하려면 `rds_task_status` 저장 프로시저를 호출합니다. 파라미터를 제공하지 않으면 이 저장 프로시저는 모든 작업의 상태를 반환합니다. 작업 상태는 약 2분마다 업데이트됩니다. 작업 기록은 36일 동안 보존됩니다.

사용법

```
exec msdb.dbo.rds_task_status
 [@db_name='database_name'],
 [@task_id=ID_number];
```

다음 파라미터는 선택적입니다.

- `@db_name` – 작업 상태를 표시할 데이터베이스의 이름입니다.
- `@task_id` – 작업 상태를 표시할 작업의 ID입니다.

예시

Example 특정 작업의 상태 나열

```
exec msdb.dbo.rds_task_status @task_id=5;
```

Example 특정 데이터베이스 및 작업의 상태 나열

```
exec msdb.dbo.rds_task_status
@db_name='my_database',
@task_id=5;
```

Example 특정 데이터베이스의 모든 작업 및 상태 나열

```
exec msdb.dbo.rds_task_status @db_name='my_database';
```

Example 현재 인스턴스의 모든 작업 및 상태 나열

```
exec msdb.dbo.rds_task_status;
```

응답

`rds_task_status` 저장 프로시저는 다음과 같은 열을 반환합니다.

열	설명
task_id	작업의 ID입니다.

task_type	입력 파라미터에 따라 다음과 같은 작업 유형:
	• 백업 작업의 경우:
	  • BACKUP_DB – 전체 데이터베이스 백업
	  • BACKUP_DB_DIFFERENTIAL – 차등 데이터베이스 백업
	• 복원 작업의 경우:
	  • RESTORE_DB – RECOVERY로 전체 데이터베이스 복원
	  • RESTORE_DB_NORECOVERY – NORECOVERY로 전체 데이터베이스 복원
	  • RESTORE_DB_DIFFERENTIAL – RECOVERY로 차등 데이터베이스 복원
	  • RESTORE_DB_DIFFERENTIAL_NORECOVERY – NORECOVERY로 차등 데이터베이스 복원
	  • RESTORE_DB_LOG – RECOVERY로 로그 복원
	  • RESTORE_DB_LOG_NORECOVERY – NORECOVERY로 로그 복원
	• 복원을 마치는 작업의 경우:
	  • FINISH_RESTORE – 복원을 마치고 데이터베이스 열기
	
	다음 복원 작업이 완료되고 데이터베이스가 열린 후 Amazon RDS가 데이터베이스의 최초 스냅샷을 만듭니다.

열	설명
	• RESTORE_DB • RESTORE_DB_DIFFERENTIAL • RESTORE_DB_LOG • FINISH_RESTORE
database_name	작업이 연결되어 있는 데이터베이스의 이름입니다.
% complete	백분율로 나타낸 작업의 진행률입니다.
duration (mins)	작업에 소요된 시간입니다(분).
lifecycle	작업의 상태입니다. 가능한 상태는 다음과 같습니다. • CREATED – rds_backup_database 또는 rds_restore_database 를 호출하는 즉시 작업이 생성되고 상태는 CREATED로 설정됩니다. • IN_PROGRESS – 백업 또는 복원 작업이 시작된 후 상태가 로 설정됩니다.IN_PROGRESS CREATED에서 IN_PROGRESS 로 상태가 변경되려면 최대 5분이 걸릴 수 있습니다. • SUCCESS – 백업 또는 복원 작업이 완료된 후 상태가 로 설정됩니다.SUCCESS • ERROR – 백업 또는 복원 작업이 실패한 후 상태가 로 설정됩니다.ERROR 오류에 대한 자세한 내용은 task_info 열을 참조하십시오. • CANCEL_REQUESTED – rds_cancel_task 를 호출하는 즉시 작업의 상태가 CANCEL_REQUESTED 로 설정됩니다. • CANCELLED – 작업이 성공적으로 취소된 뒤에는 작업 상태가 로 설정됩니다.CANCELLED

열	설명
task_info	작업에 대한 추가 정보입니다. 데이터베이스를 백업하거나 복원하는 동안 오류가 발생하면 이 열에 오류에 대한 정보가 포함됩니다. 가능한 오류 목록 및 완화 전략은 문제 해결 단원을 참조하십시오.
last_updated	작업 상태를 마지막으로 업데이트한 날짜와 시간입니다. 상태는 5% 진행 후마다 업데이트됩니다.
created_at	작업을 생성한 날짜와 시간입니다.
S3_object_arn	백업 또는 복원 중인 파일의 Amazon S3 접두사 및 이름을 나타내는 ARN입니다.
overwrite_s3_backup_file	백업 작업을 호출할 때 지정된 @overwrite_s3_backup_file 파라미터의 값입니다. 자세한 내용은 데이터베이스 백업 단원을 참조하십시오.
KMS_master_key_arn	암호화(백업용) 및 복호화(복원용)에 사용되는 KMS 키의 ARN입니다.
filepath	기본 백업 및 복원 작업에는 해당되지 않음
overwrite_file	기본 백업 및 복원 작업에는 해당되지 않음

백업 파일 압축

Amazon S3 버킷 용량 절약을 위해 백업 파일을 압축할 수 있습니다. 백업 파일 압축에 대한 자세한 내용은 Microsoft 설명서의 Backup Compression을 참조하십시오.

백업 파일 압축은 다음 데이터베이스 에디션에 지원됩니다.

- Microsoft SQL Server Enterprise Edition
- Microsoft SQL Server Standard Edition

백업 파일 압축을 설정하려면 다음 코드를 실행하십시오.

```
exec rdsadmin.dbo.rds_set_configuration 'S3 backup compression', 'true';
```

백업 파일 압축을 해제하려면 다음 코드를 실행하십시오.

```
exec rdsadmin.dbo.rds_set_configuration 'S3 backup compression', 'false';
```

문제 해결

다음은 기본 백업 및 복원을 사용할 때 생길 수 있는 문제들입니다.

문제	문제 해결 제안
데이터베이스 백업/복원 옵션이 아직 활성화되지 않았거나 활성화되는 중입니다. 나중에 다시 시도하세요.	SQLSERVER_BACKUP_RESTORE 옵션을 DB 인스턴스와 연결된 DB 옵션 그룹에 추가해야 합니다. 자세한 내용은 기본 백업 및 복원 옵션 추가 섹션을 참조하세요.
객체 '*rds_backup_database*', 데이터베이스 'msdb', 스키마 'dbo'에서 EXECUTE 권한이 거부되었습니다.	저장 프로시저를 실행할 때 마스터 사용자를 사용하고 있는지 확인합니다. 마스터 사용자로 로그인한 후에도 이 오류가 발생하면 관리 사용자 권한이 잘못 정렬되었기 때문일 수 있습니다. 마스터 사용자를 재설정하려면 AWS Management Console을 사용합니다. Amazon RDS for SQL Server 마스터 사용자의 db_owner 역할 멤버십 재설정을(를) 참조하세요.
객체 '*rds_restore_database*', 데이터베이스 'msdb', 스키마 'dbo'에서 EXECUTE 권한이 거부되었습니다.	저장 프로시저를 실행할 때 마스터 사용자를 사용하고 있는지 확인합니다. 마스터 사용자로 로그인한 후에도 이 오류가 발생하면 관리 사용자 권한이 잘못 정렬되었기 때문일 수 있습니다. 마스터 사용자를 재설정하려면 AWS Management Console을 사용합니다. Amazon RDS for SQL Server 마스터 사용자의 db_owner 역할 멤버십 재설정을(를) 참조하세요.
액세스 거부됨	백업 또는 복원 프로세스는 백업 파일에 액세스할 수 없습니다. 이것의 원인이 되는 문제는 일반적으로 다음과 같습니다. • 올바르지 않은 버킷 참조. 올바르지 않은 형식을 사용하여 버킷 참조. ARN을 사용하지 않고 파일 이름 참조. •

문제	문제 해결 제안
	버킷 파일에 올바르지 않은 권한. 예컨대 지금 액세스를 시도하는 계정과 다른 계정에서 생성한 경우 올바른 권한을 추가하십시오. • 올바르지 않거나 완전하지 않은 IAM 정책. IAM 역할에는 올바른 버전을 비롯하여 필요한 요소가 모두 포함되어야 합니다. 이에 대해서는 기본 백업 및 복원 기능을 사용하여 SQL Server 데이터베이스 가져오기 및 내보내기에 강조 표시되어 있습니다.
압축 기능이 있는 백업 데이터베이스는 <edition_name> Edition에서 지원되지 않습니다.	백업 파일 압축은 Microsoft SQL Server Enterprise Edition 및 Standard Edition에서만 지원됩니다. 자세한 내용은 백업 파일 압축 섹션을 참조하세요.
키 <ARN>이 존재하지 않습니다.	암호화된 백업을 복원하려고 시도했으나 유효한 암호화 키를 제공하지 않았습니다. 암호화 키를 확인한 후 다시 시도하십시오. 자세한 내용은 데이터베이스 복원 섹션을 참조하세요.
올바른 유형 및 덮어쓰기 속성을 사용하여 작업을 다시 실행하세요(Please reissue task with correct type and overwrite property)	데이터베이스 백업을 시도할 때 이미 존재하는 파일 이름을 입력하고 덮어쓰기 속성을 false로 설정하면 저장 작업이 실패합니다. 이 오류를 해결하려면 존재하지 않는 파일 이름을 입력하거나 덮어쓰기 속성을 true로 설정하십시오. 자세한 내용은 데이터베이스 백업 섹션을 참조하세요. 데이터베이스를 복원할 때 잘못하여 `rds_backup_database` 저장 프로시저를 호출했을 수도 있습니다. 이 경우 대신 `rds_restore_database` 저장 프로시저를 호출하십시오. 자세한 내용은 데이터베이스 복원 섹션을 참조하세요. 데이터베이스를 복원할 때 `rds_restore_database` 저장 프로시저를 호출한 경우, 유효한 백업 파일 이름을 입력했는지 확인하십시오. 자세한 내용은 기본 백업 및 복원 사용 섹션을 참조하세요.

문제	문제 해결 제안
RDS 인스턴스와 동일한 리전에 있는 버킷을 지정하세요.	Amazon RDS DB 인스턴스와 다른 AWS 리전에서는 Amazon S3 버킷으로 백업하거나 복원할 수 없습니다. Amazon S3 복제를 사용하여 백업 파일을 올바른 AWS 리전에 복사할 수 있습니다. 자세한 내용은 Amazon S3 설명서의 리전 간 복제 단원을 참조하십시오.
지정된 버킷이 존재하지 않습니다	버킷 및 파일에 대해 올바른 ARN을 올바른 형식으로 입력했는지 확인하십시오. 자세한 내용은 기본 백업 및 복원 사용 섹션을 참조하세요.
사용자 <ARN>이 리소스 <ARN>에 대해 수행할 권한이 없습니다	암호화된 작업을 요청했으나 올바른 AWS KMS 권한을 제공하지 않았습니다. 올바른 권한이 있는지 확인하거나 올바른 권한을 추가하십시오. 자세한 내용은 기본 백업 및 복원 설정 섹션을 참조하세요.
복원 작업은 10개 이상의 백업 파일에서 복원할 수 없습니다. 일치하는 파일 수를 줄이고 다시 시도하세요	복원하려는 파일 수를 줄입니다. 필요한 경우 개별 파일을 더 크게 만들 수 있습니다.
데이터베이스의 '*database_name*'이 이미 있습니다. 대/소문자 또는 액센트만 다른 두 데이터베이스는 허용되지 않습니다. 다른 데이터베이스 이름을 선택합니다.	기존 데이터베이스와 이름이 같은 데이터베이스는 복원할 수 없습니다. 데이터베이스 이름은 고유합니다.

다른 방법으로 SQL Server 데이터 가져오기 및 내보내기

다음으로, 스냅샷을 사용하여 Microsoft SQL Server 데이터를 Amazon RDS로 가져오는 방법에 대한 정보를 찾을 수 있습니다. 스냅샷을 사용하여 SQL Server를 실행 중인 RDS DB 인스턴스에서 데이터를 내보내는 방법에 대한 정보도 찾을 수 있습니다.

이 방법이 가능한 시나리오라면 기본 백업 및 복원 기능을 사용하여 Amazon RDS 안팎으로 데이터를 옮기는 것이 더 쉽습니다. 자세한 내용은 기본 백업 및 복원 기능을 사용하여 SQL Server 데이터베이스 가져오기 및 내보내기 섹션을 참조하세요.

> **Note**
>
> Amazon RDS for Microsoft SQL Server는 `msdb` 데이터베이스로 데이터 가져오기를 지원하지 않습니다.

스냅샷을 사용하여 RDS for SQL Server로 데이터 가져오기

스냅샷을 사용하여 SQL Server DB 인스턴스로 데이터를 가져오려면

1. DB 인스턴스를 만듭니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
2. 애플리케이션이 대상 DB 인스턴스에 액세스하지 못하게 차단합니다.

   데이터를 가져오는 동안 DB 인스턴스에 대한 액세스를 금지하면 데이터 전송이 더 빨라집니다. 또한, 다른 애플리케이션이 DB 인스턴스에 동시에 쓸 수 없는 경우 데이터가 로드되는 동안 충돌에 대해 걱정할 필요가 없습니다. 뭔가 잘못되어 이전의 데이터베이스 스냅샷으로 롤백해야 할 때 잃게 되는 유일한 변경 내용은 가져온 데이터입니다. 문제를 해결한 후 이 데이터를 다시 가져올 수 있습니다.

   DB 인스턴스 액세스 제어에 대한 자세한 정보는 보안 그룹을 통한 액세스 제어을(를) 참조하십시오.
3. 대상 데이터베이스의 스냅샷을 만듭니다.

   대상 데이터베이스가 이미 데이터로 채워져 있으면 데이터베이스의 스냅샷을 만든 후에 데이터를 가져오는 것이 좋습니다. 데이터 가져오기에 문제가 있거나 변경 내용을 취소하려는 경우 스냅샷을 사용하여 데이터베이스를 이전 상태로 복원할 수 있습니다. 데이터베이스 스냅샷에 대한 자세한 정보는 Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성을(를) 참조하십시오.

> **Note**
>
> 데이터베이스 스냅샷을 만들 때, 백업이 진행되는 동안 데이터베이스에 대한 I/O 작업이 잠시(밀리초) 중단됩니다.

4. 대상 데이터베이스에서 자동 백업을 비활성화합니다.

   자동 백업이 비활성화되어 있을 때는 Amazon RDS가 트랜잭션을 로그에 기록하지 않기 때문에 대상 DB 인스턴스에서 자동 백업을 비활성화하면 데이터를 가져오는 동안 성능이 향상됩니다. 하지만 몇 가지 고려해야 할 사항이 있습니다. 특정 시점으로 복구를 수행하려면 자동 백업이 필요합니다. 따라서 데이터를 가져오는 동안 데이터베이스를 특정 시점으로 복원할 수 없습니다. 또한 보관을 선택하지 않을 경우 DB 인스턴스에서 생성된 자동 백업 데이터가 지워집니다.

   자동 백업을 유지하도록 선택하면 실수로 데이터가 삭제되는 것을 방지할 수 있습니다. 또한 Amazon RDS는 데이터베이스 인스턴스 속성을 각 자동 백업과 함께 저장하여 쉽게 복구할 수 있도록 지원합니다. 이 옵션을 사용하면 데이터베이스 인스턴스가 삭제된 이후에도 백업 보존 기간 중 특정 시점으로 데이터베이스 인스턴스를 복원할 수 있습니다. 자동 백업은 활성 데이터베이스 인스턴스에서와 마찬가지로 지정된 백업 기간이 종료되면 자동으로 삭제됩니다.

   또한 이전 스냅샷을 사용하여 데이터베이스를 복구할 수 있고, 지금까지 만든 스냅샷은 전부 그대로 사용할 수 있습니다. 자동 백업에 대한 자세한 내용은 백업 소개 단원을 참조하십시오.
5. 해당될 경우 외래 키 제약 조건을 비활성화합니다.

   외래 키 제약 조건을 비활성화할 필요가 있는 경우 다음 스크립트로 비활성화할 수 있습니다.

```
--Disable foreign keys on all tables
    DECLARE @table_name SYSNAME;
    DECLARE @cmd NVARCHAR(MAX);
    DECLARE table_cursor CURSOR FOR SELECT name FROM sys.tables;

    OPEN table_cursor;
    FETCH NEXT FROM table_cursor INTO @table_name;

    WHILE @@FETCH_STATUS = 0 BEGIN
      SELECT @cmd = 'ALTER TABLE '+QUOTENAME(@table_name)+' NOCHECK CONSTRAINT
 ALL';
      EXEC (@cmd);
      FETCH NEXT FROM table_cursor INTO @table_name;
    END
```

```
    CLOSE table_cursor;
    DEALLOCATE table_cursor;

    GO
```

6. 해당될 경우 인덱스를 삭제합니다.
7. 트리거가 있는 경우 비활성화합니다.

   트리거를 비활성화해야 하는 경우 다음 스크립트로 비활성화할 수 있습니다.

```
--Disable triggers on all tables
    DECLARE @enable BIT = 0;
    DECLARE @trigger SYSNAME;
    DECLARE @table SYSNAME;
    DECLARE @cmd NVARCHAR(MAX);
    DECLARE trigger_cursor CURSOR FOR SELECT trigger_object.name trigger_name,
     table_object.name table_name
    FROM sysobjects trigger_object
    JOIN sysobjects table_object ON trigger_object.parent_obj = table_object.id
    WHERE trigger_object.type = 'TR';

    OPEN trigger_cursor;
    FETCH NEXT FROM trigger_cursor INTO @trigger, @table;

    WHILE @@FETCH_STATUS = 0 BEGIN
      IF @enable = 1
         SET @cmd = 'ENABLE ';
      ELSE
         SET @cmd = 'DISABLE ';

      SET @cmd = @cmd + ' TRIGGER dbo.'+QUOTENAME(@trigger)+' ON
 dbo.'+QUOTENAME(@table)+' ';
      EXEC (@cmd);
      FETCH NEXT FROM trigger_cursor INTO @trigger, @table;
    END

    CLOSE trigger_cursor;
    DEALLOCATE trigger_cursor;

    GO
```

8. 원본 SQL Server 인스턴스에서 대상 DB 인스턴스로 가져오려는 로그인을 쿼리합니다.

SQL Server는 `master` 데이터베이스에 로그인과 암호를 저장합니다. Amazon RDS는 `master` 데이터베이스에 액세스 권한을 부여하지 않기 때문에, 로그인 정보와 암호를 대상 DB 인스턴스로 직접 가져올 수는 없습니다. 대신, 원본 SQL Server 인스턴스에서 `master` 데이터베이스를 쿼리하여 DDL(데이터 정의 언어) 파일을 생성해야 합니다. 이 파일에는 대상 DB 인스턴스에 추가하려는 모든 로그인 정보와 암호가 포함되어야 합니다. 이 파일에는 전송하려는 역할 멤버십 및 권한도 포함되어야 합니다.

`master` 데이터베이스 쿼리에 대한 자세한 정보는 Microsoft 기술 자료의 SQL Server 인스턴스 간에 로그인 및 암호 전송을 참조하세요.

스크립트의 출력은 대상 DB 인스턴스에서 실행할 수 있는 또 다른 스크립트입니다. 기술 자료 문서에 있는 스크립트에 다음 코드가 있을 경우

```
p.type IN
```

`p.type`이 나타나는 곳에서는 모두 다음 코드를 대신 사용합니다.

```
p.type = 'S'
```

9. 데이터 가져오기의 방법을 사용하여 데이터를 가져옵니다.
10. 애플리케이션에 대상 DB 인스턴스에 대한 액세스 권한을 부여합니다.

    데이터 가져오기가 완료되면 가져오기 중에 차단한 애플리케이션에 DB 인스턴스 액세스 권한을 부여할 수 있습니다. DB 인스턴스 액세스 제어에 대한 자세한 정보는 보안 그룹을 통한 액세스 제어을(를) 참조하십시오.
11. 대상 DB 인스턴스에서 자동 백업을 활성화합니다.

    자동 백업에 대한 자세한 내용은 백업 소개 단원을 참조하십시오.
12. 외래 키 제약 조건을 활성화합니다.

    앞서 외래 키 제약 조건을 비활성화한 경우 다음 스크립트로 활성화할 수 있습니다.

```
--Enable foreign keys on all tables
    DECLARE @table_name SYSNAME;
    DECLARE @cmd NVARCHAR(MAX);
    DECLARE table_cursor CURSOR FOR SELECT name FROM sys.tables;

    OPEN table_cursor;
    FETCH NEXT FROM table_cursor INTO @table_name;
```

```
    WHILE @@FETCH_STATUS = 0 BEGIN
      SELECT @cmd = 'ALTER TABLE '+QUOTENAME(@table_name)+' CHECK CONSTRAINT ALL';
      EXEC (@cmd);
      FETCH NEXT FROM table_cursor INTO @table_name;
    END

    CLOSE table_cursor;
    DEALLOCATE table_cursor;
```

13. 인덱스가 있는 경우 인덱스를 활성화합니다.
14. 트리거가 있는 경우 트리거를 활성화합니다.

    앞서 트리거를 비활성화했다면 다음 스크립트로 활성화할 수 있습니다.

```
--Enable triggers on all tables
    DECLARE @enable BIT = 1;
    DECLARE @trigger SYSNAME;
    DECLARE @table SYSNAME;
    DECLARE @cmd NVARCHAR(MAX);
    DECLARE trigger_cursor CURSOR FOR SELECT trigger_object.name trigger_name,
     table_object.name table_name
    FROM sysobjects trigger_object
    JOIN sysobjects table_object ON trigger_object.parent_obj = table_object.id
    WHERE trigger_object.type = 'TR';

    OPEN trigger_cursor;
    FETCH NEXT FROM trigger_cursor INTO @trigger, @table;

    WHILE @@FETCH_STATUS = 0 BEGIN
      IF @enable = 1
         SET @cmd = 'ENABLE ';
      ELSE
         SET @cmd = 'DISABLE ';

      SET @cmd = @cmd + ' TRIGGER dbo.'+QUOTENAME(@trigger)+' ON
 dbo.'+QUOTENAME(@table)+' ';
      EXEC (@cmd);
      FETCH NEXT FROM trigger_cursor INTO @trigger, @table;
    END

    CLOSE trigger_cursor;
    DEALLOCATE trigger_cursor;
```

데이터 가져오기

Microsoft SQL Server Management Studio는 Express Edition을 제외한 모든 Microsoft SQL Server 버전에 포함되는 그래픽 SQL Server 클라이언트입니다. SQL Server Management Studio Express는 Microsoft에서 무료 다운로드로 사용할 수 있습니다. 다운로드에 관해서는 Microsoft 웹 사이트을(를) 참조하십시오.

> **Note**
>
> SQL Server Management Studio는 Windows 기반 애플리케이션으로만 사용할 수 있습니다.

SQL Server Management Studio에는 SQL Server DB 인스턴스로 데이터를 가져오는 데 유용한 다음 도구가 포함됩니다.

- 스크립트 생성 및 게시 마법사
- 가져오기 및 내보내기 마법사
- 대량 복사

스크립트 생성 및 게시 마법사

스크립트 생성 및 게시 마법사는 데이터베이스의 스키마, 데이터 자체 또는 두 가지를 모두 포함한 스크립트를 만듭니다. 로컬 SQL Server 배포에서 데이터베이스에 대한 스크립트를 생성할 수 있습니다. 그런 다음, 포함된 정보를 Amazon RDS DB 인스턴스로 전송하는 스크립트를 실행할 수 있습니다.

> **Note**
>
> 1GiB 이상의 데이터베이스인 경우, 데이터베이스 스키마만 스크립팅하는 것이 더 효율적입니다. 그런 다음 가져오기 및 내보내기 마법사 또는 SQL Server의 대량 복사 기능을 사용하여 데이터를 전송합니다.

스크립트 생성 및 게시 마법사에 대한 자세한 정보는 Microsoft SQL Server 문서을(를) 참조하십시오.

마법사에서는 특히 [스크립팅 옵션 설정] 페이지의 고급 옵션에 주의하여 스크립트에 포함하려는 모든 것이 선택되어 있는지 확인하십시오. 예를 들어 데이터베이스 트리거는 기본적으로 스크립트에 포함되지 않습니다.

스크립트가 생성되고 저장되면 SQL Server Management Studio를 사용하여 DB 인스턴스에 연결한 후 스크립트를 실행할 수 있습니다.

가져오기 및 내보내기 마법사

가져오기 및 내보내기 마법사는 특별한 통합 서비스 패키지를 만드는데, 이 패키지를 사용하여 로컬 SQL Server 데이터베이스에서 대상 DB 인스턴스로 데이터를 복사할 수 있습니다. 이 마법사에서는 대상 DB 인스턴스로 어떤 테이블, 심지어 테이블 내에 있는 어떤 튜플을 복사할지 필터링할 수 있습니다.

> **Note**
>
> 가져오기 및 내보내기 마법사는 큰 데이터 집합에 매우 효과적이지만, 로컬 배포 위치에서 원격으로 데이터를 내보내는 가장 빠른 방법은 아닐 수도 있습니다. 훨씬 더 빠른 방법을 원한다면, SQL Server 대량 복사 기능을 고려하십시오.

가져오기 및 내보내기 마법사에 대한 자세한 내용은 Microsoft SQL Server 문서를 참조하십시오.

이 마법사의 [대상 선택] 페이지에서 다음을 수행합니다.

- [서버 이름]에 DB 인스턴스의 엔드포인트 이름을 입력합니다.
- 서버 인증 모드를 위해 [SQL Server 인증 사용]을 선택합니다.
- [사용자 이름] 및 [암호]에 DB 인스턴스용으로 만든 마스터 사용자의 자격 증명을 입력합니다.

대량 복사

SQL Server 대량 복사 기능은 원본 데이터베이스에서 DB 인스턴스로 데이터를 복사하는 효율적인 수단입니다. 대량 복사는 ASCII 파일과 같은 데이터 파일에 사용자가 지정하는 데이터를 쓰는 기능입니다. 대량 복사를 다시 실행하여 파일의 내용을 대상 DB 인스턴스에 쓸 수 있습니다.

이 섹션에서는 모든 SQL Server 버전에 포함되어 있는 bcp 유틸리티를 사용합니다. 대량 가져오기 및 내보내기 작업에 대한 자세한 정보는 Microsoft SQL Server 문서를 참조하십시오.

> **Note**
>
> 대량 복사 기능을 사용하기 전에, 우선 데이터베이스 스키마를 대상 DB 인스턴스로 가져와야 합니다. 이 주제의 앞 부분에서 설명한 스크립트 생성 및 게시 마법사는 이 목적으로 사용하기에 훌륭한 도구입니다.

다음 명령은 로컬 SQL Server 인스턴스에 연결합니다. 그러면 기존 SQL Server 배포의 C:\ 루트 디렉터리에 지정된 테이블의 탭으로 구분된 파일이 생성됩니다. 테이블은 정규화된 이름으로 지정되고, 텍스트 파일은 복사되는 테이블과 이름이 같습니다.

```
bcp dbname.schema_name.table_name out C:\table_name.txt -n -S localhost -U username -
P password -b 10000
```

앞에 나온 코드에는 다음 옵션이 포함됩니다.

- `-n`은 대량 복사에서 복사되는 데이터의 기본 데이터 형식을 사용할 것임을 지정합니다.
- `-S`는 bcp 유틸리티가 연결할 SQL Server 인스턴스를 지정합니다.
- `-U`는 SQL Server 인스턴스에 로그인할 계정의 사용자 이름을 지정합니다.
- `-P`는 로 지정된 사용자의 암호를 지정합니다.`-U`
- `-b`는 가져온 데이터의 배치당 행 수를 지정합니다.

> **Note**
>
> 가져오기 작업에 중요한 다른 파라미터가 있을 수 있습니다. 예를 들어, 자격 증명 값에 속한 `-E` 파라미터가 필요할 수 있습니다. 자세한 내용은 Microsoft SQL Server 문서에서 bcp 유틸리티 관련 명령줄 구문에 대한 상세 설명을 참조하십시오.

예를 들어 기본 스키마 `store`를 사용하는 `dbo`라는 이름의 데이터베이스에 `customers`라는 이름의 테이블이 있다고 가정합시다. 암호가 `admin`인 사용자 계정 `insecure`이 `customers` 테이블에서 10,000개의 행을 `customers.txt`라는 파일로 복사합니다.

```
bcp store.dbo.customers out C:\customers.txt -n -S localhost -U admin -P insecure -b
 10000
```

데이터 파일을 생성한 후 비슷한 명령을 사용하여 DB 인스턴스에 데이터를 업로드할 수 있습니다. 사전에 대상 DB 인스턴스에서 데이터베이스 및 스키마를 생성하십시오. 출력 파일을 지정하는 `in` 대신 입력 파일을 지정하는 `out` 인수를 사용합니다. 로컬 SQL Server 인스턴스를 지정하기 위해 localhost를 사용하는 대신, DB 인스턴스의 엔드포인트를 지정합니다. 1433 이외의 포트를 사용하는 경우 그 포트도 지정합니다. 사용자 이름과 암호는 DB 인스턴스에 대한 마스터 사용자와 암호가 됩니다. 구문은 다음과 같습니다.

```
bcp dbname.schema_name.table_name
```

```
     in C:\table_name.txt -n -S endpoint,port -U master_user_name -
P master_user_password -b 10000
```

이전 예제를 진행하기 위해, 마스터 사용자 이름이 admin이고 암호가 insecure라고 가정합니다. DB 인스턴스의 엔드포인트는 rds.ckz2kqd4qsn1.us-east-1.rds.amazonaws.com이고, 포트 4080을 사용합니다. 명령은 다음과 같습니다.

```
bcp store.dbo.customers in C:\customers.txt -n -S rds.ckz2kqd4qsn1.us-
east-1.rds.amazonaws.com,4080 -U admin -P insecure -b 10000
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

RDS for SQL Server에서 데이터 내보내기

다음 옵션 중 하나를 선택하여 RDS for SQL Server DB 인스턴스에서 데이터를 내보낼 수 있습니다.

- 전체 백업 파일(.bak)을 사용한 기본 데이터베이스 백업 – .bak 파일을 사용하여 데이터베이스를 백업하는 과정은 철저히 최적화되어 있으며, 대개의 경우 데이터를 가장 빨리 내보내는 방법입니다. 자세한 내용은 기본 백업 및 복원 기능을 사용하여 SQL Server 데이터베이스 가져오기 및 내보내기 섹션을 참조하세요.
- SQL Server 가져오기 및 내보내기 마법사 – 자세한 내용은 SQL Server 가져오기 및 내보내기 마법사 섹션을 참조하세요.
- SQL Server 스크립트 생성 및 게시 마법사와 bcp 유틸리티 – 자세한 내용은 SQL Server 스크립트 생성 및 게시 마법사와 bcp 유틸리티 섹션을 참조하세요.

SQL Server 가져오기 및 내보내기 마법사

SQL Server 가져오기 및 내보내기 마법사를 사용하여 RDS for SQL Server DB 인스턴스에서 다른 데이터 스토어로 하나 이상의 테이블, 보기 또는 쿼리를 복사할 수 있습니다. 대상 데이터 스토어가 SQL Server가 아닌 경우 이 방법이 가장 좋습니다. 자세한 내용은 SQL Server 문서의 SQL Server 가져오기 및 내보내기 마법사를 참조하십시오.

SQL Server 가져오기 및 내보내기 마법사는 Microsoft SQL Server Management Studio의 일부로 제공됩니다. 이 그래픽 SQL Server 클라이언트는 Express Edition을 제외한 모든 Microsoft SQL Server 버

전에 포함됩니다. SQL Server Management Studio는 Windows 기반 애플리케이션으로만 사용할 수 있습니다. SQL Server Management Studio Express는 Microsoft에서 무료 다운로드로 사용할 수 있습니다. 다운로드에 관해서는 Microsoft 웹 사이트을(를) 참조하십시오.

SQL Server 가져오기 및 내보내기 마법사를 사용하여 데이터를 내보내려면

1. SQL Server Management Studio에서 RDS for SQL Server DB 인스턴스에 연결합니다. 이 작업을 수행하는 자세한 방법은 Microsoft SQL Server DB 인스턴스에 연결 단원을 참조하십시오.
2. [개체 탐색기]에서 [데이터베이스]를 확장하고, 원본 데이터베이스에 대한 컨텍스트 메뉴를 마우스 오른쪽 버튼을 클릭하여 열고, [작업]을 선택한 다음, [데이터 내보내기]를 선택합니다. 그러면 마법사가 표시됩니다.
3. [데이터 소스 선택] 페이지에서 다음을 수행합니다.
   a. Data source(데이터 원본)에 **SQL Server Native Client 11.0**을 선택합니다.
   b. [서버 이름(Server name)] 상자에 RDS for SQL Server DB 인스턴스의 엔드포인트가 표시되는지 확인합니다.
   c. [SQL 서버 인증 사용]을 선택합니다. [사용자 이름(User name)] 및 [암호(Password)]에서 DB 인스턴스의 마스터 사용자 이름과 암호를 입력합니다.
   d. [데이터베이스] 상자에 데이터를 내보내려는 원본 데이터베이스가 표시되는지 확인합니다.
   e. 다음을 선택합니다.
4. [대상 선택] 페이지에서 다음을 수행합니다.
   a. Destination(대상)에 **SQL Server Native Client 11.0**을 선택합니다.

      > **Note**
      >
      > 다른 대상 데이터 원본을 사용할 수 있습니다. 여기에는 .NET Framework 데이터 공급자, OLE DB 공급자, SQL Server Native Client 공급자, ADO.NET 공급자, Microsoft Office Excel, Microsoft Office Access 및 Flat File 소스 등이 포함됩니다. 이러한 데이터 원본 중 하나를 대상으로 선택한 경우 나머지 4단계를 건너뛰십시오. 다음에 제공할 연결 정보에 대한 자세한 내용은 SQL Server 문서의 대상 선택을 참조하십시오.

   b. [서버 이름]에 대상 SQL Server DB 인스턴스의 서버 이름을 입력합니다.
   c. 알맞은 인증 유형을 선택합니다. 필요한 경우 사용자 이름과 암호를 입력합니다.
   d. [데이터베이스]에서 대상 데이터베이스의 이름을 선택하거나, [새로 만들기]를 선택하여 내보낸 데이터를 포함하는 새 데이터베이스를 만듭니다.

새로 만들기를 선택하는 경우, 제공할 데이터베이스 정보에 대한 자세한 내용은 SQL Server 문서의 데이터베이스 생성을 참조하십시오.

e. 다음을 선택합니다.

5. [테이블 복사 또는 쿼리] 페이지에서 [하나 이상의 테이블 또는 뷰에서 데이터 복사] 또는 [전송 데이터를 지정할 쿼리 작성]을 선택합니다. 다음을 선택합니다.
6. [전송 데이터를 지정할 쿼리 작성]을 선택한 경우 [원본 쿼리 지정] 페이지가 표시됩니다. SQL 쿼리를 입력하거나 붙여 넣은 다음, [구문 분석]을 선택하여 확인합니다. 쿼리 유효성 검사가 끝나면 [다음]을 선택합니다.
7. [원본 테이블 및 뷰 선택] 페이지에서 다음을 수행합니다.

   a. 내보내려는 테이블과 뷰를 선택하거나 사용자가 입력한 쿼리가 선택되어 있는지 확인합니다.

   b. [매핑 편집]을 선택하고 데이터베이스 및 열 매핑 정보를 지정합니다. 자세한 내용은 SQL Server 문서의 열 매핑을 참조하십시오.

   c. (선택 사항) 내보낼 데이터의 미리 보기를 확인하려면 테이블, 뷰 또는 쿼리를 선택한 후 [미리 보기]를 선택합니다.

   d. 다음을 선택합니다.

8. [패키지 실행] 페이지에서 [즉시 실행]이 선택되어 있는지 확인합니다. 다음을 선택합니다.
9. [마법사 완료] 페이지에서 데이터 내보내기 세부 정보가 예상한 대로인지 확인합니다. [마침]을 클릭합니다.
10. [실행이 성공했습니다.] 페이지에서 [닫기]를 선택합니다.

SQL Server 스크립트 생성 및 게시 마법사와 bcp 유틸리티

SQL Server 스크립트 생성 및 게시 마법사를 사용하여 데이터베이스 전체 또는 선택한 개체만을 위한 스크립트를 작성할 수 있습니다. 대상 SQL Server DB 인스턴스에서 이런 스크립트를 실행하여 스크립팅된 객체를 다시 만들 수 있습니다. 그런 다음, bcp 유틸리티를 사용하여 선택한 객체에 대한 데이터를 대상 DB 인스턴스로 대량으로 내보낼 수 있습니다. (테이블 이외의 객체를 포함한) 전체 데이터베이스를 이동하거나 두 SQL Server DB 인스턴스 사이에서 대량의 데이터를 이동하려는 경우에 이 마법사를 선택하는 것이 최선입니다. bcp 명령줄 구문에 대한 전체 설명은 Microsoft SQL Server 문서의 bcp 유틸리티를 참조하십시오.

SQL Server 스크립트 생성 및 게시 마법사는 Microsoft SQL Server Management Studio의 일부로 제공됩니다. 이 그래픽 SQL Server 클라이언트는 Express Edition을 제외한 모든 Microsoft SQL Server 버전에 포함됩니다. SQL Server Management Studio는 Windows 기반 애플리케이션으로만 사용할 수

있습니다. SQL Server Management Studio Express는 Microsoft에서 무료 다운로드로 사용할 수 있습니다.

SQL Server 스크립트 생성 및 게시 마법사와 bcp 유틸리티를 사용하여 데이터를 내보내려면

1. SQL Server Management Studio에서 RDS for SQL Server DB 인스턴스에 연결합니다. 이 작업을 수행하는 자세한 방법은 Microsoft SQL Server DB 인스턴스에 연결 단원을 참조하십시오.
2. [개체 탐색기]에서 [데이터베이스] 노드를 확장하고 스크립팅하려는 데이터베이스를 선택합니다.
3. 스크립트 파일을 만들려면 SQL Server 문서의 스크립트 생성 및 게시 마법사에 설명되어 있는 지침을 따르십시오.
4. SQL Server Management Studio에서 대상 SQL Server DB 인스턴스에 연결합니다.
5. 객체 탐색기(Object Explorer)에서 대상 SQL Server DB 인스턴스를 선택한 상태에서 파일(File) 메뉴에서 열기(Open)를 선택하고, 파일(File)을 선택한 후 스크립트 파일을 엽니다.
6. 전체 데이터베이스를 스크립팅한 경우 스크립트에서 CREATE DATABASE 문을 검토하십시오. 데이터베이스가 원하는 위치 및 파라미터로 작성되고 있는지 확인하십시오. 자세한 정보는 SQL Server 문서의 CREATE DATABASE를 참조하십시오.
7. 스크립트에서 데이터베이스 사용자를 만들 경우 대상 DB 인스턴스에 해당 사용자에 대한 서버 로그인이 존재하는지 확인합니다. 존재하지 않을 경우 해당 사용자에 대한 로그인을 만듭니다. 그렇지 않으면, 데이터베이스 사용자를 만들기 위해 스크립팅된 명령이 실패합니다. 자세한 정보는 SQL Server 문서의 로그인 생성을 참조하십시오.
8. SQL 편집기 메뉴에서 !Execute를 선택하여 스크립트 파일을 실행하고 데이터베이스 개체를 생성합니다. 스크립트가 완료되면 모든 데이터베이스 개체가 예상한 대로 존재하는지 확인합니다.
9. bcp 유틸리티를 사용하여 RDS for SQL Server DB 인스턴스에서 파일로 데이터를 내보냅니다. 명령 프롬프트를 열고 다음 명령을 입력합니다.

```
bcp database_name.schema_name.table_name out data_file -n -S aws_rds_sql_endpoint -
U username -P password
```

앞에 나온 코드에는 다음 옵션이 포함됩니다.

- table_name은 대상 데이터베이스에서 다시 만들어 이제 데이터를 채우려는 테이블 중 하나의 이름입니다.
- data_file은 만들 데이터 파일의 전체 경로와 이름입니다.
- -n은 대량 복사에서 복사되는 데이터의 기본 데이터 형식을 사용할 것임을 지정합니다.
- -S는 데이터를 내보낼 원본 SQL Server DB 인스턴스를 지정합니다.

- -U는 SQL Server DB 인스턴스에 연결할 때 사용할 사용자 이름을 지정합니다.
- -P는 로 지정된 사용자의 암호를 지정합니다.-U

다음은 명령의 예시입니다.

```
bcp world.dbo.city out C:\Users\JohnDoe\city.dat -n -S sql-jdoe.1234abcd.us-
west-2.rds.amazonaws.com,1433 -U JohnDoe -P ClearTextPassword
```

내보낼 모든 테이블에 대한 데이터 파일을 확보할 때까지 이 단계를 반복하십시오.

10. SQL Server 문서의 대량 데이터 가져오기 준비에 설명되어 있는 지침에 따라 대량 데이터 가져오기를 위해 대상 DB 인스턴스를 준비하십시오.
11. SQL Server 문서의 대량 가져오기 및 내보내기 작업 정보에 설명되어 있는 성능 및 기타 문제를 고려한 후 사용할 대량 가져오기 방법을 결정하십시오.
12. bcp 유틸리티를 사용하여 생성한 데이터 파일에서 데이터를 대량으로 가져옵니다. 그러려면 11 단계에서 결정한 바에 따라, SQL Server 문서의 bcp 유틸리티를 사용하여 대량 데이터 가져오기 및 내보내기 또는 BULK INSERT 또는 OPENROWSET(BULK...)를 사용하여 대량 데이터 가져오기에 설명되어 있는 지침에 따르십시오.

Linux의 BCP 유틸리티를 사용하여 데이터 가져오기 및 내보내기

Bulk Copy Program(BCP) 유틸리티는 RDS for SQL Server DB 인스턴스와 데이터 파일 간에 대량의 데이터를 전송하는 효율적인 방법을 제공합니다. Linux 환경의 BCP를 사용하여 일괄 데이터 작업을 수행할 수 있으므로 데이터 마이그레이션, ETL 프로세스 및 정기적인 데이터 전송에 유용합니다.

BCP는 파일에서 SQL Server 테이블로 데이터를 가져오고 SQL Server 테이블에서 파일로 데이터를 내보내는 것을 모두 지원합니다. 이는 구분된 텍스트 파일을 포함한 다양한 형식의 정형 데이터를 전송하는 데 특히 효과적입니다.

사전 조건

Linux에서 RDS for SQL Server DB 인스턴스와 함께 BCP를 사용하기 전에 다음 사항이 있는지 확인합니다.

- RDS for SQL Server DB 인스턴스에 네트워크로 연결된 Linux 환경
- 다음을 포함하여 Linux 시스템에 설치된 Microsoft SQL Server 명령줄 도구:
  - sqlcmd - SQL Server 명령줄 쿼리 도구
  - bcp - Bulk Copy Program 유틸리티
- RDS for SQL Server DB 인스턴스에 유효한 자격 증명
- SQL Server 포트(일반적으로 1433)에서 연결을 허용하도록 보안 그룹을 통해 구성된 네트워크 액세스
- 수행하려는 작업에 대한 적절한 데이터베이스 권한

Linux에 SQL Server 명령줄 도구 설치

Linux의 BCP를 사용하려면 Microsoft SQL Server 명령줄 도구를 설치해야 합니다. 특정 Linux 배포에 대한 자세한 설치 지침은 다음 Microsoft 설명서를 참조하세요.

- Install sqlcmd and bcp the SQL Server command-line tools on Linux
- bcp utility - BCP 유틸리티에 대한 전체 참조 문서

설치 후 다음을 실행하여 PATH에서 도구를 사용할 수 있는지 확인합니다.

```
bcp -v
sqlcmd -?
```

RDS for SQL Server에서 데이터 내보내기

BCP를 사용하여 RDS for SQL Server DB 인스턴스의 데이터를 Linux 시스템의 파일로 내보낼 수 있습니다. 이는 백업 생성, 데이터 분석 또는 마이그레이션을 위한 데이터 준비에 유용합니다.

기본 내보내기 구문

BCP를 사용하여 데이터를 내보내는 기본 구문은 다음과 같습니다.

```
bcp database.schema.table out output_file -S server_name -U username -P password
 [options]
```

위치:

- `database.schema.table` - 정규화된 테이블 이름
- `output_file` - 출력 파일의 경로 및 이름
- `server_name` - RDS for SQL Server 엔드포인트
- `username` - 데이터베이스 사용자 이름
- `password` - 데이터베이스 암호

내보내기 예시

다음 예시에서는 `sales` 데이터베이스의 `customers` 테이블에서 데이터를 내보냅니다.

```
bcp sales.dbo.customers out /home/user/customers.txt \
    -S mydb.cluster-abc123.us-east-1.rds.amazonaws.com \
    -U admin \
    -P mypassword \
    -c \
    -t "|" \
    -r "\n"
```

이 명령은 다음을 수행합니다.

- `customers` 테이블에서 데이터 내보내기
- 출력을 `/home/user/customers.txt`에 저장
- 문자 형식(`-c`) 사용
- 파이프(|)를 필드 구분 기호(`-t "|"`)로 사용
- 줄 바꿈을 행 구분 기호(`-r "\n"`)로 사용

RDS for SQL Server로 데이터 가져오기

BCP를 사용하여 Linux 시스템의 파일에서 RDS for SQL Server DB 인스턴스로 데이터를 가져올 수 있습니다. 이는 데이터 마이그레이션, 테스트 데이터 로드 또는 정기적인 데이터 업데이트에 유용합니다.

기본 가져오기 구문

BCP를 사용하여 데이터를 가져오는 기본 구문은 다음과 같습니다.

```
bcp database.schema.table in input_file -S server_name -U username -P password
 [options]
```

위치:

- `database.schema.table` - 정규화된 대상 테이블 이름
- `input_file` - 입력 파일의 경로 및 이름
- `server_name` - RDS for SQL Server 엔드포인트
- `username` - 데이터베이스 사용자 이름
- `password` - 데이터베이스 암호

가져오기 예시

다음 예시에서는 파일에서 `customers` 테이블로 데이터를 가져옵니다.

```
bcp sales.dbo.customers in /home/user/customers.txt \
    -S mydb.cluster-abc123.us-east-1.rds.amazonaws.com \
    -U admin \
    -P mypassword \
    -c \
    -t "|" \
    -r "\n" \
    -b 1000
```

이 명령은 다음을 수행합니다.

- 데이터를 `customers` 테이블로 가져오기
- `/home/user/customers.txt`에서 데이터를 읽습니다.
- 문자 형식(`-c`) 사용

- 파이프(|)를 필드 구분 기호(`-t "|"`)로 사용
- 줄 바꿈을 행 구분 기호(`-r "\n"`)로 사용
- 1,000개 행의 배치로 데이터 처리(`-b 1000`)

일반적인 BCP 옵션

BCP는 데이터 형식 지정 및 전송 동작을 제어하는 다양한 옵션을 제공합니다. 다음 표에 일반적으로 사용되는 옵션이 설명되어 있습니다.

옵션	설명
`-c`	모든 열에 문자 데이터 유형 사용
`-n`	기본 데이터베이스 데이터 유형 사용
`-t`	필드 구분 기호 지정(기본값은 탭)
`-r`	행 구분 기호 지정(기본값은 줄 바꿈)
`-b`	일괄 작업의 배치 크기 지정
`-F`	내보내거나 가져올 첫 번째 행 지정
`-L`	내보내거나 가져올 마지막 행 지정
`-e`	거부된 행을 캡처할 오류 파일 지정
`-f`	데이터 형식 지정을 위한 형식 파일 지정
`-q`	객체 이름에 따옴표로 묶인 식별자 사용

모범 사례 및 고려 사항

Linux에서 RDS for SQL Server와 함께 BCP를 사용하는 경우 다음 모범 사례를 고려하세요.

- 배치 처리 사용 - 대규모 데이터세트의 경우 `-b` 옵션을 사용하여 데이터를 배치로 처리합니다. 이렇게 하면 성능이 개선되고 오류 복구가 개선됩니다.
- 적절한 오류 처리 - `-e` 옵션을 사용하여 분석을 위해 오류 정보와 거부된 행을 별도의 파일로 캡처합니다.

- 적절한 데이터 형식 선택 - 교차 플랫폼 호환성을 위해 문자 형식(-c)을 사용하거나 소스와 대상이 모두 SQL Server인 경우 더 나은 성능을 위해 기본 형식(-n)을 사용합니다.
- 자격 증명 보호 - 암호를 명령줄에 직접 입력하지 마세요. 적절한 권한이 있는 환경 변수 또는 구성 파일을 사용하는 것이 좋습니다.
- 작은 데이터세트로 테스트 - 대량의 데이터를 처리하기 전에 작은 데이터세트로 BCP 명령을 테스트하여 형식 지정 및 연결을 확인합니다.
- 네트워크 연결 모니터링 - 특히 대규모 데이터 전송의 경우 안정적인 네트워크 연결을 보장합니다. 장기 실행 작업에는 `screen` 또는 `tmux`와 같은 도구를 사용하는 것이 좋습니다.
- 데이터 무결성 검증 - 데이터 전송 후 행 수와 샘플 데이터를 확인하여 작업이 성공적으로 완료되었는지 확인합니다.

일반적인 문제 해결

다음 표에는 Linux의 BCP 및 해당 솔루션을 사용할 때 발생할 수 있는 일반적인 문제가 설명되어 있습니다.

문제	Solution
연결 제한 시간 초과 또는 네트워크 오류	Amazon RDS 엔드포인트, 보안 그룹 설정 및 네트워크 연결을 확인합니다. Linux 시스템에서 SQL Server 포트(일반적으로 1433)에 액세스할 수 있는지 확인합니다.
인증 실패 횟수	사용자 이름과 암호를 확인합니다. 데이터베이스 사용자에게 수행 중인 작업에 대한 적절한 권한이 있는지 확인합니다.
데이터 형식 오류	필드와 행 구분 기호를 확인합니다. 데이터 형식이 BCP에서 요구하는 것과 일치하는지 확인합니다. 복잡한 데이터 구조에 형식 파일을 사용합니다.
권한 거부 오류	데이터베이스 사용자에게 대상 테이블에 대한 가져오기의 경우 `INSERT` 권한, 내보내기의 경우 `SELECT` 권한이 있는지 확인합니다.
대용량 파일 처리 문제	`-b` 옵션과 함께 배치 처리를 사용합니다. 성능 향상 및 오류 복구를 위해 대용량 파일을 더 작은 묶음으로 분할하는 것이 좋습니다.

문제	Solution
문자 인코딩 문제	데이터 파일이 호환되는 문자 인코딩을 사용하는지 확인합니다. 문자 형식에 -c 옵션을 사용하거나 적절한 코드 페이지를 지정합니다.

Amazon RDS에서 Microsoft SQL Server용 읽기 전용 복제본 작업

일반적으로 읽기 전용 복제본을 사용하여 Amazon RDS DB 인스턴스 간 복제를 구성합니다. 읽기 전용 복제본에 대한 일반적인 정보는 DB 인스턴스 읽기 전용 복제본 작업 단원을 참조하십시오.

이 단원에서는 Amazon RDS for SQL Server의 읽기 전용 복제본 작업에 대한 특정 정보를 찾을 수 있습니다.

- 데이터베이스 사용자 및 객체를 SQL Server 읽기 전용 복제본과 동기화
- SQL Server 읽기 전용 복제본 문제 해결

SQL Server용 읽기 전용 복제본 구성

DB 인스턴스를 복제용 원본 인스턴스로 사용하려면 원본 DB 인스턴스의 자동 백업을 활성화해야 합니다. 이렇게 하려면 백업 보존 기간을 0 이외의 값으로 설정합니다. 이 유형의 배포를 설정하면 자동 백업도 강제로 활성화됩니다.

SQL Server 읽기 복제본을 만들 때 기본 DB 인스턴스의 운영 중단이 필요하지 않습니다. Amazon RDS는 서비스 중단 없이 소스 DB 및 읽기 전용 복제본에 필요한 파라미터와 권한을 설정합니다. 원본 DB 인스턴스를 캡처한 스냅샷이 읽기 전용 복제본이 됩니다. 읽기 전용 복제본을 삭제하더라도 중단은 발생하지 않습니다.

원본 DB 인스턴스 하나에서 최대 15개까지 읽기 전용 복제본을 생성할 수 있습니다. 효과적인 복제를 위해서는 읽기 전용 복제본도 각각 소스 DB 인스턴스와 동일한 양의 컴퓨팅 및 스토리지 리소스를 갖도록 구성해야 합니다. 원본 DB 인스턴스를 확장하는 경우 읽기 전용 복제본도 확장합니다.

소스 DB 인스턴스의 SQL Server DB 엔진 버전과 모든 읽기 전용 복제본은 동일해야 합니다. Amazon RDS는 유지 관리 기간과 상관없이 읽기 전용 복제본을 업그레이드한 후 즉시 기본 인스턴스를 업그레이드합니다. DB 엔진 버전 업그레이드에 대한 자세한 내용은 Microsoft SQL Server DB 엔진 업그레이드 단원을 참조하세요.

읽기 전용 복제본이 원본에서 변경 사항을 받아 적용하려면 충분한 컴퓨팅 및 스토리지 리소스가 있어야 합니다. 읽기 전용 복제본이 컴퓨팅, 네트워크 또는 스토리지 리소스 용량에 도달하면 읽기 전용 복제본은 해당 원본에서 변경 사항을 수신하거나 적용하는 것을 중지합니다. 읽기 전용 복제본의 스토리지 및 CPU 리소스를 해당 원본 및 다른 읽기 전용 복제본과 별도로 수정할 수 있습니다.

읽기 전용 복제본 생성 방법에 대한 자세한 내용은 읽기 전용 복제본 생성 섹션을 참조하세요.

SQL Server의 읽기 전용 복제본 제한 사항

Amazon RDS의 SQL Server 읽기 전용 복제본에는 다음 제한 사항이 적용됩니다.

- 읽기 전용 복제본은 SQL Server Enterprise Edition(EE) 엔진에서만 사용할 수 있습니다.
- 읽기 복제본은 SQL Server 버전 2016~2022에서 사용할 수 있습니다.
- 원본 DB 인스턴스 하나에서 최대 15개까지 읽기 전용 복제본을 생성할 수 있습니다. 소스 DB 인스턴스에 읽기 전용 복제본이 5개 이상인 경우 복제가 지연될 수 있습니다.
- 읽기 전용 복제본은 4개 이상의 vCPU가 있는 DB 인스턴스 클래스에서 실행되는 DB 인스턴스에만 사용할 수 있습니다.
- 읽기 전용 복제본은 인스턴스 클래스 유형 및 가용성 모드에 따라 최대 100개의 데이터베이스를 지원합니다. 읽기 전용 복제본으로 자동 복제하려면 소스 DB 인스턴스에 데이터베이스를 생성해야 합니다. 복제할 데이터베이스를 하나씩 선택할 수는 없습니다. 자세한 내용은 Microsoft SQL Server DB 인스턴스의 한도 섹션을 참조하세요.
- 읽기 전용 복제본에서 데이터베이스를 삭제할 수 없습니다. 데이터베이스를 삭제하려면 `rds_drop_database` 저장 프로시저를 통해 소스 DB 인스턴스에서 삭제합니다. 자세한 내용은 Amazon RDS for Microsoft SQL Server DB 인스턴스 데이터베이스 삭제 섹션을 참조하세요.
- 소스 DB 인스턴스가 Transparent Data Encryption(TDE)을 사용하는 경우 읽기 전용 복제본 역시 TDE를 자동으로 구성합니다.

  소스 DB 인스턴스가 KMS 키를 사용하여 데이터를 암호화할 경우 동일한 리전의 읽기 전용 복제본은 동일한 KMS 키를 사용합니다. 크로스 리전 읽기 전용 복제본의 경우 읽기 전용 복제본을 생성할 때 읽기 전용 복제본의 리전에서 KMS 키를 지정해야 합니다. 읽기 전용 복제본의 KMS는 변경할 수 없습니다.
- 읽기 전용 복제본은 생성된 가용 영역에 상관없이 소스 DB 인스턴스와 동일한 시간대 및 데이터 정렬 기준을 갖습니다.
- 다음은 Amazon RDS for SQL Server에서 지원되지 않습니다.
  - 읽기 전용 복제본의 백업 보존
  - 읽기 전용 복제본의 특정 시점으로 복구
  - 읽기 전용 복제본의 수동 스냅샷
  - 다중 AZ 읽기 전용 복제본
  - 읽기 전용 복제본의 읽기 전용 복제본 생성
  - 읽기 전용 복제본에 대한 사용자 로그인 동기화

- Amazon RDS for SQL Server는 개입을 통해 원본 DB 인스턴스와 해당 읽기 전용 복제본 간의 긴 복제본 지연 시간을 완화하지 않습니다. 원본 DB 인스턴스와 해당 읽기 전용 복제본이 운영 로드에 맞게 컴퓨팅 파워 및 스토리지 면에서 제대로 크기가 조정되었는지 확인합니다.
- AWS GovCloud(미국 동부) 리전과 AWS GovCloud(미국-서부) 리전 간에 복제할 수 있지만 AWS GovCloud (US) Regions 내부 또는 외부로 복제할 수는 없습니다.

RDS for SQL Server 복제본에 대한 옵션 고려 사항

RDS for SQL Server 복제본을 생성하기 전에 다음 요구 사항, 제한 사항 및 권장 사항을 확인하세요.

- SQL Server 복제본이 소스 DB 인스턴스와 동일한 리전에 있는 경우 해당 복제본이 원본 DB 인스턴스와 동일한 옵션 그룹에 속해 있는지 확인하세요. 원본 옵션 그룹 또는 원본 옵션 그룹 멤버십에 대한 수정 사항은 복제본으로 전파됩니다. 이러한 변경 사항은 복제본의 유지 관리 기간과 상관없이 원본 DB 인스턴스에 적용된 직후 복제본에 적용됩니다.

  옵션 그룹에 대한 자세한 내용은 옵션 그룹 작업 단원을 참조하십시오.
- SQL Server 리전 간 복제본을 생성하면 Amazon RDS가 전용 옵션 그룹을 생성합니다.

  전용 옵션 그룹에서 SQL Server 리전 간 복제본을 제거할 수 없습니다. 다른 DB 인스턴스는 SQL Server 리전 간 복제본에 전용 옵션 그룹을 사용할 수 없습니다.

  다음 옵션은 복제된 옵션입니다. SQL Server 리전 간 읽기 전용 복제본에 복제된 옵션을 추가하려면 원본 DB 인스턴스의 옵션 그룹에 추가하세요. 이 옵션은 모든 원본 DB 인스턴스의 복제본에도 설치됩니다.

  - `TDE`

  다음 옵션은 복제되지 않은 옵션입니다. 전용 옵션 그룹에 복제되지 않은 옵션을 추가하거나 제거할 수 있습니다.

  - `MSDTC`
  - `SQLSERVER_AUDIT`
  - 리전 간 읽기 전용 복제본에서 `SQLSERVER_AUDIT` 옵션을 활성화하려면 리전 간 읽기 전용 복제본의 전용 옵션 그룹과 원본 인스턴스의 옵션 그룹에 `SQLSERVER_AUDIT` 옵션을 추가하세요. SQL Server 리전 간 읽기 전용 복제본의 원본 인스턴스에 `SQLSERVER_AUDIT` 옵션을 추가하면 원본 인스턴스의 각 리전 간 읽기 전용 복제본에 서버 수준 감사 객체 및 서버 수준 감사 사양을 생성할 수 있습니다. 리전 간 읽기 전용 복제본에 액세스하여 완성된 감사 로그를 Amazon S3 버킷에 업로드할 수 있도록 하려면 전용 옵션 그룹에 `SQLSERVER_AUDIT` 옵션을 추가하고 옵션 설정을 구성하세요. 감사 파일의 대상으로 사용하는 Amazon S3 버킷은 리전 간 읽기 전용 복제본과

같은 리전에 있어야 합니다. 각 리전 간 읽기 전용 복제본에 대한 SQLSERVER_AUDIT 옵션의 옵션 설정을 개별적으로 수정하여 각 읽기 전용 복제본이 해당 리전의 Amazon S3 버킷에 액세스하도록 할 수 있습니다.

다음 옵션은 읽기 전용 복제본에서 지원되지 않습니다.

- SSRS
- SSAS
- SSIS

다음 옵션은 리전 간 읽기 전용 복제본에서 부분적으로 지원됩니다.

- SQLSERVER_BACKUP_RESTORE
- SQL Server 리전 간 복제본의 원본 DB 인스턴스에는 SQLSERVER_BACKUP_RESTORE 옵션이 있을 수 있지만 원본 DB 인스턴스의 모든 리전 간 복제본을 삭제하기 전에는 원본 DB 인스턴스에서 기본 복원을 수행할 수 없습니다. 리전 간 복제본을 생성하는 동안 기존의 모든 기본 복원 작업이 취소됩니다. 전용 옵션 그룹에는 SQLSERVER_BACKUP_RESTORE 옵션을 추가할 수 없습니다.

  기본 백업 및 복원에 대한 자세한 내용은 기본 백업 및 복원 기능을 사용하여 SQL Server 데이터베이스 가져오기 및 내보내기 섹션을 참조하세요.

SQL Server 리전 간 읽기 전용 복제본을 승격하면 승격된 복제본은 옵션 관리를 포함해 다른 SQL Server DB 인스턴스와 동일하게 작동합니다. 옵션 그룹에 대한 자세한 내용은 옵션 그룹 작업 단원을 참조하세요.

데이터베이스 사용자 및 객체를 SQL Server 읽기 전용 복제본과 동기화

읽기 전용 복제본을 생성할 때 프라이머리 DB 인스턴스에 있는 모든 로그인, 사용자 지정 서버 역할, SQL 에이전트 작업 또는 기타 서버 수준 객체가 새로 생성된 읽기 전용 복제본에 있어야 합니다. 하지만 읽기 전용 복제본을 생성한 후에 프라이머리 DB 인스턴스에 생성된 서버 수준 객체는 자동으로 복제되지 않으므로 읽기 전용 복제본에서 수동으로 생성해야 합니다.

데이터베이스 사용자는 프라이머리 DB 인스턴스에서 읽기 전용 복제본으로 자동 복제됩니다. 읽기 전용 복제본 데이터베이스는 읽기 전용 모드이므로 데이터베이스 사용자의 보안 식별자(SID)를 데이터베이스에서 업데이트할 수 없습니다. 따라서 읽기 전용 복제본에서 SQL 로그인을 생성할 때는 해당 로그인의 SID가 프라이머리 DB 인스턴스의 해당 SQL 로그인 SID와 일치하는지 확인해야 합니다. SQL 로그인의 SID를 동기화하지 않으면 읽기 전용 복제본의 데이터베이스에 액세스할 수 없습니다. Windows Active Directory(AD) 인증 로그인은 SQL Server가 Active Directory에서 SID를 가져오기 때문에 이 문제가 발생하지 않습니다.

프라이머리 DB 인스턴스에서 읽기 전용 복제본으로 SQL 로그인 동기화

1. 프라이머리 DB 인스턴스에 연결합니다.
2. 프라이머리 DB 인스턴스에 새 SQL 로그인을 생성합니다.

```
USE [master]
GO
CREATE LOGIN TestLogin1
WITH PASSWORD = 'REPLACE WITH PASSWORD';
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

3. 데이터베이스에 SQL 로그인을 위한 새 데이터베이스 사용자를 생성합니다.

```
USE [REPLACE WITH YOUR DB NAME]
GO
CREATE USER TestLogin1 FOR LOGIN TestLogin1;
GO
```

4. 프라이머리 DB 인스턴스에서 새로 생성한 SQL 로그인의 SID를 확인합니다.

```
SELECT name, sid FROM sys.server_principals WHERE name =  TestLogin1;
```

5. 읽기 전용 복제본에 연결합니다. 새 SQL 로그인을 생성합니다.

```
CREATE LOGIN TestLogin1 WITH PASSWORD = 'REPLACE WITH PASSWORD', SID=[REPLACE WITH
 sid FROM STEP #4];
```

또는 읽기 전용 복제본 데이터베이스에 액세스할 수 있는 경우 다음과 같이 분리된 사용자를 수정할 수 있습니다.

1. 읽기 전용 복제본에 연결합니다.
2. 데이터베이스에서 분리된 사용자를 식별합니다.

```
USE [REPLACE WITH YOUR DB NAME]
GO
EXEC sp_change_users_login 'Report';
```

```
GO
```

3. 분리된 데이터베이스 사용자를 위한 새 SQL 로그인을 생성합니다.

```
CREATE LOGIN TestLogin1 WITH PASSWORD = 'REPLACE WITH PASSWORD', SID=[REPLACE WITH
 sid FROM STEP #2];
```

예시

```
CREATE LOGIN TestLogin1 WITH PASSWORD = 'TestPa$$word#1',
 SID=[0x1A2B3C4D5E6F7G8H9I0J1K2L3M4N5O6P];
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

SQL Server 읽기 전용 복제본 문제 해결

Amazon CloudWatch에서 Amazon RDS ReplicaLag 지표를 보고 복제 지연을 모니터링할 수 있습니다. 복제 지연 시간에 대한 자세한 내용은 읽기 전용 복제본 모니터링 단원을 참조하십시오.

복제 지연 시간이 너무 긴 경우 다음 쿼리를 사용하여 지연 시간에 대한 정보를 얻을 수 있습니다.

```
SELECT AR.replica_server_name
     , DB_NAME (ARS.database_id) 'database_name'
     , AR.availability_mode_desc
     , ARS.synchronization_health_desc
     , ARS.last_hardened_lsn
     , ARS.last_redone_lsn
     , ARS.secondary_lag_seconds
FROM sys.dm_hadr_database_replica_states ARS
INNER JOIN sys.availability_replicas AR ON ARS.replica_id = AR.replica_id
--WHERE DB_NAME(ARS.database_id) = 'database_name'
ORDER BY AR.replica_server_name;
```

Amazon RDS for Microsoft SQL Server의 다중 AZ 배포

다중 AZ 배포는 DB 인스턴스를 위해 향상된 가용성, 데이터 내구성 및 내결함성을 제공합니다. 계획된 데이터베이스 유지 관리 또는 예기치 않은 서비스 중단이 발생할 경우, Amazon RDS가 최신 보조 DB 인스턴스로 자동으로 장애 조치를 수행합니다. 이 기능을 통해 수동 개입 없이 데이터베이스 작업을 빠르게 재개할 수 있습니다. 기본 인스턴스 및 예비 인스턴스는 동일한 엔드포인트를 사용합니다. 이 엔드포인트의 물리적 네트워크 주소는 장애 조치 프로세스의 일환으로 보조 복제본으로 전환됩니다. 장애 조치가 발생하는 경우 애플리케이션을 다시 구성할 필요가 없습니다.

Amazon RDS는 SQL Server 데이터베이스 미러링(DBM) 또는 상시 가동 가용성 그룹(AG)을 사용하여 Microsoft SQL Server에 대한 다중 AZ 배포를 지원합니다. Amazon RDS는 다중 AZ 배포의 상태를 모니터링하고 유지합니다. 문제가 발생하면 RDS는 이상 있는 DB 인스턴스를 복구하고, 동기화를 재설정하며, 장애 조치를 시작합니다. 대기 및 기본 인스턴스가 완벽히 동기화되어 있는 경우에만 장애 조치가 이루어집니다. 사용자가 따로 관리할 것이 없습니다.

SQL Server 다중 AZ를 설정하면 RDS가 DBM 또는 AG를 사용하도록 인스턴스의 모든 데이터베이스를 자동으로 구성합니다. Amazon RDS에서 프라이머리 DB 인스턴스, 감시 인스턴스, 세컨더리 DB 인스턴스를 자동으로 처리합니다. 구성이 자동화되어 있으므로 RDS에서는 사용자가 배포하는 SQL Server의 버전에 따라 DBM 또는 상시 가동 AG를 선택합니다.

Amazon RDS는 다음 SQL Server 버전에 상시 가동 AG를 통한 다중 AZ를 지원합니다.

- SQL Server 2022:
  - Standard Edition
  - Enterprise Edition
- SQL Server 2019:
  - Standard Edition 15.00.4073.23 이상
  - Enterprise Edition
- SQL Server 2017:
  - Standard Edition 14.00.3401.7 이상
  - Enterprise Edition 14.00.3049.1 이상
- SQL Server 2016: Enterprise Edition 13.00.5216.0 이상

Amazon RDS는 앞서 언급한 버전을 제외한 다음 SQL Server 버전에 대해 DBM를 통한 다중 AZ를 지원합니다.

- SQL Server 2019: Standard Edition 15.00.4043.16
- SQL Server 2017: Standard 및 Enterprise Edition
- SQL Server 2016: Standard 및 Enterprise Edition

다음 SQL 쿼리를 사용하여 SQL Server DB 인스턴스가 단일 AZ, DBM 기능이 있는 다중 AZ 또는 상시 가동 AG 기능이 있는 다중 AZ인지 확인할 수 있습니다.

```
SELECT CASE WHEN dm.mirroring_state_desc IS NOT NULL THEN 'Multi-AZ (Mirroring)'
    WHEN dhdrs.group_database_id IS NOT NULL THEN 'Multi-AZ (AlwaysOn)'
    ELSE 'Single-AZ'
    END 'high_availability'
FROM sys.databases sd
LEFT JOIN sys.database_mirroring dm ON sd.database_id = dm.database_id
LEFT JOIN sys.dm_hadr_database_replica_states dhdrs ON sd.database_id =
 dhdrs.database_id AND dhdrs.is_local = 1
WHERE DB_NAME(sd.database_id) = 'rdsadmin';
```

출력은 다음과 유사합니다.

```
high_availability
Multi-AZ (AlwaysOn)
```

다중 AZ를 Microsoft SQL Server DB 인스턴스에 추가

AWS Management Console을 사용하여 새로운 SQL Server DB 인스턴스를 만들 때, 데이터베이스 미러링(DBM) 또는 상시 작동 AG를 사용하는 다중 AZ를 추가할 수 있습니다. 이렇게 하려면 다중 AZ 배포에서 Yes (Mirroring / Always On)(예(미러링/상시 작동)를 선택합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성을 참조하세요.

콘솔을 사용하여 기존 SQL Server DB 인스턴스를 수정할 때 DB 인스턴스 수정 페이지의 다중 AZ 배포에서 예(미러링/상시 작동)를 선택하여 DBM 또는 AG를 사용하는 다중 AZ를 추가할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정을 참조하세요.

> **Note**
>
> DB 인스턴스가 상시 작동 가용성 그룹(AG)이 아닌 데이터베이스 미러링(DBM)을 실행 중인 경우 다중 AZ를 추가하기 전에 인 메모리 최적화를 비활성화해야 할 수도 있습니다. DB 인스턴스에서 SQL Server 2016 또는 2017 Enterprise Edition을 실행하고 있으며 인 메모리 최적화가 활성화된 경우 다중 AZ를 추가하기 전에 DBM으로 인 메모리 최적화를 비활성화합니다.

DB 인스턴스에서 AG를 실행 중인 경우에는 이 단계가 필요하지 않습니다.

Microsoft SQL Server DB 인스턴스에서 다중 AZ 제거

AWS Management Console를 사용하여 기존 SQL Server DB 인스턴스를 수정할 때 DBM 또는 AG가 있는 다중 AZ를 제거할 수 있습니다. DB 인스턴스 수정 페이지의 다중 AZ 배포에서 아니요(미러링/항상 켜짐)를 선택하면 됩니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

Microsoft SQL Server 다중 AZ 배포의 제한, 참고 및 권장 사항

다음은 RDS for SQL Server DB 인스턴스에서 다중 AZ 배포 작업 시 알아두어야 할 몇 가지 제한 사항입니다.

- 교차 리전 다중 AZ는 지원되지 않습니다.
- 다중 AZ 배포에서는 RDS for SQL Server DB 인스턴스를 중지할 수 없습니다.
- 보조 DB 인스턴스가 데이터베이스 읽기 작업을 허용하도록 구성할 수 없습니다.
- 상시 작동 가용성 그룹(AG)을 사용하는 다중 AZ는 인 메모리 최적화를 지원합니다.
- 상시 작동 가용성 그룹(AG)이 있는 다중 AZ는 가용성 그룹 리스너에 대한 Kerberos 인증을 지원하지 않습니다. 이는 리스너에 SPN(서비스 보안 주체 이름)이 없기 때문입니다.
- SQL Server 다중 AZ 배포에 있는 SQL Server DB 인스턴스의 데이터베이스는 이름을 변경할 수 없습니다. 그러한 인스턴스의 데이터베이스 이름을 바꿔야 하는 경우, 먼저 DB 인스턴스의 다중 AZ를 끈 후 데이터베이스 이름을 바꿉니다. 마지막으로 DB 인스턴스의 다중 AZ를 다시 켭니다.
- 전체 복구 모델을 사용하여 백업한 다중 AZ DB 인스턴스만 복원할 수 있습니다.
- 다중 AZ 배포에서는 SQL Server 에이전트 작업이 10,000개로 제한됩니다.

  한도를 늘려야 할 경우 지원에 문의하여 할당량 증대를 요청하세요. [지원 센터(AWS Support Center)] 페이지를 열고 필요한 경우, 로그인한 다음 [사례 생성(Create Case)]을 선택합니다. Service Limit increase(서비스 한도 증가)를 선택합니다. 양식을 작성하고 제출합니다.
- SQL Server 다중 AZ 배포에 있는 SQL Server DB 인스턴스에 오프라인 데이터베이스를 가질 수 없습니다.
- RDS for SQL Server는 보조 인스턴스에 MSDB 데이터베이스 권한을 복제하지 않습니다. 보조 인스턴스에서 이러한 권한이 필요한 경우 수동으로 다시 생성해야 합니다.

다음은 RDS for SQL Server DB 인스턴스에서 다중 AZ 배포 작업 시 알아두어야 할 몇 가지 참고 사항입니다.

- Amazon RDS는 상시 가동 AG 가용성 그룹 리스너 엔드포인트를 표시합니다. 이 엔드포인트는 콘솔에 표시되며, `DescribeDBInstances` API 작업에 의해 엔드포인트 필드의 항목으로 반환됩니다.
- Amazon RDS는 가용성 그룹 다중 서브넷 장애 조치를 지원합니다.
- Virtual Private Cloud(VPC)에서 SQL Server DB 인스턴스와 함께 SQL Server 다중 AZ를 사용하려면 먼저 별개의 가용 영역 2개 이상에 서브넷이 있는 DB 서브넷 그룹을 만들어야 합니다. 그런 다음 SQL Server DB 인스턴스의 기본 복제본에 DB 서브넷 그룹을 할당합니다.
- DB 인스턴스를 다중 AZ 배포로 수정할 때 수정하는 동안 인스턴스의 상태는 [수정 중(modifying)]입니다. Amazon RDS는 대기 인스턴스를 생성하고 프라이머리 DB 인스턴스를 백업합니다. 프로세스가 완료되면 기본 DB 인스턴스의 상태가 사용 가능이 됩니다.
- 다중 AZ 배포가 같은 노드 상의 모든 데이터베이스를 유지 관리합니다. 기본 호스트의 데이터베이스가 장애 조치하는 경우, 모든 SQL Server 데이터베이스가 하나의 원자 단위로 대기 호스트로 장애 조치합니다. Amazon RDS는 새로운 정상 호스트를 프로비저닝하고 비정상 호스트를 대체합니다.
- DBM 또는 AG를 사용하는 다중 AZ는 예비 복제본 하나를 지원합니다.
- 사용자, 로그인 및 권한은 보조에 자동으로 복제됩니다. 이를 다시 생성할 필요가 없습니다. 사용자 정의 서버 역할은 다중 AZ 배포에 Always On AG를 사용하는 DB 인스턴스에서만 복제됩니다.
- 다중 AZ 배포에서 RDS for SQL Server는 SQL Server 로그인을 생성하여 상시 작동 AG 또는 데이터베이스 미러링을 허용합니다. RDS는 `db_<dbiResourceId>_node1_login`, `db_<dbiResourceId>_node2_login`, `db_<dbiResourceId>_witness_login` 패턴으로 로그인을 생성합니다.
- RDS for SQL Server는 읽기 전용 복제본에 대한 액세스를 허용하기 위해 SQL Server 로그인을 생성합니다. RDS는 `db_<readreplica_dbiResourceId>_node_login` 패턴으로 로그인을 생성합니다.
- 다중 AZ 배포에서 SQL Server 에이전트 작업은 작업 복제 기능이 켜져 있을 때 기본 호스트에서 보조 호스트로 복제됩니다. 자세한 내용은 SQL Server 에이전트 작업 복제 켜기을 참조하세요.
- 동기식 데이터 복제로 인해 단일 가용 영역에서 표준 DB 인스턴스 배포와 비교했을 때 지연 시간이 증가할 수 있습니다.
- 장애 조치 시간은 복구 프로세스 완료에 걸리는 시간의 영향을 받습니다. 트랜잭션이 크면 장애 조치 시간이 늘어납니다.
- SQL Server 다중 AZ 배포에서 장애 조치를 사용하여 재부팅하면 기본 DB 인스턴스만 재부팅됩니다. 장애 조치 후에는 기본 DB 인스턴스가 새 보조 DB 인스턴스가 됩니다. 다중 AZ 인스턴스의 경우 파라미터가 업데이트되지 않을 수 있습니다. 장애 조치 없이 재부팅하는 경우 기본 DB 인스턴스와 보조 DB 인스턴스가 모두 재부팅되고 재부팅된 후 파라미터가 업데이트됩니다. DB 인스턴스가 응답하지 않는 경우 장애 조치 없이 재부팅하는 것이 좋습니다.

다음은 RDS for Microsoft SQL Server DB 인스턴스에서 다중 AZ 배포 작업 시 알아두어야 할 몇 가지 권장 사항입니다.

- 프로덕션 또는 프로덕션 이전 환경에서 사용되는 데이터베이스의 경우 다음 옵션을 권장합니다.
  - 고가용성을 위한 다중 AZ 배포
  - 빠르고 일관적인 성능을 위한 “프로비저닝된 IOPS”
  - “범용”이 아닌 “메모리 최적화”
- 보조 인스턴스에 대해 가용 영역(AZ)을 선택할 수 없으므로 애플리케이션 호스트를 배포할 경우 이 점을 고려하십시오. 데이터베이스를 다른 AZ로 장애 조치할 수 있으며, 애플리케이션 호스트가 데이터베이스와 다른 AZ에 있을 수도 있습니다. 이러한 이유로 지정된 AWS 리전의 모든 AZ에서 애플리케이션 호스트의 균형을 조정하는 것이 좋습니다.
- 최상의 성능을 위해, 큰 데이터 로드 작업 중에는 데이터베이스 미러링 또는 상시 작동 AG를 활성화하지 마십시오. 데이터 로드를 최대한 빠르게 수행하려면 데이터 로드를 완료한 후에 DB 인스턴스를 다중 AZ 배포로 변환합니다.
- SQL Server 데이터베이스에 액세스하는 애플리케이션에 연결 오류를 포착하는 예외 처리 기능이 있어야 합니다. 다음 코드 샘플에 통신 오류를 포착하는 try/catch 블록이 표시되어 있습니다. 이 예제에서 `break` 문은 연결에 성공하면 `while` 루프를 종료하지만 예외가 발생하면 10회까지 다시 시도합니다.

```
int RetryMaxAttempts = 10;
int RetryIntervalPeriodInSeconds = 1;
int iRetryCount = 0;
while (iRetryCount < RetryMaxAttempts)
{
   using (SqlConnection connection = new SqlConnection(DatabaseConnString))
   {
      using (SqlCommand command = connection.CreateCommand())
      {
         command.CommandText = "INSERT INTO SOME_TABLE VALUES ('SomeValue');";
         try
         {
            connection.Open();
            command.ExecuteNonQuery();
            break;
         }
         catch (Exception ex)
         {
            Logger(ex.Message);
            iRetryCount++;
```

```
         }
         finally {
            connection.Close();
         }
      }
   }
   Thread.Sleep(RetryIntervalPeriodInSeconds * 1000);
}
```

- 다중 AZ 인스턴스로 작업할 때는 `Set Partner Off` 명령을 사용하지 마십시오. 예를 들어 다음을 수행하지 마십시오.

```
--Don't do this
ALTER DATABASE db1 SET PARTNER off
```

- 복구 모드를 `simple`로 설정하지 마십시오. 예를 들어 다음을 수행하지 마십시오.

```
--Don't do this
ALTER DATABASE db1 SET RECOVERY simple
```

- 이러한 설정은 대기 미러에 적용할 수 없으므로 다중 AZ DB 인스턴스에 새 로그인을 만들 때 `DEFAULT_DATABASE` 파라미터를 사용하지 마십시오. 예를 들어 다음을 수행하지 마십시오.

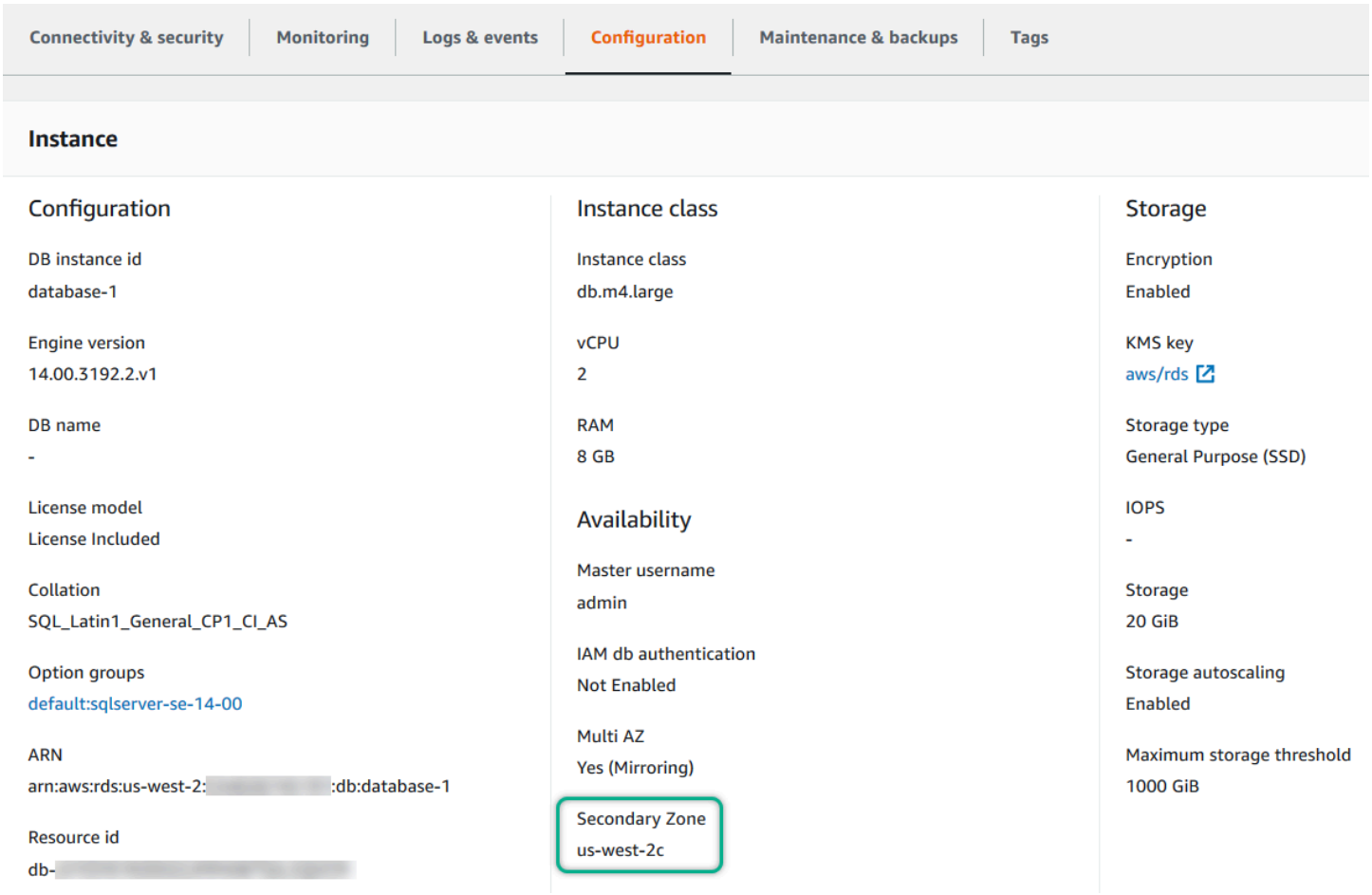
```
--Don't do this
CREATE LOGIN [test_dba] WITH PASSWORD=foo, DEFAULT_DATABASE=[db2]
```

  또한 다음 작업을 수행하지 마십시오.

```
--Don't do this
ALTER LOGIN [test_dba] WITH DEFAULT_DATABASE=[db3]
```

보조의 위치 확인

AWS Management Console을 사용하여 보조 복제본의 위치를 확인할 수 있습니다. VPC에서 기본 DB 인스턴스를 설정할 경우 보조의 위치를 알아야 합니다.



AWS CLI 명령 `describe-db-instances` 또는 RDS API 작업 `DescribeDBInstances`를 사용하여 보조의 가용 영역을 확인할 수도 있습니다. 출력에는 대기 미러가 있는 보조 AZ가 표시됩니다.

데이터베이스 미러링에서 상시 작동 가용성 그룹으로 마이그레이션

Microsoft SQL Server Enterprise Edition의 14.00.3049.1 버전에서는 상시 작동 가용성 그룹(AG)이 기본적으로 사용됩니다.

데이터베이스 미러링(DBM)에서 AG로 마이그레이션하려면 먼저 버전을 확인하십시오. 버전이 13.00.5216.0 이전인 DB 인스턴스를 사용하는 경우, 13.00.5216.0으로 패치하도록 인스턴스를 수정합니다. Enterprise Edition 14.00.3049.1 이전 버전의 DB 인스턴스를 사용하는 경우, 14.00.3049.1 이상으로 패치하도록 인스턴스를 수정합니다.

AG를 사용하도록 미러링된 DB 인스턴스를 업그레이드하려면 먼저 업그레이드를 실행하고 다중 AZ를 제거하도록 인스턴스를 수정한 다음 다시 수정하여 다중 AZ를 추가하십시오. 그러면 인스턴스가 상시 작동 AG를 사용하도록 변환됩니다.

Amazon RDS의 Microsoft SQL Server 추가 기능

다음 섹션에서는 Microsoft SQL Server DB 엔진을 실행하는 Amazon RDS 인스턴스의 강화에 대한 정보를 찾을 수 있습니다.

주제

- RDS for SQL Server에서 SQL Server 로그인에 암호 정책 사용
- Amazon RDS for SQL Server DB 인스턴스와 Amazon S3 통합
- Amazon RDS for SQL Server에 Database Mail 사용
- Amazon RDS for SQL Server의 tempdb 데이터베이스에 대한 인스턴스 스토어 지원
- Amazon RDS for Microsoft SQL Server에 확장 이벤트 사용
- RDS for SQL Server를 사용하여 트랜잭션 로그 백업에 액세스

RDS for SQL Server에서 SQL Server 로그인에 암호 정책 사용

Amazon RDS를 사용하면 Microsoft SQL Server를 실행하는 Amazon RDS DB 인스턴스에 암호 정책을 설정할 수 있습니다. 이를 사용하여 SQL Server 인증으로 DB 인스턴스를 인증하는 로그인의 복잡성, 길이 및 잠금 요구 사항을 설정할 수 있습니다.

주요 용어

로그인

SQL Server에서는 데이터베이스 인스턴스에 대해 인증할 수 있는 서버 수준 보안 주체를 로그인이라고 합니다. 다른 데이터베이스 엔진에서는 이 보안 주체를 사용자로 지칭할 수 있습니다. RDS for SQL Server에서는 SQL Server 인증 또는 Windows 인증을 사용하여 로그인을 인증할 수 있습니다.

SQL Server 로그인

SQL Server 인증을 사용하여 사용자 이름과 암호를 통해 인증하는 로그인은 SQL Server 로그인입니다. DB 파라미터를 통해 구성한 암호 정책은 SQL Server 로그인에만 적용됩니다.

Windows 로그인

Windows 보안 주체를 기반으로 하고 Windows 인증을 사용하여 인증하는 로그인은 Windows 로그인입니다. Active Directory에서 Windows 로그인에 대한 암호 정책을 구성할 수 있습니다. 자세한 내용은 RDS for SQL Server를 사용하여 Active Directory 작업 단원을 참조하십시오.

각 로그인에 대한 정책 활성화 및 비활성화

각 SQL Server 로그인에는 `CHECK_POLICY` 및 `CHECK_EXPIRATION` 플래그가 있습니다. 기본적으로 새 로그인은 `ON`으로 설정된 `CHECK_POLICY`와 `OFF`로 설정된 `CHECK_EXPIRATION`으로 생성됩니다.

로그인에 대해 `CHECK_POLICY`가 활성화된 경우 RDS for SQL Server는 복잡성 및 최소 길이 요구 사항을 기준으로 암호의 유효성을 검사합니다. 잠금 정책 또한 적용됩니다. `CHECK_POLICY` 및 `CHECK_EXPIRATION`을 활성화하기 위한 예제 T-SQL 문:

```
ALTER LOGIN [master_user] WITH CHECK_POLICY = ON, CHECK_EXPIRATION = ON;
```

`CHECK_EXPIRATION`이 활성화된 경우 암호에는 암호 수명 정책이 적용됩니다. `CHECK_POLICY` 및 `CHECK_EXPIRATION` 설정 여부를 확인하기 위한 T-SQL 문:

```
SELECT name, is_policy_checked, is_expiration_checked FROM sys.sql_logins;
```

암호 정책 파라미터

모든 암호 정책 파라미터는 동적이므로 DB를 재부팅하지 않아도 적용됩니다. 다음 표에는 SQL Server 로그인에 대한 암호 정책을 수정하기 위해 설정할 수 있는 DB 파라미터가 나열되어 있습니다.

DB 파라미터	설명	허용된 값	기본 값	

rds.password_complexity_enabled	SQL Server 로그인을 위한 암호를 만들거나 변경할 때는 암호 복잡성 요구 사항을 충족해야 합니다. 다음 제약 조건을 충족해야 합니다.
	• 암호에는 다음 중 세 가지 범주의 문자가 포함되어야 합니다.
	  • 라틴 소문자(a~z)
	  • 라틴 대문자(A~Z)
	  • 영숫자 이외의 문자(예: 느낌표(!), 달러 기호($), 숫자 기호(#) 또는 퍼센트(%)).
	• 암호에 사용자의 계정 이름이 포함되지 않음
		0,1	0	

DB 파라미터	설명	허용된 값	기본 값	
rds.password_min_length	SQL Server 로그인을 위한 암호에 필요한 최소 문자 수입니다.	0~14	0	
rds.password_min_age	사용자가 SQL Server 로그인 암호를 변경할 수 있기 전에 사용해야 하는 최소 일 수입니다. 0으로 설정되어 있는 경우 암호를 즉시 변경할 수 있습니다.	0~998	0	
rds.password_max_age	SQL Server 로그인 암호를 사용할 수 있는 최대 일 수입니다. 이 기간이 지나면 사용자가 암호를 변경해야 합니다. 0으로 설정되어 있는 경우 암호가 만료되지 않습니다.	0~999	42	
rds.password_lockout_threshold	SQL Server 로그인이 잠기기 전에 연속으로 실패할 수 있는 로그인 시도 횟수입니다.	0~999	0	

DB 파라미터	설명	허용된 값	기본 값
rds.password_lockout_duration	잠긴 SQL Server 로그인이 잠금 해제될 때까지 기다려야 하는 시간(분)입니다.	1~60	10
rds.password_lockout_reset_counter_after	로그인 시도 실패 후 로그인 시도 실패 카운터가 0으로 재설정되기 전에 경과해야 하는 시간(분)입니다.	1~60	10

> **Note**
>
> SQL Server 암호 정책에 대한 자세한 내용은 암호 정책을 참조하세요.
> 암호 복잡성 및 최소 길이 정책은 포함된 데이터베이스의 DB 사용자에게도 적용됩니다. 자세한 내용은 포함된 데이터베이스를 참조하세요.

암호 정책 매개변수에는 다음 제약 조건이 적용됩니다.

- `rds.password_max_age`가 0으로 설정되어 있지 않는 한 `rds.password_min_age` 파라미터는 `rds.password_max_age parameter`보다 작아야 합니다.
- `rds.password_lockout_reset_counter_after` 파라미터는 `rds.password_lockout_duration` 파라미터보다 작거나 같아야 합니다.
- `rds.password_lockout_threshold`가 0으로 설정되어 있으면 `rds.password_lockout_duration` 및 `rds.password_lockout_reset_counter_after`는 적용되지 않습니다.

기존 로그인에 대한 고려 사항

인스턴스의 암호 정책을 수정한 후에는 로그인을 위한 기존 암호가 새로운 암호 복잡성 및 길이 요구 사항을 기준으로 소급 평가되지 않습니다. 새 암호만 새 정책을 기준으로 검증됩니다.

SQL Server는 수명 요구 사항을 기준으로 기존 암호를 평가합니다.

암호 정책이 수정되는 즉시 암호가 만료될 수도 있습니다. 예를 들어 로그인에 CHECK_EXPIRATION이 활성화되어 있고 100일 전에 암호를 마지막으로 변경했으며 `rds.password_max_age` 파라미터를 5일로 설정한 경우 암호가 즉시 만료되므로 다음 로그인 시도 시 암호를 변경해야 합니다.

> **Note**
>
> RDS for SQL Server는 암호 기록 정책을 지원하지 않습니다. 기록 정책은 로그인이 이전에 사용한 암호를 재사용하지 못하도록 합니다.

다중 AZ 배포에 대한 고려 사항

다중 AZ 인스턴스의 로그인 시도 실패 카운터 및 잠금 상태는 노드 간에 복제되지 않습니다. 다중 AZ 인스턴스가 장애 조치될 때 로그인이 잠기는 경우 새 노드에서 로그인이 이미 잠금 해제되어 있을 수 있습니다.

마스터 로그인에 대한 암호 고려 사항

RDS for SQL Server DB 인스턴스를 생성하면 마스터 사용자 암호가 암호 정책을 기준으로 평가되지 않습니다. 마스터 사용자에게 작업을 수행할 때, 특히 `ModifyDBInstance` 명령에서 `MasterUserPassword`를 설정할 때도 새 마스터 암호는 암호를 기준으로 평가되지 않습니다. 두 경우 모두 암호 정책을 충족하지 않는 마스터 사용자 암호를 설정할 수 있으며, 이 경우에도 작업은 성공합니다. 정책이 충족되지 않으면 RDS는 강력한 암호를 설정하라는 권장 사항이 포함된 RDS 이벤트를 발생시키려고 시도합니다. 마스터 사용자에는 강력한 암호만 사용해야 합니다.

마스터 사용자 암호가 암호 정책 요구 사항을 충족하지 않는 경우 RDS는 다음 이벤트 메시지를 생성하려고 시도합니다.

- 마스터 사용자가 생성되었지만 암호가 암호 정책의 최소 길이 요구 사항을 충족하지 않습니다. 더 강력한 암호를 사용해 보세요.
- 마스터 사용자가 생성되었지만 암호가 암호 정책의 복잡성 요구 사항을 충족하지 않습니다. 더 강력한 암호를 사용해 보세요.

- 마스터 사용자 암호가 재설정되었지만 암호가 암호 정책의 최소 길이 요구 사항을 충족하지 않습니다. 더 강력한 암호를 사용해 보세요.
- 마스터 사용자 암호가 재설정되었지만 암호가 암호 정책의 복잡성 요구 사항을 충족하지 않습니다. 더 강력한 암호를 사용해 보세요.

기본적으로 마스터 사용자는 CHECK_POLICY 및 CHECK_EXPIRATION으로 생성되고 OFF로 설정됩니다. 마스터 사용자에게 암호 정책을 적용하려면 DB 인스턴스 생성 후 마스터 사용자에 대해 이러한 플래그를 수동으로 활성화해야 합니다. 이러한 플래그를 활성화한 후에는 SQL Server에서 직접 마스터 사용자 암호를 수정하여(예: T-SQL 문 또는 SSMS를 통해) 암호 정책을 기준으로 새 암호의 유효성을 검사합니다.

> **Note**
>
> 마스터 사용자가 잠긴 경우 ModifyDBInstance 명령으로 마스터 사용자 암호를 재설정하여 사용자를 잠금 해제할 수 있습니다.

마스터 사용자 암호 수정

ModifyDBInstance 명령을 사용하여 마스터 사용자 암호를 수정할 수 있습니다.

> **Note**
>
> 마스터 사용자 암호를 재설정하면 RDS는 마스터 사용자에 대한 다양한 권한을 재설정하므로 마스터 사용자는 특정 권한을 잃을 수 있습니다. 또한 마스터 사용자 암호를 재설정하면 마스터 사용자의 잠금이 해제됩니다(잠겨 있던 경우).

RDS는 새 마스터 사용자 암호의 유효성을 검사하고 암호가 정책을 충족하지 않는 경우 RDS 이벤트를 내보내려고 시도합니다. RDS는 암호 정책을 충족하지 않는 경우에도 암호를 설정합니다.

Amazon RDS for SQL Server DB 인스턴스와 Amazon S3 통합

Amazon RDS for SQL Server를 실행 중인 DB 인스턴스와 Amazon S3 버킷 사이에서 파일을 전송할 수 있습니다. 그러면 BULK INSERT와 같은 SQL Server 기능으로 Amazon S3를 사용할 수 있습니다. 예를 들어 Amazon S3의 .csv, .xml, .txt 및 기타 파일을 DB 인스턴스 호스트로 다운로드하고 `D:\S3\`에서 데이터베이스로 데이터를 가져올 수 있습니다. 모든 파일은 DB 인스턴스 기반으로 `D:\S3\`에 저장됩니다.

다음과 같은 제한이 적용됩니다.

> **Note**
>
> RDS 호스트와 S3 간의 트래픽은 S3를 사용하는 모든 SQL Server 기능의 경우 RDS 내부 VPC의 VPC 엔드포인트를 통해 라우팅됩니다. 이 트래픽은 RDS 인스턴스 엔드포인트 ENI를 사용하지 않습니다. S3 버킷 정책은 네트워킹 조건에 따라 RDS 트래픽을 제한할 수 없습니다.

- 다중 AZ 인스턴스에서 장애 조치 후 `D:\S3` 폴더의 파일이 예비 복제본에서 삭제됩니다. 자세한 내용은 S3 통합에 대한 다중 AZ 제한 사항 섹션을 참조하세요.
- DB 인스턴스와 S3 버킷은 같은 AWS 리전에 있어야 합니다.
- 한 번에 둘 이상의 S3 통합 작업을 실행하는 경우 작업은 병렬이 아닌 순차적으로 실행됩니다.

> **Note**
>
> S3 통합 작업은 기본 백업 및 복원 작업과 동일한 대기열을 공유합니다. 이 대기열에서는 언제든 최대 두 개의 작업만 진행할 수 있습니다. 따라서 두 개의 기본 백업 및 복원 작업을 실행하면 S3 통합 작업이 차단됩니다.

- 복원된 인스턴스에서 S3 통합 기능을 다시 활성화해야 합니다. S3 통합은 소스 인스턴스에서 복원된 인스턴스로 전파되지 않습니다. `D:\S3`의 파일은 복원된 인스턴스에서 삭제됩니다.
- DB 인스턴스로 다운로드하는 파일은 100개로 제한됩니다. 즉, `D:\S3\`에 있는 파일이 100개를 초과할 수 없습니다.
- 파일 확장명이 없거나, 파일 확장명이 .abf, .asdatabase, .bcp, .configsettings, .csv, .dat, .deploymentoptions, .deploymenttargets, .fmt, .info, .is인 파일만 다운로드할 수 있습니다.
- S3 버킷의 소유자는 관련 AWS Identity and Access Management(IAM) 역할과 동일해야 합니다. 따라서 교차 계정 S3 통합은 지원되지 않습니다.

- S3 버킷은 공개할 수 없습니다.
- RDS에서 S3로의 업로드 파일 크기는 파일당 50GB로 제한됩니다.
- S3에서 RDS로의 다운로드 파일 크기는 S3에서 지원하는 최대 크기로 제한됩니다.

주제

- RDS for SQL Server와 S3를 통합하기 위한 사전 요구 사항
- RDS for SQL Server와 S3 통합 활성화
- RDS for SQL Server와 Amazon S3 간 파일 전송
- RDS DB 인스턴스의 파일 나열
- RDS DB 인스턴스의 파일 삭제
- 파일 전송 작업 상태 모니터링
- 작업 취소
- S3 통합에 대한 다중 AZ 제한 사항
- RDS for SQL Server와 S3 통합 비활성화

Amazon S3의 파일 작업에 대한 자세한 내용은 Amazon Simple Storage Service 시작하기를 참조하십시오.

RDS for SQL Server와 S3를 통합하기 위한 사전 요구 사항

시작하기 전에, 사용하려는 S3 버킷을 찾거나 생성하십시오. 또한 RDS DB 인스턴스가 S3 버킷에 액세스할 수 있도록 권한을 추가하십시오. 이 액세스를 구성하려면 IAM 정책과 IAM 역할을 모두 생성하십시오.

콘솔

Amazon S3 액세스를 위한 IAM 정책을 생성하려면

1. IAM Management Console의 탐색 창에서 정책을 선택합니다.
2. 새 정책을 생성하고 다음 단계에 대해 시각적 편집기 탭을 사용하십시오.
3. 서비스에 **S3**를 입력한 후 S3 서비스를 선택하십시오.
4. 작업에서 DB 인스턴스에 필요한 액세스 권한을 부여하려면 다음을 선택하십시오.
   - `ListAllMyBuckets` - 필수
   - `ListBucket` - 필수

- `GetBucketAcl` - 필수
- `GetBucketLocation` - 필수
- `GetObject` – S3에서 로 파일을 다운로드하는 데 필요`D:\S3\`
- `PutObject` – `D:\S3\`에서 S3로 파일을 업로드하는 데 필요
- `ListMultipartUploadParts` – `D:\S3\`에서 S3로 파일을 업로드하는 데 필요
- `AbortMultipartUpload` – `D:\S3\`에서 S3로 파일을 업로드하는 데 필요

5. 리소스의 경우, 표시되는 옵션은 이전 단계에서 선택한 작업에 따라 다릅니다. 버킷, 객체 또는 두 항목 모두의 옵션이 표시될 수 있습니다. 이들 각각에 대해 적절한 Amazon 리소스 이름(ARN)을 추가하십시오.

   버킷의 경우 사용할 버킷의 ARN을 추가하십시오. 예를 들어, 버킷 이름이 *amzn-s3-demo-bucket*인 경우 ARN을 `arn:aws:s3:::`*amzn-s3-demo-bucket*으로 설정하세요.

   객체의 경우, 버킷의 ARN을 입력한 후 다음 중 하나를 선택하십시오.

   - 지정된 버킷의 모든 파일에 대한 액세스 권한을 부여하려면 버킷 이름 및 객체 이름에 대해 동일하게 모두를 선택하십시오.
   - 버킷의 특정 파일 또는 폴더에 대한 액세스 권한을 부여하려면 SQL Server가 액세스할 특정 버킷과 객체의 ARN을 제공하십시오.

6. 정책 생성을 완료할 때까지 콘솔의 지침을 따르십시오.

   위의 내용은 정책 설정을 위한 약어 형태의 안내서입니다. IAM 정책 생성에 대한 자세한 지침은 IAM 사용 설명서에서 IAM 정책 생성을 참조하세요.

이전 절차의 IAM 정책을 사용하는 IAM 역할을 생성하려면

1. IAM Management Console의 탐색 창에서 역할을 선택합니다.
2. 새 IAM 역할을 생성하고 콘솔에 표시되는 다음 옵션을 선택하십시오.

   - AWS 서비스
   - RDS
   - RDS – Add Role to Database(데이터베이스에 역할 추가)

   그런 다음 하단의 다음: 권한을 선택합니다.

3. Attach permissions policies(권한 정책 연결)에 이전에 생성한 IAM 정책의 이름을 입력하십시오. 그런 다음 목록에서 그 정책을 선택합니다.
4. 역할 생성을 완료할 때까지 콘솔의 지침을 따르십시오.

   위의 내용은 역할 설정을 위한 약어 형태의 안내서입니다. 역할 생성에 대한 자세한 지침은 IAM 사용 설명서에서 IAM 역할을 참조하세요.

AWS CLI

Amazon RDS에 Amazon S3 버킷에 대한 액세스 권한을 부여하려면 다음 프로세스를 사용합니다.

1. Amazon RDS에 S3 버킷 액세스 권한을 부여하는 IAM 정책을 생성합니다.
2. Amazon RDS가 S3 버킷에 액세스하기 위해 사용자 대신 가정할 수 있는 IAM 역할을 만듭니다.

   자세한 내용은 IAM 사용 설명서의 IAM 사용자에게 권한을 위임하기 위한 역할 생성을 참조하십시오.
3. 생성한 IAM 역할에 생성한 IAM 정책을 연결합니다.

IAM 정책을 생성하려면

DB 인스턴스에 필요한 액세스 권한을 부여하려면 적절한 조치를 포함하십시오.

- `ListAllMyBuckets` - 필수
- `ListBucket` - 필수
- `GetBucketAcl` - 필수
- `GetBucketLocation` - 필수
- `GetObject` – S3에서 로 파일을 다운로드하는 데 필요`D:\S3\`
- `PutObject` – `D:\S3\`에서 S3로 파일을 업로드하는 데 필요
- `ListMultipartUploadParts` – `D:\S3\`에서 S3로 파일을 업로드하는 데 필요
- `AbortMultipartUpload` – `D:\S3\`에서 S3로 파일을 업로드하는 데 필요

1. 다음 AWS CLI 명령은 이 옵션으로 `rds-s3-integration-policy`라는 IAM 정책을 만듭니다. *`amzn-s3-demo-bucket`*이라는 버킷에 대한 액세스 권한을 부여합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws iam create-policy \
  --policy-name rds-s3-integration-policy \
  --policy-document '{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": "s3:ListAllMyBuckets",
                "Resource": "*"
            },
            {
                "Effect": "Allow",
                "Action": [
                    "s3:ListBucket",
                    "s3:GetBucketAcl",
                    "s3:GetBucketLocation"
                ],
                "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
            },
            {
                "Effect": "Allow",
                "Action": [
                    "s3:GetObject",
                    "s3:PutObject",
                    "s3:ListMultipartUploadParts",
                    "s3:AbortMultipartUpload"
                ],
                "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/key_prefix/*"
            }
        ]
    }'
```

Windows의 경우:

행 끝을 인터페이스에서 지원하는 것으로 바꾸십시오(^ 대신 \). 또한 Windows에서는 \로 모든 큰 따옴표를 이스케이프해야 합니다. JSON에서 따옴표를 이스케이프하지 않으려면, 파일에 대신 저장하고 파라미터로 전달하면 됩니다.

먼저 다음 권한 정책을 사용하여 policy.json 파일을 생성하십시오.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:ListAllMyBuckets",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketACL",
                "s3:GetBucketLocation"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/key_prefix/*"
        }
    ]
}
```

그리고 다음 명령을 사용하여 정책을 만듭니다.

```
aws iam create-policy ^
     --policy-name rds-s3-integration-policy ^
     --policy-document file://file_path/assume_role_policy.json
```

2. 정책을 만든 후에 정책의 Amazon 리소스 이름(ARN)을 기록하십시오. 이후 단계에 이 ARN이 필요합니다.

IAM 역할을 만들려면

- 다음 AWS CLI 명령은 이 목적으로 `rds-s3-integration-role` IAM 역할을 생성합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws iam create-role \
    --role-name rds-s3-integration-role \
    --assume-role-policy-document '{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
             "Service": "rds.amazonaws.com"
           },
          "Action": "sts:AssumeRole"
        }
      ]
    }'
```

  Windows의 경우:

  행 끝을 인터페이스에서 지원하는 것으로 바꾸십시오(^ 대신 \). 또한 Windows에서는 \로 모든 큰 따옴표를 이스케이프해야 합니다. JSON에서 따옴표를 이스케이프하지 않으려면, 파일에 대신 저장하고 파라미터로 전달하면 됩니다.

  먼저, 다음 정책이 포함된 `assume_role_policy.json` 파일을 생성합니다.

  JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "rds.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

그러고 나서 다음 명령을 사용하여 IAM 역할을 만듭니다.

```
aws iam create-role ^
     --role-name rds-s3-integration-role ^
     --assume-role-policy-document file://file_path/assume_role_policy.json
```

Example 전역 조건 컨텍스트 키를 사용하여 IAM 역할 생성

서비스 권한을 특정 리소스로 제한하는 리소스 기반 정책의 aws:SourceArn 및 aws:SourceAccount 전역 조건 컨텍스트 키를 사용하는 것이 좋습니다. 이는 혼동된 대리자 문제를 방지하는 가장 효과적인 방법입니다.

전역 조건 컨텍스트 키를 모두 사용하고 aws:SourceArn 값에 계정 ID가 포함되도록 할 수 있습니다. 이 경우 aws:SourceAccount 값과 aws:SourceArn 값의 계정이 동일한 정책 문에서 사용될 때 동일한 계정 ID를 사용해야 합니다.

- 단일 리소스에 대한 교차 서비스 액세스를 원하는 경우 aws:SourceArn을 사용하세요.
- 해당 계정의 모든 리소스가 교차 서비스 사용과 연결되도록 허용하려는 경우 aws:SourceAccount를 사용하세요.

정책에서는 역할에 액세스하는 리소스의 전체 Amazon 리소스 이름(ARN)이 포함된 aws:SourceArn 전역 조건 컨텍스트 키를 사용해야 합니다. S3 통합의 경우 다음 예와 같이 DB 인스턴스 ARN을 포함해야 합니다.

대상 LinuxmacOS, 또는Unix:

```
aws iam create-role \
    --role-name rds-s3-integration-role \
```

```
    --assume-role-policy-document '{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
             "Service": "rds.amazonaws.com"
           },
          "Action": "sts:AssumeRole",
                "Condition": {
                    "StringEquals": {

 "aws:SourceArn":"arn:aws:rds:Region:my_account_ID:db:db_instance_identifier"
                    }
                }
        }
      ]
    }'
```

Windows의 경우:

assume_role_policy.json에 전역 조건 컨텍스트 키를 추가합니다.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "rds.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {

 "aws:SourceArn":"arn:aws:rds:Region:my_account_ID:db:db_instance_identifier"
                }
            }
```

```
        }
    ]
}
```

IAM 역할에 IAM 정책 연결

- 다음 AWS CLI 명령은 정책을 `rds-s3-integration-role`이라는 역할에 연결합니다. *your-policy-arn*을 이전 단계에서 기록한 정책 ARN으로 바꾸세요.

  Example

  대상 LinuxmacOS, 또는Unix:

  ```
  aws iam attach-role-policy \
      --policy-arn your-policy-arn \
      --role-name rds-s3-integration-role
  ```

  Windows의 경우:

  ```
  aws iam attach-role-policy ^
      --policy-arn your-policy-arn ^
      --role-name rds-s3-integration-role
  ```

RDS for SQL Server와 S3 통합 활성화

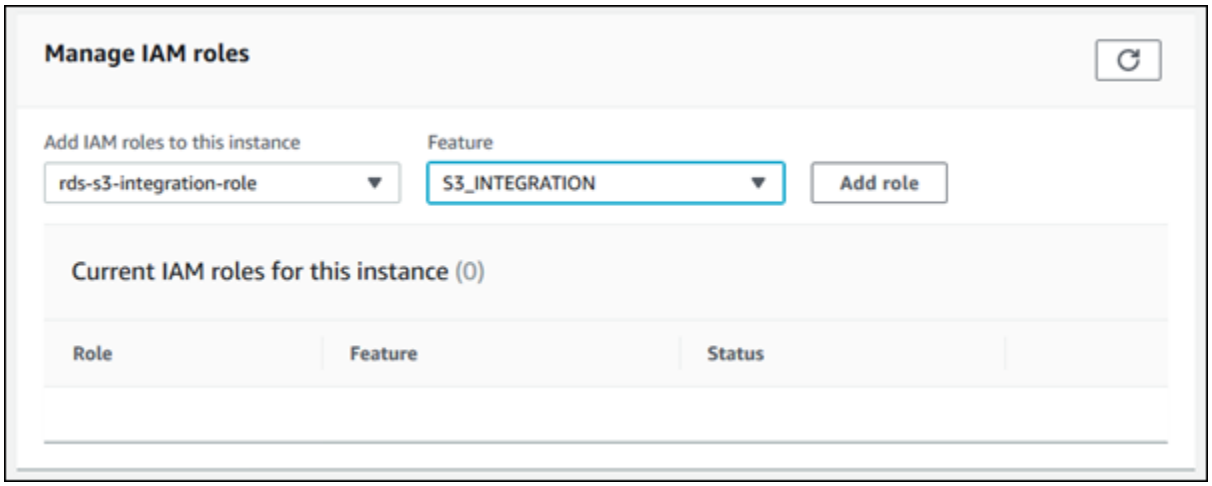
다음 섹션에서는 Amazon S3와 Amazon RDS for SQL Server의 통합을 활성화하는 방법을 배울 수 있습니다. S3 통합 작업을 위해, `S3_INTEGRATION` feature-name 파라미터를 사용하기 전에 생성한 IAM 역할과 DB 인스턴스를 연결해야 합니다.

> **Note**
>
> DB 인스턴스에 IAM 역할을 추가하려면 DB 인스턴스 상태가 사용 가능이어야 합니다.

콘솔

IAM 역할을 DB 인스턴스와 연결하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 세부 정보를 표시하고자 하는 RDS for SQL Server DB 인스턴스 이름을 선택합니다.
3. Connectivity & security(연결성 및 보안) 탭에 있는 Manage IAM roles(IAM 역할 관리) 섹션의 이 인스턴스에 IAM 역할 추가에서 추가할 IAM 역할을 선택합니다.
4. 기능에서 S3_INTEGRATION을 선택하십시오.



5. [Add role]을 선택합니다.

AWS CLI

RDS for SQL Server DB 인스턴스에 IAM 역할을 추가하려면

- 다음 AWS CLI 명령은 *mydbinstance*라는 RDS for SQL Server DB 인스턴스에 IAM 역할을 추가합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds add-role-to-db-instance \
    --db-instance-identifier mydbinstance \
    --feature-name S3_INTEGRATION \
    --role-arn your-role-arn
```

Windows의 경우:

```
aws rds add-role-to-db-instance ^
    --db-instance-identifier mydbinstance ^
    --feature-name S3_INTEGRATION ^
    --role-arn your-role-arn
```

*your-role-arn*을 이전 단계에서 기록한 역할 ARN으로 바꿉니다. `S3_INTEGRATION` 옵션에 대해 `--feature-name`을 지정해야 합니다.

RDS for SQL Server와 Amazon S3 간 파일 전송

Amazon RDS 저장 프로시저를 사용하여 Amazon S3와 RDS DB 인스턴스 간에 파일을 다운로드하고 업로드할 수 있습니다. Amazon RDS 저장 프로시저를 사용하여 RDS 인스턴스의 파일을 나열하고 삭제할 수도 있습니다.

S3에서 다운로드 및 업로드한 파일은 `D:\S3` 폴더에 저장됩니다. 파일에 액세스하는 데 사용할 수 있는 유일한 폴더입니다. 다운로드하는 동안 대상 폴더를 포함할 때 생성되는 하위 폴더에 파일을 구성할 수 있습니다.

저장 프로시저 중에는 S3 버킷과 파일에 Amazon 리소스 이름(ARN)을 제공해야 하는 것도 있습니다. ARN의 형식은 `arn:aws:s3:::amzn-s3-demo-bucket/file_name`입니다. Amazon S3에서는 ARN에 계정 번호나 AWS 리전을 요구하지 않습니다.

S3 통합 작업은 순차적으로 실행되며 기본 백업 및 복원 작업과 동일한 대기열을 공유합니다. 이 대기열에서는 언제든 최대 두 개의 작업만 진행할 수 있습니다. 작업이 처리를 시작하는 데 최대 5분이 걸릴 수 있습니다.

Amazon S3 버킷의 파일을 SQL Server DB 인스턴스로 다운로드

S3 버킷에서 RDS for SQL Server DB 인스턴스로 파일을 다운로드하려면 다음 파라미터와 함께 Amazon RDS 저장 프로시저 `msdb.dbo.rds_download_from_s3`를 사용합니다.

파라미터 이름	데이터 형식	기본값	필수	설명
`@s3_arn_of_file`	NVARCHAR	–	필수	다운로드할 파일의 S3 ARN(예: `arn:aws:s`

파라미터 이름	데이터 형식	기본값	필수	설명
				3::: *amzn-s3-demo-bucket* /mydata.csv)
@rds_file_path	NVARCHAR	–	선택	RDS 인스턴스의 파일 경로입니다. 지정하지 않으면 파일 경로는 D:\S3*<filename in s3>*입니다. RDS는 절대 경로와 상대 경로를 지원합니다. 하위 폴더를 생성하려면 파일 경로에 포함하십시오.
@overwrite_file	INT	0	선택	기존 파일을 덮어씁니다. 0 = 덮어쓰지 않음 1 = 덮어쓰기

파일 확장명이 없는 파일과 파일 확장명이 .bcp, .csv, .dat, .fmt, .info, .lst, .tbl, .txt 및 .xml인 파일을 다운로드할 수 있습니다.

> **Note**
>
> SQL Server Integration Services가 활성화된 경우 파일 확장명이 .ispac인 파일을 다운로드할 수 있습니다. SSIS 활성화에 대한 자세한 내용은 SQL Server Integration Services 단원을 참조하십시오.
> SQL Server Analysis Services가 활성화된 경우 파일 확장명이 .abf, .asdatabase, .configsettings, .deploymentoptions, .deploymenttargets 및 .xmla인 파일을 다운로드할 수 있습니다. SSAS 활성화에 대한 자세한 내용은 SQL Server Analysis Services 단원을 참조하십시오.

다음 예는 S3에서 파일을 다운로드하는 저장 프로시저를 보여줍니다.

```
exec msdb.dbo.rds_download_from_s3
     @s3_arn_of_file='arn:aws:s3:::amzn-s3-demo-bucket/bulk_data.csv',
     @rds_file_path='D:\S3\seed_data\data.csv',
     @overwrite_file=1;
```

예제 rds_download_from_s3 작업은 폴더가 없는 경우에 seed_data에 D:\S3\라는 폴더를 만듭니다. 그런 다음 이 예제에서는 S3의 원본 파일 bulk_data.csv를 DB 인스턴스에 data.csv라는 새 파일로 다운로드합니다. 파일이 이전에 존재한 경우, @overwrite_file 파라미터가 1로 설정되어 있으므로 덮어쓰기됩니다.

SQL Server DB 인스턴스의 파일을 Amazon S3 버킷으로 업로드

RDS for SQL Server DB 인스턴스의 파일을 S3 버킷에 업로드하려면 다음 파라미터와 함께 Amazon RDS 저장 프로시저 msdb.dbo.rds_upload_to_s3를 사용합니다.

파라미터 이름	데이터 형식	기본값	필수	설명
@s3_arn_of_file	NVARCHAR	–	필수	S3에 생성할 파일의 S3 ARN(예: arn:aws:s3::: *amzn-s3-demo-bucket* /mydata.csv)
@rds_file_path	NVARCHAR	–	필수	S3에 업로드할 파일의 파일 경로입니다. 절대 경로와 상대 경로가 지원됩니다.
@overwrite_file	INT	–	선택	기존 파일을 덮어씁니다. 0 = 덮어쓰지 않음 1 = 덮어쓰기

다음 예제에서는 data.csv에서 지정된 위치의 D:\S3\seed_data\라는 파일을 ARN이 지정한 S3 버킷의 파일 new_data.csv에 업로드합니다.

```
exec msdb.dbo.rds_upload_to_s3
  @rds_file_path='D:\S3\seed_data\data.csv',
  @s3_arn_of_file='arn:aws:s3:::amzn-s3-demo-bucket/new_data.csv',
  @overwrite_file=1;
```

S3에 파일이 이전에 존재한 경우, @overwrite_file 파라미터가 1로 설정되어 있으므로 덮어쓰기됩니다.

RDS DB 인스턴스의 파일 나열

DB 인스턴스에서 사용 가능한 파일을 나열하려면 저장 프로시저와 함수를 모두 사용하십시오. 먼저, 다음 저장 프로시저를 실행하여 D:\S3\의 파일에서 파일 세부 정보를 수집하십시오.

```
exec msdb.dbo.rds_gather_file_details;
```

저장 프로시저는 작업 ID를 반환합니다. 다른 작업과 마찬가지로 이 저장 프로시저는 비동기적으로 실행됩니다. 작업 상태가 SUCCESS가 되는 즉시, 다음과 같이 rds_fn_list_file_details 함수의 작업 ID를 사용하여 D:\S3\의 기존 파일과 디렉터리를 나열할 수 있습니다.

```
SELECT * FROM msdb.dbo.rds_fn_list_file_details(TASK_ID);
```

이 rds_fn_list_file_details 함수는 다음 열이 포함된 테이블을 반환합니다.

출력 파라미터	설명
filepath	파일의 절대 경로(예: D:\S3\mydata.csv)
size_in_bytes	파일 크기(바이트 단위)
last_modified_utc	UTC 형식의 마지막 수정 날짜 및 시간
is_directory	항목이 디렉터리인지 나타내는 옵션(true/false)

RDS DB 인스턴스의 파일 삭제

DB 인스턴스에서 사용 가능한 파일을 삭제하려면 다음 파라미터와 함께 Amazon RDS 저장 프로시저 msdb.dbo.rds_delete_from_filesystem을 사용하십시오.

파라미터 이름	데이터 형식	기본값	필수	설명
@rds_file_path	NVARCHAR	–	필수	삭제할 파일의 파일 경로입니다. 절대 경로와 상대 경로가 지원됩니다.
@force_delete	INT	0	선택	디렉터리를 삭제하려면 이 플래그를 포함하고 1로 설정해야 합니다. 1 = 디렉터리 삭제 파일을 삭제하는 경우 이 파라미터는 무시됩니다.

디렉터리를 삭제하려면 @rds_file_path가 백슬래시(\)로 끝나야 하며 @force_delete가 1로 설정되어야 합니다.

다음 예제에서는 D:\S3\delete_me.txt 파일을 삭제합니다.

```
exec msdb.dbo.rds_delete_from_filesystem
    @rds_file_path='D:\S3\delete_me.txt';
```

다음은 D:\S3\example_folder\ 디렉터리를 삭제하는 예제입니다.

```
exec msdb.dbo.rds_delete_from_filesystem
    @rds_file_path='D:\S3\example_folder\',
    @force_delete=1;
```

파일 전송 작업 상태 모니터링

S3 통합 작업의 상태를 추적하려면 rds_fn_task_status 함수를 호출하십시오. 두 가지 파라미터가 필요합니다. 첫 번째 파라미터는 S3 통합에 적용되지 않기 때문에 항상 NULL이어야 합니다. 두 번째 파라미터는 작업 ID를 수락합니다.

모든 작업 목록을 보려면 다음 예와 같이 첫 번째 파라미터를 NULL로, 두 번째 파라미터를 0으로 설정하십시오.

```
SELECT * FROM msdb.dbo.rds_fn_task_status(NULL,0);
```

특정 작업을 수행하려면 다음 예와 같이 첫 번째 파라미터를 NULL로, 두 번째 파라미터를 작업 ID로 설정하십시오.

```
SELECT * FROM msdb.dbo.rds_fn_task_status(NULL,42);
```

rds_fn_task_status 함수는 다음 정보를 반환합니다.

출력 파라미터	설명
task_id	작업의 ID입니다.
task_type	S3 통합의 경우 작업 유형은 다음과 같을 수 있습니다. • DOWNLOAD_FROM_S3 • UPLOAD_TO_S3 • LIST_FILES_ON_DISK • DELETE_FILES_ON_DISK
database_name	S3 통합 작업에는 적용되지 않습니다.
% complete	백분율로 나타낸 작업의 진행률입니다.
duration(mins)	작업에 소요된 시간입니다(분).
lifecycle	작업의 상태입니다. 가능한 상태는 다음과 같습니다. • CREATED – S3 통합 저장 프로시저 중 하나를 호출하면 작업이 생성되고 상태가 로 설정됩니다.CREATED • IN_PROGRESS – 작업이 시작되면 상태가 로 설정됩니다.IN_PROGRESS CREATED에서 IN_PROGRESS 로 상태가 변경되려면 최대 5분이 걸릴 수 있습니다.

출력 파라미터	설명
	• SUCCESS – 작업이 완료되면 상태가 로 설정됩니다.SUCCESS • ERROR – 작업이 실패하면 상태가 로 설정됩니다.ERROR 오류에 대한 자세한 내용은 task_info 열을 참조하십시오. • CANCEL_REQUESTED – rds_cancel_task 를 호출하는 즉시 작업의 상태가 CANCEL_REQUESTED 로 설정됩니다. • CANCELLED – 작업이 성공적으로 취소된 뒤에는 작업 상태가 로 설정됩니다.CANCELLED
task_info	작업에 대한 추가 정보입니다. 처리 중에 오류가 발생하면 이 열에 오류 정보가 포함됩니다.
last_updated	작업 상태를 마지막으로 업데이트한 날짜와 시간입니다.
created_at	작업을 생성한 날짜와 시간입니다.
S3_object_arn	다운로드하거나 업로드할 S3 객체의 ARN입니다.
overwrite_S3_backup_file	S3 통합 작업에는 적용되지 않습니다.
KMS_master_key_arn	S3 통합 작업에는 적용되지 않습니다.
filepath	RDS DB 인스턴스의 파일 경로입니다.
overwrite_file	기존 파일이 덮어쓰기되는지 나타내는 옵션입니다.
task_metadata	S3 통합 작업에는 적용되지 않습니다.

작업 취소

S3 통합 작업을 취소하려면 msdb.dbo.rds_cancel_task 파라미터와 함께 task_id 저장 프로시저를 사용하십시오. 진행 중인 작업 삭제 및 나열은 취소할 수 없습니다. 다음은 작업을 취소하는 요청 예제입니다.

```
exec msdb.dbo.rds_cancel_task @task_id = 1234;
```

모든 작업 및 해당 작업 ID에 대한 개요를 보려면 rds_fn_task_status에 설명된 파일 전송 작업 상태 모니터링 함수를 사용하십시오.

S3 통합에 대한 다중 AZ 제한 사항

다중 AZ 인스턴스에서 장애 조치 후 D:\S3 폴더의 파일이 예비 복제본에서 삭제됩니다. 예를 들어 인스턴스 클래스 변경 또는 엔진 버전 업그레이드와 같은 DB 인스턴스 수정 시 계획된 장애 조치를 수행할 수 있습니다. 또는 주 인스턴스 중단 시 계획되지 않은 장애 조치를 수행할 수 있습니다.

> **Note**
>
> 파일 저장에 D:\S3 폴더를 사용하지 않는 것이 좋습니다. 가장 좋은 방법은 생성한 파일을 Amazon S3에 업로드하여 지속성 있게 만들고 데이터를 가져와야 할 때 파일을 다운로드하는 것입니다.

마지막 장애 조치 시간을 확인하려면 msdb.dbo.rds_failover_time 저장 프로시저를 사용합니다. 자세한 내용은 Amazon RDS for SQL Server의 마지막 장애 조치 시간 결정 섹션을 참조하세요.

Example 최근 장애 조치 없음

이 예에서는 오류 로그에 최근 장애 조치가 없는 경우의 출력을 보여 줍니다. 2020-04-29 23:59:00.01 이후로 장애 조치가 발생하지 않았습니다.

따라서 이 시간 이후에 다운로드되고 rds_delete_from_filesystem 저장 프로시저를 사용하여 삭제되지 않은 모든 파일은 현재 호스트에서 계속 액세스할 수 있습니다. 이 시간 이전에 다운로드한 파일도 사용 가능할 수 있습니다.

errorlog_available_from	recent_failover_time

errorlog_available_from	recent_failover_time
2020-04-29 23:59:00.0100000	null

Example 최근 장애 조치

이 예에서는 오류 로그에 장애 조치가 있는 경우의 출력을 보여 줍니다. 가장 최근의 장애 조치는 2020-05-05 18:57:51.89에 있었습니다.

이 시간 이후에 다운로드되고 `rds_delete_from_filesystem` 저장 프로시저를 사용하여 삭제되지 않은 모든 파일은 현재 호스트에서 계속 액세스할 수 있습니다.

errorlog_available_from	recent_failover_time
2020-04-29 23:59:00.0100000	2020-05-05 18:57:51.8900000

RDS for SQL Server와 S3 통합 비활성화

다음으로, Amazon S3와 Amazon RDS for SQL Server의 통합을 비활성화하는 방법을 배울 수 있습니다. S3 통합을 비활성화해도 `D:\S3\`의 파일은 삭제되지 않습니다.

> **Note**
>
> DB 인스턴스에서 IAM 역할을 삭제하려면 DB 인스턴스 상태가 `available`이어야 합니다.

콘솔

DB 인스턴스에서 IAM 역할 연결을 해제하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 세부 정보를 표시하고자 하는 RDS for SQL Server DB 인스턴스 이름을 선택합니다.
3. Connectivity & security(연결성 및 보안) 탭에 있는 Manage IAM roles(IAM 역할 관리) 섹션에서 삭제할 IAM 역할을 선택합니다.
4. Delete(삭제)를 선택합니다.

AWS CLI

RDS for SQL Server DB 인스턴스에서 IAM 역할을 제거하려면

- 다음 AWS CLI 명령은 *mydbinstance*라는 RDS for SQL Server DB 인스턴스에서 IAM 역할을 삭제합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds remove-role-from-db-instance \
    --db-instance-identifier mydbinstance \
    --feature-name S3_INTEGRATION \
    --role-arn your-role-arn
```

  Windows의 경우:

```
aws rds remove-role-from-db-instance ^
    --db-instance-identifier mydbinstance ^
    --feature-name S3_INTEGRATION ^
    --role-arn your-role-arn
```

  *your-role-arn*을 `--feature-name` 옵션에 적합한 IAM 역할 ARN으로 바꿉니다.

Amazon RDS for SQL Server에 Database Mail 사용

Database Mail을 사용하여 Amazon RDS for SQL Server 데이터베이스 인스턴스에서 사용자에게 이메일 메시지를 보낼 수 있습니다. 메시지에는 파일 및 쿼리 결과가 포함될 수 있습니다. Database Mail은 다음 구성 요소를 포함합니다.

- 구성 및 보안 객체 – 이 객체는 프로파일과 계정을 생성하며 msdb 데이터베이스에 저장됩니다.
- 메시징 객체 – 이 객체에는 메시지를 보내는 데 사용되는 sp_send_dbmail 저장 프로시저, 그리고 메시지에 대한 정보가 들어 있는 데이터 구조가 포함됩니다. 이들 객체는 msdb 데이터베이스에 저장됩니다.
- 로깅 및 감사 객체 – Database Mail은 msdb 데이터베이스 및 Microsoft Windows 애플리케이션 이벤트 로그에 로깅 정보를 기록합니다.
- Database Mail 실행 파일 – DatabaseMail.exe은 msdb 데이터베이스의 대기열에서 데이터를 읽고 이메일 메시지를 전송합니다.

RDS는 Web, Standard 및 Enterprise Edition의 모든 SQL Server 버전에 대해 데이터베이스 Database Mail지원합니다.

제한 사항

SQL Server DB 인스턴스에 Database Mail을 사용할 경우 다음 제한 사항이 적용됩니다.

- Database Mail은 SQL Server Express Edition에 대해 지원되지 않습니다.
- Database Mail 구성 파라미터 수정은 지원되지 않습니다. 사전 설정(기본값) 값을 보려면 sysmail_help_configure_sp 저장 프로시저를 사용합니다.
- 첨부 파일은 완전하게 지원되지 않습니다. 자세한 내용은 첨부 파일 작업 섹션을 참조하세요.
- 최대 첨부 파일 크기는 1MB입니다.
- Database Mail에는 다중 AZ DB 인스턴스에 대한 추가 구성이 필요합니다. 자세한 내용은 다중 AZ 배포에 대한 고려 사항 섹션을 참조하세요.
- 미리 정의된 운영자에게 이메일 메시지를 보내도록 SQL Server 에이전트를 구성하는 것은 지원되지 않습니다.

Database Mail 활성화

DB 인스턴스에 대해 Database Mail을 활성화하려면 다음 프로세스를 따릅니다.

1. 새 파라미터 그룹을 생성해야 합니다.
2. 파라미터 그룹을 수정하여 `database mail xps` 파라미터를 1로 설정합니다.
3. 파라미터 그룹을 DB 인스턴스에 연결합니다.

Database Mail의 파라미터 그룹 생성

DB 인스턴스의 SQL Server 에디션 및 버전에 해당하는 `database mail xps` 파라미터의 파라미터 그룹을 생성합니다.

> **Note**
>
> 기존 파라미터 그룹을 수정할 수도 있습니다. Database Mail을 활성화하는 파라미터 수정의 절차를 따르십시오.

콘솔

다음 예에서는 SQL Server Standard Edition 2016에 대한 파라미터 그룹을 생성합니다.

파라미터 그룹을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.
3. [Create parameter group]을 선택합니다.
4. 서브넷 그룹 생성 창에서 다음을 수행합니다.
   a. 파라미터 그룹 패밀리에서 sqlserver-se-13.0을 선택합니다.
   b. 그룹 이름에 파라미터 그룹의 식별자(예: **`dbmail-sqlserver-se-13`**)를 입력합니다.
   c. 설명에 **`Database Mail XPs`**를 입력합니다.
5. 생성(Create)을 선택합니다.

CLI

다음 예에서는 SQL Server Standard Edition 2016에 대한 파라미터 그룹을 생성합니다.

파라미터 그룹을 생성하려면

- 다음 명령 중 하나를 사용합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-parameter-group \
    --db-parameter-group-name dbmail-sqlserver-se-13 \
    --db-parameter-group-family "sqlserver-se-13.0" \
    --description "Database Mail XPs"
```

  Windows의 경우:

```
aws rds create-db-parameter-group ^
    --db-parameter-group-name dbmail-sqlserver-se-13 ^
    --db-parameter-group-family "sqlserver-se-13.0" ^
    --description "Database Mail XPs"
```

Database Mail을 활성화하는 파라미터 수정

SQL Server 에디션 및 DB 인스턴스의 버전에 해당하는 파라미터 그룹의 `database mail xps` 파라미터를 수정합니다.

Database Mail을 활성화하려면 `database mail xps` 파라미터를 1로 설정합니다.

콘솔

다음 예에서는 SQL Server Standard Edition 2016에 대해 생성한 파라미터 그룹을 수정합니다.

파라미터 그룹을 수정하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.
3. 파라미터 그룹(예: ssis-sqlserver-se-13)을 선택합니다.
4. 파라미터에서 파라미터 목록을 **`mail`**로 필터링합니다.
5. [database mail xps]를 선택합니다.

6. 파라미터 편집을 선택합니다.
7. **1**를 입력합니다.
8. Save changes(변경 사항 저장)를 선택합니다.

CLI

다음 예에서는 SQL Server Standard Edition 2016에 대해 생성한 파라미터 그룹을 수정합니다.

파라미터 그룹을 수정하려면

- 다음 명령 중 하나를 사용합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name dbmail-sqlserver-se-13 \
    --parameters "ParameterName='database mail
 xps',ParameterValue=1,ApplyMethod=immediate"
```

  Windows의 경우:

```
aws rds modify-db-parameter-group ^
    --db-parameter-group-name dbmail-sqlserver-se-13 ^
    --parameters "ParameterName='database mail
 xps',ParameterValue=1,ApplyMethod=immediate"
```

파라미터 그룹과 DB 인스턴스 연결

AWS Management Console 또는 AWS CLI를 사용하여 Database Mail 파라미터 그룹을 DB 인스턴스와 연결할 수 있습니다.

콘솔

Database Mail 파라미터 그룹을 신규 또는 기존 DB 인스턴스와 연결할 수 있습니다.

- 새 DB 인스턴스의 경우 인스턴스를 시작할 때 연결합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.

- 기존 DB 인스턴스의 경우 인스턴스를 수정하여 연결합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

CLI

Database Mail 파라미터 그룹을 신규 또는 기존 DB 인스턴스와 연결할 수 있습니다.

Database Mail 파라미터 그룹을 사용하여 DB 인스턴스를 생성하려면

- 파라미터 그룹을 생성할 때 사용한 것과 동일한 DB 엔진 유형과 메이저 버전을 지정합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --db-instance-class db.m5.2xlarge \
    --engine sqlserver-se \
    --engine-version 13.00.5426.0.v1 \
    --allocated-storage 100 \
    --manage-master-user-password \
    --master-username admin \
    --storage-type gp2 \
    --license-model li
    --db-parameter-group-name dbmail-sqlserver-se-13
```

  Windows의 경우:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --db-instance-class db.m5.2xlarge ^
    --engine sqlserver-se ^
    --engine-version 13.00.5426.0.v1 ^
    --allocated-storage 100 ^
    --manage-master-user-password ^
    --master-username admin ^
    --storage-type gp2 ^
    --license-model li ^
    --db-parameter-group-name dbmail-sqlserver-se-13
```

DB 인스턴스를 수정하고 Database Mail 파라미터 그룹을 연결하려면

- 다음 명령 중 하나를 사용합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --db-parameter-group-name dbmail-sqlserver-se-13 \
    --apply-immediately
```

  Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --db-parameter-group-name dbmail-sqlserver-se-13 ^
    --apply-immediately
```

Database Mail 구성

Database Mail을 구성하려면 다음 태스크를 수행합니다.

1. Database Mail 프로파일을 생성합니다.
2. Database Mail 계정을 생성합니다.
3. Database Mail 계정을 Database Mail 프로파일에 추가합니다.
4. Database Mail 프로파일에 사용자를 추가합니다.

> **Note**
>
> Database Mail을 구성하려면 `execute` 데이터베이스의 저장 프로시저에 대한 `msdb` 권한이 있는지 확인합니다.

Database Mail 프로파일 생성

Database Mail 프로파일을 생성하려면 sysmail_add_profile_sp 저장 프로시저를 사용합니다. 다음 예제에서는 Notifications라는 프로파일을 생성합니다.

프로파일을 생성하려면

- 다음 SQL 문을 사용합니다.

```
USE msdb
GO

EXECUTE msdb.dbo.sysmail_add_profile_sp
    @profile_name         = 'Notifications',
    @description          = 'Profile used for sending outgoing notifications using
 Amazon SES.';
GO
```

Database Mail 계정 생성

Database Mail 계정을 생성하려면 sysmail_add_account_sp 저장 프로시저를 사용합니다. 다음 예에서는 Amazon Simple Email Service를 사용하여 프라이빗 VPC 내의 RDS for SQL Server DB 인스턴스에서 SES라는 이름의 계정을 생성합니다.

Amazon SES를 사용하려면 다음 파라미터가 필요합니다.

- @email_address - Amazon SES에서 확인된 자격 증명입니다. 자세한 내용은 Amazon SES에서 확인된 자격 증명을 참조하세요.
- @mailserver_name - Amazon SES SMTP 엔드포인트입니다. 자세한 내용은 Amazon SES SMTP 엔드포인트에 연결을 참조하세요.
- @username - Amazon SES SMTP 사용자 이름입니다. 자세한 내용은 Amazon SES SMTP 자격 증명 받기를 참조하세요.

  AWS Identity and Access Management 사용자 이름을 사용하지 마세요.
- @password - Amazon SES SMTP 암호입니다. 자세한 내용은 Amazon SES SMTP 자격 증명 받기를 참조하세요.

계정을 생성하려면

- 다음 SQL 문을 사용합니다.

```
USE msdb
GO

EXECUTE msdb.dbo.sysmail_add_account_sp
    @account_name        = 'SES',
    @description         = 'Mail account for sending outgoing notifications.',
    @email_address       = 'nobody@example.com',
    @display_name        = 'Automated Mailer',
    @mailserver_name     = 'vpce-0a1b2c3d4e5f-01234567.email-smtp.us-
west-2.vpce.amazonaws.com',
    @port                = 587,
    @enable_ssl          = 1,
    @username            = 'Smtp_Username',
    @password            = 'Smtp_Password';
GO
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 보안 인증 정보를 지정하는 것이 좋습니다.

Database Mail 프로파일에 Database Mail 계정 추가

Database Mail 계정을 Database Mail 프로파일에 추가하려면 sysmail_add_profileaccount_sp 저장 프로시저를 사용합니다. 다음 예에서는 SES 프로파일에 Notifications 계정을 추가합니다.

프로파일에 계정을 추가하려면

- 다음 SQL 문을 사용합니다.

```
USE msdb
GO

EXECUTE msdb.dbo.sysmail_add_profileaccount_sp
    @profile_name        = 'Notifications',
    @account_name        = 'SES',
```

```
    @sequence_number     = 1;
GO
```

Database Mail 프로파일에 사용자 추가

msdb 데이터베이스 보안 주체에 Database Mail 프로파일을 사용할 권한을 부여하려면 sysmail_add_principalprofile_sp 저장 프로시저를 사용합니다. 보안 주체는 SQL Server 리소스를 요청할 수 있는 엔터티입니다. 데이터베이스 보안 주체는 SQL Server 인증 사용자, Windows 인증 사용자 또는 Windows 인증 그룹에 매핑되어야 합니다.

다음 예에서는 Notifications 프로파일에 대한 퍼블릭 액세스 권한을 부여합니다.

프로파일에 사용자를 추가하려면

- 다음 SQL 문을 사용합니다.

```
USE msdb
GO

EXECUTE msdb.dbo.sysmail_add_principalprofile_sp
    @profile_name       = 'Notifications',
    @principal_name     = 'public',
    @is_default         = 1;
GO
```

Amazon RDS Database Mail에 대한 저장 프로시저 및 함수

Microsoft는 계정 및 프로필 생성, 나열, 업데이트, 삭제하는 등, Database Mail을 사용할 수 있도록 저장 프로시저를 제공합니다. 또한 RDS는 다음 표에 나와 있는 Database Mail용 저장 프로시저 및 함수도 제공합니다.

프로시저/함수	설명
rds_fn_sysmail_allitems	다른 사용자가 전송한 메시지를 포함하여 전송된 메시지를 표시합니다.
rds_fn_sysmail_event_log	다른 사용자가 전송한 메시지에 대한 이벤트를 포함하여 이벤트를 표시합니다.

프로시저/함수	설명
rds_fn_sysmail_mailattachments	다른 사용자가 전송한 메시지의 첨부 파일을 포함하여 첨부 파일을 표시합니다.
rds_sysmail_control	메일 대기열을 시작하고 중지합니다(DatabaseMail.exe 프로세스).
rds_sysmail_delete_mailitems_sp	Database Mail 내부 테이블에서 모든 사용자가 보낸 이메일 메시지를 삭제합니다.

Database Mail을 사용하여 이메일 메시지 보내기

sp_send_dbmail 저장 프로시저를 사용하여 Database Mail을 통해 이메일 메시지를 보낼 수 있습니다.

사용량

```
EXEC msdb.dbo.sp_send_dbmail
@profile_name = 'profile_name',
@recipients = 'recipient1@example.com[; recipient2; ... recipientn]',
@subject = 'subject',
@body = 'message_body',
[@body_format = 'HTML'],
[@file_attachments = 'file_path1; file_path2; ... file_pathn'],
[@query = 'SQL_query'],
[@attach_query_result_as_file = 0|1]';
```

다음 파라미터는 필수 파라미터입니다.

- @profile_name – 메시지를 보낼 Database Mail 프로파일의 이름입니다.
- @recipients – 메시지를 보낼 이메일 주소의 세미콜론으로 구분된 목록입니다.
- @subject – 메시지의 제목입니다.
- @body – 메시지의 본문입니다. 선언된 변수를 본문으로 사용할 수도 있습니다.

다음 파라미터는 선택적입니다.

- @body_format – 이 파라미터는 HTML 형식으로 이메일을 보내도록 선언된 변수와 함께 사용됩니다.

- `@file_attachments` – 메시지 첨부 파일의 세미콜론으로 구분된 목록입니다. 파일 경로는 절대 경로여야 합니다.
- `@query` – 실행할 SQL 쿼리입니다. 쿼리 결과는 파일로 첨부되거나 메시지 본문에 포함될 수 있습니다.
- `@attach_query_result_as_file` – 쿼리 결과를 파일로 첨부할지 여부를 나타냅니다. 아니요(No)인 경우 0, 예(Yes)인 경우 1로 설정합니다. 기본값은 0입니다.

예시

다음 예에서는 이메일 메시지를 보내는 방법을 보여 줍니다.

Example 한 명의 수신자에게 메시지 전송

```
USE msdb
GO

EXEC msdb.dbo.sp_send_dbmail
     @profile_name       = 'Notifications',
     @recipients         = 'nobody@example.com',
     @subject            = 'Automated DBMail message - 1',
     @body               = 'Database Mail configuration was successful.';
GO
```

Example 여러 수신자에게 메시지 전송

```
USE msdb
GO

EXEC msdb.dbo.sp_send_dbmail
     @profile_name       = 'Notifications',
     @recipients         = 'recipient1@example.com;recipient2@example.com',
     @subject            = 'Automated DBMail message - 2',
     @body               = 'This is a message.';
GO
```

Example SQL 쿼리 결과를 첨부 파일로 전송

```
USE msdb
GO
```

```
EXEC msdb.dbo.sp_send_dbmail
     @profile_name       = 'Notifications',
     @recipients         = 'nobody@example.com',
     @subject            = 'Test SQL query',
     @body               = 'This is a SQL query test.',
     @query              = 'SELECT * FROM abc.dbo.test',
     @attach_query_result_as_file = 1;
GO
```

Example HTML 형식으로 메시지 전송

```
USE msdb
GO

DECLARE @HTML_Body as NVARCHAR(500) = 'Hi, <h4> Heading </h4> </br> See the report. <b>
 Regards </b>';

EXEC msdb.dbo.sp_send_dbmail
     @profile_name       = 'Notifications',
     @recipients         = 'nobody@example.com',
     @subject            = 'Test HTML message',
     @body               = @HTML_Body,
     @body_format        = 'HTML';
GO
```

Example 데이터베이스에서 특정 이벤트가 발생할 때 트리거를 사용하여 메시지 전송

```
USE AdventureWorks2017
GO
IF OBJECT_ID ('Production.iProductNotification', 'TR') IS NOT NULL
DROP TRIGGER Purchasing.iProductNotification
GO

CREATE TRIGGER iProductNotification ON Production.Product
   FOR INSERT
   AS
   DECLARE @ProductInformation nvarchar(255);
   SELECT
   @ProductInformation = 'A new product, ' + Name + ', is now available for $' +
 CAST(StandardCost AS nvarchar(20)) + '!'
   FROM INSERTED i;
```

```
EXEC msdb.dbo.sp_send_dbmail
     @profile_name       = 'Notifications',
     @recipients         = 'nobody@example.com',
     @subject            = 'New product information',
     @body               = @ProductInformation;
GO
```

메시지, 로그 및 첨부 파일 보기

RDS 저장 프로시저를 사용하여 메시지, 이벤트 로그 및 첨부 파일을 볼 수 있습니다.

모든 이메일 메시지를 보려면

- 다음 SQL 쿼리를 사용합니다.

```
SELECT * FROM msdb.dbo.rds_fn_sysmail_allitems(); --WHERE sent_status='sent' or
 'failed' or 'unsent'
```

모든 이메일 이벤트 로그를 보려면

- 다음 SQL 쿼리를 사용합니다.

```
SELECT * FROM msdb.dbo.rds_fn_sysmail_event_log();
```

모든 이메일 첨부 파일을 보려면

- 다음 SQL 쿼리를 사용합니다.

```
SELECT * FROM msdb.dbo.rds_fn_sysmail_mailattachments();
```

메시지 삭제

`rds_sysmail_delete_mailitems_sp` 저장 프로시저를 사용하여 메시지를 삭제합니다.

> **Note**
>
> RDS는 DBMail 기록 데이터의 크기가 1GB에 도달하고 보존 기간이 24시간 이상인 경우 메일 테이블 항목을 자동으로 삭제합니다.

메일 항목을 장기간 보존하려는 경우 아카이빙할 수 있습니다. 자세한 내용은 Microsoft 설명서의 Database Mail 메시지 및 이벤트 로그를 아카이빙하기 위한 SQL Server 에이전트 작업 생성을 참조하세요.

모든 이메일 메시지를 삭제하려면

- 다음 SQL 문을 사용합니다.

```
DECLARE @GETDATE datetime
SET @GETDATE = GETDATE();
EXECUTE msdb.dbo.rds_sysmail_delete_mailitems_sp @sent_before = @GETDATE;
GO
```

특정 상태의 이메일 메시지를 모두 삭제하려면

- 실패한 메시지를 모두 삭제하려면 다음 SQL 문을 사용합니다.

```
DECLARE @GETDATE datetime
SET @GETDATE = GETDATE();
EXECUTE msdb.dbo.rds_sysmail_delete_mailitems_sp @sent_status = 'failed';
GO
```

메일 대기열 시작 및 중지

다음 지침을 사용하여 DB 메일 대기열을 시작하고 중지합니다.

주제

- 메일 대기열 시작
- 메일 대기열 중지

메일 대기열 시작

`rds_sysmail_control` 저장 프로시저를 사용하여 Database Mail 프로세스를 시작합니다.

> **Note**
>
> Database Mail을 활성화하면 메일 대기열이 자동으로 시작됩니다.

메일 대기열을 시작하려면

- 다음 SQL 문을 사용합니다.

```
EXECUTE msdb.dbo.rds_sysmail_control start;
GO
```

메일 대기열 중지

`rds_sysmail_control` 저장 프로시저를 사용하여 Database Mail 프로세스를 중지합니다.

메일 대기열을 중지하려면

- 다음 SQL 문을 사용합니다.

```
EXECUTE msdb.dbo.rds_sysmail_control stop;
GO
```

첨부 파일 작업

다음 첨부 파일 확장명은 RDS on SQL Server에서 보낸 Database Mail 메시지에서 지원되지 않습니다. .ade, .adp, .apk, .appx, .appxbundle, .bat, .bak, .cab, .chm, .cmd, .com, .cpl, .dll, .dmg, .exe, .hta, .inf1, .in

Database Mail은 현재 사용자의 Microsoft Windows 보안 컨텍스트를 사용하여 파일에 대한 액세스를 제어합니다. SQL Server 인증을 사용하여 로그인하는 사용자는 `@file_attachments` 저장 프로시저와 함께 `sp_send_dbmail` 파라미터를 사용하여 파일을 첨부할 수 없습니다. Windows에서는 SQL Server가 원격 컴퓨터에서 다른 원격 컴퓨터로 자격 증명을 제공할 수 없습니다. 따라서 SQL Server를 실행하는 컴퓨터가 아닌 다른 컴퓨터에서 이 명령을 실행하면 Database Mail이 네트워크 공유에서 파일을 연결할 수 없습니다.

하지만 SQL Server 에이전트 작업을 사용하면 파일을 첨부할 수 있습니다. SQL Server 에이전트에 대한 자세한 내용은 Microsoft 설명서에서 Amazon RDS에 SQL Server Agent 사용 및 SQL Server 에이전트를 참조하세요.

다중 AZ 배포에 대한 고려 사항

다중 AZ DB 인스턴스에 Database Mail을 구성하면 해당 구성이 보조 인스턴스에 자동으로 전파되지 않습니다. 다중 AZ 인스턴스를 단일 AZ 인스턴스로 변환하고 Database Mail을 구성한 다음 DB 인스턴스를 다시 다중 AZ로 변환하는 것이 좋습니다. 그러면 기본 노드와 보조 노드 모두에 Database Mail 구성이 적용됩니다.

Database Mail이 구성된 다중 AZ 인스턴스에서 읽기 전용 복제본을 생성하는 경우 해당 복제본은 구성을 상속하지만 SMTP 서버에 대한 암호는 상속하지 않습니다. 해당 암호로 Database Mail 계정을 업데이트합니다.

SMTP(포트 25) 제한 사항 제거

기본적으로 AWS는 RDS for SQL Server DB 인스턴스에 대한 SMTP(포트 25)의 아웃바운드 트래픽을 차단합니다. 이는 탄력적 네트워크 인터페이스 소유자의 정책에 따라 스팸을 방지하기 위해 수행됩니다. 필요한 경우 이 제한사항을 제거할 수 있습니다. 자세한 내용은 에서 Amazon EC2 인스턴스 또는 Lamda 함수에서 포트 25와 관련한 제한을 없애려면 어떻게 해야 하나요?를 참조하세요.

Amazon RDS for SQL Server의 tempdb 데이터베이스에 대한 인스턴스 스토어 지원

인스턴스 스토어는 DB 인스턴스에 블록 수준의 임시 스토리지를 제공합니다. 스토리지는 호스트 컴퓨터에 물리적으로 연결된 디스크에 위치합니다. 이러한 디스크에는 SSD(솔리드 스테이트 드라이브)를 기반으로 하는 NVMe(Non-Volatile Memory Express) 인스턴스 스토리지가 있습니다. 이 스토리지는 짧은 지연 시간, 매우 높은 임의 I/O 성능, 높은 순차 읽기 처리량에 최적화되어 있습니다.

인스턴스 스토어에 `tempdb` 데이터 파일과 `tempdb` 로그 파일을 배치하면 Amazon EBS를 기반으로 하는 표준 스토리지에 비해 읽기 및 쓰기 지연 시간을 줄일 수 있습니다.

> **Note**
>
> SQL Server 데이터베이스 파일 및 데이터베이스 로그 파일은 인스턴스 스토어에 저장되지 않습니다.

인스턴스 스토어 활성화

RDS가 다음 인스턴스 클래스 중 하나로 DB 인스턴스를 프로비저닝하면 `tempdb` 데이터베이스가 인스턴스 스토어에 자동으로 배치됩니다.

- db.m5d
- db.r5d
- db.x2iedn

인스턴스 스토어를 활성화하려면 다음 중 하나를 수행합니다.

- 이러한 인스턴스 유형 중 하나를 사용하여 SQL Server DB 인스턴스를 생성합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
- 기존 SQL Server DB 인스턴스 중 하나를 사용할 수 있도록 수정합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

인스턴스 스토어는 이러한 인스턴스 유형 중 하나 이상이 지원되는 모든 AWS 리전에서 사용 가능합니다. `db.m5d` 및 `db.r5d` 인스턴스 클래스에 대한 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요. Amazon RDS for SQL Server에서 지원하는 인스턴스 클래스에 대한 자세한 내용은 Microsoft SQL Server를 위한 DB 인스턴스 클래스 지원 단원을 참조하세요.

파일 위치와 크기 고려 사항

인스턴스 스토어가 없는 인스턴스에서는 RDS가 tempdb 데이터 및 로그 파일을 D:\rdsdbdata\DATA 디렉터리에 저장합니다. 두 파일은 기본적으로 8MB에서 시작합니다.

인스턴스 스토어가 있는 인스턴스에서는 RDS가 tempdb 데이터 및 로그 파일을 T:\rdsdbdata\DATA 디렉터리에 저장합니다.

tempdb에는 하나의 데이터 파일(tempdb.mdf)과 하나의 로그 파일(templog.ldf)만 있지만, templog.ldf는 기본적으로 8MB에서 시작하며, tempdb.mdf는 인스턴스 스토리지 용량의 80% 이상에서 시작합니다. 스토리지 용량의 20% 또는 200GB 중 더 적은 용량에서 자유롭게 시작할 수 있습니다. 여러 tempdb 데이터 파일은 80% 디스크 공간을 균등하게 분할하지만 로그 파일은 항상 초기 크기가 8MB입니다.

예를 들어 DB 인스턴스 클래스를 db.m5.2xlarge에서 db.m5d.2xlarge로 수정하면 tempdb 데이터 파일 크기가 각각 8MB에서 총 234GB로 늘어납니다.

> **Note**
>
> 인스턴스 스토어(tempdb)의 T:\rdsdbdata\DATA 데이터 및 로그 파일 외에, 데이터 볼륨(tempdb)에도 D:\rdsdbdata\DATA 데이터 및 로그 파일을 추가로 생성할 수 있습니다. 해당 파일의 초기 크기는 항상 8MB입니다.

백업 고려 사항

오랫동안 백업을 보존해야 할 수 있으므로 시간이 지나면서 비용이 발생합니다. tempdb 데이터와 로그 블록은 워크로드에 따라 매우 자주 변경될 수 있습니다. 이렇게 하면 DB 스냅샷 크기가 크게 증가할 수 있습니다.

tempdb이(가) 인스턴스 스토어에 있는 경우 스냅샷에 임시 파일이 포함되지 않습니다. 즉, EBS 전용 스토리지에 비해 스냅샷 크기가 더 작고 더 적은 여유 백업 할당량을 사용합니다.

디스크 가득 참 오류

인스턴스 스토어에서 사용 가능한 공간을 모두 사용하면 다음과 같은 오류가 발생할 수 있습니다.

- 데이터베이스 'tempdb'에 대한 트랜잭션 로그가 'ACTIVE_TRANSACTION' 때문에 가득 찼습니다.
- PRIMARY' 파일 그룹이 가득 찼기 때문에 데이터베이스 'tempdb'에 있는 객체 'dbo.SORT 임시 실행 스토리지: 140738941419520’에 대한 공간을 할당할 수 없습니다. 불필요한 파일을 삭제하거나, 파

일 그룹에서 객체를 삭제하거나, 파일 그룹에 파일을 추가하거나, 파일 그룹의 기존 파일에 대해 자동 증가를 설정하여 디스크 공간을 생성합니다.

인스턴스 스토어가 가득 차면 다음 중 하나 이상을 수행할 수 있습니다.

- 워크로드 또는 `tempdb` 사용 방식을 조정합니다.
- DB 인스턴스 클래스를 더 많은 NVMe 스토리지와 함께 사용하도록 확장합니다.
- 인스턴스 스토어 사용을 중지하고, EBS 스토리지만 있는 인스턴스 클래스를 사용합니다.
- EBS 볼륨에서 `tempdb`에 대한 보조 데이터 또는 로그 파일을 추가하여 혼합 모드를 사용합니다.

인스턴스 스토어 제거

인스턴스 스토어를 제거하려면 db.m5, db.r5 또는 db.x1e등 인스턴스 스토어를 지원하지 않는 인스턴스 유형을 사용하도록 SQL Server DB 인스턴스를 수정합니다.

> **Note**
>
> 인스턴스 스토어를 제거하면 임시 파일이 `D:\rdsdbdata\DATA` 디렉터리로 이동되고 크기가 8MB로 줄어듭니다.

Amazon RDS for Microsoft SQL Server에 확장 이벤트 사용

Microsoft SQL Server의 확장 이벤트를 사용하여 Amazon RDS for SQL Server에 대한 디버깅 및 문제 해결 정보를 캡처할 수 있습니다. 확장 이벤트는 Microsoft에서 더 이상 사용되지 않는 SQL 추적 및 서버 프로파일러를 대체합니다. 확장 이벤트는 프로파일러 추적과 유사하지만 추적되는 이벤트를 보다 세부적으로 제어할 수 있습니다. Amazon RDS에서 확장 이벤트는 SQL Server 버전 2016 이상에 대해 지원됩니다. 자세한 내용은 Microsoft 설명서의 Extended events overview를 참조하세요.

확장 이벤트는 Amazon RDS for SQL Server에서 마스터 사용자 권한이 있는 사용자에 대해 자동으로 활성화됩니다.

주제

- 제한 및 권장 사항
- RDS for SQL Server에 대한 확장 이벤트 구성
- 다중 AZ 배포에 대한 고려 사항
- 확장 이벤트 파일 쿼리

제한 및 권장 사항

RDS for SQL Server에서 확장 이벤트를 사용하는 경우 다음과 같은 제한 사항이 적용됩니다.

- 확장 이벤트는 Enterprise 및 Standard Edition에 대해서만 지원됩니다.
- 기본 확장 이벤트 세션은 변경할 수 없습니다.
- 세션 메모리 파티션 모드를 `NONE`으로 설정해야 합니다.
- 세션 이벤트 보존 모드는 `ALLOW_SINGLE_EVENT_LOSS` 또는 `ALLOW_MULTIPLE_EVENT_LOSS`일 수 있습니다.
- ETW(Event Tracing for Windows) 대상은 지원되지 않습니다.
- 파일 대상이 `D:\rdsdbdata\log` 디렉터리에 있어야 합니다.
- 페어 매칭 대상의 경우 `respond_to_memory_pressure` 속성을 1로 설정합니다.
- 링 버퍼 대상 메모리는 4MB를 초과할 수 없습니다.
- 다음은 지원되지 않는 작업입니다.
  - `debug_break`
  - `create_dump_all_threads`

  - `create_dump_single_threads`
- `rpc_completed` 이벤트는 다음 버전 이상에서 지원됩니다. 15.0.4083.2, 14.0.3370.1, 13.0.5865.1, 12.0.6433.1, 11.0.7507.2.

RDS for SQL Server에 대한 확장 이벤트 구성

RDS for SQL Server에서 확장 이벤트 세션의 특정 파라미터 값을 구성할 수 있습니다. 다음 표에서는 구성 가능한 파라미터에 대해 설명합니다.

파라미터 이름	설명
`xe_session_max_memory`	이벤트 버퍼링을 위해 세션에 할당할 최대 메모리 양을 지 에 해당합니다.
`xe_session_max_event_size`	큰 이벤트에 허용되는 최대 메모리 크기를 지정합니다. 이 합니다.
`xe_session_max_dispatch_latency`	이벤트가 확장 이벤트 세션 대상으로 전달되기 전에 메모 세션의 `max_dispatch_latency` 설정에 해당합니다.
`xe_file_target_size`	파일 대상의 최대 크기를 지정합니다. 이 값은 파일 대상의
`xe_file_retention`	이벤트 세션의 파일 대상에서 생성된 파일의 보존 시간(일

> **Note**
>
> `xe_file_retention`을 0으로 설정하면 SQL Server가 해당 파일에 대한 잠금을 해제한 후에 .xel 파일이 자동으로 제거됩니다. .xel 파일이 `xe_file_target_size`에 설정된 크기 제한에 도달할 때마다 잠금이 해제됩니다.

`rdsadmin.dbo.rds_show_configuration` 저장 프로시저를 사용하여 이러한 파라미터의 현재 값을 표시할 수 있습니다. 예를 들어 `xe_session_max_memory`의 현재 설정을 보려면 다음 SQL 문을 사용합니다.

```
exec rdsadmin.dbo.rds_show_configuration 'xe_session_max_memory'
```

`rdsadmin.dbo.rds_set_configuration` 저장 프로시저를 사용하여 설정을 수정할 수 있습니다. 예를 들어 `xe_session_max_memory`를 4MB로 설정하려면 다음 SQL 문을 사용합니다.

```
exec rdsadmin.dbo.rds_set_configuration 'xe_session_max_memory', 4
```

다중 AZ 배포에 대한 고려 사항

기본 DB 인스턴스에서 확장 이벤트 세션을 생성하면 예비 복제본으로 전파되지 않습니다. 장애 조치하고, 새 기본 DB 인스턴스에서 확장 이벤트 세션을 생성할 수 있습니다. 또는 다중 AZ 구성을 제거한 다음 다시 추가하여 확장 이벤트 세션을 대기 복제본에 전파할 수 있습니다. RDS는 대기 복제본에서 기본이 아닌 모든 확장 이벤트 세션을 중지하므로, 이러한 세션은 대기 복제본에서 리소스를 소모하지 않습니다. 따라서 예비 복제본이 기본 DB 인스턴스가 된 후에는 새 기본 복제본에서 확장 이벤트 세션을 수동으로 시작해야 합니다.

> **Note**
>
> 이 방법은 Always On 가용성 그룹과 데이터베이스 미러링에 모두 적용됩니다.

SQL Server 에이전트 작업을 사용하여 대기 복제본을 추적하고, 대기 복제본이 기본 복제본이 될 경우 세션을 시작할 수도 있습니다. 예를 들어 SQL Server 에이전트 작업 단계에서 다음 쿼리를 사용하여 기본 DB 인스턴스의 이벤트 세션을 다시 시작합니다.

```
BEGIN
    IF (DATABASEPROPERTYEX('rdsadmin','Updateability')='READ_WRITE'
    AND DATABASEPROPERTYEX('rdsadmin','status')='ONLINE'
    AND (DATABASEPROPERTYEX('rdsadmin','Collation') IS NOT NULL OR
 DATABASEPROPERTYEX('rdsadmin','IsAutoClose')=1)
    )
    BEGIN
        IF NOT EXISTS (SELECT 1 FROM sys.dm_xe_sessions WHERE name='xe1')
            ALTER EVENT SESSION xe1 ON SERVER STATE=START
        IF NOT EXISTS (SELECT 1 FROM sys.dm_xe_sessions WHERE name='xe2')
            ALTER EVENT SESSION xe2 ON SERVER STATE=START
    END
END
```

이 쿼리는 해당 세션이 중지 상태인 경우, 기본 DB 인스턴스에서 이벤트 세션 `xe1` 및 `xe2`를 다시 시작합니다. 이 쿼리에 편리한 간격으로 일정을 추가할 수도 있습니다.

확장 이벤트 파일 쿼리

SQL Server Management Studio 또는 `sys.fn_xe_file_target_read_file` 함수를 사용하여 파일 대상을 사용하는 확장 이벤트의 데이터를 볼 수 있습니다. 이 함수에 대한 자세한 내용은 Microsoft 설명서의 sys.fn_xe_file_target_read_file(Transact-SQL)을 참조하세요.

확장 이벤트 파일 대상은 RDS for SQL Server의 `D:\rdsdbdata\log` 디렉터리에만 파일을 쓸 수 있습니다.

예를 들어 이름이 `xe`로 시작하는 확장 이벤트 세션의 모든 파일의 내용을 나열하려면 다음 SQL 쿼리를 사용합니다.

```
SELECT * FROM sys.fn_xe_file_target_read_file('d:\rdsdbdata\log\xe*', null,null,null);
```

RDS for SQL Server를 사용하여 트랜잭션 로그 백업에 액세스

RDS for SQL Server 트랜잭션 로그 백업에 액세스하면 데이터베이스의 트랜잭션 로그 백업 파일을 나열하고 대상 Amazon S3 버킷에 복사할 수 있습니다. Amazon S3 버킷의 트랜잭션 로그 백업을 복사하면 전체 및 차등 데이터베이스 백업과 함께 사용하여 특정 시점 데이터베이스 복원을 수행할 수 있습니다. RDS 저장 프로시저를 사용하여 트랜잭션 로그 백업에 대한 액세스를 설정하고, 사용 가능한 트랜잭션 로그 백업을 나열하고, 이를 Amazon S3 버킷에 복사합니다.

트랜잭션 로그 백업에 대한 액세스가 제공하는 기능과 이점은 다음과 같습니다.

- RDS for SQL Server DB 인스턴스에 있는 데이터베이스의 사용 가능한 트랜잭션 로그 백업의 메타데이터를 나열하고 볼 수 있습니다.
- RDS for SQL Server에서 대상 Amazon S3 버킷으로 사용 가능한 트랜잭션 로그 백업을 복사합니다.
- 전체 DB 인스턴스를 복원할 필요 없이 특정 시점 데이터베이스 복원을 수행합니다. DB 인스턴스 특정 시점 복원에 대한 자세한 내용은 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원을 참조하세요.

가용성 및 지원

트랜잭션 로그 백업에 대한 액세스는 모든 AWS 리전에서 지원합니다. 트랜잭션 로그 백업에 대한 액세스는 Amazon RDS에서 지원되는 모든 버전의 Microsoft SQL Server에서 사용할 수 있습니다.

요구 사항

트랜잭션 로그 백업에 대한 액세스를 활성화하려면 먼저 다음 요구 사항을 충족해야 합니다.

- DB 인스턴스에서 자동 백업을 활성화하고 백업 보존을 1일 이상의 값으로 설정해야 합니다. 자동 백업 활성화 및 보존 정책 구성에 대한 자세한 내용은 자동 백업 활성화을 참조하세요.
- Amazon S3 버킷이 소스 DB 인스턴스와 동일한 계정 및 리전에 있어야 합니다. 트랜잭션 로그 백업에 대한 액세스를 활성화하기 전에 트랜잭션 로그 백업 파일에 사용할 기존 Amazon S3 버킷을 선택하거나 새 버킷을 생성합니다.
- Amazon RDS가 트랜잭션 로그 파일을 복사할 수 있도록 Amazon S3 버킷 권한 정책을 다음과 같이 구성해야 합니다.
  1. 버킷의 객체 계정 소유권 속성을 Bucket Owner Preferred(버킷 소유자 선호)로 설정합니다.
  2. 다음 정책을 추가합니다. 기본적으로 정책이 없으므로 버킷 ACL(액세스 제어 목록)을 사용하여 버킷 정책을 편집하고 추가합니다.

다음 예에서는 ARN을 사용하여 리소스를 지정합니다. 서비스 권한을 특정 리소스로 제한하는 리소스 기반 신뢰 관계의 SourceArn 및 SourceAccount 전역 조건 컨텍스트 키를 사용하는 것이 좋습니다. ARN 작업에 대한 자세한 내용은 Amazon 리소스 이름(ARN) 및 Amazon RDS의 Amazon 리소스 이름(ARN)을 참조하세요.

Example 트랜잭션 로그 백업 액세스를 위한 Amazon S3 권한 정책 예

JSON

```
   {
   "Version": "2012-10-17",
   "Statement": [
       {
           "Sid": "Only allow writes to my bucket with bucket owner full
 control",
           "Effect": "Allow",
           "Principal": {
               "Service": "backups.rds.amazonaws.com"
           },
           "Action": "s3:PutObject",
           "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/{customer_path}/*",
           "Condition": {
               "StringEquals": {
                   "s3:x-amz-acl": "bucket-owner-full-control",
                   "aws:sourceAccount": "{customer_account}",
                   "aws:sourceArn": "{db_instance_arn}"
               }
           }
       }
   ]
}
```

- Amazon S3 버킷에 액세스하기 위한 AWS Identity and Access Management(IAM) 역할. 이미 IAM 역할이 있으면 그 역할을 사용하면 됩니다. AWS Management Console을 사용하여 SQLSERVER_BACKUP_RESTORE 옵션을 추가할 때 새 IAM 역할이 생성되도록 선택할 수도 있습니다. 또는 수동으로 역할을 새로 만들 수 있습니다. SQLSERVER_BACKUP_RESTORE를 사용하여 IAM 역할을 생성하고 구성하는 방법에 대한 자세한 내용은 기본 백업 및 복원을 위한 IAM 역할 수동으로 만들기을 참조하세요.

- DB 인스턴스의 옵션 그룹에 SQLSERVER_BACKUP_RESTORE 옵션을 추가해야 합니다. SQLSERVER_BACKUP_RESTORE 옵션 추가 방법에 대한 자세한 내용은 SQL Server에서 기본 백업 및 복원 지원을 참조하세요.

  > **Note**
  >
  > DB 인스턴스에 스토리지 암호화가 활성화된 경우 기본 백업 및 복원 옵션 그룹에 제공된 IAM 역할에 AWS KMS(KMS) 작업 및 키를 제공해야 합니다.

  필요에 따라 rds_restore_log 저장 프로시저를 사용하여 특정 시점 데이터베이스 복원을 수행하려는 경우, 기본 백업 및 복원 옵션 그룹과 트랜잭션 로그 백업에 대한 액세스에 동일한 Amazon S3 경로를 사용하는 것이 좋습니다. 이 방법을 사용하면 Amazon RDS가 옵션 그룹의 역할을 맡아 복원 로그 기능을 수행할 때 동일한 Amazon S3 경로에서 트랜잭션 로그 백업을 검색할 수 있습니다.
- DB 인스턴스가 암호화된 경우 암호화 유형(AWS 관리형 키 또는 고객 관리형 키)과 관계없이 IAM 역할 및rds_tlog_backup_copy_to_S3 저장 프로시저에 고객 관리형 KMS 키를 제공해야 합니다.

제한 및 권장 사항

트랜잭션 로그 백업에 대한 액세스에는 다음과 같은 제한 및 권장 사항이 있습니다.

- 백업 보존 기간이 1~35일로 구성된 DB 인스턴스의 경우 최대 지난 7일간의 트랜잭션 로그 백업을 나열하고 복사할 수 있습니다.
- 트랜잭션 로그 백업 액세스에 사용되는 Amazon S3 버킷은 소스 DB 인스턴스와 동일한 계정 및 리전에 있어야 합니다. 계정 간 및 리전 간 복사는 지원되지 않습니다.
- 하나의 Amazon S3 버킷만 트랜잭션 로그 백업을 복사할 대상으로 구성할 수 있습니다. rds_tlog_copy_setup 저장 프로시저를 사용하여 새 대상 Amazon S3 버킷을 선택할 수 있습니다. 새 대상 Amazon S3 버킷 선택에 대한 자세한 내용은 트랜잭션 로그 백업에 대한 액세스 설정을 참조하세요.
- RDS 인스턴스에 스토리지 암호화가 활성화되지 않은 경우 rds_tlog_backup_copy_to_S3 저장 프로시저를 사용할 때 KMS 키를 지정할 수 없습니다.
- 다중 계정 복사는 지원되지 않습니다. 복사에 사용되는 IAM 역할은 DB 인스턴스 소유자 계정 내의 Amazon S3 버킷에 대한 쓰기 액세스만 허용합니다.
- RDS for SQL Server DB 인스턴스에서는 유형에 관계없이 두 개의 동시 작업만 실행할 수 있습니다.

- 지정된 시간에 단일 데이터베이스에 대해 하나의 복사 작업만 실행할 수 있습니다. DB 인스턴스에 있는 여러 데이터베이스의 트랜잭션 로그 백업을 복사하려면 각 데이터베이스마다 별도의 복사 작업을 사용하세요.
- Amazon S3 버킷에 동일한 이름으로 이미 존재하는 트랜잭션 로그 백업을 복사하는 경우 기존 트랜잭션 로그 백업을 덮어씁니다.
- 기본 DB 인스턴스의 트랜잭션 로그 백업에 액세스할 수 있는 저장 프로시저만 실행할 수 있습니다. RDS for SQL Server 읽기 전용 복제본이나 다중 AZ DB 클러스터의 보조 인스턴스에서는 이러한 저장 프로시저를 실행할 수 없습니다.
- `rds_tlog_backup_copy_to_S3` 저장 프로시저가 실행되는 동안 RDS for SQL Server DB 인스턴스를 재부팅하면 DB 인스턴스가 다시 온라인 상태가 될 때 작업이 처음부터 자동으로 다시 시작됩니다. 재부팅 전에 작업이 실행되는 동안 Amazon S3 버킷에 복사된 모든 트랜잭션 로그 백업이 덮어쓰기됩니다.
- Microsoft SQL Server 시스템 데이터베이스와 `RDSAdmin` 데이터베이스는 트랜잭션 로그 백업에 액세스하도록 구성할 수 없습니다.
- SSE-KMS로 암호화된 버킷에 복사하는 것은 지원되지 않습니다.

트랜잭션 로그 백업에 대한 액세스 설정

트랜잭션 로그 백업에 대한 액세스를 설정하려면 요구 사항 섹션의 요구 사항 목록을 완료한 다음 `rds_tlog_copy_setup` 저장 프로시저를 실행합니다. 이 프로시저를 통해 DB 인스턴스 수준에서 트랜잭션 로그 백업 기능에 액세스할 수 있습니다. DB 인스턴스의 개별 데이터베이스마다 이를 실행할 필요는 없습니다.

> **Important**
>
> 각 데이터베이스의 SQL Server 내에서 트랜잭션 로그 백업에 대한 액세스를 구성하고 사용할 수 있는 `db_owner` 역할을 데이터베이스 사용자에게 부여해야 합니다.

Example 사용법:

```
exec msdb.dbo.rds_tlog_copy_setup
@target_s3_arn='arn:aws:s3:::amzn-s3-demo-bucket/myfolder';
```

다음 파라미터는 필수입니다.

- `@target_s3_arn` - 트랜잭션 로그 백업 파일을 복사할 대상 Amazon S3 버킷의 ARN입니다.

Example Amazon S3 대상 버킷 설정 예:

```
exec msdb.dbo.rds_tlog_copy_setup @target_s3_arn='arn:aws:s3:::amzn-s3-demo-logging-bucket/mytestdb1';
```

구성을 검증하려면 `rds_show_configuration` 저장 프로시저를 호출합니다.

Example 구성 검증 예:

```
exec rdsadmin.dbo.rds_show_configuration @name='target_s3_arn_for_tlog_copy';
```

트랜잭션 로그 백업에 대한 액세스를 수정하여 다른 Amazon S3 버킷을 가리키도록 하려면 현재 Amazon S3 버킷 값을 확인하고 `@target_s3_arn`의 새 값을 사용하여 `rds_tlog_copy_setup` 저장 프로시저를 다시 실행하면 됩니다.

Example 트랜잭션 로그 백업 액세스가 구성된 기존 Amazon S3 버킷 보기 예

```
exec rdsadmin.dbo.rds_show_configuration @name='target_s3_arn_for_tlog_copy';
```

Example 새 대상 Amazon S3 버킷으로 업데이트 예

```
exec msdb.dbo.rds_tlog_copy_setup @target_s3_arn='arn:aws:s3:::amzn-s3-demo-logging-bucket1/mynewfolder';
```

사용 가능한 트랜잭션 로그 백업 나열

RDS for SQL Server를 사용하면 전체 복구 모델과 백업 보존 기간이 1일 이상으로 설정된 DB 인스턴스를 사용하도록 구성된 데이터베이스는 트랜잭션 로그 백업이 자동으로 활성화됩니다. 트랜잭션 로그 백업에 대한 액세스를 활성화하면 최대 7일간의 해당 트랜잭션 로그 백업을 Amazon S3 버킷에 복사할 수 있습니다.

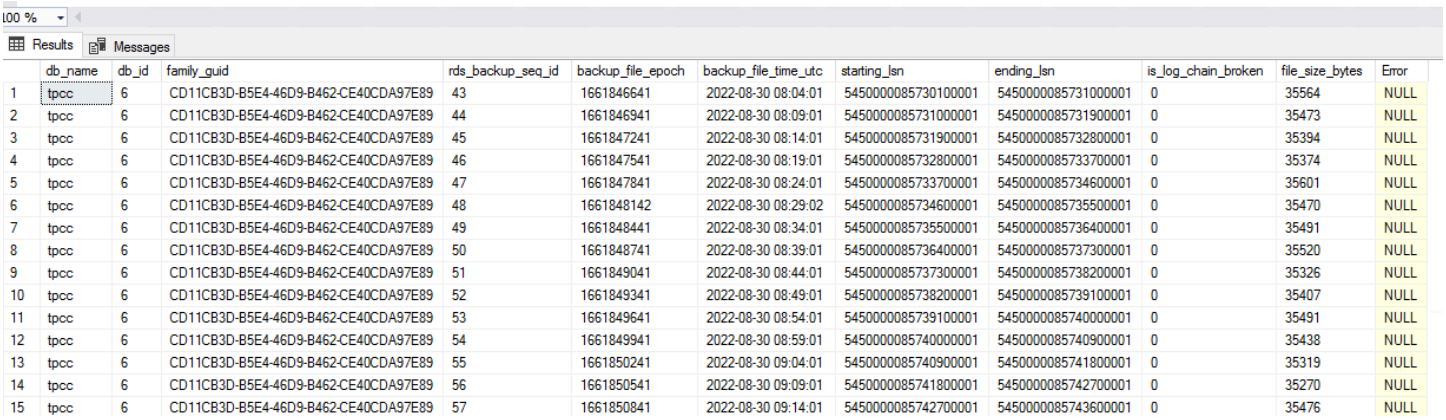
트랜잭션 로그 백업에 대한 액세스를 활성화한 후에는 이를 사용하여 사용 가능한 트랜잭션 로그 백업 파일을 나열하고 복사할 수 있습니다.

트랜잭션 로그 백업 나열

개별 데이터베이스에 사용할 수 있는 모든 트랜잭션 로그 백업을 나열하려면 `rds_fn_list_tlog_backup_metadata` 함수를 호출합니다. 함수를 호출할 때 `ORDER BY` 또는 `WHERE` 절을 사용할 수 있습니다.

Example 사용 가능한 트랜잭션 로그 백업 파일 나열 및 필터링 예

```
SELECT * from msdb.dbo.rds_fn_list_tlog_backup_metadata('mydatabasename');
SELECT * from msdb.dbo.rds_fn_list_tlog_backup_metadata('mydatabasename') WHERE
 rds_backup_seq_id = 3507;
SELECT * from msdb.dbo.rds_fn_list_tlog_backup_metadata('mydatabasename') WHERE
 backup_file_time_utc > '2022-09-15 20:44:01' ORDER BY backup_file_time_utc DESC;
```



100 %

Results Messages

	db_name	db_id	family_guid	rds_backup_seq_id	backup_file_epoch	backup_file_time_utc	starting_lsn	ending_lsn	is_log_chain_broken	file_size_bytes	Error
1	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	43	1661846641	2022-08-30 08:04:01	5450000085730100001	5450000085731000001	0	35564	NULL
2	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	44	1661846941	2022-08-30 08:09:01	5450000085731000001	5450000085731900001	0	35473	NULL
3	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	45	1661847241	2022-08-30 08:14:01	5450000085731900001	5450000085732800001	0	35394	NULL
4	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	46	1661847541	2022-08-30 08:19:01	5450000085732800001	5450000085733700001	0	35374	NULL
5	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	47	1661847841	2022-08-30 08:24:01	5450000085733700001	5450000085734600001	0	35601	NULL
6	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	48	1661848142	2022-08-30 08:29:02	5450000085734600001	5450000085735500001	0	35470	NULL
7	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	49	1661848441	2022-08-30 08:34:01	5450000085735500001	5450000085736400001	0	35491	NULL
8	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	50	1661848741	2022-08-30 08:39:01	5450000085736400001	5450000085737300001	0	35520	NULL
9	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	51	1661849041	2022-08-30 08:44:01	5450000085737300001	5450000085738200001	0	35326	NULL
10	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	52	1661849341	2022-08-30 08:49:01	5450000085738200001	5450000085739100001	0	35407	NULL
11	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	53	1661849641	2022-08-30 08:54:01	5450000085739100001	5450000085740000001	0	35491	NULL
12	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	54	1661849941	2022-08-30 08:59:01	5450000085740000001	5450000085740900001	0	35438	NULL
13	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	55	1661850241	2022-08-30 09:04:01	5450000085740900001	5450000085741800001	0	35319	NULL
14	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	56	1661850541	2022-08-30 09:09:01	5450000085741800001	5450000085742700001	0	35270	NULL
15	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	57	1661850841	2022-08-30 09:14:01	5450000085742700001	5450000085743600001	0	35476	NULL

`rds_fn_list_tlog_backup_metadata` 함수는 다음과 같은 출력을 반환합니다.

열 명칭	데이터 유형	설명
db_name	sysname	트랜잭션 로그 백업을 나열하기 위해 제공된 데이터베이스 이름입니다.
db_id	int	입력 파라미터 db_name의 내부 데이터베이스 식별자입니다.
family_guid	uniqueidentifier	생성 시 원본 데이터베이스의 고유 ID입니다. 이 값은 데이터베이스가 복원될 때 동일하게 유지되며 다른 데이터베이스 이름으로 복원되더라도 마찬가지입니다.

열 명칭	데이터 유형	설명
rds_backup_seq_id	int	RDS가 각 트랜잭션 로그 백업 파일의 시퀀스 번호를 유지하기 위해 내부적으로 사용하는 ID입니다.
backup_file_epoch	bigint	트랜잭션 백업 파일이 생성된 에포크 시간입니다.
backup_file_time_utc	datetime	backup_file_epoch 값의 UTC 시간 변환 값입니다.
starting_lsn	numeric(25,0)	트랜잭션 로그 백업 파일의 첫 번째 또는 가장 오래된 로그 레코드의 로그 시퀀스 번호입니다.
ending_lsn	numeric(25,0)	트랜잭션 로그 백업 파일의 마지막 또는 다음 로그 레코드의 로그 시퀀스 번호입니다.
is_log_chain_broken	bit	현재 트랜잭션 로그 백업 파일과 이전 트랜잭션 로그 백업 파일 간의 로그 체인이 끊어졌는지 여부를 나타내는 부울 값입니다.
file_size_bytes	bigint	트랜잭션 백업 세트의 크기(바이트)입니다.
Error	varchar(4000)	rds_fn_list_tlog_backup_metadata 함수에서 예외가 발생하는 경우의 오류 메시지입니다. 예외가 없는 경우 NULL입니다.

트랜잭션 로그 백업 복사

개별 데이터베이스의 사용 가능한 트랜잭션 로그 백업 세트를 Amazon S3 버킷에 복사하려면 rds_tlog_backup_copy_to_S3 저장 프로시저를 호출합니다. rds_tlog_backup_copy_to_S3 저장 프로시저는 트랜잭션 로그 백업을 복사하는 새 작업을 시작합니다.

> **Note**
>
> rds_tlog_backup_copy_to_S3 저장 프로시저는 is_log_chain_broken 속성에 대한 검증 없이 트랜잭션 로그 백업을 복사합니다. 따라서 rds_tlog_backup_copy_to_S3 저장

프로시저를 실행하기 전에 끊어지지 않은 로그 체인을 수동으로 확인해야 합니다. 자세한 설명은 트랜잭션 로그 백업 로그 체인 검증을 참조하세요.

Example **rds_tlog_backup_copy_to_S3** 저장 프로시저 사용 예

```
exec msdb.dbo.rds_tlog_backup_copy_to_S3
 @db_name='mydatabasename',
 [@kms_key_arn='arn:aws:kms:region:account-id:key/key-id'],
 [@backup_file_start_time='2022-09-01 01:00:15'],
 [@backup_file_end_time='2022-09-01 21:30:45'],
 [@starting_lsn=149000000112100001],
 [@ending_lsn=149000000120400001],
 [@rds_backup_starting_seq_id=5],
 [@rds_backup_ending_seq_id=10];
```

다음 입력 파라미터를 사용할 수 있습니다.

파라미터	설명
@db_name	트랜잭션 로그 백업을 복사할 데이터베이스의 이름입니다.
@kms_key_arn	고객 관리형 KMS 키. AWS 관리형 KMS 키로 DB 인스턴스를 암호화하는 경우 고객 관리형 키를 생성해야 합니다. 고객 관리형 키로 DB 인스턴스를 암호화하는 경우 동일한 KMS 키 ARN을 사용할 수 있습니다.
@backup_file_start_time	rds_fn_list_tlog_backup_metadata 함수의 [backup_file_time_utc] 열에서 제공된 UTC 타임스탬프입니다.
@backup_file_end_time	rds_fn_list_tlog_backup_metadata 함수의 [backup_file_time_utc] 열에서 제공된 UTC 타임스탬프입니다.
@starting_lsn	rds_fn_list_tlog_backup_metadata 함수의 [starting_lsn] 열에서 제공된 LSN(로그 시퀀스 번호)입니다.
@ending_lsn	rds_fn_list_tlog_backup_metadata 함수의 [ending_lsn] 열에서 제공된 LSN(로그 시퀀스 번호)입니다.

파라미터	설명
@rds_backup_starting_seq_id	rds_fn_list_tlog_backup_metadata 함수의 [rds_backup_seq_id] 열에서 제공된 시퀀스 ID입니다.
@rds_backup_ending_seq_id	rds_fn_list_tlog_backup_metadata 함수의 [rds_backup_seq_id] 열에서 제공된 시퀀스 ID입니다.

시간, LSN 또는 시퀀스 ID 파라미터 세트를 지정할 수 있습니다. 한 세트의 파라미터만 필요합니다.

모든 세트에 하나의 파라미터만 지정할 수도 있습니다. 예를 들어 backup_file_end_time 파라미터의 값만 제공하면 7일 한도 내에서 해당 시간 이전에 사용 가능한 모든 트랜잭션 로그 백업 파일이 Amazon S3 버킷에 복사됩니다.

다음은 rds_tlog_backup_copy_to_S3 저장 프로시저의 유효한 입력 파라미터 조합입니다.

제공된 파라미터	예상 결과
exec msdb.dbo.rds_tlog_backup_copy_to_S3 @db_name = 'testdb1', @backup_file_start_time='2022-08-23 00:00:00', @backup_file_end_time='2022-08-30 00:00:00';	제공된 backup_file_start_time 과 backup_file_end_time 범위 사이에 존재하는 지난 7일간의 트랜잭션 로그 백업을 복사합니다. 이 예에서 저장 프로시저는 '2022-08-23 00:00:00' 과 '2022-08-30 00:00:00' 사이에 생성된 트랜잭션 로그 백업을 복사합니다.
exec msdb.dbo.rds_tlog_	제공된 backup_file_start_

제공된 파라미터	예상 결과
`backup_copy_to_S3 @db_name = 'testdb1', @backup_file_start_time='2022-08-23 00:00:00';`	time 에서 시작하는 지난 7일간의 트랜잭션 로그 백업을 복사합니다. 이 예에서 저장 프로시저는 '2022-08-23 00:00:00'부터 최신 트랜잭션 로그 백업까지의 트랜잭션 로그 백업을 복사합니다.
`exec msdb.dbo.rds_tlog_backup_copy_to_S3 @db_name = 'testdb1', @backup_file_end_time='2022-08-30 00:00:00';`	제공된 backup_file_end_time 까지의 지난 7일간의 트랜잭션 로그 백업을 복사합니다. 이 예에서 저장 프로시저는 '2022-08-23 00:00:00'부터 '2022-08-30 00:00:00'까지의 트랜잭션 로그 백업을 복사합니다.

제공된 파라미터	예상 결과
`exec msdb.dbo.rds_tlog_backup_copy_to_S3 @db_name='testdb1', @starting_lsn =14900000 00040007, @ending_lsn = 149000000 0050009;`	제공된 `starting_lsn` 과 `ending_lsn` 범위 사이의 지난 7일간의 사용 가능한 트랜잭션 로그 백업을 복사합니다. 이 예에서 저장 프로시저는 LSN 범위가 1490000000040007에서 149000000 0050009 사이인 지난 7일간의 트랜잭션 로그 백업을 복사합니다.
`exec msdb.dbo.rds_tlog_backup_copy_to_S3 @db_name='testdb1', @starting_lsn =14900000 00040007;`	제공된 `starting_lsn` 에서 시작하는 지난 7일간의 사용 가능한 트랜잭션 로그 백업을 복사합니다. 이 예에서 저장 프로시저는 LSN 1490000000040007부터 최신 트랜잭션 로그 백업까지의 트랜잭션 로그 백업을 복사합니다.

제공된 파라미터	예상 결과
`exec msdb.dbo.rds_tlog_backup_copy_to_S3 @db_name='testdb1', @ending_lsn =14900000 00050009;`	제공된 ending_lsn 까지 지난 7일간의 사용 가능한 트랜잭션 로그 백업을 복사합니다. 이 예에서 저장 프로시저는 지난 7일부터 시작해 lsn 14900000 0050009까지의 트랜잭션 로그 백업을 복사합니다.
`exec msdb.dbo.rds_tlog_backup_copy_to_S3 @db_name='testdb1', @rds_backup_starting_seq_id= 2000, @rds_backup_ending_seq_id= 5000;`	제공된 rds_backup_starting_seq_id 과 rds_backup_ending_seq_id 범위 사이에 존재하는 지난 7일간의 사용 가능한 트랜잭션 로그 백업을 복사합니다. 이 예에서 저장 프로시저는 지난 7일부터 시작하여 seq_id 2000부터 seq_id 5000까지의 제공된 rds 백업 시퀀스 id 범위 내에 있는 트랜잭션 로그 백업을 복사합니다.

제공된 파라미터	예상 결과
`exec msdb.dbo.rds_tlog_backup_copy_to_S3 @db_name='testdb1', @rds_backup_starting_seq_id= 2000;`	제공된 `rds_backup_starting_seq_id` 에서 시작하는 지난 7일간의 사용 가능한 트랜잭션 로그 백업을 복사합니다. 이 예에서 저장 프로시저는 seq_id 2000부터 시작하여 최신 트랜잭션 로그 백업까지의 트랜잭션 로그 백업을 복사합니다.
`exec msdb.dbo.rds_tlog_backup_copy_to_S3 @db_name='testdb1', @rds_backup_ending_seq_id= 5000;`	제공된 `rds_backup_ending_seq_id` 까지 지난 7일간의 사용 가능한 트랜잭션 로그 백업을 복사합니다. 이 예에서 저장 프로시저는 지난 7일부터 시작하여 seq_id 5000까지의 트랜잭션 로그 백업을 복사합니다.

제공된 파라미터	예상 결과
`exec msdb.dbo.rds_tlog_backup_copy_to_S3 @db_name='testdb1', @rds_backup_starting_seq_id= 2000; @rds_backup_ending_seq_id= 2000;`	지난 7일 내에 사용 가능한 경우 제공된 rds_backup_starting_seq_id 와 함께 단일 트랜잭션 로그 백업을 복사합니다. 이 예에서 저장 프로시저는 seq_id가 2000인 단일 트랜잭션 로그 백업을 복사합니다(지난 7일 내에 존재하는 경우).

트랜잭션 로그 백업 로그 체인 검증

트랜잭션 로그 백업 액세스가 구성된 데이터베이스는 자동 백업 보존이 활성화되어 있어야 합니다. 자동 백업 보존은 DB 인스턴스의 데이터베이스를 FULL 복구 모델로 설정합니다. 데이터베이스의 특정 시점 복원을 지원하려면 데이터베이스 복구 모델을 변경하지 마세요. 로그 체인이 끊어질 수 있습니다. 데이터베이스를 FULL 복구 모델로 설정한 상태로 유지하는 것이 좋습니다.

트랜잭션 로그 백업을 복사하기 전에 로그 체인을 수동으로 검증하려면 rds_fn_list_tlog_backup_metadata 함수를 호출하고 is_log_chain_broken 열의 값을 검토하세요. 값이 '1'이면 현재 로그 백업과 이전 로그 백업 간 로그 체인이 끊어졌음을 나타냅니다.

다음 예는 rds_fn_list_tlog_backup_metadata 저장 프로시저의 출력에서 끊어진 로그 체인을 보여 줍니다.

rds_sequence_id	first_lsn	last_lsn	is_log_chain_broken
43	90023	90457	0
44	90457	90985	0
45	90987	92034	1

일반적 로그 체인에서는 주어진 rds_sequence_id의 first_lsn 로그 시퀀스 번호(LSN) 값이 이전 rds_sequence_id의 last_lsn 값과 일치해야 합니다. 이미지에서 rds_sequence_id 45의 first_lsn 값 90987은 이전 rds_sequence_id 44의 last_lsn 값 90985와 일치하지 않습니다.

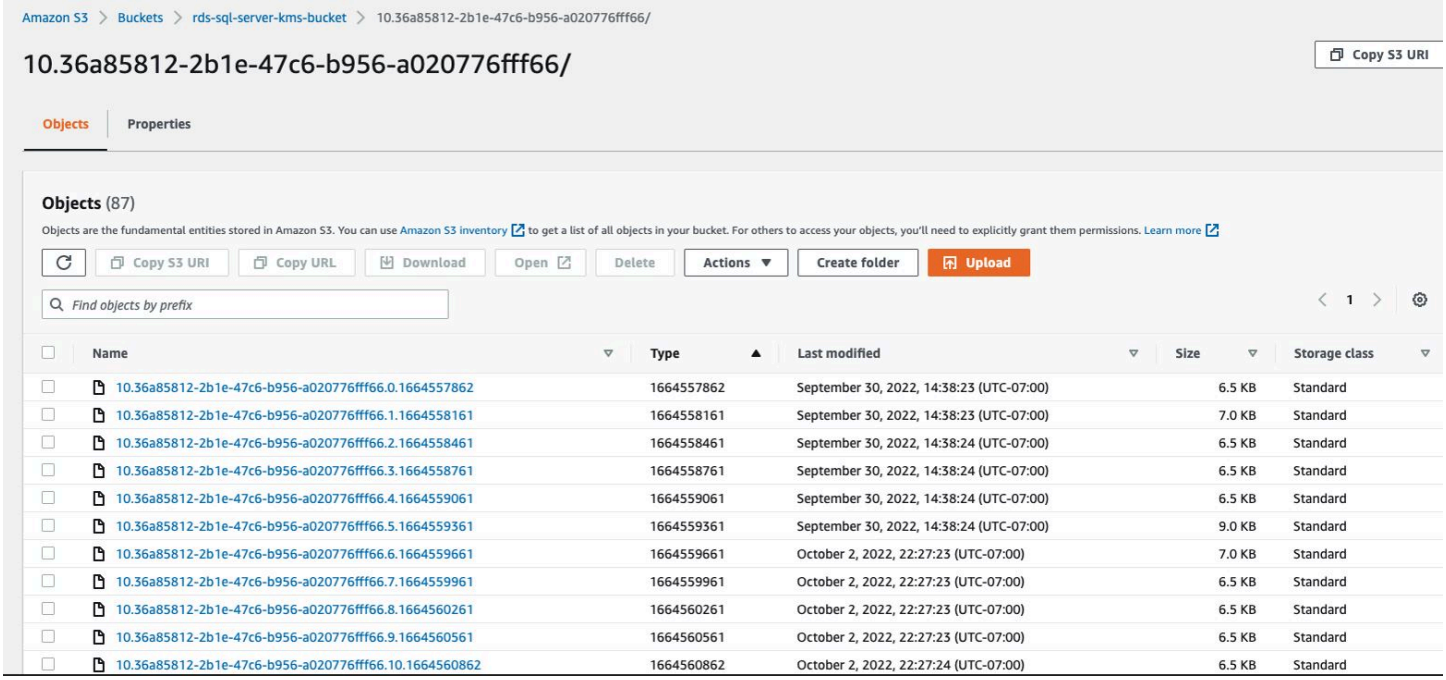
SQL Server 트랜잭션 로그 아키텍처 및 로그 시퀀스 번호에 대한 자세한 내용은 Microsoft SQL Server 설명서의 트랜잭션 로그 논리적 아키텍처를 참조하세요.

Amazon S3 버킷 폴더 및 파일 구조

트랜잭션 로그 백업은 Amazon S3 버킷 내에서 다음과 같은 표준 구조 및 명명 규칙을 사용합니다.

- 각 데이터베이스의 `target_s3_arn` 경로 아래에 `{db_id}.{family_guid}`와 같은 이름 지정 구조의 새 폴더가 생성됩니다.
- 폴더 내에서 트랜잭션 로그 백업의 파일 이름 구조는 `{db_id}.{family_guid}.{rds_backup_seq_id}.{backup_file_epoch}`와 같습니다.
- `rds_fn_list_tlog_backup_metadata` 함수를 사용하여 `family_guid,db_id,rds_backup_seq_id and backup_file_epoch` 세부 정보를 볼 수 있습니다.

다음 예는 Amazon S3 버킷에 있는 트랜잭션 로그 백업 세트의 폴더 및 파일 구조를 보여 줍니다.



작업 상태 추적

복사 작업의 상태를 추적하려면 `rds_task_status` 저장 프로시저를 호출합니다. 파라미터를 제공하지 않으면 이 저장 프로시저는 모든 작업의 상태를 반환합니다.

Example 사용법:

```
exec msdb.dbo.rds_task_status
  @db_name='database_name',
  @task_id=ID_number;
```

다음 파라미터는 선택적입니다.

- `@db_name` – 작업 상태를 표시할 데이터베이스의 이름입니다.
- `@task_id` – 작업 상태를 표시할 작업의 ID입니다.

Example 특정 작업의 상태 나열 예:

```
exec msdb.dbo.rds_task_status @task_id=5;
```

Example 특정 데이터베이스 및 작업의 상태 나열 예:

```
exec msdb.dbo.rds_task_status@db_name='my_database',@task_id=5;
```

Example 특정 데이터베이스의 모든 작업 및 상태 나열 예:

```
exec msdb.dbo.rds_task_status @db_name='my_database';
```

Example 현재 DB 인스턴스의 모든 작업 및 상태 나열 예:

```
exec msdb.dbo.rds_task_status;
```

작업 취소

실행 중인 작업을 취소하려면 `rds_cancel_task` 저장 프로시저를 호출합니다.

Example 사용법:

```
exec msdb.dbo.rds_cancel_task @task_id=ID_number;
```

다음 파라미터는 필수입니다.

- `@task_id` – 취소할 작업의 ID입니다. `rds_task_status` 저장 프로시저를 호출하여 작업 ID를 볼 수 있습니다.

실행 중인 작업 보기 및 취소에 대한 자세한 내용은 기본 백업 및 복원 기능을 사용하여 SQL Server 데이터베이스 가져오기 및 내보내기를 참조하세요.

트랜잭션 로그 백업 액세스 문제 해결

다음은 트랜잭션 로그 백업 액세스에 저장 프로시저를 사용할 때 발생할 수 있는 문제입니다.

저장 프로시저	오류 메시지	문제	문제 해결 제안
rds_tlog_copy_setup	이 DB 인스턴스에서 백업이 비활성화되었습니다. 보존 기간이 '1' 이상인 DB 인스턴스 백업을 활성화하고 다시 시도하십시오.	DB 인스턴스에서 자동 백업이 활성화되지 않았습니다.	DB 인스턴스 백업 보존은 최소 1일 이상의 보존 기간으로 활성화해야 합니다. 자동 백업 활성화 및 백업 보존 구성에 대한 자세한 내용은 백업 보관 기간을 참조하세요.
rds_tlog_copy_setup	rds_tlog_copy_setup 저장 프로시저를 실행하는 동안 오류가 발생했습니다. RDS 엔드포인트에 다시 연결하고 다	내부 오류가 발생했습니다.	RDS 엔드포인트에 다시 연결하고 `rds_tlog_copy_setup` 저장 프로시저를 다시 실행합니다.

저장 프로시저	오류 메시지	문제	문제 해결 제안
	시 시도하십시오.		
rds_tlog_copy_setup	트랜잭션 내에서 rds_tlog_backup_copy_setup 저장 프로시저를 실행하는 것은 지원되지 않습니다. 세션에 진행 중인 트랜잭션이 없는지 확인하고 다시 시도하십시오.	BEGIN 및 END를 사용하여 트랜잭션 내에서 저장 프로시저를 시도했습니다.	rds_tlog_copy_setup 저장 프로시저를 실행할 때는 BEGIN 및 END를 사용하지 마세요.
rds_tlog_copy_setup	입력 파라미터 @target_s3_arn 의 S3 버킷 이름에는 공백 이외의 문자가 한 개 이상 포함되어야 합니다.	입력 파라미터 @target_s3_arn 에 잘못된 값이 제공되었습니다.	입력 파라미터 @target_s3_arn 이 완전한 Amazon S3 버킷 ARN을 지정하는지 확인하세요.

저장 프로시저	오류 메시지	문제	문제 해결 제안
rds_tlog_copy_setup	SQLSERVER_BACKUP_RESTORE 옵션이 활성화되지 않았거나 활성화되는 중입니다. 이 옵션을 활성화하거나 나중에 다시 시도하십시오.	SQLSERVER_BACKUP_RESTORE 옵션이 DB 인스턴스에서 활성화되지 않았거나 방금 활성화되어 내부 활성화 보류 중입니다.	요구 사항 섹션에 지정된 대로 SQLSERVER_BACKUP_RESTORE 옵션을 활성화합니다. 몇 분 기다린 후 rds_tlog_copy_setup 저장 프로시저를 다시 실행합니다.
rds_tlog_copy_setup	입력 파라미터 @target_s3_arn 의 대상 S3 arn은 비어 있거나 null일 수 없습니다.	입력 파라미터 @target_s3_arn 에 NULL 값이 제공되었거나 값이 제공되지 않았습니다.	입력 파라미터 @target_s3_arn 이 완전한 Amazon S3 버킷 ARN을 지정하는지 확인하세요.
rds_tlog_copy_setup	입력 파라미터 @target_s3_arn 의 대상 S3 arn은 arn:aws로 시작해야 합니다.	입력 파라미터 @target_s3_arn 이 앞에 arn:aws 없이 제공되었습니다.	입력 파라미터 @target_s3_arn 이 완전한 Amazon S3 버킷 ARN을 지정하는지 확인하세요.

저장 프로시저	오류 메시지	문제	문제 해결 제안
rds_tlog_copy_setup	대상 S3 ARN이 이미 제공된 값으로 설정되어 있습니다.	`rds_tlog_copy_setup` 저장 프로시저가 이전에 실행되었고 Amazon S3 버킷 ARN으로 구성되었습니다.	트랜잭션 로그 백업에 액세스할 수 있도록 Amazon S3 버킷 값을 수정하려면 다른 `target S3 ARN`을 제공하세요.
rds_tlog_copy_setup	트랜잭션 로그 백업에 대한 액세스를 활성화하기 위한 자격 증명을 생성할 수 없습니다. `rds_tlog_copy_setup` 와 함께 제공된 S3 경로 ARN을 확인하고 나중에 다시 시도하십시오.	트랜잭션 로그 백업 액세스를 활성화하기 위해 자격 증명을 생성하는 동안 지정되지 않은 오류가 발생했습니다.	설정 구성을 검토하고 다시 시도하세요.

저장 프로시저	오류 메시지	문제	문제 해결 제안
rds_tlog_copy_setup	보류 중인 작업이 있는 동안에는 rds_tlog_copy_setup 저장 프로시저를 실행할 수 없습니다. 보류 중인 작업이 완료될 때까지 기다렸다가 다시 시도하십시오.	한 번에 두 개의 작업만 실행할 수 있습니다. 완료를 기다리고 있는 보류 중인 작업이 있습니다.	보류 중인 작업을 보고 완료될 때까지 기다리세요. 작업 상태 모니터링에 대한 자세한 내용은 작업 상태 추적을 참조하세요.
rds_tlog_backup_copy_to_S3	작업 ID: %d의 데이터베이스: %s에 대해 T-log 백업 파일 복사 작업이 이미 실행되었습니다. 나중에 다시 시도하십시오.	지정된 데이터베이스에 대해 언제든 하나의 복사 작업만 실행할 수 있습니다. 완료를 기다리고 있는 보류 중인 복사 작업이 있습니다.	보류 중인 작업을 보고 완료될 때까지 기다리세요. 작업 상태 모니터링에 대한 자세한 내용은 작업 상태 추적을 참조하세요.

저장 프로시저	오류 메시지	문제	문제 해결 제안
rds_tlog_backup_copy_to_S3	다음 세 가지 파라미터 세트 중 하나 이상을 제공해야 합니다. SET-1:(@backup_file_start_time, @backup_file_end_time) | SET-2:(@starting_lsn, @ending_lsn) | SET-3:(@rds_backup_starting_seq_id, @rds_backup_ending_seq_id)	세 가지 파라미터 세트가 모두 제공되지 않았거나 제공된 파라미터 세트에 필수 파라미터가 누락되었습니다.	시간, lsn 또는 시퀀스 ID 파라미터를 지정할 수 있습니다. 이 세 가지 파라미터 세트 중 한 세트가 필요합니다. 필수 파라미터에 대한 자세한 내용은 트랜잭션 로그 백업 복사를 참조하세요.
rds_tlog_backup_copy_to_S3	인스턴스에서 백업이 비활성화되었습니다. 백업을 활성화하고 잠시 후 다시 시도하십시오.	DB 인스턴스에서 자동 백업이 활성화되지 않았습니다.	자동 백업 활성화 및 백업 보존 구성에 대한 자세한 내용은 백업 보관 기간을 참조하세요.

저장 프로시저	오류 메시지	문제	문제 해결 제안
rds_tlog_backup_copy_to_S3	지정된 데이터베이스 %s을(를) 찾을 수 없습니다.	입력 파라미터 @db_name에 제공된 값이 DB 인스턴스의 데이터베이스 이름과 일치하지 않습니다.	올바른 데이터베이스 이름을 사용합니다. 모든 데이터베이스를 이름순으로 나열하려면 `SELECT * from sys.databases` 를 실행합니다.
rds_tlog_backup_copy_to_S3	SQL Server 시스템 데이터베이스 또는 rdsadmin 데이터베이스에 대해 rds_tlog_backup_copy_to_S3 저장 프로시저를 실행할 수 없습니다.	입력 파라미터 @db_name에 제공된 값이 SQL Server 시스템 데이터베이스 이름 또는 RDSAdmin 데이터베이스와 일치합니다.	다음 데이터베이스는 트랜잭션 로그 백업 액세스에 사용할 수 없습니다. `master, model, msdb, tempdb, RDSAdmin.`
rds_tlog_backup_copy_to_S3	입력 파라미터 @db_name의 데이터베이스 이름은 비어 있거나 null일 수 없습니다.	입력 파라미터 @db_name에 제공된 값이 비어 있거나 `NULL`입니다.	올바른 데이터베이스 이름을 사용합니다. 모든 데이터베이스를 이름순으로 나열하려면 `SELECT * from sys.databases` 를 실행합니다.

저장 프로시저	오류 메시지	문제	문제 해결 제안
rds_tlog_backup_copy_to_S3	rds_tlog_backup_copy_setup 저장 프로시저를 실행하려면 DB 인스턴스 백업 보존 기간을 1 이상으로 설정해야 합니다.	DB 인스턴스에서 자동 백업이 활성화되지 않았습니다.	자동 백업 활성화 및 백업 보존 구성에 대한 자세한 내용은 백업 보관 기간을 참조하세요.
rds_tlog_backup_copy_to_S3	저장 프로시저 rds_tlog_backup_copy_to_S3를 실행하는 동안 오류가 발생했습니다. RDS 엔드포인트에 다시 연결하고 다시 시도하십시오.	내부 오류가 발생했습니다.	RDS 엔드포인트에 다시 연결하고 `rds_tlog_backup_copy_to_S3` 저장 프로시저를 다시 실행합니다.

저장 프로시저	오류 메시지	문제	문제 해결 제안
rds_tlog_backup_copy_to_S3	다음 세 가지 파라미터 세트 중 하나만 제공할 수 있습니다. SET-1:(@backup_file_start_time, @backup_file_end_time) | SET-2:(@starting_lsn, @ending_lsn) | SET-3:(@rds_backup_starting_seq_id, @rds_backup_ending_seq_id)	여러 파라미터 세트가 제공되었습니다.	시간, lsn 또는 시퀀스 ID 파라미터를 지정할 수 있습니다. 이 세 가지 파라미터 세트 중 한 세트가 필요합니다. 필수 파라미터에 대한 자세한 내용은 트랜잭션 로그 백업 복사를 참조하세요.

저장 프로시저	오류 메시지	문제	문제 해결 제안
rds_tlog_backup_copy_to_S3	트랜잭션 내에서 rds_tlog_backup_copy_to_S3 저장 프로시저를 실행하는 것은 지원되지 않습니다. 세션에 진행 중인 트랜잭션이 없는지 확인하고 다시 시도하십시오.	BEGIN 및 END를 사용하여 트랜잭션 내에서 저장 프로시저를 시도했습니다.	`rds_tlog_backup_copy_to_S3` 저장 프로시저를 실행할 때는 BEGIN 및 END를 사용하지 마세요.

저장 프로시저	오류 메시지	문제	문제 해결 제안
rds_tlog_backup_copy_to_S3	제공된 파라미터가 트랜잭션 백업 로그 보존 기간을 벗어납니다. 사용 가능한 트랜잭션 로그 백업 파일을 나열하려면 rds_fn_list_tlog_backup_메타데이터 함수를 실행합니다.	제공된 입력 파라미터에 대해 복사본 보존 기간에 맞는 사용 가능한 트랜잭션 로그 백업이 없습니다.	유효한 파라미터 세트를 사용하여 다시 시도하세요. 필수 파라미터에 대한 자세한 내용은 트랜잭션 로그 백업 복사를 참조하세요.

저장 프로시저	오류 메시지	문제	문제 해결 제안
rds_tlog_backup_copy_to_S3	요청을 처리하는 동안 권한 오류가 발생했습니다. 버킷이 DB 인스턴스와 동일한 계정 및 지역에 있는지 확인하고, 공개 설명서의 템플릿과 비교하여 S3 버킷 정책 권한을 확인합니다.	제공된 S3 버킷 또는 해당 정책 권한에서 문제가 감지되었습니다.	트랜잭션 로그 백업에 대한 액세스 설정이 올바른지 확인하세요. S3 버킷의 설정 요구 사항에 대한 자세한 내용은 요구 사항을 참조하세요.
rds_tlog_backup_copy_to_S3	RDS 읽기 전용 복제본 인스턴스에서 `rds_tlog_backup_copy_to_S3` 저장 프로시저를 실행하는 것은 허용되지 않습니다.	RDS 읽기 전용 복제본 인스턴스에서 저장 프로시저를 시도했습니다.	RDS 기본 DB 인스턴스에 연결하여 `rds_tlog_backup_copy_to_S3` 저장 프로시저를 실행합니다.

저장 프로시저	오류 메시지	문제	문제 해결 제안
rds_tlog_backup_copy_to_S3	입력 파라미터 @starting_lsn 의 LSN은 @ending_lsn 보다 작아야 합니다.	입력 파라미터 @starting_lsn 에 제공된 값이 입력 파라미터 @ending_lsn 에 제공된 값보다 큽니다.	입력 파라미터 @starting_lsn 에 제공된 값이 입력 파라미터 @ending_lsn 에 제공된 값보다 작은지 확인합니다.
rds_tlog_backup_copy_to_S3	rds_tlog_backup_copy_to_S3 저장 프로시저는 소스 데이터베이스의 db_owner 역할 멤버만 수행할 수 있습니다.	제공된 db_name에서 rds_tlog_backup_copy_to_S3 저장 프로시저를 실행하려고 시도하는 계정에 db_owner 역할이 부여되지 않았습니다.	저장 프로시저를 실행하는 계정에 제공된 db_name에 대한 db_owner 역할이 허용되었는지 확인하세요.
rds_tlog_backup_copy_to_S3	입력 파라미터 @rds_backup_starting_seq_id 의 시퀀스 ID는 @rds_backup_ending_seq_id 이하여야 합니다.	입력 파라미터 @rds_backup_starting_seq_id 에 제공된 값이 입력 파라미터 @rds_backup_ending_seq_id 에 제공된 값보다 큽니다.	입력 파라미터 @rds_backup_starting_seq_id 에 제공된 값이 입력 파라미터 @rds_backup_ending_seq_id 에 제공된 값보다 작은지 확인합니다.

저장 프로시저	오류 메시지	문제	문제 해결 제안
rds_tlog_backup_copy_to_S3	SQLSERVER_BACKUP_RESTORE 옵션이 활성화되지 않았거나 활성화되는 중입니다. 이 옵션을 활성화하거나 나중에 다시 시도하십시오.	SQLSERVER_BACKUP_RESTORE 옵션이 DB 인스턴스에서 활성화되지 않았거나 방금 활성화되어 내부 활성화 보류 중입니다.	요구 사항 섹션에 지정된 대로 SQLSERVER_BACKUP_RESTORE 옵션을 활성화합니다. 몇 분 기다린 후 rds_tlog_backup_copy_to_S3 저장 프로시저를 다시 실행합니다.
rds_tlog_backup_copy_to_S3	입력 매개변수 @backup_file_start_time 의 시작 시간은 @backup_file_end_time 보다 작아야 합니다.	입력 파라미터 @backup_file_start_time 에 제공된 값이 입력 파라미터 @backup_file_end_time 에 제공된 값보다 큽니다.	입력 파라미터 @backup_file_start_time 에 제공된 값이 입력 파라미터 @backup_file_end_time 에 제공된 값보다 작은지 확인합니다.
rds_tlog_backup_copy_to_S3	액세스 권한이 부족하여 요청을 처리할 수 없었습니다. 기능에 대한 설정 및 권한을 확인하십시오.	Amazon S3 버킷 권한에 문제가 있거나 제공된 Amazon S3 버킷이 다른 계정 또는 리전에 있을 수 있습니다.	Amazon S3 버킷 정책 권한이 RDS 액세스를 허용하도록 승인되었는지 확인합니다. Amazon S3 버킷이 소스 DB 인스턴스와 동일한 계정 및 리전에 있는지 확인합니다.

저장 프로시저	오류 메시지	문제	문제 해결 제안
rds_tlog_backup_copy_to_S3	스토리지 암호화되지 않은 인스턴스의 경우 KMS Key ARN을 저장 프로시저의 입력 파라미터로 제공할 수 없습니다.	DB 인스턴스에서 스토리지 암호화가 활성화되지 않은 경우 입력 파라미터 `@kms_key_arn` 을 제공하면 안 됩니다.	`@kms_key_arn` 의 입력 파라미터를 제공하지 마세요.
rds_tlog_backup_copy_to_S3	암호화된 스토리지 인스턴스의 경우 KMS Key ARN을 저장 프로시저의 입력 파라미터로 제공해야 합니다.	DB 인스턴스에서 스토리지 암호화가 활성화된 경우 입력 파라미터 `@kms_key_arn` 을 제공해야 합니다.	트랜잭션 로그 백업에 사용할 Amazon S3 버킷의 ARN과 일치하는 값을 `@kms_key_arn` 의 입력 파라미터에 제공합니다.

저장 프로시저	오류 메시지	문제	문제 해결 제안
rds_tlog_backup_copy_to_S3	`rds_tlog_backup_copy_to_S3` 저장 프로시저를 실행하기 전에 `rds_tlog_copy_setup` 저장 프로시저를 실행하고 `@target_s3_arn` 을 설정해야 합니다.	`rds_tlog_backup_copy_to_S3` 저장 프로시저를 실행하기 전에 트랜잭션 로그 백업에 대한 액세스 설정 절차가 완료되지 않았습니다.	`rds_tlog_backup_copy_to_S3` 저장 프로시저를 실행하기 전에 `rds_tlog_copy_setup` 저장 프로시저를 실행합니다. 트랜잭션 로그 백업 액세스 설정 프로시저 실행에 대한 자세한 내용은 트랜잭션 로그 백업에 대한 액세스 설정을 참조하세요.

Microsoft SQL Server 데이터베이스 엔진의 옵션

이 단원에서는 Microsoft SQL Server DB 엔진을 실행하는 Amazon RDS 인스턴스에 사용할 수 있는 옵션에 대한 설명을 볼 수 있습니다. 이러한 옵션을 활성화하려면 먼저 옵션 그룹에 추가한 다음 옵션 그룹을 DB 인스턴스에 연결해야 합니다. 자세한 내용은 옵션 그룹 작업 섹션을 참조하세요.

SSL, Microsoft Windows 인증 및 Amazon S3 통합과 같은 RDS 옵션 그룹을 통해 추가되지 않은 선택적 기능을 찾으려면 Amazon RDS의 Microsoft SQL Server 추가 기능 단원을 참조하십시오.

Amazon RDS는 Microsoft SQL Server DB 인스턴스에 대해 다음 옵션을 지원합니다.

옵션	옵션 ID	엔진 버전
Oracle OLEDB 포함 연결된 서버	`OLEDB_ORACLE`	SQL Server Enterprise Edition SQL Server Standard Edition
기본 백업 및 복원	`SQLSERVER_BACKUP_RESTORE`	SQL Server Enterprise Edition SQL Server Standard Edition SQL Server Web Edition SQL Server Express Edition
투명한 데이터 암호화	`TRANSPARENT_DATA_ENCRYPTION` (RDS 콘솔) `TDE`(AWS CLI 및 RDS API)	SQL Server 2016–2022 Enterprise Edition SQL Server 2022 Standard Edition

옵션	옵션 ID	엔진 버전
SQL Server Audit	SQLSERVER_AUDIT	SQL Server 2016부터는 RDS에서 모든 버전의 SQL Server가 서버 수준 감사를 지원하고 Enterprise Edition은 데이터베이스 수준 감사도 지원합니다. SQL Server SQL Server 2016(13.x) SP1부터는 모든 버전이 서버 수준 및 데이터베이스 수준 감사를 모두 지원합니다. 자세한 내용은 SQL Server 설명서의 SQL Server Audit(데이터베이스 엔진)를 참조하십시오.
SQL Server Analysis Services	SSAS	SQL Server Enterprise Edition SQL Server Standard Edition
SQL Server Integration Services	SSIS	SQL Server Enterprise Edition SQL Server Standard Edition

옵션	옵션 ID	엔진 버전
SQL Server Reporting Services	SSRS	SQL Server Enterprise Edition SQL Server Standard Edition
Microsoft Distributed Transaction Coordinator	MSDTC	RDS에서는 SQL Server 2016부터 모든 버전의 SQL Server가 분산 트랜잭션을 지원합니다.

SQL Server 버전 및 에디션에 사용할 수 있는 옵션 나열

`describe-option-group-options` AWS CLI명령을 사용하여 SQL Server 버전 및 에디션에 사용할 수 있는 옵션과 해당 옵션의 설정을 나열할 수 있습니다.

다음 예에서는 SQL Server 2019 Enterprise Edition에 대한 옵션 및 옵션 설정을 보여 줍니다. `--engine-name` 옵션은 필수입니다.

```
aws rds describe-option-group-options --engine-name sqlserver-ee --major-engine-version
 15.00
```

출력은 다음과 유사합니다.

```
{
    "OptionGroupOptions": [
        {
            "Name": "MSDTC",
            "Description": "Microsoft Distributed Transaction Coordinator",
            "EngineName": "sqlserver-ee",
            "MajorEngineVersion": "15.00",
            "MinimumRequiredMinorEngineVersion": "4043.16.v1",
            "PortRequired": true,
            "DefaultPort": 5000,
            "OptionsDependedOn": [],
            "OptionsConflictsWith": [],
```

```
            "Persistent": false,
            "Permanent": false,
            "RequiresAutoMinorEngineVersionUpgrade": false,
            "VpcOnly": false,
            "OptionGroupOptionSettings": [
                {
                    "SettingName": "ENABLE_SNA_LU",
                    "SettingDescription": "Enable support for SNA LU protocol",
                    "DefaultValue": "true",
                    "ApplyType": "DYNAMIC",
                    "AllowedValues": "true,false",
                    "IsModifiable": true,
                    "IsRequired": false,
                    "MinimumEngineVersionPerAllowedValue": []
                },
        ...

        {
            "Name": "TDE",
            "Description": "SQL Server - Transparent Data Encryption",
            "EngineName": "sqlserver-ee",
            "MajorEngineVersion": "15.00",
            "MinimumRequiredMinorEngineVersion": "4043.16.v1",
            "PortRequired": false,
            "OptionsDependedOn": [],
            "OptionsConflictsWith": [],
            "Persistent": true,
            "Permanent": false,
            "RequiresAutoMinorEngineVersionUpgrade": false,
            "VpcOnly": false,
            "OptionGroupOptionSettings": []
        }
    ]
}
```

Amazon RDS for SQL Server의 Oracle OLEDB 포함 연결된 서버 지원

RDS for SQL Server의 OLEDB에 대한 Oracle Provider를 지원하는 연결된 서버를 사용하면 Oracle 데이터베이스의 외부 데이터 소스에 액세스할 수 있습니다. 원격 Oracle 데이터 소스의 데이터를 읽고 RDS for SQL Server DB 인스턴스 외부에 있는 원격 Oracle 데이터베이스 서버를 상대로 명령을 실행할 수 있습니다. Oracle OLEDB 포함 연결된 서버를 사용하면 다음을 수행할 수 있습니다.

- SQL Server 이외의 데이터 소스에 바로 액세스
- 데이터를 옮기지 않고도 동일한 쿼리를 이용해 다양한 Oracle 데이터 소스를 쿼리
- 엔터프라이즈 에코시스템 전반의 데이터 소스에 대한 분산 쿼리, 업데이트, 명령 및 트랜잭션 실행
- Microsoft 비즈니스 인텔리전스 제품군(SSIS, SSRS, SSAS) 내에서 Oracle 데이터베이스로의 연결 통합
- Oracle 데이터베이스에서 RDS for SQL Server로의 마이그레이션

기존 또는 새 RDS for SQL Server DB 인스턴스에서 Oracle용 연결된 서버를 하나 이상 활성화할 수 있습니다. 그런 다음 외부 Oracle 데이터 소스를 DB 인스턴스와 통합할 수 있습니다.

목차

- 지원되는 버전 및 리전
- 제한 및 권장 사항
- Oracle이 포함된 연결된 서버 활성화
  - OLEDB_ORACLE용 옵션 그룹 생성
  - 옵션 그룹에 OLEDB_ORACLE 옵션 추가
  - OLEDB_ORACLE 버전 옵션을 다른 버전으로 수정
  - 옵션 그룹을 DB 인스턴스와 연결
- OLEDB 제공자 속성 수정
- OLEDB 드라이버 속성 수정
- Oracl이 포함된 연결된 서버 비활성화

지원되는 버전 및 리전

RDS for SQL Server는 다음 버전의 SQL Server Standard 및 Enterprise Edition에 대해 모든 리전에서 Oracle OLEDB가 포함된 연결된 서버를 지원합니다.

- SQL Server 2022, 모든 버전
- SQL Server 2019, 모든 버전
- SQL Server 2017, 모든 버전

Oracle OLEDB가 포함된 연결된 서버는 다음 Oracle Database 버전에서 지원됩니다.

- Oracle Database 21c, 모든 버전
- Oracle Database 19c, 모든 버전
- Oracle Database 18c, 모든 버전

Oracle OLEDB가 포함된 연결된 서버는 다음 OLEDB Oracle 드라이버 버전에서 지원됩니다.

- 21.7
- 21.16

제한 및 권장 사항

Oracle OLEDB가 포함된 연결된 서버에 적용되는 다음과 같은 제한 및 권장 사항을 염두에 두십시오.

- 각 RDS for SQL Server DB인스턴스의 보안 그룹에 적절한 TCP 포트를 추가하여 네트워크 트래픽을 허용하십시오. 예를 들어 EC2 Oracle DB 인스턴스와 RDS for SQL Server DB 인스턴스 간에 연결된 서버를 구성하는 경우, EC2 Oracle DB 인스턴스의 IP 주소에서 오는 트래픽을 허용해야 합니다. 또한 SQL Server가 데이터베이스 통신을 수신하는 데 사용하는 포트에서 트래픽을 허용해야 합니다. 보안 그룹에 대한 자세한 정보는 보안 그룹을 통한 액세스 제어 단원을 참조하세요.
- 옵션 그룹에서 `OLEDB_ORACLE` 옵션을 켜거나, 해제하거나, 수정한 후에는 RDS for SQL Server DB 인스턴스를 재부팅합니다. 옵션 그룹 상태에 이러한 이벤트의 `pending_reboot`와 필수 여부가 표시됩니다. AlwaysOn 또는 미러링 옵션이 활성화된 RDS for SQL Server 다중 AZ 인스턴스의 경우 새 인스턴스 생성 또는 복원 작업 후 인스턴스를 재부팅하면 장애 조치가 수행될 수 있습니다.
- Oracle 데이터 소스의 사용자 이름과 암호를 사용한 단순 인증만 지원됩니다.
- Open Database Connectivity(ODBC) 드라이버는 지원되지 않습니다. 위에 나열된 OLEDB 드라이버 버전만 지원됩니다.
- 분산 트랜잭션(XA)은 지원됩니다. 분산 트랜잭션을 활성화하려면 DB 인스턴스의 Option Group(옵션 그룹)에서 `MSDTC` 옵션을 켜고 XA 트랜잭션이 켜져 있는지 확인합니다. 자세한 내용은 RDS for SQL Server에서 Microsoft Distributed Transaction Coordinator 지원 섹션을 참조하세요.

- 연결 문자열의 바로가기로 사용할 데이터 소스 이름(DSN) 생성은 지원되지 않습니다.
- OLEDB 드라이버 추적은 지원되지 않습니다. SQL Server 확장 이벤트를 사용하여 OLEDB 이벤트를 추적할 수 있습니다. 자세한 내용은 RDS for SQL Server에서 확장 이벤트 설정 섹션을 참조하세요.
- SQL Server Management Studio(SSMS) 를 사용하여 Oracle 연결된 서버의 카탈로그 폴터에 액세스하는 작업은 지원되지 않습니다.

Oracle이 포함된 연결된 서버 활성화

OLEDB_ORACLE 옵션을 RDS for SQL Server DB 인스턴스에 추가하여 Oracle이 포함된 연결된 서버를 활성화합니다. 다음 프로세스를 사용합니다.

1. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 선택합니다.
2. [OLEDB_ORACLE] 옵션을 옵션 그룹에 추가합니다.
3. 사용할 OLEDB 드라이버 버전을 선택합니다.
4. 옵션 그룹을 DB 인스턴스에 연결합니다.
5. DB 인스턴스를 재부팅합니다.

OLEDB_ORACLE용 옵션 그룹 생성

Oracle이 포함된 연결된 서버를 사용하려면 사용할 DB 인스턴스의 SQL Server 에디션 및 버전에 해당하는 옵션 그룹을 생성하거나 수정합니다. 이 절차를 완료하려면 AWS Management Console 또는 AWS CLI를 사용합니다.

콘솔

다음 절차에서는 SQL Server Standard Edition 2019에 대한 옵션 그룹을 생성합니다.

옵션 그룹을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 그룹 생성을 선택합니다.
4. 보안 그룹 생성 창에서 다음과 같이 합니다.

a. [이름(Name)]에 AWS 계정 내에서 쉽게 식별할 수 있는 옵션 그룹 이름을 입력합니다(예: **oracle-oledb-se-2019**). 이름은 글자, 숫자 및 하이픈만 사용 가능합니다.
b. 설명에 옵션 그룹에 대한 간단한 설명을 입력합니다(예: **OLEDB_ORACLE option group for SQL Server SE 2019**). 이 설명은 표시 용도로만 사용됩니다.
c. 엔진에 대해 sqlserver-se를 선택합니다.
d. Major engine version(메이저 엔진 버전)에 15.00을 선택합니다.

5. 생성(Create)을 선택합니다.

CLI

다음 절차에서는 SQL Server Standard Edition 2019에 대한 옵션 그룹을 생성합니다.

옵션 그룹을 생성하려면

- 다음 명령 중 하나를 실행합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds create-option-group \
    --option-group-name oracle-oledb-se-2019 \
    --engine-name sqlserver-se \
    --major-engine-version 15.00 \
    --option-group-description "OLEDB_ORACLE option group for SQL Server SE 2019"
```

  Windows의 경우:

```
aws rds create-option-group ^
    --option-group-name oracle-oledb-se-2019 ^
    --engine-name sqlserver-se ^
    --major-engine-version 15.00 ^
    --option-group-description "OLEDB_ORACLE option group for SQL Server SE 2019"
```

옵션 그룹에 **OLEDB_ORACLE** 옵션 추가

그런 다음 AWS Management Console 또는 AWS CLI를 사용하여 OLEDB_ORACLE 옵션을 옵션 그룹에 추가합니다.

콘솔

OLEDB_ORACLE 옵션을 추가하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 이 예제에서는 방금 생성한 옵션 그룹인 oracle-oledb-se-2019를 선택합니다.
4. 옵션 추가를 선택합니다.
5. Option details(옵션 세부 정보)에서 Option name(옵션 이름)으로 OLEDB_ORACL를 선택합니다.
6. 버전에서 설치할 OLEDB Oracle 드라이버의 버전을 선택합니다.
7. 예약에서 옵션을 즉시 추가할지 또는 다음 유지 관리 기간에 추가할지를 선택합니다.
8. 옵션 추가를 선택합니다.

CLI

OLEDB_ORACLE 옵션을 추가하려면

- [OLEDB_ORACLE] 옵션을 옵션 그룹에 추가합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds add-option-to-option-group \
    --option-group-name oracle-oledb-se-2019 \
    --options OptionName=OLEDB_ORACLE, OptionVersion=21.16 \
    --apply-immediately
```

  Windows의 경우:

```
aws rds add-option-to-option-group ^
    --option-group-name oracle-oledb-se-2019 ^
    --options OptionName=OLEDB_ORACLE, OptionVersion=21.16 ^
    --apply-immediately
```

OLEDB_ORACLE 버전 옵션을 다른 버전으로 수정

OLEDB_ORACLE 옵션 버전을 다른 버전으로 수정하려면 AWS Management Console 또는 AWS CLI를 사용합니다.

콘솔

OLEDB_ORACLE 옵션을 수정하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. OLEDB_ORACLE 옵션이 있는 옵션 그룹을 선택합니다(이전 예시의 경우 oledb-se-2019).
4. 옵션 수정(Modify option)을 선택합니다.
5. Option details(옵션 세부 정보)에서 Option name(옵션 이름)으로 OLEDB_ORACL를 선택합니다.
6. 버전에서 사용하려는 OLEDB Oracle 드라이버의 버전을 선택합니다.
7. 예약에서 옵션을 즉시 수정할지 또는 다음 유지 관리 기간에 수정할지를 선택합니다.
8. 옵션 수정(Modify option)을 선택합니다.

CLI

OLEDB_ORACLE 옵션 버전을 수정하려면 사용할 옵션 그룹 및 옵션 버전과 함께 `rds add-option-to-option-group` AWS CLI 명령을 사용합니다.

OLEDB_ORACLE 옵션을 수정하는 방법

- Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds add-option-to-option-group \
    --option-group-name oracle-oledb-se-2019 \
    --options OptionName=OLEDB_ORACLE, OptionVersion=21.7 \
    --apply-immediately
```

  Windows의 경우:

```
aws rds add-option-to-option-group ^
    --option-group-name oracle-oledb-se-2019 ^
```

```
    --options OptionName=OLEDB_ORACLE, OptionVersion=21.7 ^
    --apply-immediately
```

옵션 그룹을 DB 인스턴스와 연결

OLEDB_ORACLE 옵션 그룹 및 파라미터 그룹을 DB 인스턴스와 연결하려면 AWS Management Console 또는 AWS CLI를 사용합니다.

콘솔

Oracle용 연결된 서버 활성화를 완료하려면 OLEDB_ORACLE 옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스와 연결합니다.

- 새 DB 인스턴스의 경우 인스턴스를 시작할 때 이러한 그룹을 연결합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
- 기존 DB 인스턴스의 경우 인스턴스를 수정하여 그룹을 연결합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

CLI

OLEDB_ORACLE 옵션 그룹 및 파라미터 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스와 연결할 수 있습니다.

OLEDB_ORACLE 옵션 그룹 및 파라미터 그룹을 사용하여 인스턴스를 생성하려면

- 옵션 그룹을 생성할 때 사용한 것과 동일한 DB 엔진 유형과 메이저 버전을 지정합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance \
    --db-instance-identifier mytestsqlserveroracleoledbinstance \
    --db-instance-class db.m5.2xlarge \
    --engine sqlserver-se \
    --engine-version 15.0.4236.7.v1 \
    --allocated-storage 100 \
    --manage-master-user-password \
    --master-username admin \
    --storage-type gp2 \
    --license-model li \
```

```
    --domain-iam-role-name my-directory-iam-role \
    --domain my-domain-id \
    --option-group-name oracle-oledb-se-2019 \
    --db-parameter-group-name my-parameter-group-name
```

Windows의 경우:

```
aws rds create-db-instance ^
    --db-instance-identifier mytestsqlserveroracleoledbinstance ^
    --db-instance-class db.m5.2xlarge ^
    --engine sqlserver-se ^
    --engine-version 15.0.4236.7.v1 ^
    --allocated-storage 100 ^
    --manage-master-user-password ^
    --master-username admin ^
    --storage-type gp2 ^
    --license-model li ^
    --domain-iam-role-name my-directory-iam-role ^
    --domain my-domain-id ^
    --option-group-name oracle-oledb-se-2019 ^
    --db-parameter-group-name my-parameter-group-name
```

인스턴스를 수정하여 **OLEDB_ORACLE** 옵션 그룹을 연결하려면

- 다음 명령 중 하나를 실행합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mytestsqlserveroracleoledbinstance \
    --option-group-name oracle-oledb-se-2019 \
    --db-parameter-group-name my-parameter-group-name \
    --apply-immediately
```

  Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mytestsqlserveroracleoledbinstance ^
    --option-group-name oracle-oledb-se-2019 ^
```

```
    --db-parameter-group-name my-parameter-group-name ^
    --apply-immediately
```

OLEDB 제공자 속성 수정

OLEDB 공급자의 속성을 보고 변경할 수 있습니다. 이 작업은 master 사용자만 수행할 수 있습니다. DB 인스턴스에서 생성한 모든 Oracle용 연결 서버는 관련 OLEDB 공급자와 동일한 속성을 사용합니다. sp_MSset_oledb_prop 저장 프로시저를 호출하여 OLEDB 공급자의 속성을 변경합니다.

OLEDB 공급자 속성을 변경하려면

```
USE [master]
GO
EXEC sp_MSset_oledb_prop N'OraOLEDB.Oracle', N'AllowInProcess', 1
EXEC sp_MSset_oledb_prop N'OraOLEDB.Oracle', N'DynamicParameters', 0
GO
```

다음 속성을 수정할 수 있습니다.

속성 이름	권장 값(1 = 켜기, 0 = 해제)	설명
Dynamic parameter	1	파라미터화된 쿼리에서 ('? '로 표시하는) SQL 플레이스홀더 허용
Nested queries	1	FROM 절에 중첩된 SELECT 문(예: 하위 쿼리)을 허용합니다.
Level zero only	0	기본 수준 OLEDB 인터페이스만 공급자에 대해 호출됩니다.
Allow inprocess	1	이 기능을 켜면 Microsoft SQL Server는 공급자를 진행 중 서버로 인스턴스화할 수 있습니다. Oracle 연결된 서버를 사용하려면 이 속성을 1로 설정합니다.
Non transacted updates	0	0이 아닌 경우 SQL Server는 업데이트를 허용합니다.

속성 이름	권장 값(1 = 켜기, 0 = 해제)	설명
Index as access path	False	0이 아닌 경우 SQL Server는 공급자의 인덱스를 사용하여 데이터를 가져옵니다.
Disallow adhoc access	False	설정하면 SQL Server는 OLEDB 공급자를 대상으로 한 패스스루 쿼리 실행을 허용하지 않습니다. 이 옵션을 선택해도 되지만, 패스스루 쿼리를 실행하는 것이 적절할 때도 있습니다.
Supports LIKE operator	1	공급자가 LIKE 키워드를 사용하는 쿼리를 지원한다는 뜻입니다.

OLEDB 드라이버 속성 수정

Oracle용 연결된 서버를 만들 때 OLEDB 드라이버의 속성을 보고 변경할 수 있습니다. 이 작업은 master 사용자만 수행할 수 있습니다. 드라이버 속성은 원격 Oracle 데이터 소스를 작업할 때 OLEDB 드라이버가 데이터를 처리하는 방법을 정의합니다. 드라이버 속성은 DB 인스턴스에서 생성한 각 Oracle 연결된 서버에만 적용됩니다. master.dbo.sp_addlinkedserver 저장 프로시저를 호출하여 OLEDB 드라이버의 속성을 변경합니다.

예: 연결된 서버를 만들고 OLEDB 드라이버 FetchSize 속성을 변경하려면

```
EXEC master.dbo.sp_addlinkedserver
@server = N'Oracle_link2',
@srvproduct=N'Oracle',
@provider=N'OraOLEDB.Oracle',
@datasrc=N'my-oracle-test.cnetsipka.us-west-2.rds.amazonaws.com:1521/ORCL',
@provstr='FetchSize=200'
GO
```

```
EXEC master.dbo.sp_addlinkedsrvlogin
@rmtsrvname=N'Oracle_link2',
@useself=N'False',
@locallogin=NULL,
@rmtuser=N'master',
```

```
@rmtpassword='Test#1234'
GO
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

Oracl이 포함된 연결된 서버 비활성화

Oracle이 포함된 연결된 서버를 비활성화하려면 해당 옵션 그룹에서 `OLEDB_ORACLE` 옵션을 제거합니다.

> **Important**
>
> 옵션을 제거해도 DB 인스턴스의 기존 연결된 서버 구성은 삭제되지 않습니다. DB 인스턴스에서 제거하려면 수동으로 삭제해야 합니다.
> 제거 후 `OLEDB_ORACLE` 옵션을 다시 활성화하여 DB 인스턴스에서 이전에 구성한 연결된 서버 구성을 재사용할 수 있습니다.

콘솔

다음 절차에서는 `OLEDB_ORACLE` 옵션을 제거합니다.

옵션 그룹에서 OLEDB_ORACLE 옵션을 제거하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. `OLEDB_ORACLE` 옵션이 있는 옵션 그룹을 선택합니다(이전 예제의 경우 `oracle-oledb-se-2019`).
4. 옵션 삭제를 선택합니다.
5. Deletion options(옵션 삭제)에서 Options to delete(삭제할 옵션)에 OLEDB_ORACLE를 선택합니다.
6. Apply immediately(즉시 적용)에서 Yes(예)를 선택하여 옵션을 즉시 삭제하거나 No(아니오)를 선택하여 다음 유지 관리 기간에 삭제합니다.

7. 삭제를 선택합니다.

CLI

다음 절차에서는 OLEDB_ORACLE 옵션을 제거합니다.

옵션 그룹에서 OLEDB_ORACLE 옵션을 제거하려면

- 다음 명령 중 하나를 실행합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds remove-option-from-option-group \
    --option-group-name oracle-oledb-se-2019 \
    --options OLEDB_ORACLE \
    --apply-immediately
```

  Windows의 경우:

```
aws rds remove-option-from-option-group ^
    --option-group-name oracle-oledb-se-2019 ^
    --options OLEDB_ORACLE ^
    --apply-immediately
```

RDS for SQL Server에서 Teradata ODBC와 연결된 서버

RDS for SQL Server에서 Teradata ODBC 드라이버와 연결된 서버를 지원하므로 Teradata 데이터베이스의 외부 데이터 소스에 액세스할 수 있습니다. RDS for SQL Server 인스턴스 외부에 있는 원격 Teradata 데이터베이스 서버에서 데이터를 읽고 명령을 실행할 수 있습니다. Teradata ODBC와 연결된 서버를 사용하여 다음 기능을 활성화합니다.

- SQL Server 이외의 데이터 소스에 바로 액세스합니다.
- 데이터를 옮기지 않고도 동일한 쿼리를 이용해 다양한 Teradata 데이터 소스를 쿼리합니다.
- 엔터프라이즈 에코시스템 전반의 데이터 소스에 대한 분산 쿼리, 업데이트, 명령 및 트랜잭션을 실행합니다.
- Microsoft 비즈니스 인텔리전스 제품군(SSIS, SSRS, SSAS) 내에서 Teradata 데이터베이스로의 연결을 통합합니다.
- Teradata 데이터베이스에서 RDS for SQL Server로의 마이그레이션

기존 또는 새 RDS for SQL Server DB 인스턴스에서 Teradata용 연결된 서버를 하나 이상 활성화할 수 있습니다. 그런 다음 외부 Teradata 데이터 소스를 DB 인스턴스와 통합할 수 있습니다.

주제

- 지원되는 버전 및 리전
- 제한 및 권장 사항
- 다중 AZ 배포에 대한 고려 사항
- Teradata와 연결된 서버 활성화
- Teradata와 연결된 서버 만들기
- Teradata에 연결된 서버 비활성화

지원되는 버전 및 리전

RDS for SQL Server는 다음 버전의 SQL Server Standard 및 Enterprise Edition에 대해 모든 AWS 리전에서 Teradata ODBC와 연결된 서버를 지원합니다.

- SQL Server 2022, 모든 버전
- SQL Server 2019, 모든 버전
- SQL Server 2017, 모든 버전

다음 Teradata 데이터베이스 버전은 RDS for SQL Server와의 연결을 지원합니다.

- Teradata 17.20, 모든 버전

제한 및 권장 사항

Teradata ODBC와 연결된 서버에는 다음 제한 사항이 적용됩니다.

- RDS for SQL Server는 Teradata 데이터 소스에 대해 사용자 이름과 암호를 사용한 단순 인증만 지원합니다.
- RDS for SQL Server는 Teradata ODBC 드라이버 버전 17.20.0.33만 지원합니다.
- RDS for SQL Server는 연결 문자열의 바로 가기로 사용할 데이터 소스 이름(DSN) 생성을 지원하지 않습니다.
- RDS for SQL Server는 ODBC 드라이버 추적을 지원하지 않습니다. SQL Server 확장 이벤트를 사용하여 OLEDB 이벤트를 추적하세요. 자세한 내용은 RDS for SQL Server에서 확장 이벤트 설정 섹션을 참조하세요.
- RDS for SQL Server는 SQL Server Management Studio(SSMS)를 사용하여 Teradata에 연결된 서버의 카탈로그 폴터에 액세스하는 작업을 지원하지 않습니다.

Teradata ODBC와 연결된 서버를 사용할 때는 다음 권장 사항을 고려하세요.

- 각 RDS for SQL Server DB인스턴스의 보안 그룹에 적절한 TCP 포트를 추가하여 네트워크 트래픽을 허용하십시오. EC2 Teradata DB 인스턴스와 RDS for SQL Server DB 인스턴스 간에 연결된 서버를 구성하는 경우, EC2 Teradata DB 인스턴스의 IP 주소에서 오는 트래픽을 허용해야 합니다. 또한 RDS for SQL Server DB 인스턴스가 데이터베이스 통신을 수신하는 데 사용하는 포트에서 트래픽을 허용해야 합니다. 보안 그룹에 대한 자세한 정보는 보안 그룹을 통한 액세스 제어 단원을 참조하세요.
- 분산 트랜잭션(XA)은 지원됩니다. 분산 트랜잭션을 활성화하려면 DB 인스턴스의 옵션 그룹에서 `MSDTC` 옵션을 켜고 XA 트랜잭션이 켜져 있는지 확인합니다. 자세한 내용은 RDS for SQL Server에서 Microsoft Distributed Transaction Coordinator 지원 섹션을 참조하세요.
- 연결된 Teradata ODBC는 Teradata Server에서 구성된 경우 SSL/TLS를 지원합니다. 자세한 내용은 Teradata Vantage에서 TLS 연결 활성화를 참조하세요.

다중 AZ 배포에 대한 고려 사항

RDS for SQL Server는 현재 다중 AZ 배포에서 연결된 서버를 미러링된 데이터베이스 서버(또는 상시 작동 가용성 그룹 보조 서버)에 복제하지 않습니다. 미러링 또는 상시 작동을 추가하도록 구성을 변경하기 전에 연결된 서버를 추가하면 연결된 서버가 기존 연결된 서버에 대해 복사됩니다.

또는 기본 인스턴스에서 연결된 서버를 만들고 고가용성 서버 인스턴스로 장애 조치한 다음 연결된 서버를 다시 만들어 두 RDS for SQL Server 인스턴스 모두에 연결된 서버가 있도록 할 수 있습니다.

Teradata와 연결된 서버 활성화

ODBC_TERADATA 옵션을 RDS for SQL Server DB 인스턴스에 추가하여 Teradata와 연결된 서버를 활성화합니다. 다음 프로세스를 사용합니다.

주제

- ODBC_TERADATA용 옵션 그룹 만들기
- 옵션 그룹에 ODBC_TERADATA 옵션 추가
- ODBC_TERADATA 옵션을 DB 인스턴스와 연결

ODBC_TERADATA용 옵션 그룹 만들기

Teradata와 연결된 서버를 사용하려면 사용할 DB 인스턴스의 SQL Server 에디션 및 버전에 해당하는 옵션 그룹을 만들거나 수정합니다. 이 절차를 완료하려면 AWS Management Console 또는 AWS CLI를 사용합니다.

콘솔

다음 절차를 사용하여 SQL Server Standard Edition 2019에 대한 옵션 그룹을 만듭니다.

옵션 그룹을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 그룹 생성을 선택합니다.
4. 보안 그룹 생성 창에서 다음과 같이 합니다.
    a. 이름에 AWS 계정 계정 내에서 쉽게 식별할 수 있는 옵션 그룹 이름을 입력합니다(예: `teradata-odbc-se-2019`). 이름은 글자, 숫자 및 하이픈만 사용 가능합니다.

b. 설명에 옵션 그룹에 대한 간단한 설명을 입력합니다.

c. 엔진에 대해 sqlserver-se를 선택합니다.

d. Major engine version(메이저 엔진 버전)에 15.00을 선택합니다.

5. 생성(Create)을 선택합니다.

AWS CLI

다음 절차에서는 SQL Server Standard Edition 2019에 대한 옵션 그룹을 생성합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-option-group \
    --option-group-name teradata-odbc-se-2019 \
    --engine-name sqlserver-se \
    --major-engine-version 15.00 \
    --option-group-description "ODBC_TERADATA option group for SQL Server SE 2019"
```

Example

Windows의 경우:

```
aws rds create-option-group ^
    --option-group-name teradata-odbc-se-2019 ^
    --engine-name sqlserver-se ^
    --major-engine-version 15.00 ^
    --option-group-description "ODBC_TERADATA option group for SQL Server SE 2019"
```

옵션 그룹에 **ODBC_TERADATA** 옵션 추가

그런 다음 AWS Management Console 또는 AWS CLI를 사용하여 ODBC_Teradata 옵션을 옵션 그룹에 추가합니다.

콘솔

다음 절차를 사용하여 SQL Server Standard Edition 2019에 대한 옵션 그룹을 만듭니다.

ODBC_TERADATA 옵션을 추가하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 새 옵션 그룹을 선택합니다.
4. 옵션 추가를 선택합니다.
5. 옵션 세부 정보에서 다음을 수행합니다.
   a. 옵션 이름으로 ODBC_TERADATA를 선택합니다.
   b. 옵션 버전으로 `17.20.33.00`을 입력합니다.
6. 예약에서 옵션을 즉시 추가할지 또는 다음 유지 관리 기간에 추가할지를 선택합니다.
7. 옵션 추가를 선택합니다.

AWS CLI

다음 절차에서는 `ODBC_TERADATA` 옵션을 옵션 그룹에 추가합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds add-option-to-option-group \
    --option-group-name teradata-odbc-se-2019 \
    --options "OptionName=ODBC_TERADATA,OptionVersion=17.20.33.00" \
    --apply-immediately
```

Example

Windows의 경우:

```
aws rds add-option-to-option-group ^
    --option-group-name teradata-odbc-se-2019 ^
    --options "OptionName=ODBC_TERADATA,OptionVersion=17.20.33.00" ^
    --apply-immediately
```

ODBC_TERADATA 옵션을 DB 인스턴스와 연결

AWS Management Console 또는 AWS CLI를 사용하여 ODBC_TERADATA 옵션 그룹을 DB 인스턴스와 연결할 수 있습니다.

콘솔

Teradata용 연결된 서버 활성화를 완료하려면 옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스와 연결합니다.

- 새 DB 인스턴스의 경우 인스턴스를 시작할 때 연결합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
- 기존 DB 인스턴스의 경우 인스턴스를 수정하여 연결합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

AWS CLI

옵션 그룹을 생성할 때 사용한 것과 동일한 DB 엔진 유형과 메이저 버전을 지정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance \
    --db-instance-identifier mytestsqlserverteradataodbcinstance \
    --db-instance-class db.m5.2xlarge \
    --engine sqlserver-se \
    --engine-version 15.00 \
    --license-model license-included \
    --allocated-storage 100 \
    --master-username admin \
    --master-user-password password \
    --storage-type gp2 \
    --option-group-name teradata-odbc-se-2019
```

Windows의 경우:

```
aws rds create-db-instance ^
    --db-instance-identifier mytestsqlserverteradataodbcinstance ^
    --db-instance-class db.m5.2xlarge ^
    --engine sqlserver-se ^
    --engine-version 15.00 ^
    --license-model license-included ^
    --allocated-storage 100 ^
```

```
    --master-username admin ^
    --master-user-password password ^
    --storage-type gp2 ^
    --option-group-name teradata-odbc-se-2019
```

인스턴스를 수정하여 새로운 옵션 그룹을 연결하려면:

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mytestsqlserverteradataodbcinstance \
    --option-group-name teradata-odbc-se-2019 \
    --apply-immediately
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mytestsqlserverteradataodbcinstance ^
    --option-group-name teradata-odbc-se-2019 ^
    --apply-immediately
```

Teradata와 연결된 서버 만들기

Teradata와 연결된 서버를 만들려면 다음 명령을 실행합니다.

```
EXECUTE master.dbo.sp_addlinkedserver
    @server = N'LinkedServer_NAME',
    @srvproduct=N'',
    @provider=N'MSDASQL',
    @provstr=N'"PROVIDER=MSDASQL;DRIVER={Teradata Database ODBC Driver 17.20};
                DBCName=Server;UID=user_name;PWD=user_password;
                UseDataEncryption=YES/NO;SSLMODE=PREFER/ALLOW/DISABLE>;"',
    @catalog='database'
```

```
EXECUTE master.dbo.sp_addlinkedsrvlogin
    @rmtsrvname = N'LinkedServer_NAME',
    @locallogin = NULL ,
    @useself = N'False',
    @rmtuser = N'user_name',
    @rmtpassword = N'user_password'
```

위 명령의 예는 다음과 같습니다.

```
EXECUTE master.dbo.sp_addlinkedserver
    @server = N'LinkedServerToTeradata',
    @srvproduct=N'',
    @provider=N'MSDASQL',
    @provstr=N'"PROVIDER=MSDASQL;DRIVER={Teradata Database ODBC Driver 17.20};
                DBCName=my-teradata-test.cnetsipka.us-west-2.rds.amazonaws.com;
                UID=master;
                PWD=Test#1234;
                UseDataEncryption=YES;
                SSLMODE=PREFER;"',
    @catalog='MyTestTeradataDB'

EXECUTE master.dbo.sp_addlinkedsrvlogin
    @rmtsrvname = N'LinkedServerToTeradata',
    @locallogin = NULL ,
    @useself = N'False',
    @rmtuser = N'master',
    @rmtpassword = N'Test#1234'
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

Teradata에 연결된 서버 비활성화

Teradata와 연결된 서버를 비활성화하려면 해당 옵션 그룹에서 ODBC_TERADATA 옵션을 제거합니다.

> **Important**
>
> 옵션을 제거해도 DB 인스턴스의 연결된 서버 구성은 삭제되지 않습니다. DB 인스턴스에서 제거하려면 수동으로 삭제해야 합니다.
> 제거 후 ODBC_TERADATA를 재활성화하여 DB 인스턴스에서 이전에 구성한 연결된 서버 구성을 재사용할 수 있습니다.

콘솔

옵션 그룹에서 ODBC_TERADATA 옵션을 제거합니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. ODBC_TERADATA 옵션을 사용하여 옵션 그룹을 선택합니다.
4. 삭제를 선택합니다.
5. 삭제 옵션에서 삭제할 옵션 아래의 ODBC_TERADATA를 선택합니다.
6. Apply immediately(즉시 적용)에서 Yes(예)를 선택하여 옵션을 즉시 삭제하거나 No(아니오)를 선택하여 다음 유지 관리 기간에 삭제합니다.
7. 삭제를 선택합니다.

AWS CLI

다음 절차에서는 ODBC_TERADATA 옵션을 제거합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds remove-option-from-option-group \
    --option-group-name teradata-odbc-se-2019 \
    --options ODBC_TERADATA \
    --apply-immediately
```

Windows의 경우:

```
aws rds remove-option-from-option-group ^
    --option-group-name teradata-odbc-se-2019 ^
    --options ODBC_TERADATA ^
    --apply-immediately
```

SQL Server에서 기본 백업 및 복원 지원

SQL Server 데이터베이스에 대한 기본 백업 및 복원을 사용하여 온프레미스 데이터베이스의 차등 백업 또는 전체 백업을 생성하고 Amazon S3에 백업 파일을 저장할 수 있습니다. 그런 다음 SQL Server를 실행하는 기존 Amazon RDS DB 인스턴스로 복원할 수 있습니다. RDS for SQL Server 데이터베이스를 백업하고 Amazon S3에 저장하고 다른 위치에 복원할 수도 있습니다. 또한 온프레미스 서버 또는 SQL Server를 실행 중인 다른 Amazon RDS DB 인스턴스로 백업을 복원할 수 있습니다. 자세한 내용은 기본 백업 및 복원 기능을 사용하여 SQL Server 데이터베이스 가져오기 및 내보내기 섹션을 참조하세요.

Amazon RDS는 차등 및 전체 백업 파일(.bak 파일)을 사용하여 Microsoft SQL Server 데이터베이스에 기본 백업 및 복원을 할 수 있도록 지원합니다.

기본 백업 및 복원 옵션 추가

기본 백업 및 복원 옵션을 DB 인스턴스에 추가하는 일반적인 프로세스는 다음과 같습니다.

1. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 복사 또는 수정합니다.
2. [SQLSERVER_BACKUP_RESTORE] 옵션을 옵션 그룹에 추가합니다.
3. AWS Identity and Access Management(IAM) 역할을 옵션과 연결합니다. 데이터베이스 백업을 저장하려면 IAM 역할에 S3 버킷에 대한 액세스 권한이 있어야 합니다.

   즉, `arn:aws:iam::`*`account-id`*`:role/`*`role-name`* 형식으로 유효한 Amazon 리소스 이름(ARN)을 설정하는 옵션이 있어야 합니다. 자세한 내용은 AWS 일반 참조의 Amazon 리소스 이름(ARN)을 참조하세요.

   또한 IAM 역할에는 신뢰 관계와 권한 정책이 연결되어 있어야 합니다. RDS는 신뢰 관계를 사용하여 역할을 수임할 수 있으며 권한 정책은 역할이 수행할 수 있는 작업을 정의합니다. 자세한 내용은 기본 백업 및 복원을 위한 IAM 역할 수동으로 만들기 섹션을 참조하세요.
4. 옵션 그룹을 DB 인스턴스에 연동시킵니다.

기본 백업 및 복원 옵션을 추가한 후 DB 인스턴스를 다시 시작할 필요가 없습니다. 옵션 그룹이 활성화되는 즉시 백업 및 복원을 시작할 수 있습니다.

콘솔

기본 백업 및 복원 옵션을 추가하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 사용합니다. 사용자 지정 DB 옵션 그룹을 생성하는 방법에 대한 자세한 내용은 옵션 그룹 생성 단원을 참조하십시오.

   기존 옵션 그룹을 사용하려면 다음 단계로 건너뛰십시오.
4. 옵션 그룹에 SQLSERVER_BACKUP_RESTORE 옵션을 추가합니다. 옵션 추가에 대한 자세한 내용은 옵션 그룹에 옵션 추가 섹션을 참조하세요.
5. 다음 중 하나를 수행하십시오.

   - 기존 IAM 역할 및 Amazon S3 설정을 사용하려면 IAM 역할에 대해 기존 IAM 역할을 선택합니다. 기존 IAM 역할을 사용하는 경우 RDS는 이 역할에 구성된 Amazon S3 설정을 사용합니다.
   - 새 역할을 생성하고 새 Amazon S3 설정을 구성하려면 다음을 수행하세요.
     1. IAM 역할에서 새 역할 생성을 선택합니다.
     2. S3 버킷(S3 bucket)에서 목록의 S3 버킷을 선택합니다.
     3. S3 접두사(선택 사항)(S3 prefix (optional))에서 Amazon S3 버킷에 저장된 파일에 사용할 접두사를 지정합니다.

        이 접두사에 파일 경로를 포함할 수 있지만 필수는 아닙니다. 접두사를 제공하면 RDS가 해당 접두사를 모든 백업 파일에 첨부합니다. 그런 다음 RDS는 복원 중에 접두사를 사용하여 관련 파일을 식별하고 관련 없는 파일을 무시합니다. 예를 들어 백업 파일을 보관하는 것 이외의 목적으로 S3 버킷을 사용할 수 있습니다. 이 경우 접두사를 사용하여 RDS가 특정 폴더와 해당 하위 폴더에서만 기본 백업 및 복원을 수행하도록 할 수 있습니다.

        접두사를 비워 두면 RDS가 접두사를 사용하여 백업 파일 또는 복원할 파일을 식별하지 않습니다. 결과적으로 다중 파일 복원 중에 RDS는 S3 버킷의 모든 폴더에 있는 모든 파일을 복원하려고 시도합니다.
     4. 암호화 활성화(Enable Encryption) 체크박스를 선택하여 백업 파일을 암호화합니다. 백업 파일을 암호화하지 않도록 하려면 확인란의 선택을 취소한 상태로 둡니다(기본값).

암호화 활성화(Enable encryption)를 선택한 경우, AWS KMS key을 위한 암호화 키를 선택합니다. 암호화 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서에서 시작하기를 참조하세요.

6. 옵션 추가를 선택합니다.
7. 옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스에 적용합니다:
   - 새 DB 인스턴스의 경우, 인스턴스를 시작할 때 옵션 그룹을 적용하십시오. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
   - 기존 DB 인스턴스의 경우, 해당 인스턴스를 수정하고 새 옵션 그룹을 연결하여 옵션 그룹을 적용하십시오. 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하세요.

CLI

이 절차에서는 다음과 같이 가정합니다.

- SQLSERVER_BACKUP_RESTORE 옵션을 이미 존재하는 옵션 그룹에 추가하려고 합니다. 옵션 추가에 대한 자세한 내용은 옵션 그룹에 옵션 추가 섹션을 참조하세요.
- 이 옵션을 이미 존재하고 백업 저장을 위해 S3 버킷에 액세스할 수 있는 IAM 역할과 연결합니다.
- 이미 존재하는 DB 인스턴스에 옵션 그룹을 적용하려고 합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

기본 백업 및 복원 옵션을 추가하려면

1. [SQLSERVER_BACKUP_RESTORE] 옵션을 옵션 그룹에 추가합니다.

   Example

   대상 LinuxmacOS, 또는Unix:

```
aws rds add-option-to-option-group \
 --apply-immediately \
 --option-group-name mybackupgroup \
 --options "OptionName=SQLSERVER_BACKUP_RESTORE, \
   OptionSettings=[{Name=IAM_ROLE_ARN,Value=arn:aws:iam::account-id:role/role-
name}]"
```

   Windows의 경우:

```
aws rds add-option-to-option-group ^
 --option-group-name mybackupgroup ^
 --options "[{\"OptionName\": \"SQLSERVER_BACKUP_RESTORE\", ^
 \"OptionSettings\": [{\"Name\": \"IAM_ROLE_ARN\", ^
 \"Value\": \"arn:aws:iam::account-id:role/role-name"}]}]" ^
 --apply-immediately
```

> **Note**
>
> Windows 명령 프롬프트를 사용하는 경우 백슬래시(\)를 접두사로 추가하여 JSON 코드에서 큰 따옴표(")를 이스케이프해야 합니다.

2. 옵션 그룹을 DB 인스턴스에 적용합니다.

   Example

   대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
 --db-instance-identifier mydbinstance \
 --option-group-name mybackupgroup \
 --apply-immediately
```

   Windows의 경우:

```
aws rds modify-db-instance ^
 --db-instance-identifier mydbinstance ^
 --option-group-name mybackupgroup ^
 --apply-immediately
```

기본 백업 및 복원 옵션 설정 수정

기본 백업 및 복원 옵션을 활성화한 후 옵션의 설정을 수정할 수 있습니다. 옵션 설정을 변경하는 방법에 대한 자세한 내용은 옵션 설정 수정을(를) 참조하십시오.

기본 백업 및 복원 옵션 제거

DB 인스턴스에서 옵션을 제거하여 기본 백업 및 복원 기능을 끌 수 있습니다. 기본 백업 및 복원 옵션을 제거한 후 DB 인스턴스를 다시 시작할 필요가 없습니다.

DB 인스턴스에서 기본 백업 및 복원 옵션을 제거하려면 다음 중 하나를 수행합니다.

- 소속 옵션 그룹에서 옵션을 제거합니다. 이 변경은 해당 옵션 그룹을 사용하는 모든 DB 인스턴스에 영향을 미칩니다. 자세한 내용은 옵션 그룹에서 옵션 제거 섹션을 참조하세요.
- DB 인스턴스를 수정하고, 기본 백업 및 복원 옵션이 포함되지 않은 다른 옵션 그룹을 지정합니다. 이 변경은 단일 DB 인스턴스에 영향을 미칩니다. 기본(빈) 옵션 그룹을 지정하거나 다른 사용자 지정 옵션 그룹을 지정할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

SQL Server에서 TDE(투명한 데이터 암호화) 지원

Amazon RDS는 Microsoft SQL Server를 실행하는 DB 인스턴스에 저장된 데이터를 암호화하기 위하여 Transparent Data Encryption(TDE) 이용을 지원합니다. TDE는 스토리지에 데이터를 쓰기 전에 자동으로 데이터를 암호화한 뒤에 데이터를 스토리지에서 읽을 때 다시 자동으로 복호화합니다.

Amazon RDS는 다음 SQL Server 버전에 TDE를 지원합니다.

- SQL Server 2022 Standard Edition 및 Enterprise Edition
- SQL Server 2019 Standard Edition 및 Enterprise Edition
- SQL Server 2017 Enterprise Edition
- SQL Server 2016 Enterprise Edition

SQL Server의 TDE는 2계층 키 아키텍처를 사용하여 암호화 키 관리 기능을 지원합니다. 데이터베이스 마스터 키에서 생성된 인증서는 데이터 암호화 키를 보호하는 데 사용됩니다. 데이터베이스 암호화 키는 사용자 데이터베이스의 데이터를 실제로 암호화 및 복호화합니다. Amazon RDS는 데이터베이스 마스터 키와 TDE 인증서를 백업하고 관리합니다.

Transparent Data Encryption은 중요한 데이터를 암호화해야 하는 경우에 사용됩니다. 예를 들어 타사에 데이터 파일과 백업을 제공하거나 보안 관련 규정 준수 문제를 해결하고 싶을 수 있습니다. `model` 또는 `master` 데이터베이스와 같은 SQL Server의 시스템 데이터베이스를 암호화할 수 없습니다.

투명한 데이터 암호화에 대한 자세한 설명은 본 문서의 범위에서 벗어나지만, 보안과 관련하여 각 암호화 알고리즘과 키의 장단점은 잘 알고 있어야 합니다. SQL Server의 투명한 데이터 암호화(TDE)에 대한 자세한 내용은 Microsoft 설명서의 투명한 데이터 암호화(TDE)를 참조하세요.

주제

- RDS for SQL Server에 대한 TDE 활성화
- RDS for SQL Server의 데이터 암호화
- RDS for SQL Server에서 TDE 인증서 백업 및 복원
- 온프레미스 데이터베이스에 대한 TDE 인증서 백업 및 복원
- RDS for SQL Server에 대한 TDE 비활성화

RDS for SQL Server에 대한 TDE 활성화

RDS for SQL Server DB 인스턴스에서 투명한 데이터 암호화를 활성화하려면 해당 DB 인스턴스와 연동되어 있는 RDS 옵션 그룹에서 TDE 옵션을 지정해야 합니다.

1. 혹시 DB 인스턴스가 TDE 옵션이 추가된 옵션 그룹과 이미 연동되어 있는지 먼저 확인합니다. DB 인스턴스와 연동되어 있는 옵션 그룹은 RDS 콘솔, describe-db-instance AWS CLI 명령 또는 API 작업 DescribeDBInstances를 사용하여 확인할 수 있습니다.
2. DB 인스턴스가 TDE가 활성화된 옵션 그룹과 연결되어 있지 않으면 2가지 옵션을 사용할 수 있습니다. 옵션 그룹을 생성하고 TDE 옵션을 추가하거나, 연결된 옵션 그룹을 수정하여 추가할 수 있습니다.

   > **Note**
   >
   > RDS 콘솔에서는 옵션의 이름이 `TRANSPARENT_DATA_ENCRYPTION`입니다. AWS CLI 및 RDS API에서는 이 이름이 `TDE`입니다.

   옵션 그룹의 생성 및 변경에 대한 자세한 내용은 옵션 그룹 작업 단원을 참조하십시오. 옵션 그룹에 옵션을 추가하는 방법에 대한 자세한 내용은 옵션 그룹에 옵션 추가 단원을 참조하십시오.
3. DB 인스턴스를 TDE 옵션이 있는 옵션 그룹과 연결합니다. DB 인스턴스와 옵션 그룹의 연동에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

옵션 그룹 고려 사항

TDE 옵션은 영구 옵션입니다. 모든 DB 인스턴스 및 백업이 옵션 그룹과 연결되어 있지 않은 경우를 제외하고 옵션 그룹에서 DB 인스턴스 및 백업을 제거할 수 없습니다. TDE 옵션을 옵션 그룹에 추가하면 이 옵션 그룹은 TDE를 사용하는 DB 인스턴스에 한해 연동이 가능합니다. 옵션 그룹의 영구 옵션에 대한 자세한 내용은 옵션 그룹 개요 단원을 참조하십시오.

TDE 옵션은 영구 옵션이기 때문에 옵션 그룹과 연결된 DB 인스턴스 사이에 충돌이 일어날 수 있습니다. 다음 상황에서 충돌이 발생할 수 있습니다.

- 현재 TDE 옵션을 설정한 옵션 그룹을 TDE 옵션을 사용하지 않는 옵션 그룹으로 변경하는 경우
- DB 스냅샷에서 TDE 옵션을 포함하는 옵션 그룹이 연결되지 않은 새 DB 인스턴스로 복원하는 경우

  이 시나리오에 대한 자세한 내용은 옵션 그룹에 대한 고려 사항를 참조하십시오.

SQL Server 성능 고려 사항

투명한 데이터 암호화를 사용하면 SQL Server DB 인스턴스의 성능에 영향을 미칠 수 있습니다.

DB 인스턴스의 데이터베이스 중 암호화된 데이터베이스가 하나 이상만 있어도 마찬가지로 암호화되지 않은 데이터베이스의 성능이 떨어질 수 있습니다. 따라서 암호화되지 않은 데이터베이스와 암호화된 데이터베이스는 별도의 DB 인스턴스에서 관리하는 것이 좋습니다.

RDS for SQL Server의 데이터 암호화

TDE 옵션을 옵션 그룹에 추가하면 Amazon RDS가 암호화 프로세스에 사용할 인증서를 생성합니다. 그러면 이 인증서를 사용하여 DB 인스턴스의 데이터베이스에 저장된 데이터를 암호화하는 SQL 문을 실행할 수 있습니다.

다음은 `RDSTDECertificateName`이라고 하는 RDS 생성 인증서를 사용하여 `myDatabase`라는 데이터베이스를 암호화하는 예제입니다.

```
---------- Turning on TDE -------------

-- Find an RDS TDE certificate to use
USE [master]
GO
SELECT name FROM sys.certificates WHERE name LIKE 'RDSTDECertificate%'
GO

USE [myDatabase]
GO
-- Create a database encryption key (DEK) using one of the certificates from the
 previous step
CREATE DATABASE ENCRYPTION KEY WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER CERTIFICATE [RDSTDECertificateName]
GO

-- Turn on encryption for the database
ALTER DATABASE [myDatabase] SET ENCRYPTION ON
GO

-- Verify that the database is encrypted
USE [master]
GO
SELECT name FROM sys.databases WHERE is_encrypted = 1
GO
SELECT db_name(database_id) as DatabaseName, * FROM sys.dm_database_encryption_keys
GO
```

TDE를 사용하여 SQL Server 데이터베이스를 암호화하는 데 걸리는 시간은 몇 가지 요인에 따라 다릅니다. 여기에는 DB 인스턴스의 크기, 인스턴스가 프로비저닝된 IOPS 스토리지를 사용하는지 여부, 데이터 양 및 기타 요소가 포함됩니다.

RDS for SQL Server에서 TDE 인증서 백업 및 복원

RDS for SQL Server는 TDE 인증서를 백업, 복원 및 삭제하기 위한 저장 프로시저를 제공합니다. RDS for SQL Server는 복원된 사용자 TDE 인증서를 보는 기능도 제공합니다.

사용자 TDE 인증서는 온프레미스에 있고 TDE가 설정된 RDS for SQL Server로 데이터베이스를 복원하는 데 사용됩니다. 이러한 인증서에는 접두사 `UserTDECertificate_`가 붙습니다. RDS는 데이터베이스를 복원한 후 사용할 수 있게 하기 전에 TDE가 활성화된 데이터베이스를 수정하여 RDS 생성 TDE 인증서를 사용합니다. 이러한 인증서에는 접두사 `RDSTDECertificate`가 붙습니다.

사용자 TDE 인증서는 `rds_drop_tde_certificate` 저장 프로시저를 사용하여 삭제하지 않는 한 RDS for SQL Server DB 인스턴스에 남아 있습니다. 자세한 내용은 복원된 TDE 인증서 삭제 섹션을 참조하세요.

사용자 TDE 인증서를 사용하여 소스 DB 인스턴스에서 다른 데이터베이스를 복원할 수 있습니다. 복원할 데이터베이스는 동일한 TDE 인증서를 사용해야 하며, TDE가 활성화되어 있어야 합니다. 동일한 인증서를 다시 가져올(복원) 필요가 없습니다.

주제

- 사전 조건
- 제한 사항
- TDE 인증서 백업
- TDE 인증서 복원
- 복원된 TDE 인증서 보기
- 복원된 TDE 인증서 삭제

사전 조건

RDS for SQL Server에 TDE 인증서를 백업하거나 복원하려면 먼저 다음 태스크를 수행해야 합니다. 처음 3개는 기본 백업 및 복원 설정에 설명되어 있습니다.

1. 백업 및 복원할 파일을 저장하기 위한 Amazon S3 버킷을 생성합니다.

   데이터베이스 백업 및 TDE 인증서 백업에는 별도의 버킷을 사용하는 것이 좋습니다.

2. 파일 백업 및 복원을 위한 IAM 역할을 생성합니다.

   IAM 역할은 AWS KMS key의 관리자이며 사용자여야 합니다.

   SQL Server 기본 백업 및 복원에 필요한 권한 외에도 IAM 역할에는 다음과 같은 권한이 필요합니다.

   - S3 버킷 리소스의 `s3:GetBucketAcl`, `s3:GetBucketLocation`, `s3:ListBucket`
   - * 리소스의 `s3:ListAllMyBuckets`

3. DB 인스턴스의 옵션 그룹에 `SQLSERVER_BACKUP_RESTORE` 옵션을 추가합니다.

   이는 `TRANSPARENT_DATA_ENCRYPTION(TDE)` 옵션에 추가됩니다.

4. 대칭 암호화 KMS 키가 있는지 확인합니다. 다음과 같은 옵션이 있습니다:

   - 계정에 기존 KMS 키가 있는 경우 사용할 수 있습니다. 별도로 조치를 취할 필요가 없습니다.
   - 계정에 사용 중이던 대칭 암호화 KMS 키가 없는 경우 AWS Key Management Service 개발자 가이드의 키 생성 지침에 따라 KMS 키를 생성합니다.

5. Amazon S3 통합을 활성화하여 DB 인스턴스와 Amazon S3 간에 파일을 전송합니다.

   Amazon S3 통합에 대한 자세한 내용은 Amazon RDS for SQL Server DB 인스턴스와 Amazon S3 통합 섹션을 참조하세요.

제한 사항

저장 프로시저를 사용하여 TDE 인증서를 백업 및 복원하는 데는 다음과 같은 제한이 있습니다.

- `SQLSERVER_BACKUP_RESTORE` 및 `TRANSPARENT_DATA_ENCRYPTION(TDE)` 옵션을 모두 DB 인스턴스에 연결된 옵션 그룹에 추가해야 합니다.
- 다중 AZ DB 인스턴스에는 TDE의 인증서 백업 및 복원이 지원되지 않습니다.
- TDE 인증서 백업 및 복원 태스크는 취소할 수 없습니다.
- RDS for SQL Server DB 인스턴스에 있는 다른 데이터베이스의 TDE 암호화에 사용자 TDE 인증서를 사용할 수 없습니다. 이를 사용하여 TDE가 활성화되어 있고 동일한 TDE 인증서를 사용하는 소스 DB 인스턴스에서 다른 데이터베이스만 복원할 수 있습니다.
- 사용자 TDE 인증서만 삭제할 수 있습니다.
- RDS에서 지원되는 최대 사용자 TDE 인증서 수는 10개입니다. 개수가 10개를 초과하면 사용하지 않는 TDE 인증서를 삭제하고 다시 시도하세요.
- 인증서 이름은 비어 있거나 null일 수 없습니다.

- 인증서를 복원할 때 인증서 이름에 RDSTDECERTIFICATE 키워드를 포함할 수 없으며, UserTDECertificate_ 접두사로 시작해야 합니다.
- @certificate_name 파라미터에는 a-z, 0-9, @, $, #, 밑줄(_) 문자만 포함할 수 있습니다.
- @certificate_file_s3_arn 파일 확장명은 .cer(대/소문자를 구분하지 않음)이어야 합니다.
- @private_key_file_s3_arn 파일 확장명은 .pvk(대/소문자를 구분하지 않음)이어야 합니다.
- 프라이빗 키 파일의 S3 메타데이터에는 x-amz-meta-rds-tde-pwd 태그가 포함되어야 합니다. 자세한 내용은 온프레미스 데이터베이스에 대한 TDE 인증서 백업 및 복원 섹션을 참조하세요.

TDE 인증서 백업

TDE 인증서를 백업하려면 rds_backup_tde_certificate 저장 프로시저를 사용하면 됩니다. 다음 구문을 사용합니다.

```
EXECUTE msdb.dbo.rds_backup_tde_certificate
    @certificate_name='UserTDECertificate_certificate_name |
 RDSTDECertificatetimestamp',
    @certificate_file_s3_arn='arn:aws:s3:::bucket_name/certificate_file_name.cer',
    @private_key_file_s3_arn='arn:aws:s3:::bucket_name/key_file_name.pvk',
    @kms_password_key_arn='arn:aws:kms:region:account-id:key/key-id',
    [@overwrite_s3_files=0|1];
```

다음 파라미터는 필수 파라미터입니다.

- @certificate_name - 백업할 TDE 인증서의 이름입니다.
- @certificate_file_s3_arn - Amazon S3의 인증서 백업 파일에 대한 대상 Amazon 리소스 이름(ARN)입니다.
- @private_key_file_s3_arn - TDE 인증서를 보호하는 프라이빗 키 파일의 대상 S3 ARN입니다.
- @kms_password_key_arn - 프라이빗 키 암호를 암호화하는 데 사용되는 대칭 KMS 키의 ARN입니다.

다음 파라미터는 선택 사항입니다.

- @overwrite_s3_files - S3의 기존 인증서 및 프라이빗 키 파일을 덮어쓸지를 나타냅니다.
  - 0 – 기존 파일을 덮어쓰지 않습니다. 이 값이 기본값입니다.

    @overwrite_s3_files를 0으로 설정하면 파일이 이미 존재할 경우 오류를 반환합니다.

- 1 – 백업 파일이 아니더라도 지정된 이름이 있는 기존 파일을 덮어씁니다.

Example TDE 인증서 백업

```
EXECUTE msdb.dbo.rds_backup_tde_certificate
    @certificate_name='RDSTDECertificate20211115T185333',
    @certificate_file_s3_arn='arn:aws:s3:::TDE_certs/mycertfile.cer',
    @private_key_file_s3_arn='arn:aws:s3:::TDE_certs/mykeyfile.pvk',
    @kms_password_key_arn='arn:aws:kms:us-
west-2:123456789012:key/AKIAIOSFODNN7EXAMPLE',
    @overwrite_s3_files=1;
```

TDE 인증서 복원

`rds_restore_tde_certificate` 저장 프로시저를 통해 사용자 TDE 인증서를 복원(가져오기)할 수 있습니다. 다음 구문을 사용합니다.

```
EXECUTE msdb.dbo.rds_restore_tde_certificate
    @certificate_name='UserTDECertificate_certificate_name',
    @certificate_file_s3_arn='arn:aws:s3:::bucket_name/certificate_file_name.cer',
    @private_key_file_s3_arn='arn:aws:s3:::bucket_name/key_file_name.pvk',
    @kms_password_key_arn='arn:aws:kms:region:account-id:key/key-id';
```

다음 파라미터는 필수 파라미터입니다.

- `@certificate_name` - 복원할 TDE 인증서의 이름입니다. 이름이 `UserTDECertificate_` 접두사로 시작해야 합니다.
- `@certificate_file_s3_arn` - TDE 인증서를 복원하는 데 사용된 백업 파일의 S3 ARN입니다.
- `@private_key_file_s3_arn` - 복원할 TDE 인증서에 대한 프라이빗 키 백업 파일의 S3 ARN입니다.
- `@kms_password_key_arn` - 프라이빗 키 암호를 암호화하는 데 사용되는 대칭 KMS 키의 ARN입니다.

Example TDE 인증서 복원

```
EXECUTE msdb.dbo.rds_restore_tde_certificate
    @certificate_name='UserTDECertificate_myTDEcertificate',
    @certificate_file_s3_arn='arn:aws:s3:::TDE_certs/mycertfile.cer',
```

```
    @private_key_file_s3_arn='arn:aws:s3:::TDE_certs/mykeyfile.pvk',
    @kms_password_key_arn='arn:aws:kms:us-
west-2:123456789012:key/AKIAIOSFODNN7EXAMPLE';
```

복원된 TDE 인증서 보기

rds_fn_list_user_tde_certificates 함수를 통해 복원된(가져온) 사용자 TDE 인증서를 볼 수 있습니다. 다음 구문을 사용합니다.

```
SELECT * FROM msdb.dbo.rds_fn_list_user_tde_certificates();
```

다음과 유사하게 출력됩니다. 여기에는 일부 열이 표시되지 않습니다.

name	certificate_id	principal_id	pvt_key_encrypt_type_c	issuer_name	cert_serial_num	thumbprint	subject	start_date	expiry_date	pvt_key_last_backup_date
UserTDCertific_tde_c	343	1	ENCRYPR_BY_MAR_KEY	AnyCompany Shippi	793e57a369fd1d9e472c32671d9caaf	0x6BB2341103 80BFE1BA2C69509 5B5	AnyCompany Shippi	2022-05 19:49:000000	2023-05 19:49:000000	NULL

복원된 TDE 인증서 삭제

사용하지 않는 복원된(가져온) 사용자 TDE 인증서를 삭제하려면 `rds_drop_tde_certificate` 저장 프로시저를 사용합니다. 다음 구문을 사용합니다.

```
EXECUTE msdb.dbo.rds_drop_tde_certificate
 @certificate_name='UserTDECertificate_certificate_name';
```

다음 파라미터는 필수입니다.

- `@certificate_name` - 삭제할 TDE 인증서의 이름입니다.

복원된(가져온) TDE 인증서만 삭제할 수 있습니다. RDS 생성 인증서는 삭제할 수 없습니다.

Example TDE 인증서 삭제

```
EXECUTE msdb.dbo.rds_drop_tde_certificate
 @certificate_name='UserTDECertificate_myTDEcertificate';
```

온프레미스 데이터베이스에 대한 TDE 인증서 백업 및 복원

온프레미스 데이터베이스에 대한 TDE 인증서를 백업한 다음 나중에 RDS for SQL Server로 복원할 수 있습니다. RDS for SQL Server TDE 인증서를 온프레미스 DB 인스턴스로 복원할 수도 있습니다.

다음 절차에서는 TDE 인증서 및 프라이빗 키를 백업합니다. 프라이빗 키는 대칭 암호화 KMS 키에서 생성된 데이터 키를 사용하여 암호화됩니다.

온프레미스 TDE 인증서를 백업하려면

1. AWS CLI generate-data-key 명령을 사용하여 데이터 키를 생성합니다.

   ```
   aws kms generate-data-key \
       --key-id my_KMS_key_ID \
       --key-spec AES_256
   ```

   다음과 유사하게 출력됩니다.

   ```
   {
   "CiphertextBlob": "AQIDAHimL2NEoAlOY6Bn7LJfnxi/OZe9kTQo/
   XQXduug1rmerwGiL7g5ux4av9GfZLxYTDATAAAAfjB8BgkqhkiG9w0B
   ```

```
BwagbzBtAgEAMGgGCSqGSIb3DQEHATAeBglghkgBZQMEAS4wEQQMyCxLMi7GRZgKqD65AgEQgDtjvZLJo2cQ31Vetng
2RezQy3sAS6ZHrCjfnfn0c65bFdhsXxjSMnudIY7AKw==",
"Plaintext": "U/fpGtmzGCYBi8A2+0/9qcRQRK2zmG/aOn939ZnKi/0=",
"KeyId": "arn:aws:kms:us-west-2:123456789012:key/1234abcd-00ee-99ff-88dd-
aa11bb22cc33"
}
```

다음 단계에서 일반 텍스트 출력을 프라이빗 키 암호로 사용합니다.

2. 다음 예와 같이 TDE 인증서를 백업합니다.

```
BACKUP CERTIFICATE myOnPremTDEcertificate TO FILE = 'D:\tde-cert-backup.cer'
WITH PRIVATE KEY (
FILE = 'C:\Program Files\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\DATA\cert-
backup-key.pvk',
ENCRYPTION BY PASSWORD = 'U/fpGtmzGCYBi8A2+0/9qcRQRK2zmG/aOn939ZnKi/0=');
```

3. Amazon S3 인증서 버킷에 인증서 백업 파일을 저장합니다.
4. 파일의 메타데이터에 다음 태그를 사용하여 프라이빗 키 백업 파일을 S3 인증서 버킷에 저장합니다.

   - 키 - `x-amz-meta-rds-tde-pwd`
   - 값 - 다음 예와 같이 데이터 키를 생성할 때의 `CiphertextBlob` 값입니다.

   ```
   AQIDAHimL2NEoAlOY6Bn7LJfnxi/OZe9kTQo/
   XQXduug1rmerwGiL7g5ux4av9GfZLxYTDATAAAAfjB8BgkqhkiG9w0B
   BwagbzBtAgEAMGgGCSqGSIb3DQEHATAeBglghkgBZQMEAS4wEQQMyCxLMi7GRZgKqD65AgEQgDtjvZLJo2cQ31Vet
   2RezQy3sAS6ZHrCjfnfn0c65bFdhsXxjSMnudIY7AKw==
   ```

다음 절차에서는 RDS for SQL Server TDE 인증서를 온프레미스 DB 인스턴스에 복원합니다. 인증서 백업, 해당 프라이빗 키 파일 및 데이터 키를 사용하여 대상 DB 인스턴스에서 TDE 인증서를 복사하고 복원합니다. 복원된 인증서는 새 서버의 데이터베이스 마스터 키로 암호화됩니다.

TDE 인증서를 복원하려면

1. Amazon S3에서 대상 인스턴스로 TDE 인증서 백업 파일과 프라이빗 키 파일을 복사합니다. Amazon S3에서 파일 복사에 대한 자세한 내용은 RDS for SQL Server와 Amazon S3 간 파일 전송 섹션을 참조하세요.
2. KMS 키로 출력 암호 텍스트를 해독하여 데이터 키의 일반 텍스트를 검색합니다. 암호 텍스트는 프라이빗 키 백업 파일의 S3 메타데이터에 있습니다.

```
aws kms decrypt \
    --key-id my_KMS_key_ID \
    --ciphertext-blob fileb://exampleCiphertextFile | base64 -d \
    --output text \
    --query Plaintext
```

다음 단계에서 일반 텍스트 출력을 프라이빗 키 암호로 사용합니다.

3. TDE 인증서를 복원하려면 다음 SQL 명령을 사용합니다.

```
CREATE CERTIFICATE myOnPremTDEcertificate FROM FILE='D:\tde-cert-backup.cer'
WITH PRIVATE KEY (FILE = N'D:\tde-cert-key.pvk',
DECRYPTION BY PASSWORD = 'plain_text_output');
```

KMS 해독에 대한 자세한 내용은 AWS CLI 명령 참조의 KMS 섹션에서 해독을 참조하세요.

TDE 인증서가 대상 DB 인스턴스에 복원된 후 해당 인증서를 사용하여 암호화된 데이터베이스를 복원할 수 있습니다.

> **Note**
>
> 동일한 TDE 인증서를 사용하여 소스 DB 인스턴스의 여러 SQL Server 데이터베이스를 암호화할 수 있습니다. 여러 데이터베이스를 대상 인스턴스로 마이그레이션하려면 연결된 TDE 인증서를 대상 인스턴스에 한 번만 복사합니다.

RDS for SQL Server에 대한 TDE 비활성화

RDS for SQL Server DB 인스턴스에서 TDE를 비활성화하려면 먼저 DB 인스턴스에 암호화된 객체가 하나도 없어야 합니다. 이렇게 하려면 객체의 암호를 해독하거나 삭제하면 됩니다. DB 인스턴스에 암호화된 객체가 남아 있을 경우에는 DB 인스턴스에서 TDE를 비활성할 수 없습니다. 콘솔을 사용하여 옵션 그룹에서 TDE 옵션을 제거하면 콘솔에 처리 중으로 표시됩니다. 또한 옵션 그룹이 암호화된 DB 인스턴스 또는 DB 스냅샷과 연결되어 있으면 오류 이벤트가 생성됩니다.

다음은 `customerDatabase`라고 하는 데이터베이스에서 TDE 암호화를 제거하는 예제입니다.

```
------------- Removing TDE ----------------

USE [customerDatabase]
```

```
GO

-- Turn off encryption of the database
ALTER DATABASE [customerDatabase]
SET ENCRYPTION OFF
GO

-- Wait until the encryption state of the database becomes 1. The state is 5
 (Decryption in progress) for a while
SELECT db_name(database_id) as DatabaseName, * FROM sys.dm_database_encryption_keys
GO

-- Drop the DEK used for encryption
DROP DATABASE ENCRYPTION KEY
GO

-- Alter to SIMPLE Recovery mode so that your encrypted log gets truncated
USE [master]
GO
ALTER DATABASE [customerDatabase] SET RECOVERY SIMPLE
GO
```

모든 객체의 암호를 해독할 때는 2가지 옵션을 사용할 수 있습니다.

1. TDE 옵션 없이 옵션 그룹과 연결되도록 DB 인스턴스를 수정할 수 있습니다.
2. 옵션 그룹에서 TDE 옵션을 제거할 수 있습니다.

SQL Server Audit

Amazon RDS에서는 기본 제공 SQL Server Audit 메커니즘을 사용하여 Microsoft SQL Server 데이터베이스를 감사할 수 있습니다. 온프레미스 데이터베이스 서버용으로 생성하는 경우와 같은 방식으로 감사 및 감사 사양을 생성할 수 있습니다.

RDS는 사용자가 제공한 IAM 역할을 사용하여 완료된 감사 로그를 S3 버킷에 업로드합니다. 보존을 활성화하면 RDS는 구성된 기간 동안 DB 인스턴스에 대한 감사 로그를 유지합니다.

자세한 내용은 Microsoft SQL Server 설명서의 SQL Server Audit(데이터베이스 엔진)를 참조하십시오.

SQL Server Audit와 데이터베이스 활동 스트림 함께 사용

RDS용 데이터베이스 활동 스트림을 사용하여 SQL Server Audit 이벤트를 Imperva, McAfee 및 IBM의 데이터베이스 활동 모니터링 도구와 통합할 수 있습니다. RDS SQL Server에 대해 데이터베이스 활동 스트림을 사용하여 감사를 수행하는 방법에 대한 자세한 내용은 Microsoft SQL Server에서의 감사 섹션을 참조하세요.

주제

- SQL Server Audit 지원
- DB 인스턴스 옵션에 SQL Server Audit 추가
- SQL Server Audit의 사용
- 감사 로그 보기
- 다중 AZ 인스턴스에서 SQL Server Audit 사용
- S3 버킷 구성
- 수동으로 SQL Server Audit에 대한 IAM 역할 생성

SQL Server Audit 지원

SQL Server 2016부터는 Amazon RDS에서 모든 버전의 SQL Server가 서버 수준 감사를 지원하고 Enterprise Edition은 데이터베이스 수준 감사도 지원합니다. SQL Server 2016(13.x) SP1부터는 모든 버전이 서버 및 데이터베이스 수준 감사를 모두 지원합니다. 자세한 내용은 SQL Server 설명서의 SQL Server Audit(데이터베이스 엔진)를 참조하십시오.

RDS는 SQL Server Audit에 대한 다음 옵션 설정 구성을 지원합니다.

옵션 설정	유효한 값	설명
IAM_ROLE_ARN	arn:aws:iam:: *account-id* :role/*role-name* 형식의 유효한 ARN(Amazon 리소스 이름).	감사 로그를 저장할 S3 버킷에 액세스 권한을 부여하는 IAM 역할의 ARN입니다. 자세한 내용은 AWS 일반 참조의 Amazon 리소스 이름(ARN)을 참조하세요.
S3_BUCKET_ARN	arn:aws:s3::: *amzn-s3-demo-bucket* 또는 arn:aws:s3::: *amzn-s3-demo-bucket* /key-prefix 형식의 유효한 ARN	감사 로그를 저장할 S3 버킷의 ARN입니다.
ENABLE_COMPRESSION	true 또는 false	감사 로그 압축을 제어합니다. 기본적으로 압축은 활성화됩니다(true로 설정).
RETENTION_TIME	0~840	SQL Server Audit 레코드가 RDS 인스턴스에 유지되는 보존 기간(시간 단위)입니다. 기본적으로 보존은 비활성화됩니다.

DB 인스턴스 옵션에 SQL Server Audit 추가

SQL Server Audit를 활성화하려면 DB 인스턴스에 옵션을 활성화하고 SQL Server 내에서 이 기능을 활성화하는 두 단계가 필요합니다. SQL Server Audit 옵션을 DB 인스턴스에 추가하는 프로세스는 다음과 같습니다.

1. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 복사 또는 수정합니다.
2. 필요한 모든 옵션을 추가하고 구성하십시오.
3. 옵션 그룹을 DB 인스턴스에 연동시킵니다.

SQL Server Audit 옵션을 추가한 후 DB 인스턴스를 다시 시작할 필요가 없습니다. 옵션 그룹이 활성화되면 S3 버킷에 감사를 생성하고 감사 로그를 저장할 수 있습니다.

DB 인스턴스의 옵션 그룹에 SQL Server Audit을 추가하고 구성하려면

1. 다음 중 하나를 선택합니다.
   - 기존 옵션 그룹을 사용합니다.
   - 사용자 지정 DB 옵션 그룹을 생성하고 해당 옵션 그룹을 사용합니다. 자세한 내용은 옵션 그룹 생성 섹션을 참조하세요.
2. 옵션 그룹에 SQLSERVER_AUDIT 옵션을 추가하고 옵션 설정을 구성합니다. 옵션 추가에 대한 자세한 내용은 옵션 그룹에 옵션 추가 섹션을 참조하세요.
   - IAM 역할에 필요한 정책의 IAM 역할이 이미 있는 경우 해당 역할을 선택할 수 있습니다. 새 IAM 역할을 만들려면 새 역할 생성을 선택하십시오. 필요한 정책에 대한 자세한 내용은 수동으로 SQL Server Audit에 대한 IAM 역할 생성 단원을 참조하십시오.
   - S3 대상 선택의 경우 사용할 S3 버킷이 이미 있는 경우 선택합니다. S3 버킷을 생성하려면 S3 버킷 새로 만들기를 선택하십시오.
   - 압축 활성화의 경우 감사 파일을 압축하려면 이 옵션을 선택한 상태로 둡니다. 압축은 기본적으로 활성화되어 있습니다. 압축을 비활성화하려면 Enable Compression(압축 활성화) 선택을 취소하십시오.
   - 감사 로그 보존의 경우 DB 인스턴스에 감사 레코드를 보존하려면 이 옵션을 선택합니다. 보존 시간을 시간 단위로 지정하십시오. 최대 보존 기간은 35일입니다.
3. 옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스에 적용합니다. 다음 중 하나를 선택합니다.
   - 새 DB 인스턴스를 생성하는 경우, 인스턴스를 시작할 때 옵션 그룹을 적용하십시오.
   - 기존 DB 인스턴스의 경우, 해당 인스턴스를 수정한 후 새 옵션 그룹을 연결하여 옵션 그룹을 적용하십시오. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

SQL Server Audit 옵션 수정

SQL Server Audit 옵션을 활성화하면 설정을 수정할 수 있습니다. 옵션 설정을 변경하는 방법에 대한 자세한 내용은 옵션 설정 수정 단원을 참조하십시오.

DB 인스턴스 옵션에서 SQL Server Audit 제거

감사를 비활성화한 다음 옵션을 삭제하여 SQL Server Audit 기능을 해제할 수 있습니다.

감사를 제거하려면

1. SQL Server 내부의 모든 감사 설정을 비활성화합니다. 감사가 실행되는 위치를 확인하려면 SQL Server 보안 카탈로그 보기를 쿼리하십시오. 자세한 내용은 Microsoft SQL Server 설명서의 보안 카탈로그 보기를 참조하십시오.
2. DB 인스턴스에서 SQL Server Audit 옵션을 삭제하십시오. 다음 중 하나를 선택합니다.
   - DB 인스턴스가 사용하는 옵션 그룹에서 SQL Server Audit 옵션을 삭제하십시오. 이 변경은 동일한 옵션 그룹을 사용하는 모든 DB 인스턴스에 영향을 미칩니다. 자세한 내용은 옵션 그룹에서 옵션 제거 섹션을 참조하세요.
   - DB 인스턴스를 수정한 다음 SQL Server Audit 옵션이 없는 옵션 그룹을 선택합니다. 이 변경은 수정하는 DB 인스턴스에만 영향을 줍니다. 기본(빈) 옵션 그룹을 지정하거나 다른 사용자 지정 옵션 그룹을 지정할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.
3. DB 인스턴스에서 SQL Server Audit 옵션을 삭제한 후 인스턴스를 다시 시작할 필요가 없습니다. S3 버킷에서 불필요한 감사 파일을 제거하십시오.

SQL Server Audit의 사용

온프레미스 데이터베이스 서버에서 제어하는 것과 동일한 방식으로 서버 감사, 서버 감사 사양 및 데이터베이스 감사 사양을 제어할 수 있습니다.

감사 생성

온프레미스 데이터베이스 서버에 대해 생성하는 것과 같은 방법으로 서버 감사를 생성합니다. 서버 감사를 생성하는 방법에 대한 자세한 내용은 Microsoft SQL Server 설명서에서 서버 감사 생성을 참조하십시오.

오류를 방지하려면 다음 제한 사항을 준수하십시오.

- 인스턴스당 지원되는 최대 50개의 서버 감사를 초과하지 마십시오.
- 이진 파일에 데이터를 쓰도록 SQL Server에 지시하십시오.
- 서버 감사 이름의 접두사로 `RDS_`를 사용하지 마십시오.
- `FILEPATH`에 `D:\rdsdbdata\SQLAudit`을 지정합니다.
- `MAXSIZE`에 2MB ~ 50MB 사이의 크기를 지정하십시오.
- `MAX_ROLLOVER_FILES` 또는 `MAX_FILES`를 구성하지 마십시오.
- SQL Server가 감사 레코드를 쓰지 못할 경우 DB 인스턴스를 종료하도록 구성하지 마십시오.

감사 사양 생성

온프레미스 데이터베이스 서버에서 생성하는 것과 동일한 방식으로 서버 감사 사양 및 데이터베이스 감사 사양을 생성합니다. 감사 사양 생성에 대한 자세한 내용은 Microsoft SQL Server 설명서에서 서버 감사 사양 생성 및 데이터베이스 감사 사양 생성을 참조하십시오.

오류를 피하려면 데이터베이스 감사 사양 또는 서버 감사 사양의 접두사로 `RDS_`를 사용하지 마십시오.

감사 로그 보기

감사 로그는 `D:\rdsdbdata\SQLAudit`에 저장됩니다.

SQL Server가 감사 로그 파일에 파일 쓰기를 완료하면—파일이 크기 제한에 도달하면—Amazon RDS가 S3 버킷에 파일을 업로드합니다. 보존을 활성화하면 Amazon RDS가 파일을 보존 폴더 `D:\rdsdbdata\SQLAudit\transmitted`로 옮깁니다.

보존 구성에 대한 자세한 내용은 DB 인스턴스 옵션에 SQL Server Audit 추가 단원을 참조하십시오.

감사 레코드는 감사 로그 파일이 업로드될 때까지 DB 인스턴스에 보관됩니다. 다음 명령을 실행하여 감사 레코드를 볼 수 있습니다.

```
SELECT   *
 FROM     msdb.dbo.rds_fn_get_audit_file
             ('D:\rdsdbdata\SQLAudit\*.sqlaudit'
             , default
             , default )
```

같은 명령을 사용하여 필터를 `D:\rdsdbdata\SQLAudit\transmitted\*.sqlaudit`로 변경하여 보존 폴더의 감사 레코드를 볼 수 있습니다.

```
SELECT   *
 FROM     msdb.dbo.rds_fn_get_audit_file
             ('D:\rdsdbdata\SQLAudit\transmitted\*.sqlaudit'
             , default
             , default )
```

다중 AZ 인스턴스에서 SQL Server Audit 사용

다중 AZ 인스턴스의 경우 감사 로그 파일을 Amazon S3으로 보내는 프로세스는 단일 AZ 인스턴스의 프로세스와 유사합니다. 그러나 몇 가지 중요한 차이점이 있습니다.

- 데이터베이스 감사 사양 객체는 모든 노드에 복제됩니다.
- 서버 감사 및 서버 감사 사양은 보조 노드에 복제되지 않습니다. 대신, 수동으로 생성하거나 수정해야 합니다.

두 노드에서 서버 감사 사양 또는 서버 감사를 캡처하려면

1. 기본 노드에서 서버 감사 또는 서버 감사 사양을 생성하십시오.
2. 보조 노드로 장애 조치하고 보조 노드에서 동일한 이름과 GUID로 서버 감사 또는 서버 감사 사양을 생성합니다. `AUDIT_GUID` 파라미터를 사용하여 GUID를 지정합니다.

S3 버킷 구성

감사 로그 파일은 DB 인스턴스에서 S3 버킷으로 자동 업로드됩니다. 감사 파일의 대상으로 사용하는 S3 버킷에는 다음 제한이 적용됩니다.

- DB 인스턴스와 동일한 AWS 리전에 있어야 합니다.
- 대중에게 공개되어서는 안 됩니다.
- 버킷 소유자는 IAM 역할 소유자여야 합니다.
- IAM 역할에는 S3 버킷 서버 측 암호화와 관련된 고객 관리형 KMS 키에 대한 권한이 있어야 합니다.

데이터를 저장하는 데 사용되는 대상 키는 *amzn-s3-demo-bucket*`/key-prefix/instance-name/audit-name/node_file-name.ext` 명명 스키마를 따릅니다.

> **Note**
>
> 버킷 이름과 키 접두사 값을 모두 `S3_BUCKET_ARN` 옵션 설정을 사용하여 설정합니다.

스키마는 다음 요소로 구성됩니다.

- ***amzn-s3-demo-bucket*** – S3 버킷의 이름입니다.
- **`key-prefix`** – 감사 로그에 사용할 사용자 지정 키 접두사.
- **`instance-name`** – Amazon RDS 인스턴스의 이름.
- **`audit-name`** – 감사의 이름.

- **node** – 감사 로그의 소스인 노드의 식별자(node1 또는 node2). 단일 AZ 인스턴스에는 하나의 노드가 있고 다중 AZ 인스턴스에는 두 개의 복제 노드가 있습니다. 기본 노드와 보조 노드의 역할은 시간이 지남에 따라 변하기 때문에 이들은 기본 노드와 보조 노드가 아닙니다. 대신, 노드 식별자는 단순한 레이블입니다.
  - **node1** – 첫 번째 복제 노드(단일 AZ에는 하나의 노드만 있음).
  - **node2** – 두 번째 복제 노드(다중 AZ에는 두 개의 노드가 있음).
- **file-name** – 대상 파일 이름. 파일 이름은 SQL Server에서 그대로 가져옵니다.
- **ext** – 파일 확장자(`zip` 또는 `sqlaudit`):
  - **zip** – 압축이 활성화된 경우(기본값).
  - **sqlaudit** – 압축이 비활성화된 경우.

수동으로 SQL Server Audit에 대한 IAM 역할 생성

일반적으로 새 옵션을 만들면 AWS Management Console은 IAM 역할 및 IAM 신뢰 정책을 생성합니다. 그러나 SQL Server Audit에서 사용할 새로운 IAM 역할을 수동으로 생성할 수 있으므로 추가 요구 사항에 맞게 사용자 지정할 수 있습니다. 그러려면 IAM 역할을 생성하고 Amazon RDS 서비스가 Amazon S3 버킷을 사용할 수 있도록 권한을 위임합니다. IAM 역할을 만들 때 신뢰 및 권한 정책을 연결합니다. 신뢰 정책은 Amazon RDS가 이 역할을 맡도록 허용합니다. 권한 정책은 이 역할이 수행할 수 있는 작업을 정의합니다. 자세한 내용은 AWS Identity and Access Management 사용 설명서에서 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.

이 섹션의 예를 사용하여 필요한 신뢰 관계와 사용 권한 정책을 생성할 수 있습니다.

다음 예는 SQL Server Audit에 대한 신뢰 관계를 보여줍니다. 이 관계는 서비스 원칙 `rds.amazonaws.com`을 사용하여 RDS가 S3 버킷에 쓸 수 있도록 허용합니다. 서비스 보안 주체는 서비스에 권한을 부여하는 데 사용되는 식별자입니다. 이러한 방식으로 `rds.amazonaws.com`에 대한 액세스를 허용할 때마다 RDS가 사용자를 대신하여 작업을 수행하도록 허용합니다. 서비스 보안 주체에 대한 자세한 내용은 AWS JSON 정책 요소: 보안 주체를 참조하세요.

Example SQL Server Audit에 대한 신뢰 관계

JSON

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
```

```
            "Effect": "Allow",
            "Principal": {
                "Service": "rds.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

서비스 권한을 특정 리소스로 제한하는 리소스 기반 신뢰 관계의 aws:SourceArn 및 aws:SourceAccount 전역 조건 컨텍스트 키를 사용하는 것이 좋습니다. 이는 혼동된 대리자 문제를 방지하는 가장 효과적인 방법입니다.

전역 조건 컨텍스트 키를 모두 사용하고 aws:SourceArn 값에 계정 ID가 포함되도록 할 수 있습니다. 이 경우 aws:SourceAccount 값과 aws:SourceArn 값의 계정이 동일한 문에서 사용될 때 동일한 계정 ID를 사용해야 합니다.

- 단일 리소스에 대한 교차 서비스 액세스를 원하는 경우 aws:SourceArn을 사용하세요.
- 해당 계정의 모든 리소스가 교차 서비스 사용과 연결되도록 허용하려는 경우 aws:SourceAccount를 사용하세요.

신뢰 정책에서는 역할에 액세스하는 리소스의 전체 Amazon 리소스 이름(ARN)이 포함된 aws:SourceArn 전역 조건 컨텍스트 키를 사용해야 합니다. SQL Server Audit의 경우 다음 예와 같이 DB 옵션 그룹과 DB 인스턴스를 모두 포함해야 합니다.

Example SQL Server Audit에 대한 전역 조건 컨텍스트 키와의 신뢰 관계

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rds.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
```

```
                "StringEquals": {
                    "aws:SourceArn": [

 "arn:aws:rds:Region:my_account_ID:db:db_instance_identifier",
                        "arn:aws:rds:Region:my_account_ID:og:option_group_name"
                    ]
                }
            }
        }
    ]
}
```

SQL Server Audit에 대한 권한 정책의 다음 예에서는 Amazon S3 버킷에 대한 ARN을 지정합니다. ARN을 사용하여 액세스 권한을 부여할 특정 계정, 사용자 또는 역할을 식별할 수 있습니다. ARN 사용에 대한 자세한 내용은 Amazon 리소스 이름(ARN)을 참조하세요.

Example SQL Server Audit에 대한 권한 정책

JSON

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Effect": "Allow",
             "Action": "s3:ListAllMyBuckets",
             "Resource": "*"
         },
         {
             "Effect": "Allow",
             "Action": [
                 "s3:ListBucket",
                 "s3:GetBucketACL",
                 "s3:GetBucketLocation"
             ],
             "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
         },
         {
             "Effect": "Allow",
             "Action": [
```

```
                "s3:PutObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/key_prefix/*"
        }
    ]
}
```

> **Note**
>
> `s3:ListAllMyBuckets` 작업은 동일한 AWS 계정이 S3 버킷과 SQL Server DB 인스턴스를 모두 소유하고 있는지 확인하는 데 필요합니다. 이 작업은 계정에 있는 버킷의 이름을 나열합니다.
> S3 버킷 네임스페이스는 글로벌입니다. 실수로 버킷을 삭제한 경우 다른 사용자가 동일한 이름의 버킷을 다른 계정에서 만들 수 있습니다. 그런 다음 SQL Server 감사 데이터가 새 버킷에 기록됩니다.

Amazon RDS for SQL Server에서 SQL Server Analysis Services 지원

Microsoft SQL Server Analysis Services(SSAS)는 Microsoft Business Intelligence(MSBI) 제품군의 일부입니다. SSAS는 SQL Server 내에 설치된 온라인 분석 처리(OLAP) 및 데이터 마이닝 도구입니다. SSAS를 사용해 데이터를 분석하여 비즈니스 의사 결정을 내릴 수 있습니다. SSAS는 비즈니스 인텔리전스 환경에서 일반적인 쿼리 및 계산에 최적화되어 있기 때문에 SQL Server 관계형 데이터베이스와 다릅니다.

기존 DB 인스턴스 또는 새 DB 인스턴스에 대해 SSAS를 설정할 수 있습니다. 이 인스턴스는 데이터베이스 엔진과 동일한 DB 인스턴스에 설치됩니다. SSAS에 대한 자세한 내용은 Microsoft Analysis Services 설명서를 참조하십시오.

Amazon RDS는 다음 버전에서 SQL Server Standard 및 Enterprise Edition용 SSAS를 지원합니다.

- 테이블 형식 모드:
  - SQL Server 2019, 버전 15.00.4043.16.v1 이상
  - SQL Server 2017, 버전 14.00.3223.3.v1 이상
  - SQL Server 2016, 버전 13.00.5426.0.v1 이상
- 다차원 모드:
  - SQL Server 2019, 버전 15.00.4153.1.v1 이상
  - SQL Server 2017, 버전 14.00.3381.3.v1 이상
  - SQL Server 2016, 버전 13.00.5882.1.v1 이상

목차

- 제한 사항
- SSAS 설정
  - SSAS용 옵션 그룹 생성
  - 옵션 그룹에 SSAS 옵션 추가
  - 옵션 그룹을 DB 인스턴스와 연결
  - VPC 보안 그룹에 대한 인바운드 액세스 허용
  - Amazon S3 통합 사용 설정
- Amazon RDS에 SSAS 프로젝트 배포
- 배포 작업의 상태 모니터링
- Amazon RDS에서 SSAS 사용

  - SSAS용 Windows 인증 사용자 설정
  - 도메인 사용자를 데이터베이스 관리자로 추가
  - SSAS 프록시 생성
  - SQL Server 에이전트를 사용하여 SSAS 데이터베이스 처리 예약
  - 프록시에서 SSAS 액세스 취소
- SSAS 데이터베이스 백업
- SSAS 데이터베이스 복원
  - DB 인스턴스를 지정된 시간으로 복원
- SSAS 모드 변경
- SSAS 해제
- SSAS 문제 해결

제한 사항

RDS for SQL Server에서 SSAS를 사용하는 경우 다음과 같은 제한 사항이 적용됩니다.

- RDS for SQL Server는 테이블 형식 또는 다차원 모드에서 SSAS 실행을 지원합니다. 자세한 내용은 Microsoft 설명서의 테이블 형식 및 다차원 솔루션 비교를 참조하세요.
- 한 번에 하나의 SSAS 모드만 사용할 수 있습니다. 모드를 변경하기 전에 모든 SSAS 데이터베이스를 삭제해야 합니다.

  자세한 내용은 SSAS 모드 변경 단원을 참조하십시오.
- 다중 AZ 인스턴스는 지원되지 않습니다.
- 인스턴스는 SSAS 인증을 위해 자체 관리형 Active Directory 또는 AWS Directory Service for Microsoft Active Directory를 사용해야 합니다. 자세한 내용은 RDS for SQL Server를 사용하여 Active Directory 작업 단원을 참조하십시오.
- 사용자에게는 SSAS 서버 관리자 액세스 권한이 부여되지 않지만 데이터베이스 수준 관리자 액세스 권한을 부여받을 수 있습니다.
- SSAS에 액세스하기 위해 지원되는 유일한 포트는 2383입니다.
- 프로젝트를 직접 배포할 수 없습니다. 이를 수행할 RDS 저장 프로시저를 제공합니다. 자세한 내용은 Amazon RDS에 SSAS 프로젝트 배포 섹션을 참조하세요.
- 배포 중 처리는 지원되지 않습니다.
- .xmla 파일을 배포에 사용하는 것은 지원되지 않습니다.

- SSAS 프로젝트 입력 파일 및 데이터베이스 백업 출력 파일은 DB 인스턴스의 `D:\S3` 폴더에만 있을 수 있습니다.

SSAS 설정

다음 프로세스를 사용하여 DB 인스턴스에 SSAS를 설정합니다.

1. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 선택합니다.
2. [SSAS] 옵션을 옵션 그룹에 추가합니다.
3. 옵션 그룹을 DB 인스턴스에 연결합니다.
4. SSAS 리스너 포트의 Virtual Private Cloud(VPC) 보안 그룹에 대한 인바운드 액세스를 허용합니다.
5. Amazon S3 통합을 설정합니다.

SSAS용 옵션 그룹 생성

AWS Management Console 또는 AWS CLI를 사용하여 SQL Server 엔진과 사용하려는 DB 인스턴스 버전에 해당하는 옵션 그룹을 생성합니다.

> **Note**
>
> 올바른 SQL Server 엔진 및 버전인 경우 기존 옵션 그룹을 사용할 수도 있습니다.

콘솔

다음 콘솔 프로시저는 SQL Server Standard 에디션 2017에 대한 옵션 그룹을 생성합니다.

옵션 그룹을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 그룹 생성을 선택합니다.
4. 보안 그룹 생성 창에서 다음을 수행합니다.
   a. [이름(Name)]에 AWS 계정 내에서 쉽게 식별할 수 있는 옵션 그룹 이름을 입력합니다(예: **`ssas-se-2017`**). 이름은 글자, 숫자 및 하이픈만 사용 가능합니다.

b. 설명에 옵션 그룹에 대한 간단한 설명을 입력합니다(예: **SSAS option group for SQL Server SE 2017**). 이 설명은 표시 용도로만 사용됩니다.

c. 엔진에 대해 sqlserver-se를 선택합니다.

d. 메이저 엔진 버전에 대해 14.00을 선택합니다.

5. 생성(Create)을 선택합니다.

CLI

다음 CLI 예제는 SQL Server Standard 에디션 2017에 대한 옵션 그룹을 생성합니다.

옵션 그룹을 생성하려면

- 다음 명령 중 하나를 사용합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds create-option-group \
    --option-group-name ssas-se-2017 \
    --engine-name sqlserver-se \
    --major-engine-version 14.00 \
    --option-group-description "SSAS option group for SQL Server SE 2017"
```

  Windows의 경우:

```
aws rds create-option-group ^
    --option-group-name ssas-se-2017 ^
    --engine-name sqlserver-se ^
    --major-engine-version 14.00 ^
    --option-group-description "SSAS option group for SQL Server SE 2017"
```

옵션 그룹에 SSAS 옵션 추가

그런 다음 AWS Management Console 또는 AWS CLI를 사용하여 SSAS 옵션을 옵션 그룹에 추가합니다.

콘솔

SSAS 옵션을 추가하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 방금 생성한 옵션 그룹을 선택합니다.
4. 옵션 추가를 선택합니다.
5. 옵션 세부 정보에서 옵션 이름으로 SSAS를 선택합니다.
6. 옵션 설정에서 다음을 수행합니다.

   a. 최대 메모리(Max memory)에 10~80 범위의 값을 입력합니다.

      최대 메모리는 SSAS가 실행 중인 요청과 우선 순위가 높은 새 요청에 대한 공간을 확보하기 위해 메모리를 보다 적극적으로 해제하기 시작하는 상한 임계값을 지정합니다. 숫자는 DB 인스턴스에 대한 총 메모리 비율입니다. 허용 값은 10–80이고, 기본값은 45입니다.

   b. 모드(Mode)에서 SSAS 서버 모드인 테이블 형식(Tabular) 또는 다차원(Multidimensional)을 선택합니다.

      모드(Mode) 옵션 설정이 표시되지 않으면 해당 AWS 리전에서 다차원 모드가 지원되지 않는 것입니다. 자세한 내용은 제한 사항 단원을 참조하십시오.

      테이블 형식(Tabular)이 기본값입니다.

   c. 보안 그룹의 경우 옵션과 연결할 VPC 보안 그룹을 선택합니다.

   > **Note**
   >
   > SSAS, 2383에 액세스하기 위한 포트가 미리 채워져 있습니다.

7. 예약에서 옵션을 즉시 추가할지 또는 다음 유지 관리 기간에 추가할지를 선택합니다.
8. 옵션 추가를 선택합니다.

CLI

SSAS 옵션을 추가하려면

1. 다음 파라미터를 사용하여 JSON 파일(예: `ssas-option.json`)을 생성합니다.

   - `OptionGroupName` – 이전에 생성했거나 선택한 옵션 그룹의 이름입니다(다음 예의 경우 `ssas-se-2017`).
   - `Port` – SSAS에 액세스하는 데 사용하는 포트입니다. 지원되는 유일한 포트는 2383입니다.
   - `VpcSecurityGroupMemberships` – RDS DB 인스턴스에 대한 VPC 보안 그룹의 멤버십입니다.
   - `MAX_MEMORY` – SSAS가 실행 중인 요청과 우선 순위가 높은 새 요청에 대한 공간을 확보하기 위해 메모리를 보다 적극적으로 해제하기 시작해야 하는 상한 임계값입니다. 숫자는 DB 인스턴스에 대한 총 메모리 비율입니다. 허용 값은 10–80이고, 기본값은 45입니다.
   - `MODE` - SSAS 서버 모드(`Tabular` 또는 `Multidimensional`)입니다. `Tabular`가 기본값입니다.

     `MODE` 옵션 설정이 유효하지 않다는 오류가 나타나면 해당 AWS 리전에서 다차원 모드가 지원되지 않는 것입니다. 자세한 내용은 제한 사항 단원을 참조하십시오.

   다음은 SSAS 옵션 설정이 있는 JSON 파일의 예입니다.

   ```
   {
   "OptionGroupName": "ssas-se-2017",
   "OptionsToInclude": [
    {
    "OptionName": "SSAS",
    "Port": 2383,
    "VpcSecurityGroupMemberships": ["sg-0abcdef123"],
    "OptionSettings": [{"Name":"MAX_MEMORY","Value":"60"},
   {"Name":"MODE","Value":"Multidimensional"}]
    }],
   "ApplyImmediately": true
   }
   ```

2. [SSAS] 옵션을 옵션 그룹에 추가합니다.

   Example

   대상 LinuxmacOS, 또는Unix:

```
aws rds add-option-to-option-group \
    --cli-input-json file://ssas-option.json \
    --apply-immediately
```

Windows의 경우:

```
aws rds add-option-to-option-group ^
    --cli-input-json file://ssas-option.json ^
    --apply-immediately
```

옵션 그룹을 DB 인스턴스와 연결

콘솔 또는 CLI를 사용하여 옵션 그룹을 DB 인스턴스와 연결할 수 있습니다.

콘솔

옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스와 연결합니다.

- 새 DB 인스턴스의 경우 인스턴스를 시작할 때 옵션 그룹을 DB 인스턴스와 연결합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
- 기존 DB 인스턴스의 경우 인스턴스를 수정하고 새 옵션 그룹을 인스턴스와 연결합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

> **Note**
>
> 기존 인스턴스를 사용하는 경우 이미 Active Directory 도메인과 AWS Identity and Access Management(IAM) 역할이 연결되어 있어야 합니다. 새 인스턴스를 생성하는 경우 기존 Active Directory 도메인 및 IAM 역할을 지정합니다. 자세한 내용은 RDS for SQL Server를 사용하여 Active Directory 작업 단원을 참조하십시오.

CLI

옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스와 연결할 수 있습니다.

> **Note**
>
> 기존 인스턴스를 사용하는 경우 이미 Active Directory 도메인과 IAM 역할이 연결되어 있어야 합니다. 새 인스턴스를 생성하는 경우 기존 Active Directory 도메인 및 IAM 역할을 지정합니다. 자세한 내용은 RDS for SQL Server를 사용하여 Active Directory 작업 단원을 참조하십시오.

옵션 그룹을 사용하는 DB 인스턴스를 생성하려면

- 옵션 그룹을 생성할 때 사용한 것과 동일한 DB 엔진 유형과 메이저 버전을 지정합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance \
    --db-instance-identifier myssasinstance \
    --db-instance-class db.m5.2xlarge \
    --engine sqlserver-se \
    --engine-version 14.00.3223.3.v1 \
    --allocated-storage 100 \
    --manage-master-user-password \
    --master-username admin \
    --storage-type gp2 \
    --license-model li \
    --domain-iam-role-name my-directory-iam-role \
    --domain my-domain-id \
    --option-group-name ssas-se-2017
```

  Windows의 경우:

```
aws rds create-db-instance ^
    --db-instance-identifier myssasinstance ^
    --db-instance-class db.m5.2xlarge ^
    --engine sqlserver-se ^
    --engine-version 14.00.3223.3.v1 ^
    --allocated-storage 100 ^
    --manage-master-user-password ^
    --master-username admin ^
    --storage-type gp2 ^
    --license-model li ^
```

```
    --domain-iam-role-name my-directory-iam-role ^
    --domain my-domain-id ^
    --option-group-name ssas-se-2017
```

DB 인스턴스를 수정하여 옵션 그룹을 연결하려면

- 다음 명령 중 하나를 사용합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier myssasinstance \
    --option-group-name ssas-se-2017 \
    --apply-immediately
```

  Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier myssasinstance ^
    --option-group-name ssas-se-2017 ^
    --apply-immediately
```

VPC 보안 그룹에 대한 인바운드 액세스 허용

DB 인스턴스와 연결된 VPC 보안 그룹에서 지정된 SSAS 리스너 포트에 대한 인바운드 규칙을 생성합니다. 보안 그룹 설정에 대한 자세한 내용은 보안 그룹을 생성하여 VPC 내부의 DB 인스턴스에 대한 액세스를 제공 단원을 참조하십시오.

Amazon S3 통합 사용 설정

배포를 위해 호스트에 모델 구성 파일을 다운로드하려면 Amazon S3 통합을 사용합니다. 자세한 내용은 Amazon RDS for SQL Server DB 인스턴스와 Amazon S3 통합 단원을 참조하십시오.

Amazon RDS에 SSAS 프로젝트 배포

RDS에서는 SQL Server Management Studio(SSMS)를 사용하여 SSAS 프로젝트를 직접 배포할 수 없습니다. 프로젝트를 배포하려면 RDS 저장 프로시저를 사용합니다.

> **Note**
>
> .xmla 파일을 배포에 사용하는 것은 지원되지 않습니다.

프로젝트를 배포하기 전에 다음 사항을 확인하십시오.

- Amazon S3 통합이 설정됩니다. 자세한 내용은 Amazon RDS for SQL Server DB 인스턴스와 Amazon S3 통합 단원을 참조하십시오.
- `Processing Option` 구성 설정이 `Do Not Process`로 설정됩니다. 이 설정은 배포 후 처리가 진행되지 않음을 의미합니다.
- *`myssasproject`*`.asdatabase` 파일과 *`myssasproject`*`.deploymentoptions` 파일이 모두 있습니다. SSAS 프로젝트를 빌드할 때 자동으로 생성됩니다.

RDS에 SSAS 프로젝트를 배포하려면

1. 다음 예제와 같이 S3 버킷에서 DB 인스턴스로 `.asdatabase`(SSAS 모델) 파일을 다운로드합니다. 다운로드 파라미터에 대한 자세한 내용은 Amazon S3 버킷의 파일을 SQL Server DB 인스턴스로 다운로드 단원을 참조하십시오.

   ```
   exec msdb.dbo.rds_download_from_s3
   @s3_arn_of_file='arn:aws:s3:::bucket_name/myssasproject.asdatabase',
   [@rds_file_path='D:\S3\myssasproject.asdatabase'],
   [@overwrite_file=1];
   ```

2. S3 버킷에서 DB 인스턴스로 `.deploymentoptions` 파일을 다운로드합니다.

   ```
   exec msdb.dbo.rds_download_from_s3
   @s3_arn_of_file='arn:aws:s3:::bucket_name/myssasproject.deploymentoptions',
   [@rds_file_path='D:\S3\myssasproject.deploymentoptions'],
   [@overwrite_file=1];
   ```

3. 프로젝트를 배포합니다.

   ```
   exec msdb.dbo.rds_msbi_task
   @task_type='SSAS_DEPLOY_PROJECT',
   @file_path='D:\S3\myssasproject.asdatabase';
   ```

배포 작업의 상태 모니터링

배포(또는 다운로드) 작업의 상태를 추적하려면 rds_fn_task_status 함수를 호출합니다. 두 가지 파라미터가 필요합니다. 첫 번째 파라미터는 SSAS에 적용되지 않기 때문에 항상 NULL이어야 합니다. 두 번째 파라미터는 작업 ID를 수락합니다.

모든 작업 목록을 보려면 다음 예와 같이 첫 번째 파라미터를 NULL로, 두 번째 파라미터를 0으로 설정하십시오.

```
SELECT * FROM msdb.dbo.rds_fn_task_status(NULL,0);
```

특정 작업을 수행하려면 다음 예와 같이 첫 번째 파라미터를 NULL로, 두 번째 파라미터를 작업 ID로 설정하십시오.

```
SELECT * FROM msdb.dbo.rds_fn_task_status(NULL,42);
```

rds_fn_task_status 함수는 다음 정보를 반환합니다.

출력 파라미터	설명
task_id	작업의 ID입니다.
task_type	SSAS의 경우 작업 유형은 다음과 같을 수 있습니다. • SSAS_DEPLOY_PROJECT • SSAS_ADD_DB_ADMIN_MEMBER • SSAS_BACKUP_DB • SSAS_RESTORE_DB
database_name	SSAS 작업에는 적용되지 않습니다.
% complete	백분율로 나타낸 작업의 진행률입니다.
duration (mins)	작업에 소요된 시간입니다(분).
lifecycle	작업의 상태입니다. 가능한 상태는 다음과 같습니다.

출력 파라미터	설명
	• CREATED – SSAS 저장 프로시저 중 하나를 호출하면 작업이 생성되고 상태가 로 설정됩니다.CREATED • IN_PROGRESS – 작업이 시작되면 상태가 로 설정됩니다.IN_PROGRESS CREATED에서 IN_PROGRESS 로 상태가 변경되려면 최대 5분이 걸릴 수 있습니다. • SUCCESS – 작업이 완료되면 상태가 로 설정됩니다.SUCCESS • ERROR – 작업이 실패하면 상태가 로 설정됩니다.ERROR 오류에 대한 자세한 내용은 task_info 열을 참조하십시오. • CANCEL_REQUESTED – rds_cancel_task 를 호출하는 즉시 작업의 상태가 CANCEL_REQUESTED 로 설정됩니다. • CANCELLED – 작업이 성공적으로 취소된 뒤에는 작업 상태가 로 설정됩니다.CANCELLED
task_info	작업에 대한 추가 정보입니다. 처리 중에 오류가 발생하면 이 열에 오류 정보가 포함됩니다. 자세한 내용은 SSAS 문제 해결 단원을 참조하십시오.
last_updated	작업 상태를 마지막으로 업데이트한 날짜와 시간입니다.
created_at	작업을 생성한 날짜와 시간입니다.

출력 파라미터	설명
S3_object_arn	SSAS 작업에는 적용되지 않습니다.
overwrite_S3_backup_file	SSAS 작업에는 적용되지 않습니다.
KMS_master_key_arn	SSAS 작업에는 적용되지 않습니다.
filepath	SSAS 작업에는 적용되지 않습니다.
overwrite_file	SSAS 작업에는 적용되지 않습니다.
task_metadata	SSAS 작업과 관련된 메타데이터입니다.

Amazon RDS에서 SSAS 사용

SSAS 프로젝트를 배포한 후 SSMS에서 OLAP 데이터베이스를 직접 처리할 수 있습니다.

RDS에서 SSAS를 사용하려면

1. SSMS에서 Active Directory 도메인의 사용자 이름과 암호를 사용하여 SSAS에 연결합니다.
2. 데이터베이스를 확장합니다. 새로 배포된 SSAS 데이터베이스가 나타납니다.
3. 연결 문자열을 찾고 사용자 이름과 암호를 업데이트하여 원본 SQL 데이터베이스에 대한 액세스 권한을 부여합니다. 이 작업은 SSAS 객체를 처리하는 데 필요합니다.

    a. 테이블형 모드의 경우 다음을 수행합니다.

        1. 연결(Connections) 탭을 확장합니다.
        2. 연결 객체의 컨텍스트 메뉴(마우스 오른쪽 버튼 클릭)를 연 다음 속성(Properties)을 선택합니다.
        3. 연결 문자열에서 사용자 이름과 암호를 업데이트합니다.

    b. 다차원 모드의 경우 다음을 수행합니다.

        1. 데이터 원본(Data Sources) 탭을 확장합니다.

2. 데이터 원본 객체의 컨텍스트 메뉴(마우스 오른쪽 버튼 클릭)를 연 다음 속성(Properties)을 선택합니다.
3. 연결 문자열에서 사용자 이름과 암호를 업데이트합니다.

4. 생성한 SSAS 데이터베이스에 대한 컨텍스트 메뉴(마우스 오른쪽 버튼 클릭)를 열고 데이터베이스 처리를 선택합니다.

   입력 데이터의 크기에 따라 처리 작업을 완료하는 데 몇 분 정도 걸릴 수 있습니다.

주제

- SSAS용 Windows 인증 사용자 설정
- 도메인 사용자를 데이터베이스 관리자로 추가
- SSAS 프록시 생성
- SQL Server 에이전트를 사용하여 SSAS 데이터베이스 처리 예약
- 프록시에서 SSAS 액세스 취소

SSAS용 Windows 인증 사용자 설정

기본 관리자 사용자(마스터 사용자라고도 함)는 다음 코드 예제를 사용하여 Windows 인증 로그인을 설정하고 필요한 절차 권한을 부여할 수 있습니다. 이렇게 하면 도메인 사용자에게 SSAS 고객 태스크를 실행하고, S3 파일 전송 절차를 사용하고, 자격 증명을 만들고, SQL Server 에이전트 프록시로 작업할 수 있는 권한이 부여됩니다. 자세한 내용은 Microsoft 설명서의 자격 증명(데이터베이스 엔진) 및 SQL Server 에이전트 프록시 생성을 참조하십시오.

필요에 따라 Windows 인증 사용자에게 다음 사용 권한 중 일부 또는 모두를 부여할 수 있습니다.

Example

```
-- Create a server-level domain user login, if it doesn't already exist
USE [master]
GO
CREATE LOGIN [mydomain\user_name] FROM WINDOWS
GO

-- Create domain user, if it doesn't already exist
USE [msdb]
GO
CREATE USER [mydomain\user_name] FOR LOGIN [mydomain\user_name]
GO
```

```
-- Grant necessary privileges to the domain user
USE [master]
GO
GRANT ALTER ANY CREDENTIAL TO [mydomain\user_name]
GO

USE [msdb]
GO
GRANT EXEC ON msdb.dbo.rds_msbi_task TO [mydomain\user_name] with grant option
GRANT SELECT ON msdb.dbo.rds_fn_task_status TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.rds_task_status TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.rds_cancel_task TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.rds_download_from_s3 TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.rds_upload_to_s3 TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.rds_delete_from_filesystem TO [mydomain\user_name] with grant
 option
GRANT EXEC ON msdb.dbo.rds_gather_file_details TO [mydomain\user_name] with grant
 option
GRANT EXEC ON msdb.dbo.sp_add_proxy TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.sp_update_proxy TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.sp_grant_login_to_proxy TO [mydomain\user_name] with grant
 option
GRANT EXEC ON msdb.dbo.sp_revoke_login_from_proxy TO [mydomain\user_name] with grant
 option
GRANT EXEC ON msdb.dbo.sp_delete_proxy TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.sp_enum_login_for_proxy to [mydomain\user_name] with grant
 option
GRANT EXEC ON msdb.dbo.sp_enum_proxy_for_subsystem TO [mydomain\user_name] with grant
 option
GRANT EXEC ON msdb.dbo.rds_sqlagent_proxy TO [mydomain\user_name] with grant option
ALTER ROLE [SQLAgentUserRole] ADD MEMBER [mydomain\user_name]
GO
```

도메인 사용자를 데이터베이스 관리자로 추가

다음과 같은 방법으로 도메인 사용자를 SSAS 데이터베이스 관리자로 추가할 수 있습니다.

- 데이터베이스 관리자는 SSMS를 사용하여 admin 권한이 있는 역할을 생성한 다음 해당 역할에 사용자를 추가할 수 있습니다.
- 다음 저장 프로시저를 사용할 수 있습니다.

```
exec msdb.dbo.rds_msbi_task
```

```
@task_type='SSAS_ADD_DB_ADMIN_MEMBER',
@database_name='myssasdb',
@ssas_role_name='exampleRole',
@ssas_role_member='domain_name\domain_user_name';
```

다음 파라미터는 필수 파라미터입니다.

- `@task_type` – MSBI 작업의 유형입니다(이 경우 `SSAS_ADD_DB_ADMIN_MEMBER`).
- `@database_name` – 관리자 권한을 부여할 SSAS 데이터베이스의 이름입니다.
- `@ssas_role_name` – SSAS 데이터베이스 관리자 역할 이름입니다. 역할이 아직 존재하지 않으면 역할이 생성됩니다.
- `@ssas_role_member` – 관리자 역할에 추가할 SSAS 데이터베이스 사용자입니다.

SSAS 프록시 생성

SQL Server 에이전트를 사용하여 SSAS 데이터베이스 처리를 예약하려면 SSAS 자격 증명과 SSAS 프록시를 생성합니다. Windows 인증 사용자로 다음 절차를 실행합니다.

SSAS 자격 증명 생성

- 프록시에 대한 자격 증명을 생성합니다. 이렇게 하려면 SSMS 또는 다음 SQL 문을 사용하면 됩니다.

```
USE [master]
GO
CREATE CREDENTIAL [SSAS_Credential] WITH IDENTITY = N'mydomain\user_name', SECRET =
 N'mysecret'
GO
```

> **Note**
>
> `IDENTITY`는 도메인 인증 로그인이어야 합니다. *`mysecret`*를 도메인 인증 로그인에 대한 암호로 바꿉니다.

SSAS 프록시 생성

1. 다음 SQL 문을 사용하여 프록시를 생성합니다.

```
USE [msdb]
GO
EXEC msdb.dbo.sp_add_proxy
 @proxy_name=N'SSAS_Proxy',@credential_name=N'SSAS_Credential',@description=N''
GO
```

2. 다음 SQL 문을 사용하여 프록시에 대한 액세스 권한을 다른 사용자에게 부여합니다.

```
USE [msdb]
GO
EXEC msdb.dbo.sp_grant_login_to_proxy
 @proxy_name=N'SSAS_Proxy',@login_name=N'mydomain\user_name'
GO
```

3. 다음 SQL 문을 사용하여 SSAS 하위 시스템에 프록시에 대한 액세스 권한을 부여합니다.

```
USE [msdb]
GO
EXEC msdb.dbo.rds_sqlagent_proxy
 @task_type='GRANT_SUBSYSTEM_ACCESS',@proxy_name='SSAS_Proxy',@proxy_subsystem='SSAS'
GO
```

프록시 및 프록시의 부여를 보려면

1. 다음 SQL 문을 사용하여 프록시의 피부여자를 확인합니다.

```
USE [msdb]
GO
EXEC sp_help_proxy
GO
```

2. 다음 SQL 문을 사용하여 하위 시스템 부여를 확인합니다.

```
USE [msdb]
GO
EXEC msdb.dbo.sp_enum_proxy_for_subsystem
GO
```

SQL Server 에이전트를 사용하여 SSAS 데이터베이스 처리 예약

자격 증명 및 프록시를 생성하고 SSAS에 프록시에 대한 액세스 권한을 부여한 후 SSAS 데이터베이스 처리를 예약하는 SQL Server 에이전트 작업을 생성할 수 있습니다.

SSAS 데이터베이스 처리 예약

- SSMS 또는 T-SQL을 사용하여 SQL Server 에이전트 작업을 생성합니다. 다음 예제에서는 T-SQL을 사용합니다. SSMS 또는 T-SQL을 통해 작업 일정을 추가로 구성할 수 있습니다.
  - `@command` 파라미터는 SQL Server 에이전트 작업에서 실행할 XMLA(XML for Analysis) 명령을 간략하게 설명합니다. 이 예에서는 SSAS 다차원 데이터베이스 처리를 구성합니다.
  - `@server` 파라미터는 SQL Server 에이전트 작업의 대상 SSAS 서버 이름을 간략하게 설명합니다.

    SQL Server 에이전트 작업이 있는 동일한 RDS DB 인스턴스 내에서 SSAS 서비스를 호출하려면 `localhost:2383`을 호출합니다.

    RDS DB 인스턴스 외부에서 SSAS 서비스를 호출하려면 RDS 엔드포인트를 사용합니다. RDS DB 인스턴스가 동일한 도메인에 가입되어 있는 경우 Kerberos Active Directory(AD) 엔드포인트(*your-DB-instance-name*.*your-AD-domain-name*)를 사용할 수도 있습니다. 외부 DB 인스턴스의 경우 보안 연결을 위해 RDS DB 인스턴스와 연결된 VPC 보안 그룹을 올바르게 구성해야 합니다.

  다양한 XMLA 작업을 지원하도록 쿼리를 추가로 편집할 수 있습니다. T-SQL 쿼리를 직접 수정하거나 SQL Server 에이전트 작업 생성 후 SSMS UI를 사용하여 편집합니다.

```
USE [msdb]
GO
DECLARE @jobId BINARY(16)
EXEC msdb.dbo.sp_add_job @job_name=N'SSAS_Job',
    @enabled=1,
    @notify_level_eventlog=0,
    @notify_level_email=0,
    @notify_level_netsend=0,
    @notify_level_page=0,
    @delete_level=0,
    @category_name=N'[Uncategorized (Local)]',
    @job_id = @jobId OUTPUT
GO
```

```
EXEC msdb.dbo.sp_add_jobserver
    @job_name=N'SSAS_Job',
    @server_name = N'(local)'
GO
EXEC msdb.dbo.sp_add_jobstep @job_name=N'SSAS_Job',
 @step_name=N'Process_SSAS_Object',
    @step_id=1,
    @cmdexec_success_code=0,
    @on_success_action=1,
    @on_success_step_id=0,
    @on_fail_action=2,
    @on_fail_step_id=0,
    @retry_attempts=0,
    @retry_interval=0,
    @os_run_priority=0, @subsystem=N'ANALYSISCOMMAND',
    @command=N'<Batch xmlns="http://schemas.microsoft.com/analysisservices/2003/
engine">
        <Parallel>
            <Process xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xmlns:ddl2="http://schemas.microsoft.com/analysisservices/2003/
engine/2" xmlns:ddl2_2="http://schemas.microsoft.com/analysisservices/2003/
engine/2/2"
                xmlns:ddl100_100="http://schemas.microsoft.com/
analysisservices/2008/engine/100/100" xmlns:ddl200="http://schemas.microsoft.com/
analysisservices/2010/engine/200"
                xmlns:ddl200_200="http://schemas.microsoft.com/
analysisservices/2010/engine/200/200" xmlns:ddl300="http://schemas.microsoft.com/
analysisservices/2011/engine/300"
                xmlns:ddl300_300="http://schemas.microsoft.com/
analysisservices/2011/engine/300/300" xmlns:ddl400="http://schemas.microsoft.com/
analysisservices/2012/engine/400"
                xmlns:ddl400_400="http://schemas.microsoft.com/
analysisservices/2012/engine/400/400" xmlns:ddl500="http://schemas.microsoft.com/
analysisservices/2013/engine/500"
                xmlns:ddl500_500="http://schemas.microsoft.com/
analysisservices/2013/engine/500/500">
                <Object>
                    <DatabaseID>Your_SSAS_Database_ID</DatabaseID>
                </Object>
                <Type>ProcessFull</Type>
                <WriteBackTableCreation>UseExisting</WriteBackTableCreation>
            </Process>
        </Parallel>
```

```
    </Batch>',
    @server=N'localhost:2383',
    @database_name=N'master',
    @flags=0,
    @proxy_name=N'SSAS_Proxy'
GO
```

프록시에서 SSAS 액세스 취소

다음 저장 프로시저를 사용하여 SSAS 하위 시스템에 대한 액세스를 취소하고 SSAS 프록시를 삭제할 수 있습니다.

액세스를 취소하고 프록시를 삭제하려면

1. 하위 시스템 액세스를 취소합니다.

```
USE [msdb]
GO
EXEC msdb.dbo.rds_sqlagent_proxy
 @task_type='REVOKE_SUBSYSTEM_ACCESS',@proxy_name='SSAS_Proxy',@proxy_subsystem='SSAS'
GO
```

2. 프록시에 대한 부여를 취소합니다.

```
USE [msdb]
GO
EXEC msdb.dbo.sp_revoke_login_from_proxy
 @proxy_name=N'SSAS_Proxy',@name=N'mydomain\user_name'
GO
```

3. 프록시를 삭제합니다.

```
USE [msdb]
GO
EXEC dbo.sp_delete_proxy @proxy_name = N'SSAS_Proxy'
GO
```

SSAS 데이터베이스 백업

SSAS 데이터베이스 백업 파일은 DB 인스턴스의 D:\S3 폴더에만 생성할 수 있습니다. 백업 파일을 S3 버킷으로 이동하려면 Amazon S3를 사용합니다.

다음과 같이 SSAS 데이터베이스를 백업할 수 있습니다.

- 특정 데이터베이스에 대한 admin 역할이 있는 도메인 사용자는 SSMS를 사용하여 데이터베이스를 D:\S3 폴더에 백업할 수 있습니다.

  자세한 내용은 도메인 사용자를 데이터베이스 관리자로 추가 섹션을 참조하세요.
- 다음 저장 프로시저를 사용할 수 있습니다. 이 저장 프로시저는 암호화를 지원하지 않습니다.

```
exec msdb.dbo.rds_msbi_task
@task_type='SSAS_BACKUP_DB',
@database_name='myssasdb',
@file_path='D:\S3\ssas_db_backup.abf',
[@ssas_apply_compression=1],
[@ssas_overwrite_file=1];
```

  다음 파라미터는 필수 파라미터입니다.

  - @task_type – MSBI 작업의 유형입니다(이 경우 SSAS_BACKUP_DB).
  - @database_name – 백업 대상 SSAS 데이터베이스의 이름입니다.
  - @file_path – SSAS 백업 파일의 경로입니다. .abf 확장이 필요합니다.

  다음 파라미터는 선택적입니다.

  - @ssas_apply_compression – SSAS 백업 압축을 적용할지 여부입니다. 유효한 값은 1(예) 및 0(아니오)입니다.
  - @ssas_overwrite_file – SSAS 백업 파일을 덮어쓸지 여부입니다. 유효한 값은 1(예) 및 0(아니오)입니다.

SSAS 데이터베이스 복원

다음 저장 프로시저를 사용하여 백업에서 SSAS 데이터베이스를 복원합니다.

이름이 같은 기존 SSAS 데이터베이스가 있으면 데이터베이스를 복원할 수 없습니다. 복원을 위한 저장 프로시저는 암호화된 백업 파일을 지원하지 않습니다.

```
exec msdb.dbo.rds_msbi_task
```

```
@task_type='SSAS_RESTORE_DB',
@database_name='mynewssasdb',
@file_path='D:\S3\ssas_db_backup.abf';
```

다음 파라미터는 필수 파라미터입니다.

- `@task_type` – MSBI 작업의 유형입니다(이 경우 `SSAS_RESTORE_DB`).
- `@database_name` – 복원 대상 새 SSAS 데이터베이스의 이름입니다.
- `@file_path` – SSAS 백업 파일의 경로입니다.

DB 인스턴스를 지정된 시간으로 복원

PITR(특정 시점으로 복구)은 SSAS 데이터베이스에 적용되지 않습니다. PITR을 수행하는 경우 요청된 시간 이전의 마지막 스냅샷에 있는 SSAS 데이터만 복원된 인스턴스에서 사용할 수 있습니다.

복원된 DB 인스턴스에서 SSAS 데이터베이스를 최신 상태로 유지하려면

1. SSAS 데이터베이스를 원본 인스턴스의 `D:\S3` 폴더에 백업합니다.
2. 백업 파일을 S3 버킷으로 전송합니다.
3. S3 버킷에서 복원된 인스턴스의 `D:\S3` 폴더로 백업 파일을 전송합니다.
4. 저장 프로시저를 실행하여 SSAS 데이터베이스를 복원된 인스턴스로 복원합니다.

   SSAS 프로젝트를 다시 처리하여 데이터베이스를 복원할 수도 있습니다.

SSAS 모드 변경

SSAS가 실행되는 모드를 테이블 형식 또는 다차원으로 변경할 수 있습니다. 모드를 변경하려면 AWS Management Console 또는 AWS CLI를 사용하여 SSAS 옵션의 옵션 설정을 수정합니다.

> **Important**
>
> 한 번에 하나의 SSAS 모드만 사용할 수 있습니다. 모드를 변경하기 전에 모든 SSAS 데이터베이스를 삭제해야 합니다. 그렇지 않으면 오류가 발생합니다.

콘솔

다음 Amazon RDS 콘솔 절차는 SSAS 모드를 테이블 형식으로 변경하고 MAX_MEMORY 파라미터를 70%로 설정합니다.

SSAS 옵션 수정

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 수정하려는 SSAS 옵션이 있는 옵션 그룹을 선택합니다(이전 예의 ssas-se-2017).
4. 옵션 수정(Modify option)을 선택합니다.
5. 옵션 설정을 변경합니다.
   a. 최대 메모리(Max memory)에서 **70**을 입력합니다.
   b. 모드(Mode)에서 테이블 형식(Tabular)을 선택합니다.
6. 옵션 수정(Modify option)을 선택합니다.

AWS CLI

다음 AWS CLI 예에서는 SSAS 모드를 테이블 형식으로 변경하고 MAX_MEMORY 파라미터를 70%로 설정합니다.

CLI 명령이 작동하려면 수정하지 않는 경우에도 필수 파라미터를 모두 포함해야 합니다.

SSAS 옵션 수정

- 다음 명령 중 하나를 사용합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds add-option-to-option-group \
    --option-group-name ssas-se-2017 \
    --options
 "OptionName=SSAS,VpcSecurityGroupMemberships=sg-12345e67,OptionSettings=[{Name=MAX_MEMORY,
{Name=MODE,Value=Tabular}]" \
    --apply-immediately
```

Windows의 경우:

```
aws rds add-option-to-option-group ^
    --option-group-name ssas-se-2017 ^
    --options
 OptionName=SSAS,VpcSecurityGroupMemberships=sg-12345e67,OptionSettings=[{Name=MAX_MEMORY,V
{Name=MODE,Value=Tabular}] ^
    --apply-immediately
```

SSAS 해제

SSAS를 해제하려면 해당 옵션 그룹에서 SSAS 옵션을 제거합니다.

> **Important**
>
> SSAS 옵션을 제거하기 전에 SSAS 데이터베이스를 삭제하십시오.
> SSAS 데이터베이스를 삭제하고 SSAS 옵션을 제거하기 전에 SSAS 데이터베이스를 백업하는 것이 좋습니다.

콘솔

옵션 그룹에서 SSAS 옵션을 제거하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 제거하려는 SSAS 옵션이 있는 옵션 그룹을 선택합니다(이전 예의 `ssas-se-2017`).
4. 옵션 삭제를 선택합니다.
5. 옵션 삭제에서 SSAS 또는 삭제할 옵션을 선택합니다.
6. 즉시 적용에서 옵션을 즉시 삭제하려면 예를 선택하고 다음 유지 관리 기간에 삭제하려면 아니오를 선택합니다.
7. Delete(삭제)를 선택합니다.

AWS CLI

옵션 그룹에서 SSAS 옵션을 제거하려면

- 다음 명령 중 하나를 사용합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds remove-option-from-option-group \
    --option-group-name ssas-se-2017 \
    --options SSAS \
    --apply-immediately
```

  Windows의 경우:

```
aws rds remove-option-from-option-group ^
    --option-group-name ssas-se-2017 ^
    --options SSAS ^
    --apply-immediately
```

SSAS 문제 해결

SSAS 사용 시 다음과 같은 문제가 발생할 수 있습니다.

문제	유형	문제 해결 제안
SSAS 옵션을 구성할 수 없습니다. 요청한 SSAS 모드는 *new_mode*지만 현재 DB 인스턴스에는 *number*개의 *current_mode* 데이터베이스가 있습니다. *new_mode* 모드로 전환하기 전에 기존 데이터베이스를 삭제하세요. 데이터베이스 삭제를 위해 *current_mode* 모드에 다시 액세스하려면 현재 DB 옵션 그룹을 업데이트하거나 SSAS 옵션에 대한 MODE 옵	RDS 이벤트	현재 모드를 사용하는 SSAS 데이터베이스가 여전히 있는 경우 SSAS 모드를 변경할 수 없습니다. SSAS 데이터베이스를 삭제한 다음 다시 시도하세요.

문제	유형	문제 해결 제안
션 설정 값으로 %s를 사용하여 새 옵션 그룹을 연결합니다.		
*number*개의 기존 *mode* 데이터베이스가 있어 SSAS 옵션을 제거할 수 없습니다. 모든 SSAS 데이터베이스가 삭제될 때까지 SSAS 옵션을 제거할 수 없습니다. SSAS 옵션을 다시 추가하고 모든 SSAS 데이터베이스를 삭제한 다음 다시 시도하세요.	RDS 이벤트	아직 SSAS 데이터베이스가 있는 경우 SSAS를 해제할 수 없습니다. SSAS 데이터베이스를 삭제한 다음 다시 시도하세요.
SSSAS 옵션이 사용 설정되지 않았거나 사용 설정되는 중입니다. 나중에 다시 시도해 주세요.	RDS 저장 프로시저	옵션이 해제되어 있거나 설정되고 있을 때는 SSAS 저장 프로시저를 실행할 수 없습니다.

문제	유형	문제 해결 제안
SSAS 옵션이 잘못 구성되었습니다. 옵션 그룹 구성원 자격 상태가 "동기화 중"인지 확인하고 RDS 이벤트 로그에서 관련 SSAS 구성 오류 메시지를 검토하세요. 이러한 조사 후 다시 시도하세요. 오류가 계속 발생하면 AWS Support에 문의하세요.	RDS 저장 프로시저	옵션 그룹 멤버십이 `in-sync` 상태가 아니면 SSAS 저장 프로시저를 실행할 수 없습니다. 이렇게 하면 SSAS 옵션이 잘못된 구성 상태가 됩니다. SSAS 옵션 수정으로 인해 옵션 그룹 멤버십 상태가 `failed`로 변경되는 경우 두 가지 이유가 있을 수 있습니다. 1. SSAS 데이터베이스를 삭제하지 않고 SSAS 옵션이 제거되었습니다. 2. SSAS 모드는 기존 SSAS 데이터베이스를 삭제하지 않고 테이블 형식에서 다차원으로 또는 다차원에서 테이블 형식으로 업데이트되었습니다. RDS는 한 번에 하나의 SSAS 모드만 허용하고 SSAS 데이터베이스가 있는 SSAS 옵션 제거를 지원하지 않으므로 SSAS 옵션을 다시 구성합니다. RDS 이벤트 로그에서 SSAS 인스턴스의 구성 오류를 확인하고 그에 따라 문제를 해결하세요.
배포에 실패했습니다. 변경 사항은 *deployment_file_mode* 모드에서 실행 중인 서버에만 배포할 수 있습니다. 현재 서버 모드는 *current_mode* 입니다.	RDS 저장 프로시저	다차원 서버에 테이블 형식 데이터베이스를 배포하거나 테이블 형식 서버에 다차원 데이터베이스를 배포할 수 없습니다. 올바른 모드의 파일을 사용하고 있는지 확인하고 `MODE` 옵션 설정이 적절한 값으로 설정되어 있는지 확인합니다.

문제	유형	문제 해결 제안
복원에 실패했습니다. 백업 파일은 *restore_file_mode* 모드로 실행 중인 서버에서만 복원할 수 있습니다. 현재 서버 모드는 *current_mode* 입니다.	RDS 저장 프로시저	테이블 형식 데이터베이스를 다차원 서버로 복원하거나 다차원 데이터베이스를 테이블 형식 서버로 복원할 수 없습니다. 올바른 모드의 파일을 사용하고 있는지 확인하고 MODE 옵션 설정이 적절한 값으로 설정되어 있는지 확인합니다.
복원에 실패했습니다. 백업 파일과 RDS DB 인스턴스 버전이 호환되지 않습니다.	RDS 저장 프로시저	SQL Server 인스턴스 버전과 호환되지 않는 버전으로 SSAS 데이터베이스를 복원할 수 없습니다. 자세한 내용은 Microsoft 설명서의 테이블 형식 모델에 대한 호환성 수준과다차원 데이터베이스의 호환성 수준을 참조하세요.
복원에 실패했습니다. 복원 작업에 지정된 백업 파일이 손상되었거나 SSAS 백업 파일이 아닙니다. @rds_file_path 형식이 올바른지 확인하세요.	RDS 저장 프로시저	손상된 파일을 사용하여 SSAS 데이터베이스를 복원할 수 없습니다. 파일이 손상되지 않았는지 확인합니다. 이 오류는 @rds_file_path 의 형식이 올바르지 않은 경우에도 발생할 수 있습니다 (예: D:\S3\\incorrect_format.abf 에서와 같이 이중 백슬래시가 있는 경우).
복원에 실패했습니다. 복원된 데이터베이스 이름은 예약어 또는 유효하지 않은 문자(. , ; ' ` : / \\ * | ? \" & % $! + = ( ) [ ] { } < >)를 포함하거나 100자를 초과할 수 없습니다.	RDS 저장 프로시저	복원된 데이터베이스 이름은 예약어 또는 유효하지 않은 문자를 포함하거나 100자를 초과할 수 없습니다. SSAS 객체 이름 지정 규칙은 Microsoft 설명서의 객체 명명 규칙을 참조하세요.

문제	유형	문제 해결 제안
잘못된 역할 이름이 제공되었습니다. 역할 이름은 예약된 문자열을 포함할 수 없습니다.	RDS 저장 프로시저	역할 이름은 예약된 문자열을 포함할 수 없습니다. SSAS 객체 이름 지정 규칙은 Microsoft 설명서의 객체 명명 규칙을 참조하세요.
잘못된 역할 이름이 제공되었습니다. 역할 이름은 예약된 문자(. , ; ' ` : / \\ * | ? \" & % $! + = ( ) [ ] { } < >)를 포함할 수 없습니다.	RDS 저장 프로시저	역할 이름은 예약된 문자를 포함할 수 없습니다. SSAS 객체 이름 지정 규칙은 Microsoft 설명서의 객체 명명 규칙을 참조하세요.

Amazon RDS for SQL Server에서 SQL Server Integration Services 지원

Microsoft SQL Server Integration Services(SSIS)는 광범위한 데이터 마이그레이션 작업을 수행하는 데 사용할 수 있는 구성 요소입니다. SSIS는 데이터 통합 및 워크플로우 애플리케이션용 플랫폼입니다. 이는 데이터 추출, 변환 및 로드(ETL)에 사용되는 데이터 웨어하우징 도구를 특징으로 합니다. 이 도구를 사용하여 SQL Server 데이터베이스의 유지 관리 및 다차원 큐브 데이터 업데이트를 자동화할 수도 있습니다.

SSIS 프로젝트는 XML 기반 .dtsx 파일로 저장된 패키지로 구성됩니다. 패키지에는 제어 흐름 및 데이터 흐름이 포함될 수 있습니다. 데이터 흐름을 사용하여 ETL 작업을 나타냅니다. 배포 후 패키지는 SSISDB 데이터베이스의 SQL Server에 저장됩니다. SSISDB는 전체 복구 모드의 OLTP(온라인 트랜잭션 처리) 데이터베이스입니다.

Amazon RDS for SQL Server는 RDS DB 인스턴스에서 SSRS를 직접 실행할 수 있도록 지원합니다. 기존 DB 인스턴스 또는 새 DB 인스턴스에서 SSRS를 활성화할 수 있습니다. SSIS는 데이터베이스 엔진과 동일한 DB 인스턴스에 설치됩니다.

RDS는 다음 버전에서 SQL Server Standard 및 Enterprise Edition용 SSIS를 지원합니다.

- SQL Server 2022, 모든 버전
- SQL Server 2019, 버전 15.00.4043.16.v1 이상
- SQL Server 2017, 버전 14.00.3223.3.v1 이상
- SQL Server 2016, 버전 13.00.5426.0.v1 이상

목차

- 제한 및 권장 사항
- SSIS 활성화
  - SSIS용 옵션 그룹 생성
  - 옵션 그룹에 SSIS 옵션 추가
  - SSIS용 파라미터 그룹 생성
  - SSIS용 파라미터 수정
  - 옵션 그룹 및 파라미터 그룹을 DB 인스턴스와 연결
  - S3 통합 활성화
- SSISDB에 대한 관리 권한
  - SSIS용 Windows 인증 사용자 설정

- SSIS 프로젝트 배포
- 배포 작업의 상태 모니터링
- SSIS 사용
  - SSIS 프로젝트에 대한 데이터베이스 연결 관리자 설정
  - SSIS 프록시 생성
  - SQL Server 에이전트를 사용하여 SSIS 패키지 예약
  - 프록시에서 SSIS 액세스 취소
- SSIS 데이터베이스 비활성화 및 삭제
  - SSIS 비활성화
  - SSISDB 데이터베이스 삭제

제한 및 권장 사항

RDS for SQL Server에서 SSRS를 실행하는 경우 다음과 같은 제한 및 권장 사항이 적용됩니다.

- DB 인스턴스에는 `clr enabled` 파라미터가 1로 설정된 연결된 파라미터 그룹이 있어야 합니다. 자세한 내용은 SSIS용 파라미터 수정 섹션을 참조하세요.

  > **Note**
  >
  > SQL Server 2017 또는 2019에서 `clr enabled` 파라미터를 활성화하면 DB 인스턴스에서 CLR(공통 언어 런타임)을 사용할 수 없습니다. 자세한 내용은 지원되지 않는 기능과 지원이 제한된 기능 섹션을 참조하세요.

- 다음 제어 흐름 작업이 지원됩니다.
  - 분석 서비스 DDL 실행 작업
  - 분석 서비스 처리 작업
  - 대량 삽입 작업
  - 데이터베이스 무결성 검사 작업
  - 데이터 흐름 작업
  - 데이터 마이닝 쿼리 작업
  - 데이터 프로파일링 작업
  - 패키지 실행 작업
  - SQL Server 에이전트 작업 실행 작업

  - SQL 실행 작업
  - T-SQL 문 실행 작업
  - 운영자에게 알림 작업
  - 인덱스 재구축 작업
  - 인덱스 재구성 작업
  - 데이터베이스 축소 작업
  - 데이터베이스 전송 작업
  - 작업 전송 작업
  - 로그인 전송 작업
  - SQL Server 개체 전송 작업
  - 통계 업데이트 작업
- 프로젝트 배포만 지원됩니다.
- SQL Server 에이전트를 사용하여 SSIS 패키지를 실행할 수 있습니다.
- SSIS 로그 레코드는 사용자가 만든 데이터베이스에만 삽입할 수 있습니다.
- 파일 작업에는 `D:\S3` 폴더만 사용합니다. 다른 디렉터리에 있는 파일은 삭제됩니다. 다른 파일 위치 세부 정보를 숙지하십시오.
  - SSIS 프로젝트 입력 및 출력 파일을 `D:\S3` 폴더에 배치합니다.
  - 데이터 흐름 작업의 경우 `BLOBTempStoragePath` 및 `BufferTempStoragePath`의 위치를 `D:\S3` 폴더 내의 파일로 변경합니다. 파일 경로는 `D:\S3\`로 시작되어야 합니다.
  - 파일 연결에 사용되는 모든 파라미터, 변수 및 표현식이 `D:\S3` 폴더를 가리키는지 확인합니다.
  - 다중 AZ 인스턴스에서 장애 조치 후 `D:\S3` 폴더에 SSIS에서 생성한 파일이 삭제됩니다. 자세한 내용은 S3 통합에 대한 다중 AZ 제한 사항 섹션을 참조하세요.
  - `D:\S3` 폴더에 SSIS에서 생성한 파일을 Amazon S3 버킷에 업로드하여 내구성을 높입니다.
- 열 가져오기 및 열 내보내기 변환과 데이터 흐름 작업의 스크립트 구성 요소는 지원되지 않습니다.
- SSIS 패키지 실행 시 덤프를 활성화할 수 없으며 SSIS 패키지에 데이터 탭을 추가할 수 없습니다.
- SSIS 확장 기능은 지원되지 않습니다.
- 프로젝트를 직접 배포할 수 없습니다. 이를 수행할 RDS 저장 프로시저를 제공합니다. 자세한 내용은 SSIS 프로젝트 배포 섹션을 참조하세요.
- RDS에 배포하기 위한 `DoNotSavePasswords` 보호 모드를 사용하여 SSIS 프로젝트(.ispac) 파일을 빌드합니다.
- 읽기 전용 복제본이 있는 Always On 인스턴스에서는 SSIS가 지원되지 않습니다.

- SSIS 옵션과 연결된 SSISDB 데이터베이스는 백업할 수 없습니다.
- SSIS의 다른 인스턴스에서 SSISDB 데이터베이스를 가져오고 복원하는 것은 지원되지 않습니다.
- 다른 SQL Server DB 인스턴스나 Oracle 데이터 소스에 연결할 수 있습니다. MySQL이나 PostgreSQL 같은 다른 데이터베이스 엔진에 연결하는 것은 RDS for SQL Server의 SSIS에서 지원되지 않습니다. Oracle 데이터 소스 연결에 대한 자세한 내용은 Oracle OLEDB 포함 연결된 서버 단원을 참조하세요.
- SSIS는 온프레미스 도메인에 대한 발신 신뢰가 있는 도메인 조인 인스턴스를 지원하지 않습니다. 발신 신뢰를 사용하는 경우 로컬 AWS 도메인의 계정에서 SSIS 작업을 실행합니다.

SSIS 활성화

SSIS 옵션을 DB 인스턴스에 추가하여 SSIS를 활성화합니다. 다음 프로세스를 사용합니다.

1. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 선택합니다.
2. [SSIS] 옵션을 옵션 그룹에 추가합니다.
3. 새 파라미터 그룹을 생성하거나 기존 파라미터 그룹을 선택합니다.
4. 파라미터 그룹을 수정하여 `clr enabled` 파라미터를 1로 설정합니다.
5. 옵션 그룹 및 파라미터 그룹을 DB 인스턴스와 연결합니다.
6. Amazon S3 통합을 활성화합니다.

> **Note**
>
> 이름이 SSISDB이거나 예약된 SSIS 로그인이 있는 데이터베이스가 DB 인스턴스에 이미 있는 경우 인스턴스에서 SSIS를 활성화할 수 없습니다.

SSIS용 옵션 그룹 생성

SSIS를 사용하려면 사용할 DB 인스턴스의 SQL Server 에디션 및 버전에 해당하는 옵션 그룹을 생성하거나 수정합니다. 이렇게 하려면 AWS Management Console 또는 AWS CLI를 사용합니다.

콘솔

다음 절차에서는 SQL Server Standard Edition 2016에 대한 옵션 그룹을 생성합니다.

옵션 그룹을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 그룹 생성을 선택합니다.
4. 보안 그룹 생성 창에서 다음과 같이 합니다.
   a. [이름(Name)]에 AWS 계정 내에서 쉽게 식별할 수 있는 옵션 그룹 이름을 입력합니다(예: **ssis-se-2016**). 이름은 글자, 숫자 및 하이픈만 사용 가능합니다.
   b. 설명에 옵션 그룹에 대한 간단한 설명을 입력합니다(예: **SSIS option group for SQL Server SE 2016**). 이 설명은 표시 용도로만 사용됩니다.
   c. 엔진에 대해 sqlserver-se를 선택합니다.
   d. 메이저 엔진 버전에 13.00을 선택합니다.
5. 생성(Create)을 선택합니다.

CLI

다음 절차에서는 SQL Server Standard Edition 2016에 대한 옵션 그룹을 생성합니다.

옵션 그룹을 생성하려면

- 다음 명령 중 하나를 실행합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds create-option-group \
    --option-group-name ssis-se-2016 \
    --engine-name sqlserver-se \
    --major-engine-version 13.00 \
    --option-group-description "SSIS option group for SQL Server SE 2016"
```

  Windows의 경우:

```
aws rds create-option-group ^
    --option-group-name ssis-se-2016 ^
```

```
    --engine-name sqlserver-se ^
    --major-engine-version 13.00 ^
    --option-group-description "SSIS option group for SQL Server SE 2016"
```

옵션 그룹에 SSIS 옵션 추가

그런 다음 AWS Management Console 또는 AWS CLI를 사용하여 SSIS 옵션을 옵션 그룹에 추가합니다.

콘솔

SSIS 옵션을 추가하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 이 예제에서는 방금 생성한 옵션 그룹인 ssis-se-2016을 선택합니다.
4. 옵션 추가를 선택합니다.
5. 옵션 세부 정보에서 옵션 이름으로 SSIS를 선택합니다.
6. 예약에서 옵션을 즉시 추가할지 또는 다음 유지 관리 기간에 추가할지를 선택합니다.
7. 옵션 추가를 선택합니다.

CLI

SSIS 옵션을 추가하려면

- [SSIS] 옵션을 옵션 그룹에 추가합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

  ```
  aws rds add-option-to-option-group \
      --option-group-name ssis-se-2016 \
      --options OptionName=SSIS \
      --apply-immediately
  ```

  Windows의 경우:

```
aws rds add-option-to-option-group ^
    --option-group-name ssis-se-2016 ^
    --options OptionName=SSIS ^
    --apply-immediately
```

SSIS용 파라미터 그룹 생성

SSIS를 사용할 DB 인스턴스의 SQL Server 에디션 및 버전에 해당하는 `clr enabled` 파라미터의 파라미터 그룹을 생성하거나 수정합니다.

콘솔

다음 절차에서는 SQL Server Standard Edition 2016에 대한 파라미터 그룹을 생성합니다.

파라미터 그룹을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.
3. [Create parameter group]을 선택합니다.
4. 서브넷 그룹 생성 창에서 다음을 수행합니다.
   a. 파라미터 그룹 패밀리에서 sqlserver-se-13.0을 선택합니다.
   b. 그룹 이름에 파라미터 그룹의 식별자(예: **ssis-sqlserver-se-13**)를 입력합니다.
   c. 설명에 **clr enabled parameter group**를 입력합니다.
5. 생성(Create)을 선택합니다.

CLI

다음 절차에서는 SQL Server Standard Edition 2016에 대한 파라미터 그룹을 생성합니다.

파라미터 그룹을 생성하려면

- 다음 명령 중 하나를 실행합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-parameter-group \
    --db-parameter-group-name ssis-sqlserver-se-13 \
    --db-parameter-group-family "sqlserver-se-13.0" \
    --description "clr enabled parameter group"
```

Windows의 경우:

```
aws rds create-db-parameter-group ^
    --db-parameter-group-name ssis-sqlserver-se-13 ^
    --db-parameter-group-family "sqlserver-se-13.0" ^
    --description "clr enabled parameter group"
```

SSIS용 파라미터 수정

SQL Server 에디션 및 DB 인스턴스의 버전에 해당하는 파라미터 그룹의 `clr enabled` 파라미터를 수정합니다. SSIS의 경우 `clr enabled` 파라미터를 1로 설정합니다.

콘솔

다음 절차에서는 SQL Server Standard Edition 2016에 대해 생성한 파라미터 그룹을 수정합니다.

파라미터 그룹을 수정하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.
3. 파라미터 그룹(예: ssis-sqlserver-se-13)을 선택합니다.
4. 파라미터에서 파라미터 목록을 **clr**로 필터링합니다.
5. clr 활성을 선택합니다.
6. 파라미터 편집을 선택합니다.
7. 값에서 1을 선택합니다.
8. 변경 사항 저장을 선택합니다.

CLI

다음 절차에서는 SQL Server Standard Edition 2016에 대해 생성한 파라미터 그룹을 수정합니다.

파라미터 그룹을 수정하려면

- 다음 명령 중 하나를 실행합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name ssis-sqlserver-se-13 \
    --parameters "ParameterName='clr
 enabled',ParameterValue=1,ApplyMethod=immediate"
```

  Windows의 경우:

```
aws rds modify-db-parameter-group ^
    --db-parameter-group-name ssis-sqlserver-se-13 ^
    --parameters "ParameterName='clr
 enabled',ParameterValue=1,ApplyMethod=immediate"
```

옵션 그룹 및 파라미터 그룹을 DB 인스턴스와 연결

SSIS 옵션 그룹 및 파라미터 그룹을 DB 인스턴스와 연결하려면 AWS Management Console 또는 AWS CLI를 사용합니다.

> **Note**
>
> 기존 인스턴스를 사용하는 경우 이미 Active Directory 도메인과 AWS Identity and Access Management(IAM) 역할이 연결되어 있어야 합니다. 새 인스턴스를 생성하는 경우 기존 Active Directory 도메인 및 IAM 역할을 지정합니다. 자세한 내용은 RDS for SQL Server를 사용하여 Active Directory 작업 섹션을 참조하세요.

콘솔

SSIS 활성화를 완료하려면 SSIS 옵션 그룹 및 파라미터 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스와 연결합니다.

- 새 DB 인스턴스의 경우 인스턴스를 시작할 때 이러한 그룹을 연결합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.

- 기존 DB 인스턴스의 경우 인스턴스를 수정하여 그룹을 연결합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

CLI

SSIS 옵션 그룹 및 파라미터 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스와 연결할 수 있습니다.

SSIS 옵션 그룹 및 파라미터 그룹을 사용하여 인스턴스를 생성하려면

- 옵션 그룹을 생성할 때 사용한 것과 동일한 DB 엔진 유형과 메이저 버전을 지정합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance \
    --db-instance-identifier myssisinstance \
    --db-instance-class db.m5.2xlarge \
    --engine sqlserver-se \
    --engine-version 13.00.5426.0.v1 \
    --allocated-storage 100 \
    --manage-master-user-password \
    --master-username admin \
    --storage-type gp2 \
    --license-model li \
    --domain-iam-role-name my-directory-iam-role \
    --domain my-domain-id \
    --option-group-name ssis-se-2016 \
    --db-parameter-group-name ssis-sqlserver-se-13
```

  Windows의 경우:

```
aws rds create-db-instance ^
    --db-instance-identifier myssisinstance ^
    --db-instance-class db.m5.2xlarge ^
    --engine sqlserver-se ^
    --engine-version 13.00.5426.0.v1 ^
    --allocated-storage 100 ^
    --manage-master-user-password ^
    --master-username admin ^
    --storage-type gp2 ^
    --license-model li ^
```

```
    --domain-iam-role-name my-directory-iam-role ^
    --domain my-domain-id ^
    --option-group-name ssis-se-2016 ^
    --db-parameter-group-name ssis-sqlserver-se-13
```

인스턴스를 수정하고 SSIS 옵션 그룹 및 파라미터 그룹을 연결하려면

- 다음 명령 중 하나를 실행합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier myssisinstance \
    --option-group-name ssis-se-2016 \
    --db-parameter-group-name ssis-sqlserver-se-13 \
    --apply-immediately
```

  Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier myssisinstance ^
    --option-group-name ssis-se-2016 ^
    --db-parameter-group-name ssis-sqlserver-se-13 ^
    --apply-immediately
```

S3 통합 활성화

배포를 위해 SSIS 프로젝트(.ispac) 파일을 호스트에 다운로드하려면 S3 파일 통합을 사용합니다. 자세한 내용은 Amazon RDS for SQL Server DB 인스턴스와 Amazon S3 통합 섹션을 참조하세요.

SSISDB에 대한 관리 권한

SSIS 옵션을 사용하여 인스턴스를 생성하거나 수정하면 마스터 사용자에게 ssis_admin 및 ssis_logreader 역할이 부여된 SSISDB 데이터베이스가 생성됩니다. 마스터 사용자는 SSISDB에 대해 다음 권한을 가집니다.

- ssis_admin 역할 변경

- ssis_logreader 역할 변경
- 모든 사용자 변경

마스터 사용자는 SQL 인증 사용자이므로 마스터 사용자를 사용하여 SSIS 패키지를 실행할 수 없습니다. 마스터 사용자는 이러한 권한을 사용하여 새 SSISDB 사용자를 생성하고 이를 ssis_admin 및 ssis_logreader 역할에 추가할 수 있습니다. 이렇게 하면 SSIS 사용을 위해 도메인 사용자에게 액세스 권한을 부여하는 데 유용합니다.

SSIS용 Windows 인증 사용자 설정

마스터 사용자가 다음 코드 예제를 사용하여 SSISDB에서 Windows 인증 로그인을 설정하고 필요한 절차에 대한 권한을 부여할 수 있습니다. 이렇게 하면 도메인 사용자에게 SSIS 패키지를 배포 및 실행하고, S3 파일 전송 프로시저를 사용하고, 자격 증명을 만들고, SQL Server 에이전트 프록시를 사용할 수 있는 권한이 부여됩니다. 자세한 내용은 Microsoft 설명서의 자격 증명(데이터베이스 엔진) 및 SQL Server 에이전트 프록시 생성을 참조하십시오.

> **Note**
>
> 필요에 따라 Windows 인증 사용자에게 다음 사용 권한 중 일부 또는 모두를 부여할 수 있습니다.

Example

```
-- Create a server-level SQL login for the domain user, if it doesn't already exist
USE [master]
GO
CREATE LOGIN [mydomain\user_name] FROM WINDOWS
GO

-- Create a database-level account for the domain user, if it doesn't already exist

USE [SSISDB]
GO
CREATE USER [mydomain\user_name] FOR LOGIN [mydomain\user_name]

-- Add SSIS role membership to the domain user
ALTER ROLE [ssis_admin] ADD MEMBER [mydomain\user_name]
ALTER ROLE [ssis_logreader] ADD MEMBER [mydomain\user_name]
GO
```

```
-- Add MSDB role membership to the domain user
USE [msdb]
GO
CREATE USER [mydomain\user_name] FOR LOGIN [mydomain\user_name]

-- Grant MSDB stored procedure privileges to the domain user
GRANT EXEC ON msdb.dbo.rds_msbi_task TO [mydomain\user_name] with grant option
GRANT SELECT ON msdb.dbo.rds_fn_task_status TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.rds_task_status TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.rds_cancel_task TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.rds_download_from_s3 TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.rds_upload_to_s3 TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.rds_delete_from_filesystem TO [mydomain\user_name] with grant
 option
GRANT EXEC ON msdb.dbo.rds_gather_file_details TO [mydomain\user_name] with grant
 option
GRANT EXEC ON msdb.dbo.sp_add_proxy TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.sp_update_proxy TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.sp_grant_login_to_proxy TO [mydomain\user_name] with grant
 option
GRANT EXEC ON msdb.dbo.sp_revoke_login_from_proxy TO [mydomain\user_name] with grant
 option
GRANT EXEC ON msdb.dbo.sp_delete_proxy TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.sp_enum_login_for_proxy to [mydomain\user_name] with grant
 option
GRANT EXEC ON msdb.dbo.sp_enum_proxy_for_subsystem TO [mydomain\user_name]  with grant
 option
GRANT EXEC ON msdb.dbo.rds_sqlagent_proxy TO [mydomain\user_name] WITH GRANT OPTION


-- Add the SQLAgentUserRole privilege to the domain user
USE [msdb]
GO
ALTER ROLE [SQLAgentUserRole] ADD MEMBER [mydomain\user_name]
GO

-- Grant the ALTER ANY CREDENTIAL privilege to the domain user
USE [master]
GO
GRANT ALTER ANY CREDENTIAL TO [mydomain\user_name]
GO
```

SSIS 프로젝트 배포

RDS에서는 SQL Server Management Studio(SSMS) 또는 SSIS 프로시저를 사용하여 SSIS 프로젝트를 직접 배포할 수 없습니다. Amazon S3에서 프로젝트 파일을 다운로드한 다음 배포하려면 RDS 저장 프로시저를 사용합니다.

저장 프로시저를 실행하려면 저장 프로시저를 실행할 권한을 부여한 사용자로 로그인합니다. 자세한 내용은 SSIS용 Windows 인증 사용자 설정 섹션을 참조하세요.

SSIS 프로젝트를 배포하려면

1. 프로젝트(.ispac) 파일을 다운로드합니다.

```
exec msdb.dbo.rds_download_from_s3
@s3_arn_of_file='arn:aws:s3:::bucket_name/ssisproject.ispac',
@rds_file_path='D:\S3\ssisproject.ispac',
@overwrite_file=1;
```

2. 다음 사항을 확인하여 배포 작업을 제출합니다.

   - 폴더는 SSIS 카탈로그에 있습니다.
   - 프로젝트 이름은 SSIS 프로젝트를 개발하는 동안 사용한 프로젝트 이름과 일치합니다.

```
exec msdb.dbo.rds_msbi_task
@task_type='SSIS_DEPLOY_PROJECT',
@folder_name='DEMO',
@project_name='ssisproject',
@file_path='D:\S3\ssisproject.ispac';
```

배포 작업의 상태 모니터링

배포 작업의 상태를 추적하려면 `rds_fn_task_status` 함수를 호출합니다. 두 가지 파라미터가 필요합니다. 첫 번째 파라미터는 SSIS에 적용되지 않기 때문에 항상 NULL이어야 합니다. 두 번째 파라미터는 작업 ID를 수락합니다.

모든 작업 목록을 보려면 다음 예와 같이 첫 번째 파라미터를 NULL로, 두 번째 파라미터를 0으로 설정하십시오.

```
SELECT * FROM msdb.dbo.rds_fn_task_status(NULL,0);
```

특정 작업을 수행하려면 다음 예와 같이 첫 번째 파라미터를 NULL로, 두 번째 파라미터를 작업 ID로 설정하십시오.

```
SELECT * FROM msdb.dbo.rds_fn_task_status(NULL,42);
```

rds_fn_task_status 함수는 다음 정보를 반환합니다.

출력 파라미터	설명
task_id	작업의 ID입니다.
task_type	SSIS_DEPLOY_PROJECT
database_name	SSIS 작업에는 적용되지 않습니다.
% complete	백분율로 나타낸 작업의 진행률입니다.
duration (mins)	작업에 소요된 시간입니다(분).
lifecycle	작업의 상태입니다. 가능한 상태는 다음과 같습니다. • CREATED – msdb.dbo.rds_msbi_task 저장 프로시저를 호출하면 작업이 생성되고 상태가 CREATED로 설정됩니다. • IN_PROGRESS – 작업이 시작되면 상태가 로 설정됩니다.IN_PROGRESS CREATED에서 IN_PROGRESS 로 상태가 변경되려면 최대 5분이 걸릴 수 있습니다. • SUCCESS – 작업이 완료되면 상태가 로 설정됩니다.SUCCESS • ERROR – 작업이 실패하면 상태가 로 설정됩니다.ERROR 오류에 대한 자세한 내용은 task_info 열을 참조하십시오.

출력 파라미터	설명
	• CANCEL_REQUESTED – rds_cancel_task 를 호출하는 즉시 작업의 상태가 CANCEL_REQUESTED 로 설정됩니다. • CANCELLED – 작업이 성공적으로 취소된 뒤에는 작업 상태가 로 설정됩니다.CANCELLED
task_info	작업에 대한 추가 정보입니다. 처리 중에 오류가 발생하면 이 열에 오류 정보가 포함됩니다.
last_updated	작업 상태를 마지막으로 업데이트한 날짜와 시간입니다.
created_at	작업을 생성한 날짜와 시간입니다.
S3_object_arn	SSIS 작업에는 적용되지 않습니다.
overwrite_S3_backup_file	SSIS 작업에는 적용되지 않습니다.
KMS_master_key_arn	SSIS 작업에는 적용되지 않습니다.
filepath	SSIS 작업에는 적용되지 않습니다.
overwrite_file	SSIS 작업에는 적용되지 않습니다.
task_metadata	SSIS 작업과 관련된 메타데이터입니다.

SSIS 사용

SSIS 프로젝트를 SSIS 카탈로그에 배포한 후 SSMS에서 직접 패키지를 실행하거나 SQL Server 에이전트를 사용하여 예약할 수 있습니다. SSIS 패키지를 실행하려면 Windows 인증 로그인을 사용해야 합니다. 자세한 내용은 SSIS용 Windows 인증 사용자 설정 섹션을 참조하세요.

주제

- SSIS 프로젝트에 대한 데이터베이스 연결 관리자 설정
- SSIS 프록시 생성
- SQL Server 에이전트를 사용하여 SSIS 패키지 예약
- 프록시에서 SSIS 액세스 취소

SSIS 프로젝트에 대한 데이터베이스 연결 관리자 설정

연결 관리자를 사용하는 경우 다음과 같은 유형의 인증을 사용할 수 있습니다.

- AWS 관리형 Active Directory를 사용하는 로컬 데이터베이스 연결의 경우 SQL 인증 또는 Windows 인증을 사용할 수 있습니다. Windows 인증의 경우 연결 문자열의 서버 이름으로 *DB_instance_name*.*fully_qualified_domain_name*을 사용합니다.

  예를 들어 myssisinstance.corp-ad.example.com입니다. 여기서 myssisinstance는 DB 인스턴스 이름이고 corp-ad.example.com은 정규화된 도메인 이름입니다.
- 원격 연결의 경우 항상 SQL 인증을 사용합니다.
- 자체 관리형 Active Directory를 사용하는 로컬 데이터베이스 연결의 경우 SQL 인증 또는 Windows 인증을 사용할 수 있습니다. Windows 인증의 경우 연결 문자열의 서버 이름으로 . 또는 *LocalHost*를 사용합니다.

SSIS 프록시 생성

SQL Server 에이전트를 사용하여 SSIS 패키지를 예약하려면 SSIS 자격 증명과 SSIS 프록시를 생성합니다. Windows 인증 사용자로 다음 절차를 실행합니다.

SSIS 자격 증명을 생성하려면

- 프록시에 대한 자격 증명을 생성합니다. 이렇게 하려면 SSMS 또는 다음 SQL 문을 사용하면 됩니다.

```
USE [master]
GO
CREATE CREDENTIAL [SSIS_Credential] WITH IDENTITY = N'mydomain\user_name', SECRET =
 N'mysecret'
GO
```

> **Note**
>
> IDENTITY는 도메인 인증 로그인이어야 합니다. *mysecret*를 도메인 인증 로그인에 대한 암호로 바꿉니다.
> SSISDB 기본 호스트가 변경될 때마다 새 호스트가 액세스할 수 있도록 SSIS 프록시 자격 증명을 변경합니다.

SSIS 프록시를 생성하려면

1. 다음 SQL 문을 사용하여 프록시를 생성합니다.

```
USE [msdb]
GO
EXEC msdb.dbo.sp_add_proxy
 @proxy_name=N'SSIS_Proxy',@credential_name=N'SSIS_Credential',@description=N''
GO
```

2. 다음 SQL 문을 사용하여 프록시에 대한 액세스 권한을 다른 사용자에게 부여합니다.

```
USE [msdb]
GO
EXEC msdb.dbo.sp_grant_login_to_proxy
 @proxy_name=N'SSIS_Proxy',@login_name=N'mydomain\user_name'
GO
```

3. 다음 SQL 문을 사용하여 SSIS 하위 시스템에 프록시에 대한 액세스 권한을 부여합니다.

```
USE [msdb]
GO
EXEC msdb.dbo.rds_sqlagent_proxy
 @task_type='GRANT_SUBSYSTEM_ACCESS',@proxy_name='SSIS_Proxy',@proxy_subsystem='SSIS'
GO
```

프록시 및 프록시의 부여를 보려면

1. 다음 SQL 문을 사용하여 프록시의 피부여자를 확인합니다.

```
USE [msdb]
```

```
GO
EXEC sp_help_proxy
GO
```

2. 다음 SQL 문을 사용하여 하위 시스템 부여를 확인합니다.

```
USE [msdb]
GO
EXEC msdb.dbo.sp_enum_proxy_for_subsystem
GO
```

SQL Server 에이전트를 사용하여 SSIS 패키지 예약

자격 증명 및 프록시를 생성하고 SSIS에 프록시에 대한 액세스 권한을 부여한 후 SSIS 패키지를 예약하는 SQL Server 에이전트 작업을 생성할 수 있습니다.

SSIS 패키지를 예약하려면

- SSMS 또는 T-SQL을 사용하여 SQL Server 에이전트 작업을 생성할 수 있습니다. 다음 예제에서는 T-SQL을 사용합니다.

```
USE [msdb]
GO
DECLARE @jobId BINARY(16)
EXEC msdb.dbo.sp_add_job @job_name=N'MYSSISJob',
@enabled=1,
@notify_level_eventlog=0,
@notify_level_email=2,
@notify_level_page=2,
@delete_level=0,
@category_name=N'[Uncategorized (Local)]',
@job_id = @jobId OUTPUT
GO
EXEC msdb.dbo.sp_add_jobserver @job_name=N'MYSSISJob',@server_name=N'(local)'
GO
EXEC msdb.dbo.sp_add_jobstep
 @job_name=N'MYSSISJob',@step_name=N'ExecuteSSISPackage',
@step_id=1,
@cmdexec_success_code=0,
@on_success_action=1,
@on_fail_action=2,
@retry_attempts=0,
```

```
@retry_interval=0,
@os_run_priority=0,
@subsystem=N'SSIS',
@command=N'/ISSERVER "\"\SSISDB\MySSISFolder\MySSISProject\MySSISPackage.dtsx\"" /
SERVER "\"my-rds-ssis-instance.corp-ad.company.com/\""
/Par "\"$ServerOption::LOGGING_LEVEL(Int16)\"";1 /Par
 "\"$ServerOption::SYNCHRONIZED(Boolean)\"";True /CALLERINFO SQLAGENT /REPORTING
 E',
@database_name=N'master',
@flags=0,
@proxy_name=N'SSIS_Proxy'
GO
```

프록시에서 SSIS 액세스 취소

다음 저장 프로시저를 사용하여 SSIS 하위 시스템에 대한 액세스를 취소하고 SSIS 프록시를 삭제할 수 있습니다.

액세스를 취소하고 프록시를 삭제하려면

1. 하위 시스템 액세스를 취소합니다.

```
USE [msdb]
GO
EXEC msdb.dbo.rds_sqlagent_proxy
 @task_type='REVOKE_SUBSYSTEM_ACCESS',@proxy_name='SSIS_Proxy',@proxy_subsystem='SSIS'
GO
```

2. 프록시에 대한 부여를 취소합니다.

```
USE [msdb]
GO
EXEC msdb.dbo.sp_revoke_login_from_proxy
 @proxy_name=N'SSIS_Proxy',@name=N'mydomain\user_name'
GO
```

3. 프록시를 삭제합니다.

```
USE [msdb]
GO
EXEC dbo.sp_delete_proxy @proxy_name = N'SSIS_Proxy'
```

```
GO
```

SSIS 데이터베이스 비활성화 및 삭제

다음 단계를 사용하여 SSIS 데이터베이스를 비활성화하거나 삭제합니다.

주제

- SSIS 비활성화
- SSISDB 데이터베이스 삭제

SSIS 비활성화

SSIS를 비활성화하려면 해당 옵션 그룹에서 `SSIS` 옵션을 제거합니다.

> **Important**
>
> 이 옵션을 제거해도 SSISDB 데이터베이스가 삭제되지 않으므로 SSIS 프로젝트 손실 없이 해당 옵션을 안전하게 제거할 수 있습니다.
> 제거 후 `SSIS` 옵션을 다시 활성화하여 이전에 SSIS 카탈로그에 배포된 SSIS 프로젝트를 다시 사용할 수 있습니다.

콘솔

다음 절차에서는 `SSIS` 옵션을 제거합니다.

옵션 그룹에서 SSIS 옵션을 제거하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. `SSIS` 옵션이 있는 옵션 그룹을 선택합니다(이전 예제의 경우 `ssis-se-2016`).
4. 옵션 삭제를 선택합니다.
5. 옵션 삭제에서 삭제할 옵션에 SSIS를 선택합니다.
6. 즉시 적용에서 옵션을 즉시 삭제하려면 예를 선택하고 다음 유지 관리 기간에 삭제하려면 아니오를 선택합니다.
7. Delete(삭제)를 선택합니다.

CLI

다음 절차에서는 SSIS 옵션을 제거합니다.

옵션 그룹에서 SSIS 옵션을 제거하려면

- 다음 명령 중 하나를 실행합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds remove-option-from-option-group \
    --option-group-name ssis-se-2016 \
    --options SSIS \
    --apply-immediately
```

  Windows의 경우:

```
aws rds remove-option-from-option-group ^
    --option-group-name ssis-se-2016 ^
    --options SSIS ^
    --apply-immediately
```

SSISDB 데이터베이스 삭제

SSIS 옵션을 제거한 후 SSISDB 데이터베이스는 삭제되지 않습니다. SSISDB 데이터베이스를 삭제하려면 SSIS 옵션을 제거한 후 rds_drop_ssis_database 저장 프로시저를 사용합니다.

SSIS 데이터베이스를 삭제하려면

- 다음 저장 프로시저를 사용합니다.

```
USE [msdb]
GO
EXEC dbo.rds_drop_ssis_database
GO
```

SSISDB 데이터베이스를 삭제한 후 SSIS 옵션을 다시 활성화하면 새 SSISDB 카탈로그를 얻을 수 있습니다.

Amazon RDS for SQL Server에서 SQL Server Reporting Services 지원

Microsoft SQL Server Reporting Services(SSRS)는 보고서 생성 및 배포에 사용되는 서버 기반 애플리케이션으로, SQL Server Analysis Services(SSAS) 및 SQL Server Integration Services(SSIS)를 포함하는 SQL Server 서비스 제품군의 일부입니다. SQL Server를 기반으로 하는 서비스인 SSRS를 사용하여 다양한 데이터 원본에서 데이터를 수집하고 쉽게 이해 및 분석 가능한 방식으로 제공할 수 있습니다.

Amazon RDS for SQL Server는 RDS DB 인스턴스에서 SSRS를 직접 실행할 수 있도록 지원합니다. 기존 DB 인스턴스 또는 새 DB 인스턴스에서 SSRS를 사용할 수 있습니다.

RDS는 다음 버전에서 SQL Server Standard 및 Enterprise Edition용 SSRS를 지원합니다.

- SQL Server 2022, 모든 버전
- SQL Server 2019, 버전 15.00.4043.16.v1 이상
- SQL Server 2017, 버전 14.00.3223.3.v1 이상
- SQL Server 2016, 13.00.5820.21.v1 이상 버전

목차

- 제한 및 권장 사항
- SSRS 설정
  - SSRS용 옵션 그룹 생성
  - 옵션 그룹에 SSRS 옵션 추가
  - 옵션 그룹을 DB 인스턴스와 연결
  - VPC 보안 그룹에 대한 인바운드 액세스 허용
- 보고서 서버 데이터베이스
- SSRS 로그 파일
- SSRS 웹 포털 액세스
  - RDS에서 SSL 사용
  - 도메인 사용자에게 액세스 권한 부여
  - 웹 포털 액세스
- 보고서 배포 및 보고서 데이터 소스 구성
  - SSRS에 보고서 배포
  - 보고서 데이터 소스 구성

- SSRS 이메일을 사용하여 보고서 보내기
- 시스템 수준 권한 취소
- 작업의 상태 모니터링
- SSRS 데이터베이스 비활성화 및 삭제
  - SSRS 해제
  - SSRS 데이터베이스 삭제

제한 및 권장 사항

RDS for SQL Server에서 SSRS를 실행하는 경우 다음과 같은 제한 및 권장 사항이 적용됩니다.

- 읽기 전용 복제본이 있는 DB 인스턴스에서는 SSRS를 사용할 수 없습니다.
- 인스턴스에서 자체 관리형 Active Directory 또는 SSRS용 AWS Directory Service for Microsoft Active Directory 웹 포털 및 웹 서버 인증을 사용해야 합니다. 자세한 내용은 RDS for SQL Server를 사용하여 Active Directory 작업 단원을 참조하십시오.
- SSRS 옵션으로 만든 보고 서버 데이터베이스는 백업할 수 없습니다.
- 다른 SSRS 인스턴스에서 보고서 서버 데이터베이스를 가져오고 복원하는 것은 지원되지 않습니다. 자세한 내용은 보고서 서버 데이터베이스 단원을 참조하십시오.
- 기본 SSL 포트(443)에서 수신하도록 SSRS를 구성할 수 없습니다. 허용되는 값은 1150–49511(1234, 1434, 3260, 3343, 3389, 47001 제외)입니다.
- Windows 파일 공유를 통한 구독은 지원되지 않습니다.
- 보고 서비스 구성 관리자를 사용하는 것은 지원되지 않습니다.
- 역할을 생성 및 수정하는 것은 지원되지 않습니다.
- 보고서 서버 속성을 수정하는 것은 지원되지 않습니다.
- 시스템 관리자 및 시스템 사용자 역할은 부여되지 않습니다.
- 웹 포털을 통해 시스템 수준의 역할 할당을 편집할 수 없습니다.

SSRS 설정

다음 프로세스를 사용하여 DB 인스턴스에 대해 SSRS를 켭니다.

1. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 선택합니다.
2. [SSRS] 옵션을 옵션 그룹에 추가합니다.

3. 옵션 그룹을 DB 인스턴스에 연동시킵니다.
4. SSRS 리스너 포트의 Virtual Private Cloud(VPC) 보안 그룹에 대한 인바운드 액세스를 허용합니다.

SSRS용 옵션 그룹 생성

SSRS를 사용하려면 SQL Server 엔진과 사용할 DB 인스턴스 버전에 해당하는 옵션 그룹을 생성합니다. 이렇게 하려면 AWS Management Console 또는 AWS CLI를 사용합니다.

> **Note**
>
> 올바른 SQL Server 엔진 및 버전인 경우 기존 옵션 그룹을 사용할 수도 있습니다.

콘솔

다음 절차에서는 SQL Server Standard Edition 2017에 대한 옵션 그룹을 생성합니다.

옵션 그룹을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 그룹 생성을 선택합니다.
4. 보안 그룹 생성 창에서 다음을 수행합니다.
   a. 이름에 AWS 계정 계정 내에서 쉽게 식별할 수 있는 옵션 그룹 이름을 입력합니다(예: **`ssrs-se-2017`**). 이름은 글자, 숫자 및 하이픈만 사용 가능합니다.
   b. 설명에 옵션 그룹에 대한 간단한 설명을 입력합니다(예: **`SSRS option group for SQL Server SE 2017`**). 이 설명은 표시 용도로만 사용됩니다.
   c. 엔진에 대해 sqlserver-se를 선택합니다.
   d. 메이저 엔진 버전에 대해 14.00을 선택합니다.
5. 생성(Create)을 선택합니다.

CLI

다음 절차에서는 SQL Server Standard Edition 2017에 대한 옵션 그룹을 생성합니다.

옵션 그룹을 생성하려면

- 다음 명령 중 하나를 실행합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-option-group \
    --option-group-name ssrs-se-2017 \
    --engine-name sqlserver-se \
    --major-engine-version 14.00 \
    --option-group-description "SSRS option group for SQL Server SE 2017"
```

Windows의 경우:

```
aws rds create-option-group ^
    --option-group-name ssrs-se-2017 ^
    --engine-name sqlserver-se ^
    --major-engine-version 14.00 ^
    --option-group-description "SSRS option group for SQL Server SE 2017"
```

옵션 그룹에 SSRS 옵션 추가

그런 다음 AWS Management Console 또는 AWS CLI를 사용하여 SSRS 옵션을 옵션 그룹에 추가합니다.

콘솔

SSRS 옵션을 추가하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 방금 생성한 옵션 그룹을 선택한 다음 옵션 추가를 선택합니다.
4. 옵션 세부 정보에서 옵션 이름으로 SSRS를 선택합니다.
5. 옵션 설정에서 다음을 수행합니다.

    a. SSRS 서비스가 수신할 포트를 입력합니다. 기본값은 8443입니다. 허용되는 값 목록은 제한 및 권장 사항 단원을 참조하십시오.

b. 최대 메모리에 값을 입력합니다.

   최대 메모리는 상한 임계값을 지정합니다. 이 임계값을 넘으면 보고서 서버 애플리케이션에 새 메모리 할당 요청이 수락되지 않습니다. 숫자는 DB 인스턴스에 대한 총 메모리 비율입니다. 허용되는 값은 10–80입니다.

c. 보안 그룹의 경우 옵션과 연결할 VPC 보안 그룹을 선택합니다. DB 인스턴스와 연결된 보안 그룹을 사용합니다.

6. SSRS 이메일을 사용하여 보고서를 보내려면 보고 서비스의 이메일 배달 아래에서 이메일 배달 옵션 구성 확인란을 선택한 후 다음을 수행합니다.

   a. 발신자 이메일 주소에 대해 SSRS 이메일에서 보낸 메시지의 보낸 사람 필드에 사용할 이메일 주소를 입력합니다.

      SMTP 서버에서 메일을 보낼 수 있는 권한이 있는 사용자 계정을 지정합니다.

   b. 에 대한SMTP 서버, 사용할 SMTP 서버 또는 게이트웨이를 지정합니다.

      IP 주소, 회사 인트라넷에 있는 컴퓨터의 NetBIOS 이름 또는 정규화된 도메인 이름이 될 수 있습니다.

   c. SMTP 포트에 메일 서버에 연결할 때 사용할 포트를 입력합니다. 기본값은 25입니다.

   d. 인증을 사용하려면:

      i. 인증 사용확인란을 선택합니다.

      ii. Secret Amazon 리소스 이름(ARN)에 사용자 자격 증명에 대한 AWS Secrets Manager ARN을 입력합니다.

         다음 형식을 사용합니다.

         arn:aws:secretsmanager:*Region*:*AccountId*:secret:*SecretName*-*6RandomChara*

         예:

         arn:aws:secretsmanager:*us-west-2*:*123456789012*:secret:*MySecret-a1b2c3*

         새 SFTP 비밀 생성에 대한 자세한 정보는 SSRS 이메일을 사용하여 보고서 보내기 단원을 참조하십시오.

   e. SSL을 사용하여 이메일 메시지를 암호화하려면 SSL(Secure Sockets Layer) 사용 확인란을 선택합니다.

7. 예약에서 옵션을 즉시 추가할지 또는 다음 유지 관리 기간에 추가할지를 선택합니다.
8. 옵션 추가를 선택합니다.

CLI

SSRS 옵션을 추가하려면

1. JSON 파일 (예: `ssrs-option.json`)을 새 제품으로 합니다.
   a. 다음 필수 파라미터를 설정합니다.
      - `OptionGroupName` – 이전에 생성했거나 선택한 옵션 그룹의 이름입니다(다음 예의 경우 `ssrs-se-2017`).
      - `Port` – SSRS 서비스가 수신할 포트입니다. 기본값은 8443입니다. 허용되는 값 목록은 제한 및 권장 사항 단원을 참조하십시오.
      - `VpcSecurityGroupMemberships` – RDS DB 인스턴스에 대한 VPC 보안 그룹 멤버십입니다.
      - `MAX_MEMORY` – 상한 임계값을 지정합니다. 이 임계값을 넘으면 보고서 서버 애플리케이션에 새 메모리 할당 요청이 수락되지 않습니다. 숫자는 DB 인스턴스에 대한 총 메모리 비율입니다. 허용되는 값은 10–80입니다.
   b. (선택 사항) SSRS 이메일을 사용하려면 다음 파라미터를 설정합니다.
      - `SMTP_ENABLE_EMAIL` — `true`로 설정하여 SSRS 이메일을 사용합니다. 기본값은 `false`입니다.
      - `SMTP_SENDER_EMAIL_ADDRESS` — SSRS 이메일에서 보낸 메시지의 보낸 사람 필드에 사용할 전자 메일 주소입니다. SMTP 서버에서 메일을 보낼 수 있는 권한이 있는 사용자 계정을 지정합니다.
      - `SMTP_SERVER` — 사용할 SMTP 서버 또는 게이트웨이입니다. IP 주소, 회사 인트라넷에 있는 컴퓨터의 NetBIOS 이름 또는 정규화된 도메인 이름이 될 수 있습니다.
      - `SMTP_PORT` — 메일 서버에 연결하는 데 사용할 포트입니다. 기본값은 25입니다.
      - `SMTP_USE_SSL` — `true`를 설정하여 SSL을 사용하는 이메일 메시지를 암호화합니다. 기본값은 `true`입니다.
      - `SMTP_EMAIL_CREDENTIALS_SECRET_ARN` — 사용자 자격 증명을 보관하는 Secrets Manager ARN. 다음 형식을 사용합니다.

        **arn:aws:secretsmanager:*Region*:*AccountId*:secret:*SecretName*-*6RandomCharact*

비밀 생성에 대한 자세한 내용은 SSRS 이메일을 사용하여 보고서 보내기 단원을 참조하세요.

- SMTP_USE_ANONYMOUS_AUTHENTICATION — 인증을 사용하지 않으려면 true로 설정하고 SMTP_EMAIL_CREDENTIALS_SECRET_ARN을 포함하지 마세요.

  기본값은 SMTP_ENABLE_EMAIL이 true일 때 false입니다.

다음 예에는 암호 ARN을 사용하는 SSRS 이메일 파라미터가 포함되어 있습니다.

```
{
"OptionGroupName": "ssrs-se-2017",
"OptionsToInclude": [
 {
 "OptionName": "SSRS",
 "Port": 8443,
 "VpcSecurityGroupMemberships": ["sg-0abcdef123"],
 "OptionSettings": [
            {"Name": "MAX_MEMORY","Value": "60"},
            {"Name": "SMTP_ENABLE_EMAIL","Value": "true"}
            {"Name": "SMTP_SENDER_EMAIL_ADDRESS","Value": "nobody@example.com"},
            {"Name": "SMTP_SERVER","Value": "email-smtp.us-west-2.amazonaws.com"},
            {"Name": "SMTP_PORT","Value": "25"},
            {"Name": "SMTP_USE_SSL","Value": "true"},
            {"Name": "SMTP_EMAIL_CREDENTIALS_SECRET_ARN","Value":
 "arn:aws:secretsmanager:us-west-2:123456789012:secret:MySecret-a1b2c3"}
            ]
 }],
"ApplyImmediately": true
}
```

2. [SSRS] 옵션을 옵션 그룹에 추가합니다.

   Example

   대상 LinuxmacOS, 또는Unix:

```
aws rds add-option-to-option-group \
    --cli-input-json file://ssrs-option.json \
    --apply-immediately
```

Windows의 경우:

```
aws rds add-option-to-option-group ^
    --cli-input-json file://ssrs-option.json ^
    --apply-immediately
```

옵션 그룹을 DB 인스턴스와 연결

AWS Management Console 또는 AWS CLI를 사용하여 옵션 그룹을 DB 인스턴스와 연결할 수 있습니다.

기존 DB 인스턴스를 사용하는 경우 이미 Active Directory 도메인과 AWS Identity and Access Management(IAM) 역할이 연결되어 있어야 합니다. 새 인스턴스를 생성하는 경우 기존 Active Directory 도메인 및 IAM 역할을 지정합니다. 자세한 내용은 RDS for SQL Server를 사용하여 Active Directory 작업 단원을 참조하십시오.

콘솔

옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스와 연결할 수 있습니다.

- 새 DB 인스턴스의 경우, 인스턴스를 시작할 때 옵션 그룹을 연결하십시오. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
- 기존 DB 인스턴스의 경우 인스턴스를 수정하고 새 옵션 그룹을 연결합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정을 참조하세요.

CLI

옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스와 연결할 수 있습니다.

옵션 그룹을 사용하는 DB 인스턴스를 생성하려면

- 옵션 그룹을 생성할 때 사용한 것과 동일한 DB 엔진 유형과 메이저 버전을 지정합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance \
    --db-instance-identifier myssrsinstance \
```

```
    --db-instance-class db.m5.2xlarge \
    --engine sqlserver-se \
    --engine-version 14.00.3223.3.v1 \
    --allocated-storage 100 \
    --manage-master-user-password  \
    --master-username admin \
    --storage-type gp2 \
    --license-model li \
    --domain-iam-role-name my-directory-iam-role \
    --domain my-domain-id \
    --option-group-name ssrs-se-2017
```

Windows의 경우:

```
aws rds create-db-instance ^
    --db-instance-identifier myssrsinstance ^
    --db-instance-class db.m5.2xlarge ^
    --engine sqlserver-se ^
    --engine-version 14.00.3223.3.v1 ^
    --allocated-storage 100 ^
    --manage-master-user-password ^
    --master-username admin ^
    --storage-type gp2 ^
    --license-model li ^
    --domain-iam-role-name my-directory-iam-role ^
    --domain my-domain-id ^
    --option-group-name ssrs-se-2017
```

옵션 그룹을 사용하도록 DB 인스턴스를 수정하려면

- 다음 명령 중 하나를 실행합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier myssrsinstance \
    --option-group-name ssrs-se-2017 \
    --apply-immediately
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier myssrsinstance ^
    --option-group-name ssrs-se-2017 ^
    --apply-immediately
```

VPC 보안 그룹에 대한 인바운드 액세스 허용

DB 인스턴스와 연결된 VPC 보안 그룹에 대한 인바운드 액세스를 허용하려면 지정된 SSRS 리스너 포트에 대한 인바운드 규칙을 생성합니다. 보안 그룹 설정에 대한 자세한 내용은 보안 그룹을 생성하여 VPC 내부의 DB 인스턴스에 대한 액세스를 제공 단원을 참조하십시오.

보고서 서버 데이터베이스

DB 인스턴스가 SSRS 옵션과 연결되면 DB 인스턴스에 두 개의 새 데이터베이스가 생성됩니다.

- `rdsadmin_ReportServer`
- `rdsadmin_ReportServerTempDB`

이러한 데이터베이스는 ReportServer 및 ReportServerTempDB 데이터베이스로 작동합니다. SSRS는 ReportServer 데이터베이스에 데이터를 저장하고 ReportServerTempDB 데이터베이스에 데이터를 캐시합니다. 자세한 정보는 Microsoft 문서의 보고서 서버 데이터베이스를 참조하세요.

RDS는 이러한 데이터베이스를 소유하고 관리하므로 ALTER 및 DROP 등의 데이터베이스 작업은 허용되지 않습니다. `rdsadmin_ReportServerTempDB` 데이터베이스에서는 액세스가 허용되지 않습니다. 그러나 `rdsadmin_ReportServer` 데이터베이스에서 읽기 작업을 수행할 수 있습니다.

SSRS 로그 파일

SSRS 로그 파일을 나열하고 보고 다운로드할 수 있습니다. SSRS 로그 파일은 ReportServerService_*timestamp*.log의 명명 규칙을 따릅니다. 이러한 보고서 서버 로그는 `D:\rdsdbdata\Log\SSRS` 디렉터리에 있습니다. (`D:\rdsdbdata\Log` 디렉터리는 오류 로그 및 SQL Server 에이전트 로그의 상위 디렉터리이기도 합니다.) 자세한 내용은 데이터베이스 로그 파일 보기 및 나열 단원을 참조하십시오.

기존 SSRS 인스턴스의 경우 보고서 서버 로그에 액세스하려면 SSRS 서비스를 다시 시작해야 할 수 있습니다. SSRS 옵션을 업데이트하여 서비스를 재시작하면 됩니다.

자세한 내용은 Amazon RDS for Microsoft SQL Server 로그 작업을 참조하세요.

SSRS 웹 포털 액세스

SSRS 웹 포털에 액세스하려면 다음 프로세스를 사용합니다.

1. Secure Sockets Layer(SSL) 설정
2. 도메인 사용자에게 액세스 권한을 부여합니다.
3. 브라우저 및 도메인 사용자 자격 증명을 사용하여 웹 포털에 액세스합니다.

RDS에서 SSL 사용

SSRS는 연결에 HTTPS SSL 프로토콜을 사용합니다. 이 프로토콜을 사용할 수 있도록 클라이언트 컴퓨터의 Microsoft Windows 운영 체제로 SSL 인증서를 가져옵니다.

SSL 인증서에 대한 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 단원을 참조하십시오. SQL Server에서 SSL을 사용하는 방법에 대한 자세한 내용은 ??? 단원을 참조하십시오.

도메인 사용자에게 액세스 권한 부여

새 SSRS 활성화 시 SSRS에 역할이 할당되어 있지 않습니다. 도메인 사용자 또는 사용자 그룹에 웹 포털에 대한 액세스 권한을 부여하기 위해 RDS는 저장 프로시저를 제공합니다.

웹 포털에서 도메인 사용자에게 액세스 권한을 부여하려면

- 다음 저장 프로시저를 사용합니다.

```
exec msdb.dbo.rds_msbi_task
@task_type='SSRS_GRANT_PORTAL_PERMISSION',
@ssrs_group_or_username=N'AD_domain\user';
```

도메인 사용자 또는 사용자 그룹에는 RDS_SSRS_ROLE 시스템 역할이 부여됩니다. 이 역할은 다음과 같은 시스템 수준의 작업을 수행할 수 있습니다.

- 보고서 실행
- 작업 관리
- 공유 일정 관리
- 공유 일정 보기

루트 폴더에서 항목 수준의 Content Manager 역할도 부여됩니다.

웹 포털 액세스

SSRS_GRANT_PORTAL_PERMISSION 작업이 성공적으로 완료되면 웹 브라우저를 사용하여 포털에 액세스할 수 있습니다. 웹 포털 URL의 형식은 다음과 같습니다.

```
https://rds_endpoint:port/Reports
```

위의 형식에서 각 요소는 다음과 같습니다.

- *rds_endpoint* – SSRS와 함께 사용 중인 RDS DB 인스턴스의 엔드포인트입니다.

  이 엔드포인트는 DB 인스턴스의 연결 및 보안 탭에서 찾을 수 있습니다. 자세한 내용은 Microsoft SQL Server DB 인스턴스에 연결 섹션을 참조하세요.
- *port* – SSRS 옵션에서 설정한 SSRS의 리스너 포트입니다.

웹 포털에 액세스하려면

1. 브라우저에 웹 포털 URL을 입력합니다.

   ```
   https://myssrsinstance.cg034itsfake.us-east-1.rds.amazonaws.com:8443/Reports
   ```
2. SSRS_GRANT_PORTAL_PERMISSION 작업을 통해 액세스 권한을 부여한 도메인 사용자의 자격 증명으로 로그인합니다.

보고서 배포 및 보고서 데이터 소스 구성

다음 절차에 따라 보고서를 SSRS에 배포하고 보고 데이터 소스를 구성합니다.

주제

- SSRS에 보고서 배포
- 보고서 데이터 소스 구성

SSRS에 보고서 배포

웹 포털에 대한 액세스 권한이 있으면 웹 포털에 보고서를 배포할 수 있습니다. 웹 포털의 업로드 도구를 사용하여 보고서를 업로드하거나 SQL Server 데이터 도구(SSDT)에서 직접 배포할 수 있습니다. SSDT에서 배포하는 경우 다음을 확인하세요.

- SSDT를 시작한 사용자가 SSRS 웹 포털에 액세스할 수 있습니다.
- SSRS 프로젝트 속성의 `TargetServerURL` 값이 다음과 같이 `ReportServer` 접미사가 붙은 RDS DB 인스턴스의 HTTPS 엔드포인트로 설정됩니다.

```
https://myssrsinstance.cg034itsfake.us-east-1.rds.amazonaws.com:8443/ReportServer
```

보고서 데이터 소스 구성

보고서를 SSRS에 배포한 후에는 보고서 데이터 소스를 구성해야 합니다. 보고서 데이터 소스를 구성할 때 다음 요구 사항을 충족해야 합니다.

- AWS Directory Service for Microsoft Active Directory에 연결된 RDS for SQL Server DB 인스턴스의 경우 정규화된 도메인 이름(FQDN)을 연결 문자열의 데이터 소스 이름으로 사용합니다. 예를 들어 *myssrsinstance.corp-ad.example.com*입니다. 여기서 *myssrsinstance*는 DB 인스턴스 이름이고 *corp-ad.example.com*은 정규화된 도메인 이름입니다.
- 자체 관리형 Active Directory에 연결된 RDS for SQL Server DB 인스턴스의 경우 `.` 또는 *LocalHost*를 연결 문자열의 데이터 소스 이름으로 사용합니다.

SSRS 이메일을 사용하여 보고서 보내기

SSRS에는 사용자에게 보고서를 보내는 데 사용할 수 있는 SSRS 전자 메일 확장이 포함되어 있습니다.

SSRS 이메일을 구성하려면 SSRS 옵션 설정을 사용합니다. 자세한 내용은 옵션 그룹에 SSRS 옵션 추가을 참조하세요.

SSRS 이메일을 구성한 후 보고서 서버에서 보고서를 구독할 수 있습니다. 자세한 내용은 보고 서비스를 통한 이메일 전달을 알아보려면 다음 섹션을 참조하세요.마이크로소프트 설명서에서

SSRS 이메일이 RDS에서 작동하려면 AWS Secrets Manager와의 통합이 필요합니다. Secrets Manager와 통합하려면 보안 암호를 생성합니다.

> **Note**
>
> 나중에 암호를 변경하는 경우 옵션 그룹의 SSRS 옵션도 업데이트해야 합니다.

SSRS 이메일에 보안 암호를 만들려면

1. AWS Secrets Manager 사용 설명서의 암호 생성 단계를 따릅니다.

   a. 암호 유형 선택에서 다른 유형의 암호를 선택합니다.

   b. 키/값 쌍의 경우 다음을 입력합니다.

      - **SMTP_USERNAME** - SMTP 서버에서 메일을 보낼 권한이 있는 사용자를 입력합니다.
      - **SMTP_PASSWORD** - SMTP 사용자의 암호를 입력합니다.

   c. 암호화 키의 경우 기본 AWS KMS key을 사용하지 않습니다. 기존의 키를 사용하거나 새 키를 생성합니다.

      KMS 키 정책은 다음과 같은 `kms:Decrypt` 작업을 허용해야 합니다.

```
{
    "Sid": "Allow use of the key",
    "Effect": "Allow",
    "Principal": {
        "Service": [
            "rds.amazonaws.com"
        ]
    },
    "Action": [
        "kms:Decrypt"
    ],
    "Resource": "*"
}
```

2. AWS Secrets Manager 사용 설명서의 암호에 권한 정책 연결의 단계를 따르세요. 권한 정책은 `rds.amazonaws.com` 서비스 주체에게 `secretsmanager:GetSecretValue` 작업을 제공합니다.

   혼란스러운 대리인 문제를 피하기 위해 정책의 `aws:sourceAccount` 및 `aws:sourceArn` 조건을 사용하는 것이 좋습니다. `aws:sourceAccount`에 대해 AWS 계정 및 `aws:sourceArn`의 옵션 그룹 ARN을 사용합니다. 자세한 내용은 교차 서비스 혼동된 대리자 문제 방지 단원을 감사하세요.

   다음 예는 권한 정책을 보여 줍니다.

JSON

```
{
  "Version" : "2012-10-17",
  "Statement" : [ {
    "Effect" : "Allow",
    "Principal" : {
      "Service" : "rds.amazonaws.com"
    },
    "Action" : "secretsmanager:GetSecretValue",
    "Resource" : "*",
    "Condition" : {
      "StringEquals" : {
        "aws:sourceAccount" : "123456789012"
      },
      "ArnLike" : {
        "aws:sourceArn" : "arn:aws:rds:us-west-2:123456789012:og:ssrs-
se-2017"
      }
    }
  } ]
}
```

더 많은 예제는 AWS Secrets Manager 사용 설명서의 AWS Secrets Manager에 대한 권한 정책 예를 참조하세요.

시스템 수준 권한 취소

RDS_SSRS_ROLE 시스템 역할에는 시스템 수준 역할 할당을 삭제할 수 있는 권한이 없습니다. RDS_SSRS_ROLE에서 사용자 또는 사용자 그룹을 제거하려면 역할을 부여하는 데 사용한 것과 동일한 저장 프로시저를 사용하되 SSRS_REVOKE_PORTAL_PERMISSION 작업 유형을 사용합니다.

웹 포털에 대한 도메인 사용자의 액세스 권한을 취소하려면

- 다음 저장 프로시저를 사용합니다.

```
exec msdb.dbo.rds_msbi_task
@task_type='SSRS_REVOKE_PORTAL_PERMISSION',
```

```
@ssrs_group_or_username=N'AD_domain\user';
```

이렇게 하면 RDS_SSRS_ROLE 시스템 역할에서 사용자가 삭제됩니다. 또한 사용자가 Content Manager 항목 수준 역할(역할을 가진 경우)에서 삭제됩니다.

작업의 상태 모니터링

권한 부여 또는 취소 작업의 상태를 추적하려면 rds_fn_task_status 함수를 호출합니다. 두 가지 파라미터가 필요합니다. 첫 번째 파라미터는 SSRS에 적용되지 않기 때문에 항상 NULL이어야 합니다. 두 번째 파라미터는 작업 ID를 수락합니다.

모든 작업 목록을 보려면 다음 예와 같이 첫 번째 파라미터를 NULL로, 두 번째 파라미터를 0으로 설정하십시오.

```
SELECT * FROM msdb.dbo.rds_fn_task_status(NULL,0);
```

특정 작업을 수행하려면 다음 예와 같이 첫 번째 파라미터를 NULL로, 두 번째 파라미터를 작업 ID로 설정하십시오.

```
SELECT * FROM msdb.dbo.rds_fn_task_status(NULL,42);
```

rds_fn_task_status 함수는 다음 정보를 반환합니다.

출력 파라미터	설명
task_id	작업의 ID입니다.
task_type	SSRS의 경우 작업 유형은 다음과 같을 수 있습니다. • SSRS_GRANT_PORTAL_PERMISSION • SSRS_REVOKE_PORTAL_PERMISSION
database_name	SSRS 작업에는 적용되지 않습니다.
% complete	백분율로 나타낸 작업의 진행률입니다.
duration (mins)	작업에 소요된 시간입니다(분).

출력 파라미터	설명
lifecycle	작업의 상태입니다. 가능한 상태는 다음과 같습니다. • CREATED – SSRS 저장 프로시저 중 하나를 호출하면 작업이 생성되고 상태가 로 설정됩니다.CREATED • IN_PROGRESS – 작업이 시작되면 상태가 로 설정됩니다.IN_PROGRESS CREATED에서 IN_PROGRESS 로 상태가 변경되려면 최대 5분이 걸릴 수 있습니다. • SUCCESS – 작업이 완료되면 상태가 로 설정됩니다.SUCCESS • ERROR – 작업이 실패하면 상태가 로 설정됩니다.ERROR 오류에 대한 자세한 내용은 task_info 열을 참조하십시오. • CANCEL_REQUESTED – rds_cancel_task 저장 프로시저를 호출한 후 작업의 상태가 CANCEL_REQUESTED 로 설정됩니다. • CANCELLED – 작업이 성공적으로 취소된 뒤에는 작업 상태가 로 설정됩니다.CANCELLED
task_info	작업에 대한 추가 정보입니다. 처리 중에 오류가 발생하면 이 열에 오류 정보가 포함됩니다.
last_updated	작업 상태를 마지막으로 업데이트한 날짜와 시간입니다.
created_at	작업을 생성한 날짜와 시간입니다.

출력 파라미터	설명
S3_object_arn	SSRS 작업에는 적용되지 않습니다.
overwrite_S3_backup_file	SSRS 작업에는 적용되지 않습니다.
KMS_master_key_arn	SSRS 작업에는 적용되지 않습니다.
filepath	SSRS 작업에는 적용되지 않습니다.
overwrite_file	SSRS 작업에는 적용되지 않습니다.
task_metadata	SSRS 작업과 관련된 메타데이터입니다.

SSRS 데이터베이스 비활성화 및 삭제

다음 절차에 따라 SSRS를 비활성화하고 SSRS 데이터베이스를 삭제합니다.

주제

- SSRS 해제
- SSRS 데이터베이스 삭제

SSRS 해제

SSRS를 해제하려면 해당 옵션 그룹에서 SSRS 옵션을 제거합니다. 이 옵션을 제거해도 SSRS 데이터베이스는 삭제되지 않습니다. 자세한 내용은 SSRS 데이터베이스 삭제을 참조하세요.

SSRS 옵션을 다시 추가하여 SSRS를 다시 켤 수 있어야 합니다. SSRS 데이터베이스도 삭제한 경우 동일한 DB 인스턴스에서 옵션을 읽으면 새 보고서 서버 데이터베이스가 생성됩니다.

콘솔

옵션 그룹에서 SSRS 옵션을 제거하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. SSRS 옵션이 있는 옵션 그룹을 선택합니다(이전 예제의 경우 ssrs-se-2017).
4. 옵션 삭제를 선택합니다.

5. 옵션 삭제에서 SSRS 또는 삭제할 옵션을 선택합니다.
6. 즉시 적용에서 옵션을 즉시 삭제하려면 예를 선택하고 다음 유지 관리 기간에 삭제하려면 아니오를 선택합니다.
7. Delete(삭제)를 선택합니다.

CLI

옵션 그룹에서 SSRS 옵션을 제거하려면

- 다음 명령 중 하나를 실행합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds remove-option-from-option-group \
    --option-group-name ssrs-se-2017 \
    --options SSRS \
    --apply-immediately
```

  Windows의 경우:

```
aws rds remove-option-from-option-group ^
    --option-group-name ssrs-se-2017 ^
    --options SSRS ^
    --apply-immediately
```

SSRS 데이터베이스 삭제

SSRS 옵션을 제거해도 보고서 서버 데이터베이스는 삭제되지 않습니다. 삭제하려면 다음 저장 프로시저를 사용합니다.

보고서 서버 데이터베이스를 삭제하려면 먼저 SSRS 옵션을 제거해야 합니다.

SSRS 데이터베이스를 삭제하려면

- 다음 저장 프로시저를 사용합니다.

```
exec msdb.dbo.rds_drop_ssrs_databases
```

RDS for SQL Server에서 Microsoft Distributed Transaction Coordinator 지원

분산 트랜잭션은 두 개 이상의 네트워크 호스트가 관련된 데이터베이스 트랜잭션입니다. RDS for SQL Server는 호스트 간의 분산 트랜잭션을 지원하며 여기서 단일 호스트는 다음 중 하나일 수 있습니다.

- RDS for SQL Server DB 인스턴스
- 온프레미스 SQL Server 호스트
- SQL Server가 설치된 Amazon EC2 호스트
- 분산 트랜잭션을 지원하는 데이터베이스 엔진이 있는 기타 EC2 호스트 또는 RDS DB 인스턴스

RDS에서는 SQL Server 2012(버전 11.00.5058.0.v1 이상)부터 모든 에디션의 RDS for SQL Server에서 분산 트랜잭션을 지원합니다. Microsoft Distributed Transaction Coordinator(MSDTC)를 사용하여 지원이 제공됩니다. MSDTC에 대한 자세한 내용은 Microsoft 설명서에서 Distributed Transaction Coordinator를 참조하세요.

목차

- 제한 사항
- MSDTC 활성화
  - MSDTC용 옵션 그룹 생성
  - 옵션 그룹에 MSDTC 옵션 추가
  - MSDTC용 파라미터 그룹 생성
  - MSDTC에 대한 파라미터 수정
  - 옵션 그룹 및 파라미터 그룹을 DB 인스턴스와 연결
  - MSDTC 옵션 수정
- 트랜잭션 사용
  - 분산 트랜잭션 사용
  - XA 트랜잭션 사용
  - 트랜잭션 추적 사용
- MSDTC 비활성화
- RDS for SQL Server에 대한 MSDTC 문제 해결

제한 사항

RDS for SQL Server에서 MSDTC를 사용하는 경우 다음과 같은 제한 사항이 적용됩니다.

- SQL Server 데이터베이스 미러링을 사용하는 인스턴스에서는 MSDTC가 지원되지 않습니다. 자세한 내용은 트랜잭션 - 가용성 그룹 및 데이터베이스 미러링을 참조하십시오.
- `in-doubt xact resolution` 파라미터를 1 또는 2로 설정해야 합니다. 자세한 내용은 MSDTC에 대한 파라미터 수정 단원을 참조하세요.
- MSDTC에서는 분산 트랜잭션에 참여하는 모든 호스트를 호스트 이름을 사용하여 확인할 수 있어야 합니다. RDS는 도메인에 가입된 인스턴스에 대해 이 기능을 자동으로 유지 관리합니다. 그러나 독립 실행형 인스턴스의 경우 DNS 서버를 수동으로 구성해야 합니다.
- Java Database Connectivity(JDBC) XA 트랜잭션은 SQL Server 2017 버전 14.00.3223.3 이상 및 SQL Server 2019에서 지원됩니다.
- RDS 인스턴스의 클라이언트 동적 연결 라이브러리(DLL)를 사용하는 분산 트랜잭션은 지원되지 않습니다.
- 사용자 지정 XA 동적 연결 라이브러리 사용은 지원되지 않습니다.

MSDTC 활성화

DB 인스턴스에 MSDTC를 활성화하려면 다음 프로세스를 사용합니다.

1. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 선택합니다.
2. [MSDTC] 옵션을 옵션 그룹에 추가합니다.
3. 새 파라미터 그룹을 생성하거나 기존 파라미터 그룹을 선택합니다.
4. 파라미터 그룹을 수정하여 `in-doubt xact resolution` 파라미터를 1 또는 2로 설정합니다.
5. 옵션 그룹 및 파라미터 그룹을 DB 인스턴스와 연결합니다.

MSDTC용 옵션 그룹 생성

AWS Management Console 또는 AWS CLI를 사용하여 SQL Server 엔진과 DB 인스턴스 버전에 해당하는 옵션 그룹을 생성합니다.

> **Note**
>
> 올바른 SQL Server 엔진 및 버전인 경우 기존 옵션 그룹을 사용할 수도 있습니다.

콘솔

다음 절차에서는 SQL Server Standard Edition 2016에 대한 옵션 그룹을 생성합니다.

옵션 그룹을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 그룹 생성을 선택합니다.
4. 보안 그룹 생성 창에서 다음을 수행합니다.
   a. [이름(Name)]에 AWS 계정 내에서 쉽게 식별할 수 있는 옵션 그룹 이름을 입력합니다(예: **msdtc-se-2016**). 이름은 글자, 숫자 및 하이픈만 사용 가능합니다.
   b. 설명에 옵션 그룹에 대한 간단한 설명을 입력합니다(예: **MSDTC option group for SQL Server SE 2016**). 이 설명은 표시 용도로만 사용됩니다.
   c. 엔진에 대해 sqlserver-se를 선택합니다.
   d. 메이저 엔진 버전에 13.00을 선택합니다.
5. 생성(Create)을 선택합니다.

CLI

다음 예에서는 SQL Server Standard Edition 2016에 대한 옵션 그룹을 생성합니다.

옵션 그룹을 생성하려면

- 다음 명령 중 하나를 사용합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds create-option-group \
    --option-group-name msdtc-se-2016 \
    --engine-name sqlserver-se \
    --major-engine-version 13.00 \
    --option-group-description "MSDTC option group for SQL Server SE 2016"
```

  Windows의 경우:

```
aws rds create-option-group ^
    --option-group-name msdtc-se-2016 ^
```

```
    --engine-name sqlserver-se ^
    --major-engine-version 13.00 ^
    --option-group-description "MSDTC option group for SQL Server SE 2016"
```

옵션 그룹에 MSDTC 옵션 추가

그런 다음 AWS Management Console 또는 AWS CLI를 사용하여 MSDTC 옵션을 옵션 그룹에 추가합니다.

다음 옵션 설정이 필요합니다.

- 포트 – MSDTC에 액세스하는 데 사용하는 포트입니다. 허용되는 값은 1150–49151(1234, 1434, 3260, 3343, 3389, 47001 제외)입니다. 기본값은 5000입니다.

  사용하려는 포트가 방화벽 규칙에서 활성화되어 있는지 확인합니다. 또한 필요에 따라 DB 인스턴스와 연결된 보안 그룹의 인바운드 및 아웃바운드 규칙에서 이 포트를 활성화해야 합니다. 자세한 내용은 Amazon RDS DB 인스턴스에 연결할 수 없음 단원을 참조하세요.
- 보안 그룹 – RDS DB 인스턴스에 대한 VPC 보안 그룹 멤버십입니다.
- 인증 유형 – 호스트 간의 인증 모드입니다. 지원되는 인증 유형은 다음과 같습니다.
  - 상호 – RDS 인스턴스가 통합 인증을 사용하여 상호 인증됩니다. 이 옵션을 선택하는 경우 이 옵션 그룹과 연결된 모든 인스턴스가 도메인에 가입되어 있어야 합니다.
  - 없음 – 호스트 간에 인증이 수행되지 않습니다. 프로덕션 환경에서는 이 모드를 사용하지 않는 것이 좋습니다.
- 트랜잭션 로그 크기 – MSDTC 트랜잭션 로그의 크기입니다. 허용되는 값은 4–1024MB입니다. 기본 크기는 4MB입니다.

다음 옵션 설정은 선택 사항입니다.

- 인바운드 연결 활성화 – 이 옵션 그룹과 연결된 인스턴스로의 인바운드 MSDTC 연결을 허용할지 여부를 지정합니다.
- 아웃바운드 연결 활성화 – 이 옵션 그룹과 연결된 인스턴스로부터의 아웃바운드 MSDTC 연결을 허용할지 여부를 지정합니다.
- XA 활성화 – XA 트랜잭션을 허용할지 여부를 지정합니다 . XA 프로토콜에 대한 자세한 내용은 XA 사양을 참조하세요.

- SNA LU 활성화 – 분산 트랜잭션에 SNA LU 프로토콜을 사용하도록 허용할지 여부를 지정합니다. SNA LU 프로토콜 지원에 대한 자세한 내용은 Microsoft 설명서의 IBM CICS LU 6.2 트랜잭션 관리를 참조하세요.

콘솔

MSDTC 옵션을 추가하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 방금 생성한 옵션 그룹을 선택합니다.
4. 옵션 추가를 선택합니다.
5. 옵션 세부 정보에서 옵션 이름으로 MSDTC를 선택합니다.
6. 옵션 설정에서 다음을 수행합니다.
   a. 포트에 MSDTC에 액세스하는 데 사용할 포트 번호를 입력합니다. 기본값은 5000입니다.
   b. 보안 그룹의 경우 옵션과 연결할 VPC 보안 그룹을 선택합니다.
   c. 인증 유형에서 상호 또는 없음을 선택합니다.
   d. 트랜잭션 로그 크기에 4–1024 사이의 값을 입력합니다. 기본값은 4입니다.
7. 추가 구성에서 다음을 수행합니다.
   a. 연결에서 필요에 따라 인바운드 연결 활성화 및 아웃바운드 연결 활성화를 선택합니다.
   b. 허용된 프로토콜에서 필요에 따라 XA 활성화 및 SNA LU 활성화를 선택합니다.
8. 예약에서 옵션을 즉시 추가할지 또는 다음 유지 관리 기간에 추가할지를 선택합니다.
9. 옵션 추가를 선택합니다.

   이 옵션을 추가하는 데 재부팅은 필요하지 않습니다.

CLI

MSDTC 옵션을 추가하려면

1. 다음 필수 파라미터를 사용하여 JSON 파일(예: `msdtc-option.json`)을 생성합니다.

```
{
```

```
"OptionGroupName":"msdtc-se-2016",
"OptionsToInclude": [
 {
 "OptionName":"MSDTC",
 "Port":5000,
 "VpcSecurityGroupMemberships":["sg-0abcdef123"],
 "OptionSettings":[{"Name":"AUTHENTICATION","Value":"MUTUAL"},
{"Name":"TRANSACTION_LOG_SIZE","Value":"4"}]
 }],
"ApplyImmediately": true
}
```

2. [MSDTC] 옵션을 옵션 그룹에 추가합니다.

   Example

   대상 LinuxmacOS, 또는Unix:

   ```
   aws rds add-option-to-option-group \
       --cli-input-json file://msdtc-option.json \
       --apply-immediately
   ```

   Windows의 경우:

   ```
   aws rds add-option-to-option-group ^
       --cli-input-json file://msdtc-option.json ^
       --apply-immediately
   ```

   재부팅이 필요하지 않습니다.

MSDTC용 파라미터 그룹 생성

DB 인스턴스의 SQL Server 에디션 및 버전에 해당하는 `in-doubt xact resolution` 파라미터의 파라미터 그룹을 생성하거나 수정합니다.

콘솔

다음 예에서는 SQL Server Standard Edition 2016에 대한 파라미터 그룹을 생성합니다.

파라미터 그룹을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.
3. [Create parameter group]을 선택합니다.
4. 서브넷 그룹 생성 창에서 다음을 수행합니다.
   a. 파라미터 그룹 패밀리에서 sqlserver-se-13.0을 선택합니다.
   b. 그룹 이름에 파라미터 그룹의 식별자(예: **msdtc-sqlserver-se-13**)를 입력합니다.
   c. 설명에 **in-doubt xact resolution**를 입력합니다.
5. 생성(Create)을 선택합니다.

CLI

다음 예에서는 SQL Server Standard Edition 2016에 대한 파라미터 그룹을 생성합니다.

파라미터 그룹을 생성하려면

- 다음 명령 중 하나를 사용합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-parameter-group \
    --db-parameter-group-name msdtc-sqlserver-se-13 \
    --db-parameter-group-family "sqlserver-se-13.0" \
    --description "in-doubt xact resolution"
```

  Windows의 경우:

```
aws rds create-db-parameter-group ^
    --db-parameter-group-name msdtc-sqlserver-se-13 ^
    --db-parameter-group-family "sqlserver-se-13.0" ^
    --description "in-doubt xact resolution"
```

MSDTC에 대한 파라미터 수정

SQL Server 에디션 및 DB 인스턴스의 버전에 해당하는 파라미터 그룹의 `in-doubt xact resolution` 파라미터를 수정합니다.

MSDTC에 대해 `in-doubt xact resolution` 파라미터를 다음 중 하나로 설정합니다.

- `1 - Presume commit` 모든 MSDTC 미결 트랜잭션을 커밋된 것으로 가정합니다.
- `2 - Presume abort` 모든 MSDTC 미결 트랜잭션을 중지된 것으로 가정합니다.

자세한 내용은 Microsoft 설명서의 in-doubt xact resolution 서버 구성 옵션을 참조하세요.

콘솔

다음 예에서는 SQL Server Standard Edition 2016에 대해 생성한 파라미터 그룹을 수정합니다.

파라미터 그룹을 수정하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 파라미터 그룹을 선택합니다.
3. 파라미터 그룹(예: msdtc-sqlserver-se-13)을 선택합니다.
4. 파라미터에서 파라미터 목록을 **`xact`**로 필터링합니다.
5. in-doubt xact resolution을 선택합니다.
6. 파라미터 편집을 선택합니다.
7. **1** 또는 **2**를 입력합니다.
8. Save changes(변경 사항 저장)를 선택합니다.

CLI

다음 예에서는 SQL Server Standard Edition 2016에 대해 생성한 파라미터 그룹을 수정합니다.

파라미터 그룹을 수정하려면

- 다음 명령 중 하나를 사용합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name msdtc-sqlserver-se-13 \
    --parameters "ParameterName='in-doubt xact
 resolution',ParameterValue=1,ApplyMethod=immediate"
```

Windows의 경우:

```
aws rds modify-db-parameter-group ^
    --db-parameter-group-name msdtc-sqlserver-se-13 ^
    --parameters "ParameterName='in-doubt xact
 resolution',ParameterValue=1,ApplyMethod=immediate"
```

옵션 그룹 및 파라미터 그룹을 DB 인스턴스와 연결

AWS Management Console 또는 AWS CLI를 사용하여 MSDTC 옵션 그룹 및 파라미터 그룹을 DB 인스턴스와 연결할 수 있습니다.

콘솔

MSDTC 옵션 그룹 및 파라미터 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스와 연결할 수 있습니다.

- 새 DB 인스턴스의 경우 인스턴스를 시작할 때 이러한 그룹을 연결합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
- 기존 DB 인스턴스의 경우 인스턴스를 수정하여 그룹을 연결합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

> **Note**
>
> 도메인에 가입된 기존 DB 인스턴스를 사용하는 경우 이미 Active Directory 도메인과 AWS Identity and Access Management(IAM) 역할이 연결되어 있어야 합니다. 도메인에 가입된 새 인스턴스를 생성하는 경우 기존 Active Directory 도메인 및 IAM 역할을 지정합니다. 자세한 내용은 RDS for SQL Server를 사용하여 AWS 관리형 Active Directory 작업 단원을 참조하세요.

CLI

MSDTC 옵션 그룹 및 파라미터 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스와 연결할 수 있습니다.

> **Note**
>
> 도메인에 가입된 기존 DB 인스턴스를 사용하는 경우 이미 Active Directory 도메인과 IAM 역할이 연결되어 있어야 합니다. 도메인에 가입된 새 인스턴스를 생성하는 경우 기존 Active Directory 도메인 및 IAM 역할을 지정합니다. 자세한 내용은 RDS for SQL Server를 사용하여 AWS 관리형 Active Directory 작업 섹션을 참조하세요.

MSDTC 옵션 그룹 및 파라미터 그룹과 함께 DB 인스턴스를 생성하려면

- 옵션 그룹을 생성할 때 사용한 것과 동일한 DB 엔진 유형과 메이저 버전을 지정합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --db-instance-class db.m5.2xlarge \
    --engine sqlserver-se \
    --engine-version 13.00.5426.0.v1 \
    --allocated-storage 100 \
    --manage-master-user-password \
    --master-username admin \
    --storage-type gp2 \
    --license-model li \
    --domain-iam-role-name my-directory-iam-role \
    --domain my-domain-id \
    --option-group-name msdtc-se-2016 \
    --db-parameter-group-name msdtc-sqlserver-se-13
```

  Windows의 경우:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --db-instance-class db.m5.2xlarge ^
    --engine sqlserver-se ^
    --engine-version 13.00.5426.0.v1 ^
    --allocated-storage 100 ^
    --manage-master-user-password ^
    --master-username admin ^
```

```
    --storage-type gp2 ^
    --license-model li ^
    --domain-iam-role-name my-directory-iam-role ^
    --domain my-domain-id ^
    --option-group-name msdtc-se-2016 ^
    --db-parameter-group-name msdtc-sqlserver-se-13
```

DB 인스턴스를 수정하고 MSDTC 옵션 그룹 및 파라미터 그룹을 연결하려면

- 다음 명령 중 하나를 사용합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --option-group-name msdtc-se-2016 \
    --db-parameter-group-name msdtc-sqlserver-se-13 \
    --apply-immediately
```

  Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --option-group-name msdtc-se-2016 ^
    --db-parameter-group-name msdtc-sqlserver-se-13 ^
    --apply-immediately
```

MSDTC 옵션 수정

MSDTC 옵션을 활성화한 후 해당 설정을 수정할 수 있습니다. 옵션 설정을 변경하는 방법에 대한 자세한 내용은 옵션 설정 수정 단원을 참조하십시오.

> **Note**
>
> MSDTC 옵션 설정을 일부 변경하려면 MSDTC 서비스를 다시 시작해야 합니다. 이 요구 사항은 분산 트랜잭션 실행에 영향을 줄 수 있습니다.

트랜잭션 사용

분산 트랜잭션 사용

Amazon RDS for SQL Server에서는 온프레미스에서 실행되는 분산 트랜잭션과 동일한 방식으로 분산 트랜잭션을 실행합니다.

- .NET Framework `System.Transactions` 승격 가능한 트랜잭션 사용. 이를 통해 필요할 때까지 생성을 연기함으로써 분산 트랜잭션을 최적화합니다.

  이 경우 승격이 자동으로 수행되므로 사용자가 개입할 필요가 없습니다. 트랜잭션 내에 하나의 리소스 관리자만 있는 경우 승격이 수행되지 않습니다. 암시적 트랜잭션 범위에 대한 자세한 내용은 Microsoft 설명서의 트랜잭션 범위를 사용하여 암시적 트랜잭션 구현을 참조하십시오.

  승격 가능 트랜잭션은 다음 .NET 구현에서 지원됩니다.

  - ADO.NET 2.0부터 `System.Data.SqlClient`는 SQL Server를 사용하여 승격 가능한 트랜잭션을 지원합니다. 자세한 내용은 Microsoft 설명서에서 SQL Server와의 System.Transactions 통합을 참조하십시오.
  - ODP.NET은 `System.Transactions`를 지원합니다. Oracle Database 11g 릴리스 1(버전 11.1) 이상에서 `TransactionsScope` 범위에서 열린 첫 번째 연결에 대해 로컬 트랜잭션이 생성됩니다. 두 번째 연결이 열리면 이 트랜잭션이 분산 트랜잭션으로 자동 승격됩니다. ODP.NET에서 분산 트랜잭션 지원에 대한 자세한 내용은 Microsoft 설명서의 Microsoft Distributed Transaction Coordinator 통합을 참조하십시오.
- `BEGIN DISTRIBUTED TRANSACTION` 문 사용. 자세한 내용은 Microsoft 설명서의 BEGIN DISTRIBUTED TRANSACTION(Transact-SQL)을 참조하세요.

XA 트랜잭션 사용

RDS for SQL Server 2017 버전 14.00.3223.3부터 JDBC를 사용하여 분산 트랜잭션을 제어할 수 있습니다. `Enable XA` 옵션 설정을 `true` 옵션의 `MSDTC`(으)로 설정하면 RDS가 자동으로 JDBC 트랜잭션을 활성화하고 `SqlJDBCXAUser` 역할을 `guest` 사용자에게 부여합니다. 이를 통해 JDBC를 통해 분산 트랜잭션을 실행할 수 있습니다. 코드 예제를 포함한 자세한 정보는 Microsoft 설명서의 XA 트랜잭션 이해를 참조하세요.

트랜잭션 추적 사용

RDS는 문제 해결을 위해 MSDTC 트랜잭션 추적을 제어하고 RDS DB 인스턴스에서 해당 추적을 다운로드할 수 있도록 지원합니다. 다음 RDS 저장 프로시저를 실행하여 트랜잭션 추적 세션을 제어할 수 있습니다.

```
exec msdb.dbo.rds_msdtc_transaction_tracing 'trace_action',
[@traceall='0|1'],
[@traceaborted='0|1'],
[@tracelong='0|1'];
```

다음 파라미터는 필수입니다.

- `trace_action` – 추적 작업입니다. START, STOP 또는 STATUS일 수 있습니다.

다음 파라미터는 선택적입니다.

- `@traceall` – 모든 분산 트랜잭션을 추적하려면 1로 설정합니다. 기본값은 0입니다.
- `@traceaborted` – 취소된 분산 트랜잭션을 추적하려면 1로 설정합니다. 기본값은 0입니다.
- `@tracelong` – 장기 실행 분산 트랜잭션을 추적하려면 1로 설정합니다. 기본값은 0입니다.

Example START 추적 작업

새 트랜잭션 추적 세션을 시작하려면 다음 예에 나온 문을 실행합니다.

```
exec msdb.dbo.rds_msdtc_transaction_tracing 'START',
@traceall='0',
@traceaborted='1',
@tracelong='1';
```

> **Note**
>
> 한 번에 하나의 트랜잭션 추적 세션만 활성화할 수 있습니다. 추적 세션이 활성 상태일 때 새 추적 세션 START 명령이 실행되면 오류가 반환되고 활성 추적 세션이 변경되지 않습니다.

Example STOP 추적 작업

트랜잭션 추적 세션을 중지하려면 다음 문을 실행합니다.

```
exec msdb.dbo.rds_msdtc_transaction_tracing 'STOP'
```

이 문은 활성 트랜잭션 추적 세션을 중지하고 트랜잭션 추적 데이터를 RDS DB 인스턴스의 로그 디렉터리에 저장합니다. 출력의 첫 번째 행에는 전체 결과가 포함되며 다음 줄은 작업의 세부 정보를 나타냅니다.

다음은 성공적인 추적 세션 중지의 예입니다.

```
OK: Trace session has been successfully stopped.
Setting log file to: D:\rdsdbdata\MSDTC\Trace\dtctrace.log
Examining D:\rdsdbdata\MSDTC\Trace\msdtctr.mof for message formats,  8 found.
Searching for TMF files on path: (null)
Logfile D:\rdsdbdata\MSDTC\Trace\dtctrace.log:
 OS version    10.0.14393  (Currently running on 6.2.9200)
 Start Time    <timestamp>
 End Time      <timestamp>
 Timezone is   @tzres.dll,-932 (Bias is 0mins)
 BufferSize            16384 B
 Maximum File Size     10 MB
 Buffers  Written      Not set (Logger may not have been stopped).
 Logger Mode Settings (11000002) ( circular paged
 ProcessorCount         1
Processing completed   Buffers: 1, Events: 3, EventsLost: 0 :: Format Errors: 0,
 Unknowns: 3
Event traces dumped to d:\rdsdbdata\Log\msdtc_<timestamp>.log
```

세부 정보를 사용하여 생성된 로그 파일의 이름을 쿼리할 수 있습니다. RDS DB 인스턴스에서 로그 파일을 다운로드하는 방법에 대한 자세한 내용은 Amazon RDS 로그 파일 모니터링 단원을 참조하십시오.

추적 세션 로그는 35일 동안 인스턴스에 남아 있습니다. 이전 추적 세션 로그는 자동으로 삭제됩니다.

Example STATUS 추적 작업

트랜잭션 추적 세션의 상태를 추적하려면 다음 문을 실행합니다.

```
exec msdb.dbo.rds_msdtc_transaction_tracing 'STATUS'
```

이 문은 결과 집합의 별도 행으로 다음을 출력합니다.

```
OK
SessionStatus: <Started|Stopped>
TraceAll: <True|False>
TraceAborted: <True|False>
```

```
TraceLongLived: <True|False>
```

첫 번째 줄은 작업의 전체 결과(OK 또는 ERROR)와 함께 세부 정보(해당되는 경우)를 나타냅니다. 그 다음 줄에는 추적 세션 상태에 대한 세부 정보가 표시됩니다.

- SessionStatus의 값은 다음 중 하나일 수 있습니다.
  - Started - 추적 세션이 실행 중인 경우.
  - Stopped - 실행 중인 추적 세션이 없는 경우.
- 추적 세션 플래그는 True 명령에서 설정된 방법에 따라 False 또는 START일 수 있습니다.

MSDTC 비활성화

MSDTC를 비활성화하려면 해당 옵션 그룹에서 MSDTC 옵션을 제거합니다.

콘솔

옵션 그룹에서 MSDTC 옵션을 제거하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. MSDTC 옵션이 있는 옵션 그룹을 선택합니다(이전 예제의 경우 msdtc-se-2016).
4. 옵션 삭제를 선택합니다.
5. 옵션 삭제에서 MSDTC 또는 삭제할 옵션을 선택합니다.
6. 즉시 적용에서 옵션을 즉시 삭제하려면 예를 선택하고 다음 유지 관리 기간에 삭제하려면 아니오를 선택합니다.
7. Delete(삭제)를 선택합니다.

CLI

옵션 그룹에서 MSDTC 옵션을 제거하려면

- 다음 명령 중 하나를 사용합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds remove-option-from-option-group \
    --option-group-name msdtc-se-2016 \
    --options MSDTC \
    --apply-immediately
```

Windows의 경우:

```
aws rds remove-option-from-option-group ^
    --option-group-name msdtc-se-2016 ^
    --options MSDTC ^
    --apply-immediately
```

RDS for SQL Server에 대한 MSDTC 문제 해결

경우에 따라 클라이언트 컴퓨터에서 실행되는 MSDTC와 RDS for SQL Server DB 인스턴스에서 실행되는 MSDTC 서비스 간에 연결을 설정하는 데 문제가 있을 수 있습니다. 그 경우 다음을 확인하십시오.

- DB 인스턴스와 연결된 보안 그룹의 인바운드 규칙이 올바르게 구성되어 있는지. 자세한 내용은 Amazon RDS DB 인스턴스에 연결할 수 없음 섹션을 참조하세요.
- 클라이언트 컴퓨터가 올바르게 구성되어 있는지.
- 클라이언트 컴퓨터에서 MSDTC 방화벽 규칙이 활성화되어 있는지.

클라이언트 컴퓨터를 구성하려면

1. 구성 요소 서비스를 엽니다.

   또는 서버 관리자에서 도구를 선택한 다음 구성 요소 서비스를 선택합니다.
2. 구성 요소 서비스, 컴퓨터, 내 컴퓨터, Distributed Transaction Coordinator를 차례로 확장합니다.
3. 로컬 DTC에 대한 컨텍스트 메뉴를 열고(마우스 오른쪽 버튼 클릭) 속성을 선택합니다.
4. 보안 탭을 선택합니다.
5. 다음을 모두 선택합니다.
   - 네트워크 DTC 액세스
   - 인바운드 허용
   - 아웃바운드 허용

6. 올바른 인증 모드가 선택되어 있는지 확인합니다.

   - 상호 인증 필요 – 클라이언트 시스템이 분산 트랜잭션에 참여하는 다른 노드와 동일한 도메인에 가입되었거나 도메인 간에 신뢰 관계가 구성되어 있습니다.
   - 인증 필요 없음 – 다른 모든 경우.
7. 확인을 선택하여 변경 사항을 저장합니다.
8. 서비스를 다시 시작하라는 메시지가 표시되면 예를 선택합니다.

MSDTC 방화벽 규칙을 활성화하려면

1. Windows 방화벽을 연 다음 고급 설정을 선택합니다.

   서버 관리자를 열고 도구, Windows Firewall with Advanced Security를 차례로 선택합니다.

   > **Note**
   >
   > 운영 체제에 따라 Windows 방화벽을 Windows Defender 방화벽이라고 할 수도 있습니다.
2. 왼쪽 창에서 인바운드 규칙을 선택합니다.
3. 다음 방화벽 규칙을 활성화합니다(아직 활성화되지 않은 경우).

   - DTC(Distributed Transaction Coordinator)(RPC)
   - DTC(Distributed Transaction Coordinator)(RPC)-EPMAP
   - DTC(Distributed Transaction Coordinator)(TCP-In)
4. Windows 방화벽을 닫습니다.

Amazon RDS for Microsoft SQL Server에 대한 일반 DBA 작업

이 단원에서는 Microsoft SQL Server 데이터베이스 엔진을 실행하는 DB 인스턴스의 몇 가지 일반적인 DBA 작업을 Amazon RDS에 따라 구현하는 방법에 대해 살펴봅니다. 관리되는 서비스 환경을 제공하기 위해 Amazon RDS는 DB 인스턴스에 대해 셸 액세스를 제공하지 않으며, 고급 권한을 필요로 하는 특정 시스템 절차와 테이블에 대한 액세스를 제한합니다.

> **Note**
>
> SQL Server DB 인스턴스를 이용한 작업에서는 스크립트를 실행하여 새롭게 생성한 데이터베이스를 수정할 수는 있지만 새로운 데이터베이스 모델로 사용되는 데이터베이스인 [모델] 데이터베이스는 수정할 수 없습니다.

주제

- Amazon RDS에서 실행되는 Microsoft SQL Server DB 인스턴스의 tempdb 데이터베이스에 액세스
- 데이터베이스 엔진 튜닝 관리자를 사용하여 Amazon RDS for SQL Server DB 인스턴스의 데이터베이스 워크로드 분석
- Amazon RDS for SQL Server 데이터베이스 db_owner를 rdsa 계정으로 변경
- Amazon RDS for Microsoft SQL Server의 데이터 정렬 및 문자 세트 관리
- Amazon RDS for SQL Server용 데이터베이스 사용자 생성
- Amazon RDS for SQL Server 데이터베이스에 적용할 복구 모델 결정
- Amazon RDS for SQL Server의 마지막 장애 조치 시간 결정
- 로그 시퀀스 번호 차이로 인한 시점 복구 실패 문제 해결
- Amazon RDS for SQL Server의 데이터베이스 이름 보기 거부 또는 허용
- Amazon RDS for SQL Server에 대한 대량 로드 중 빠른 삽입 비활성화
- Amazon RDS for Microsoft SQL Server DB 인스턴스 데이터베이스 삭제
- 다중 AZ 배포의 Amazon RDS for Microsoft SQL Server 데이터베이스 이름 변경
- Amazon RDS for SQL Server 마스터 사용자의 db_owner 역할 멤버십 재설정
- Amazon RDS for SQL Server에 대한 라이선스 종료 DB 인스턴스 복원
- 오프라인에서 온라인으로 Amazon RDS for SQL Server 데이터베이스 전환
- Amazon RDS for SQL Server에 변경 데이터 캡처 사용
- Amazon RDS에 SQL Server Agent 사용

- Amazon RDS for Microsoft SQL Server 로그 작업
- Amazon RDS for SQL Server에 대한 추적 및 덤프 파일 작업

Amazon RDS에서 실행되는 Microsoft SQL Server DB 인스턴스의 tempdb 데이터베이스에 액세스

Amazon RDS에서 실행되는 Microsoft SQL Server DB 인스턴스의 tempdb 데이터베이스에 액세스할 수 있습니다. Microsoft SQL Server Management Studio(SSMS)를 통한 Transact-SQL 또는 기타 표준 SQL 클라이언트 애플리케이션을 사용하여 tempdb에서 코드를 실행할 수 있습니다. DB 인스턴스 연결에 대한 자세한 내용은 Microsoft SQL Server DB 인스턴스에 연결 단원을 참조하십시오.

DB 인스턴스의 마스터 사용자는 CONTROL 데이터베이스 옵션을 수정할 수 있도록 tempdb에 대한 tempdb 액세스 권한을 부여받습니다. 마스터 사용자는 tempdb 데이터베이스의 데이터베이스 소유자가 아닙니다. 필요한 경우 마스터 사용자는 다른 사용자가 CONTROL 데이터베이스 옵션을 수정할 수 있도록 이들에게 tempdb 액세스 권한을 부여할 수 있습니다.

> **Note**
>
> tempdb 데이터베이스에서 Database Console Commands(DBCC)를 실행할 수 없습니다.

tempdb 데이터베이스 옵션 수정

Amazon RDS DB 인스턴스에 있는 tempdb 데이터베이스의 데이터베이스 옵션을 수정할 수 있습니다. 수정할 수 있는 옵션에 대한 자세한 내용은 Microsoft 설명서의 tempdb 데이터베이스를 참조하십시오.

최대 파일 크기 옵션과 같은 데이터베이스 옵션은 DB 인스턴스를 다시 시작한 후에도 유지됩니다. 데이터를 가져올 때 성능을 최적화하고 스토리지 부족을 방지하도록 데이터베이스 옵션을 수정할 수 있습니다.

데이터를 가져올 때 성능 최적화

대량의 데이터를 DB 인스턴스로 가져올 때 성능을 최적화하려면 tempdb 데이터베이스의 SIZE 및 FILEGROWTH 속성을 큰 수로 설정합니다. tempdb를 최적화하는 방법에 대한 자세한 내용은 Microsoft 설명서의 tempdb 성능 최적화를 참조하십시오.

다음은 크기를 100GB로, 파일 증가를 10%로 설정한 예입니다.

```
alter database[tempdb] modify file (NAME = N'templog', SIZE=100GB, FILEGROWTH = 10%)
```

스토리지 문제 방지

tempdb 데이터베이스에서 사용 가능한 모든 디스크 공간을 사용하는 것을 방지하려면 MAXSIZE 속성을 설정합니다. 다음은 속성을 2048MB로 설정한 예입니다.

```
alter database [tempdb] modify file (NAME = N'templog', MAXSIZE = 2048MB)
```

tempdb 데이터베이스 축소

Amazon RDS DB 인스턴스의 tempdb 데이터베이스는 두 가지 방법으로 축소할 수 있습니다. rds_shrink_tempdbfile 프로시저를 사용하거나, SIZE 속성을 설정하면 됩니다.

rds_shrink_tempdbfile 프로시저 사용

Amazon RDS 프로시저 msdb.dbo.rds_shrink_tempdbfile를 사용하여 tempdb 데이터베이스를 축소할 수 있습니다. rds_shrink_tempdbfile에 대한 CONTROL 액세스 권한이 있는 경우에만 tempdb을 호출할 수 있습니다. rds_shrink_tempdbfile을 호출해도 DB 인스턴스에 가동 중지가 발생하지 않습니다.

rds_shrink_tempdbfile 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
@temp_filename	SYSNAME	—	필수	축소할 파일의 논리적 이름입니다.
@target_size	int	null	선택 사항	파일의 새로운 크기(MB)입니다.

다음 예제에서는 tempdb 데이터베이스의 파일 이름을 가져옵니다.

```
use tempdb;
GO

select name, * from sys.sysfiles;
GO
```

다음 예제에서는 tempdb이라는 test_file 데이터베이스 파일을 축소하고 새로운 10MB 크기를 요청합니다.

```
exec msdb.dbo.rds_shrink_tempdbfile @temp_filename = N'test_file', @target_size = 10;
```

SIZE 속성 설정

tempdb 속성을 설정한 다음 DB 인스턴스를 다시 시작하여 SIZE 데이터베이스를 축소할 수도 있습니다. DB 인스턴스를 다시 시작하는 방법에 대한 자세한 내용은 DB 인스턴스 재부팅 단원을 참조하십시오.

다음은 SIZE 속성을 1024MB로 설정한 예입니다.

```
alter database [tempdb] modify file (NAME = N'templog', SIZE = 1024MB)
```

다중 AZ 배포에 대한 TempDB 구성

RDS for SQL Server DB 인스턴스가 데이터베이스 미러링(DBM) 또는 상시 작동 가용성 그룹(AG)을 사용하는 다중 AZ 배포에 있는 경우, tempdb 데이터베이스 사용에 대한 다음 고려 사항을 유념하세요.

기본 DB 인스턴스에서 보조 DB 인스턴스로 tempdb 데이터를 복제할 수 없습니다. 보조 DB 인스턴스로 장애 조치하면 보조 DB 인스턴스에 있는 tempdb는 비어 있게 됩니다.

파일 크기 조정 및 자동 증가 설정을 비롯한 tempdb 데이터베이스 옵션의 구성을 기본 DB 인스턴스에서 보조 DB 인스턴스로 동기화할 수 있습니다. tempDB 구성 동기화는 모든 RDS for SQL Server 버전에서 지원됩니다. 다음 저장 프로시저를 사용하여 tempdb 구성의 자동 동기화를 켤 수 있습니다.

```
EXECUTE msdb.dbo.rds_set_system_database_sync_objects @object_types = 'TempDbFile';
```

> **Important**
>
> rds_set_system_database_sync_objects 저장 프로시저를 사용하기 전에 보조 DB 인스턴스가 아닌 기본 DB 인스턴스에서 기본 tempdb 구성을 설정했는지 확인하세요. 보조 DB 인스턴스에서 구성을 변경한 경우 자동 동기화를 켜면 기본 tempdb 구성이 삭제될 수 있습니다.

다음 기능을 사용하면 tempdb 구성 자동 동기화가 켜져 있는지 확인할 수 있습니다.

```
SELECT * from msdb.dbo.rds_fn_get_system_database_sync_objects();
```

tempdb 구성 자동 동기화가 켜져 있는 경우 object_class 필드에 반환 값이 표시됩니다. 끄면 값이 반환되지 않습니다.

다음 함수를 사용하여 객체가 마지막으로 동기화된 시간(UTC 시간)을 찾을 수 있습니다.

```
SELECT * from msdb.dbo.rds_fn_server_object_last_sync_time();
```

예를 들어, 01:00에 tempdb 구성을 수정한 다음 rds_fn_server_object_last_sync_time 함수를 실행하는 경우 에서 반환되는 값은 01:00 last_sync_time 이후여야 하며, 이는 자동 동기화가 발생했음을 나타냅니다.

SQL Server 에이전트 작업 복제를 함께 사용하는 경우 @object_type 파라미터에 SQL 에이전트 작업과 tempdb 구성을 제공하여 둘 모두에 대해 복제를 활성화할 수 있습니다.

```
EXECUTE msdb.dbo.rds_set_system_database_sync_objects @object_types =
 'SQLAgentJob,TempDbFile';
```

SQL Server 에이전트 작업 복제에 대한 자세한 내용은 SQL Server 에이전트 작업 복제 켜기 섹션을 참조하세요.

tempdb 구성 변경 내용이 자동으로 동기화되도록 rds_set_system_database_sync_objects 저장 프로시저를 사용하는 대신 다음과 같은 수동 방법 중 하나를 사용할 수 있습니다.

> **Note**
>
> rds_set_system_database_sync_objects 저장 프로시저를 사용하여 tempdb 구성의 자동 동기화를 켤 수 있습니다. 자동 동기화를 사용하면 tempdb 구성을 변경할 때마다 이러한 수동 작업을 수행할 필요가 없습니다.

- 먼저 DB 인스턴스를 수정하고 다중 AZ를 비활성화한 다음 tempdb를 수정하고 마지막으로 다중 AZ를 다시 활성화합니다. 이 방법을 사용하는 경우 가동 중지가 없습니다.

  자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.
- 먼저 원래의 기본 인스턴스에서 tempdb를 수정한 다음, 수동으로 장애 조치를 수행하고, 마지막으로 새 기본 인스턴스에서 tempdb를 수정합니다. 이 방법을 사용하는 경우 가동 중지가 있습니다.

  자세한 내용은 DB 인스턴스 재부팅 단원을 참조하십시오.

데이터베이스 엔진 튜닝 관리자를 사용하여 Amazon RDS for SQL Server DB 인스턴스의 데이터베이스 워크로드 분석

데이터베이스 엔진 튜닝 관리자는 Microsoft에서 제공하는 클라이언트 애플리케이션으로, 데이터베이스 워크로드를 분석하고 실행하는 쿼리 종류에 따라 Microsoft SQL Server 데이터베이스에 대한 최적의 인덱스 집합을 권장합니다. SQL Server Management Studio와 마찬가지로 튜닝 어드바이저 역시 SQL Server 기반 Amazon RDS DB 인스턴스에 연결되어 있는 클라이언트 컴퓨터에서 실행됩니다. 여기서 클라이언트 컴퓨터는 기업 네트워크 내 온프레미스에서 실행되는 로컬 컴퓨터가 될 수도 있고, 혹은 Amazon RDS DB 인스턴스와 동일한 리전에서 실행되는 Amazon EC2 Windows 인스턴스가 될 수도 있습니다.

이 섹션에서는 튜닝 어드바이저의 분석 워크로드를 캡처하는 방법에 대해 살펴보겠습니다. Amazon RDS는 SQL Server 인스턴스에 대한 호스트 액세스를 제한하기 때문에 이 방법은 워크로드 캡처에 우선적으로 사용되는 프로세스입니다. 자세한 내용은 Microsoft 설명서의 데이터베이스 엔진 튜닝 관리자를 참조하세요.

튜닝 어드바이저를 사용하려면 먼저 워크로드를 어드바이저에게 제공해야 합니다. 여기서 워크로드란 튜닝하려는 데이터베이스에 실행하는 Transact-SQL 문의 집합을 말합니다. 데이터베이스 엔진 튜닝 어드바이저는 데이터베이스를 튜닝할 때 트레이스 파일, 트레이스 테이블, Transact-SQL 스크립트 또는 XML 파일을 워크로드 입력 수단으로 사용합니다. Amazon RDS에서는 클라이언트 컴퓨터의 파일이나 클라이언트 컴퓨터에 액세스할 수 있는 Amazon RDS for SQL Server DB의 데이터베이스 테이블을 워크로드로 사용할 수도 있습니다. 하지만 파일이든, 테이블이든 상관없이 튜닝하려는 데이터베이스에 대한 쿼리는 재실행에 적합한 형식에 따라 저장되어야 합니다.

튜닝 어드바이저의 효과를 극대화하려면 워크로드가 최대한 사실적이어야 합니다. 워크로드 파일이나 테이블은 DB 인스턴스에 대한 트레이스를 수행하여 생성할 수 있습니다. 트레이스가 실행 중일 때도 DB 인스턴스에 가해지는 부하를 시뮬레이션하거나 정상적인 부하로 애플리케이션을 실행할 수 있습니다.

트레이스는 클라이언트 측과 서버 측, 두 가지 유형이 있습니다. 클라이언트 측 트레이스는 설치가 더욱 쉬울 뿐만 아니라 트레이스 이벤트가 캡처되는 것을 SQL Server 프로파일러에서 실시간으로 볼 수 있습니다. 반면 서버 측 트레이스는 설치가 더욱 복잡할 뿐만 아니라 몇 가지 Transact-SQL 스크립트를 작성해야 합니다. 그뿐만 아니라 트레이스가 Amazon RDS DB 인스턴스의 파일에 기록되기 때문에 스토리지 공간을 차지하게 됩니다. DB 인스턴스는 스토리지가 가득 찬 상태가 되어 스토리지 공간이 부족할 경우 더 이상 사용할 수 없기 때문에 실행 중인 서버 측 트레이스가 사용하는 스토리지 공간을 계속해서 추적해야 합니다.

클라이언트 측 트레이스의 경우, SQL Server 프로파일러에 트레이스 데이터가 충분히 캡처되면 트레이스를 로컬 컴퓨터의 파일이나 클라이언트 컴퓨터에 액세스할 수 있는 DB 인스턴스의 데이터베이스

테이블에 저장하여 워크로드 파일을 생성할 수 있습니다. 하지만 클라이언트 측 트레이스를 사용할 때는 과도한 부하 시 트레이스가 쿼리를 모두 캡처하지 못한다는 커다란 단점이 있습니다. 이로 인해 데이터베이스 엔진 튜닝 어드바이저의 분석 효과가 약해질 수 있습니다. 따라서 과도한 부하에서 트레이스를 실행하면서 트레이스 세션 중 모든 쿼리를 캡처해야 하는 경우에는 서버 측 트레이스를 사용하는 것이 바람직합니다.

서버 측 트레이스에서는 DB 인스턴스의 트레이스 파일을 적합한 워크로드 파일로 가져오거나, 완료된 트레이스를 DB 인스턴스의 테이블에 저장할 수 있습니다. 또한 SQL Server 프로파일러를 사용해 트레이스를 로컬 컴퓨터의 파일에 저장하거나 튜닝 어드바이저를 사용해 DB 인스턴스의 트레이스 테이블을 읽을 수 있습니다.

SQL Server DB 인스턴스의 클라이언트 측 트레이스 실행

SQL Server DB 인스턴스의 클라이언트 측 트레이스를 실행하는 방법

1. SQL Server 프로파일러를 시작합니다. 이 프로파일러는 SQL Server 인스턴스 폴더 내 Performance Tools 폴더에 설치되어 있습니다. 클라이언트 측 트레이스를 시작하려면 트레이스 정의 템플릿을 로드하거나 정의해야 합니다.
2. SQL Server 프로파일러 파일 메뉴에서 New Trace(새 트레이스)를 선택합니다. [Connect to Server] 대화 상자에 DB 인스턴스 엔드포인트, 포트, 마스터 사용자 이름 그리고 트레이스를 실행할 데이터베이스의 암호를 입력합니다.
3. [Trace Properties] 대화 상자에 트레이스 이름을 입력한 다음 트레이스 정의 템플릿을 선택합니다. 기본 템플릿인 TSQL_Replay는 애플리케이션으로 제공됩니다. 이 템플릿을 편집하여 트레이스를 정의할 수 있습니다. [Trace Properties] 대화 상자의 [Events Selection] 탭 아래 있는 이벤트와 이벤트 정보를 편집합니다.

   추적 정의 템플릿 및 SQL Server 프로파일러를 사용하여 클라이언트 측 추적을 지정하는 방법에 대한 자세한 내용은 Microsoft 설명서의 데이터베이스 엔진 튜닝 관리자를 참조하세요.
4. 클라이언트 측 트레이스를 시작하여 DB 인스턴스에 대한 SQL 쿼리 실행을 실시간으로 모니터링합니다.
5. 트레이스가 완료되면 파일 메뉴에서 Stop Trace(트레이스 중지)를 선택합니다. 결과를 파일이나 DB 인스턴스의 트레이스 테이블로 저장합니다.

SQL Server DB 인스턴스의 서버 측 트레이스 실행

스크립트를 작성하여 서버 측 트레이스를 생성하는 것은 복잡할 뿐만 아니라 이 문서의 범위에서 벗어납니다. 따라서 여기서는 예제로 사용할 수 있는 샘플 스크립트만 다룹니다. 클라이언트 측 트레이스와

마찬가지로 이 트레이스의 목적 역시 데이터베이스 엔진 튜닝 어드바이저를 사용해 열 수 있도록 워크로드 파일 또는 트레이스를 생성하는 데 있습니다.

다음은 서버 측 트레이스를 시작하여 세부 정보를 워크 로드에 캡처하기 위해 요약된 예제 스크립트입니다. 트레이스가 처음에는 D:\RDSDBDATA\Log 디렉터리의 RDSTrace.trc 파일에 저장되지만 100MB마다 롤오버되어 이후부터는 RDSTrace_1.trc, RDSTrace_2.trc 등의 이름으로 트레이스 파일이 저장됩니다.

```
DECLARE @file_name NVARCHAR(245) = 'D:\RDSDBDATA\Log\RDSTrace';
DECLARE @max_file_size BIGINT = 100;
DECLARE @on BIT = 1
DECLARE @rc INT
DECLARE @traceid INT

EXEC @rc = sp_trace_create @traceid OUTPUT, 2, @file_name, @max_file_size
IF (@rc = 0) BEGIN
   EXEC sp_trace_setevent @traceid, 10, 1, @on
   EXEC sp_trace_setevent @traceid, 10, 2, @on
   EXEC sp_trace_setevent @traceid, 10, 3, @on
 . . .
   EXEC sp_trace_setfilter @traceid, 10, 0, 7, N'SQL Profiler'
   EXEC sp_trace_setstatus @traceid, 1
   END
```

다음은 트레이스를 중단하는 스크립트 예제입니다. 단, 이전 스크립트에서 생성된 트레이스는 명시적으로 트레이스가 중단되거나, 프로세스의 디스크 공간이 부족해질 때까지 계속 실행됩니다.

```
DECLARE @traceid INT
SELECT @traceid = traceid FROM ::fn_trace_getinfo(default)
WHERE property = 5 AND value = 1 AND traceid <> 1

IF @traceid IS NOT NULL BEGIN
   EXEC sp_trace_setstatus @traceid, 0
   EXEC sp_trace_setstatus @traceid, 2
END
```

서버 측 트레이스 결과는 데이터베이스 테이블로 저장한 후 fn_trace_gettable 함수를 이용하면 튜닝 어드바이저의 워크로드로 사용할 수 있습니다. 다음 명령은 RDSTrace_1.trc 등의 모든 롤오버 파일을 포함해 D:\rdsdbdata\Log 디렉터리에서 RDSTrace.trc라는 이름의 모든 파일 결과를 현재 데이터베이스에서 RDSTrace라는 이름의 테이블로 로드합니다.

```
SELECT * INTO RDSTrace
FROM fn_trace_gettable('D:\rdsdbdata\Log\RDSTrace.trc', default);
```

예를 들어 RDSTrace_1.trc 같이 특정 롤오버 파일을 테이블에 저장하려면 롤오버 파일의 이름을 지정한 다음 fn_trace_gettable의 마지막 파라미터를 default가 아닌 1로 변경합니다.

```
SELECT * INTO RDSTrace_1
FROM fn_trace_gettable('D:\rdsdbdata\Log\RDSTrace_1.trc', 1);
```

트레이스를 이용한 튜닝 어드바이저 실행

로컬 파일이든 데이터베이스 테이블이든 상관없이 트레이스가 생성되면 이제 DB 인스턴스에 대해 튜닝 어드바이저를 실행할 수 있습니다. Amazon RDS와 함께 튜닝 어드바이저를 사용하는 방법은 독립된 원격 SQL Server 인스턴스를 사용할 때와 프로세스가 동일합니다. 그 밖에 클라이언트 컴퓨터에서 튜닝 어드바이저 UI를 사용하거나 명령줄에서 dta.exe 유틸리티를 사용할 수도 있습니다. 하지만 두 가지 경우 모두 DB 인스턴스 엔드포인트를 사용하여 Amazon RDS DB 인스턴스에 연결한 후 마스터 사용자 이름과 마스터 사용자 암호를 입력해야만 튜닝 어드바이저를 사용할 수 있습니다.

다음 코드 예제는 **dta.cnazcmklsdei.us-east-1.rds.amazonaws.com** 엔드포인트에서 Amazon RDS DB 인스턴스에 대해 dta.exe 명령줄 유틸리티를 사용하는 방법을 나타냅니다. 이 예에는 마스터 사용자 이름 **admin** 및 마스터 사용자 암호 **test**가 포함되며 튜닝할 예제 데이터베이스의 이름은 **C:\RDSTrace.trc**라는 머신 이름으로 지정됩니다. 또한 예제 명령줄 코드를 보면 트레이스 세션 이름을 **RDSTrace1**로 지정하였으며, 로컬 컴퓨터에 출력되는 파일 이름을 SQL 출력 스크립트의 경우 **RDSTrace.sql**로, 결과 파일의 경우 **RDSTrace.txt**로, 그리고 분석한 XML 파일의 경우 **RDSTrace.xml**로 지정하였습니다. 마지막으로 RDSDTA 데이터베이스 이름을 **RDSTraceErrors**로 지정한 오류 테이블도 있습니다.

```
dta -S dta.cnazcmklsdei.us-east-1.rds.amazonaws.com -U admin -P test -D RDSDTA -
if C:\RDSTrace.trc -s RDSTrace1 -of C:\ RDSTrace.sql -or C:\ RDSTrace.txt -ox C:\
 RDSTrace.xml -e RDSDTA.dbo.RDSTraceErrors
```

다음 예제 명령줄 코드는 입력 워크로드가 **RDSTrace** 데이터베이스에서 **RDSDTA**라는 이름의 원격 Amazon RDS 인스턴스에 저장된 테이블이라는 점을 제외하고 모두 동일합니다.

```
dta -S dta.cnazcmklsdei.us-east-1.rds.amazonaws.com -U admin -P test -D RDSDTA -it
 RDSDTA.dbo.RDSTrace -s RDSTrace1 -of C:\ RDSTrace.sql -or C:\ RDSTrace.txt -ox C:\
 RDSTrace.xml -e RDSDTA.dbo.RDSTraceErrors
```

dta 유틸리티 명령줄 파라미터의 전체 목록은 Microsoft 설명서의 dta 유틸리티를 참조하세요.

Amazon RDS for SQL Server 데이터베이스 **db_owner**를 **rdsa** 계정으로 변경

RDS for SQL Server DB 인스턴스에서 데이터베이스를 생성하거나 복원하는 경우 Amazon RDS가 데이터베이스 소유자를 rdsa로 설정합니다. SQL Server 데이터베이스 미러링(DBM) 또는 상시 가동 가용성 그룹(AG)을 사용하는 다중 AZ 배포의 경우 Amazon RDS는 보조 DB 인스턴스의 데이터베이스 소유자를 NT AUTHORITY\SYSTEM으로 설정합니다. 보조 DB 인스턴스가 기본 역할로 승격되기 전까지는 보조 데이터베이스의 소유자를 변경할 수 없습니다. 대부분의 경우 쿼리를 실행할 때 데이터베이스 소유자를 NT AUTHORITY\SYSTEM으로 설정하는 것은 문제가 되지 않지만, 실행을 위해 승격된 권한이 필요한 sys.sp_updatestats와 같은 시스템 저장 프로시저를 실행할 때는 오류가 발생할 수 있습니다.

다음 쿼리를 사용하여 NT AUTHORITY\SYSTEM에서 소유한 데이터베이스의 소유자를 식별할 수 있습니다.

```
SELECT name FROM sys.databases WHERE SUSER_SNAME(owner_sid) = 'NT AUTHORITY\SYSTEM';
```

Amazon RDS 저장 프로시저 rds_changedbowner_to_rdsa를 사용하여 데이터베이스 소유자를 rdsa로 변경할 수 있습니다. 다음 데이터베이스는 rds_changedbowner_to_rdsa와 함께 사용할 수 없습니다. master, model, msdb, rdsadmin, rdsadmin_ReportServer, rdsadmin_ReportServerTempDB, SSISDB

데이터베이스 소유자를 rdsa로 변경하려면 rds_changedbowner_to_rdsa 저장 프로시저를 호출하고 데이터베이스 이름을 입력합니다.

Example 사용법:

```
exec msdb.dbo.rds_changedbowner_to_rdsa 'TestDB1';
```

다음 파라미터는 필수입니다.

- @db_name - 데이터베이스 소유자를 rdsa로 변경할 데이터베이스의 이름입니다.

> **Important**
>
> rds_changedbowner_to_rdsa를 사용하여 데이터베이스의 소유권을 rdsa 이외의 로그인으로 변경할 수 없습니다. 예를 들어 데이터베이스를 만들 때 사용한 로그인의 소유권을 변경

할 수 없습니다. 다른 데이터베이스 사용자를 사용하여 멤버십을 부여할 수 없을 때 마스터 사용자의 db_owner 역할에서 손실된 멤버십을 복원하려면 마스터 사용자 암호를 재설정하여 db_owner 역할의 멤버십을 확보합니다. 자세한 내용은 Amazon RDS for SQL Server 마스터 사용자의 db_owner 역할 멤버십 재설정 단원을 참조하십시오.

Amazon RDS for Microsoft SQL Server의 데이터 정렬 및 문자 세트 관리

이 주제에서는 Amazon RDS에서 Microsoft SQL Server의 데이터 정렬 및 문자 세트를 관리하는 방법에 대한 지침을 제공합니다. 데이터베이스 생성 중에 데이터 정렬을 구성하고 나중에 수정하여 언어 및 로캘 요구 사항에 따라 텍스트 데이터를 적절하게 처리하는 방법을 설명합니다. 또한 Amazon RDS의 SQL Server 환경에서 호환성 및 성능을 유지하기 위한 모범 사례를 다룹니다.

SQL Server는 여러 수준에서 콜레이션을 지원합니다. DB 인스턴스를 생성할 때 기본 서버 콜레이션을 설정합니다. 데이터베이스, 테이블 또는 열 수준에서 콜레이션을 재정의할 수 있습니다.

주제

- Microsoft SQL Server의 서버 수준 콜레이션
- Microsoft SQL Server의 데이터베이스 수준 콜레이션

Microsoft SQL Server의 서버 수준 콜레이션

Microsoft SQL Server DB 인스턴스를 생성할 때 사용하려는 서버 콜레이션을 설정할 수 있습니다. 다른 콜레이션을 선택하지 않으면 서버 수준 콜레이션이 SQL_Latin1_General_CP1_CI_AS으로 기본 설정됩니다. 모든 데이터베이스 및 데이터베이스 객체에 서버 콜레이션이 기본적으로 적용됩니다.

> **Note**
>
> DB 스냅샷에서 복원할 때 데이터 정렬을 변경할 수 없습니다.

현재 Amazon RDS에서는 다음과 같은 서버 콜레이션을 지원합니다.

콜레이션	설명
Arabic_CI_AS	아랍어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분

콜레이션	설명
Chinese_PRC_BIN2	중국어-중화인민공화국어, 이진 코드 포인트 비교 정렬
Chinese_PRC_CI_AS	중국어-중화인민공화국어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Chinese_Taiwan_Stroke_CI_AS	중국어-대만어-스트로크, 소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Danish_Norwegian_CI_AS	덴마크어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Danish_Norwegian_CI_AS_KS	덴마크-노르웨이어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Danish_Norwegian_CI_AS_KS_WS	덴마크-노르웨이어, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분
Danish_Norwegian_CI_AS_WS	덴마크-노르웨이어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 구분
Danish_Norwegian_CS_AI	덴마크-노르웨이어, 대소문자 구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분
Danish_Norwegian_CS_AI_KS	덴마크-노르웨이어, 대소문자 구분, 액센트 비구분, 일본어 가나 구분, 전자/반자 비구분
Finnish_Swedish_100_BIN	Finnish-Swedish-100, 이진 정렬
Finnish_Swedish_100_BIN2	Finnish-Swedish-100, 이진 코드 포인트 비교 정렬
Finnish_Swedish_100_CI_AI	Finnish-Swedish-100, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분
Finnish_Swedish_100_CI_AS	Finnish-Swedish-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분

콜레이션	설명
Finnish_Swedish_CI_AS	핀란드어, 스웨덴어, 스웨덴어(핀란드), 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
French_CI_AS	프랑스어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Hebrew_BIN	히브리어, 이진 정렬
Hebrew_CI_AS	히브리어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Japanese_BIN	일본어, 이진 정렬
Japanese_CI_AS	일본어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Japanese_CS_AS	일본어, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Japanese_XJIS_140_CI_AS	일본어, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 비구분
Japanese_XJIS_140_CI_AS_KS_VSS	일본어, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분
Japanese_XJIS_140_CI_AS_VSS	일본어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 보조 문자, 변형 선택기 구분
일본어_XJIS_140_CS_AS_KS_WS	일본어, 대소문자 구분, 액센트 구분, 일본어 가나 구분, 전자/반자 구분, 보조 문자, 변형 선택기 비구분

콜레이션	설명
Korean_Wansung_CI_AS	한국어-완성형, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Latin1_General_100_BIN	라틴어1-일반-100, 이진 정렬
Latin1_General_100_BIN2	라틴어1-일반-100, 이진 코드 포인트 비교 정렬
Latin1_General_100_BIN2_UTF8	Latin1-General-100, 이진 코드 포인트 비교 정렬, UTF-8 인코딩
Latin1_General_100_CI_AS	라틴어1-일반-100, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Latin1_General_100_CI_AS_SC_UTF8	Latin1-General-100, 대소문자 비구분, 액센트 구분, 보조 문자, UTF-8 인코딩
Latin1_General_BIN	라틴어1-일반, 이진 정렬
Latin1_General_BIN2	라틴어1-일반, 이진 코드 포인트 비교 정렬
Latin1_General_CI_AI	라틴어1-일반, 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분
Latin1_General_CI_AS	라틴어1-일반, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Latin1_General_CI_AS_KS	라틴어1-일반, 대소문자 비구분, 액센트 구분, 일본어 가나 구분, 전자/반자 비구분
Latin1_General_CS_AS	라틴어1-일반, 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Modern_Spanish_CI_AS	현대-스페인어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Polish_CI_AS	폴란드어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분

콜레이션	설명
SQL_1xCompat_CP850_CI_AS	라틴어1-일반, 유니코드 데이터의 경우 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비 유니코드 데이터의 경우 코드 페이지 850의 SQL Server 정렬 순서 49
SQL_Latin1_General_CP1_CI_AI	라틴어1-일반, 유니코드 데이터의 경우 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 비 유니코드 데이터의 경우 코드 페이지 1252의 SQL Server 정렬 순서 54
SQL_Latin1_General_CP1_CI_AS(기본값)	라틴어1-일반, 유니코드 데이터의 경우 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비 유니코드 데이터의 경우 코드 페이지 1252의 SQL Server 정렬 순서 52
SQL_Latin1_General_CP1_CS_AS	라틴어1-일반, 유니코드 데이터의 경우 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비 유니코드 데이터의 경우 코드 페이지 1252의 SQL Server 정렬 순서 51
SQL_Latin1_General_CP437_CI_AI	라틴어1-일반, 유니코드 데이터의 경우 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 비 유니코드 데이터의 경우 코드 페이지 437의 SQL Server 정렬 순서 34
SQL_Latin1_General_CP850_BIN	라틴어1-일반, 유니코드 데이터의 경우 이진 정렬 순서, 비유니코드 데이터의 경우 코드 페이지 850의 SQL Server 정렬 순서 40
SQL_Latin1_General_CP850_BIN2	라틴어1-일반, 유니코드 데이터의 경우 이진 코드 포인트 비교, 비 유니코드 데이터의 경우 코드 페이지 850의 SQL Server 정렬 순서 40

콜레이션	설명
SQL_Latin1_General_CP850_CI_AI	라틴어1-일반, 유니코드 데이터의 경우 대소문자 비구분, 액센트 비구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 850의 SQL Server 정렬 순서 44
SQL_Latin1_General_CP850_CI_AS	라틴어1-일반, 유니코드 데이터의 경우 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비 유니코드 데이터의 경우 코드 페이지 850의 SQL Server 정렬 순서 42
SQL_Latin1_General_CP1256_CI_AS	라틴어1-일반, 유니코드 데이터의 경우 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비 유니코드 데이터의 경우 코드 페이지 1256의 SQL Server 정렬 순서 146
SQL_Latin1_General_CP1255_CS_AS	Latin1-General, 유니코드 데이터의 경우 대소문자 구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분, 비유니코드 데이터의 경우 코드 페이지 1255의 SQL Server 정렬 순서 137
Thai_CI_AS	프랑스어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분
Turkish_CI_AS	터키어, 대소문자 비구분, 액센트 구분, 일본어 가나 비구분, 전자/반자 비구분

콜레이션을 선택하려면 다음을 수행하세요.

- Amazon RDS 콘솔을 사용하는 경우 새 DB 인스턴스를 생성할 때 Additional configuration(추가 구성)을 선택한 다음 Collation(데이터 정렬) 필드에 데이터 정렬을 입력합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
- AWS CLI를 사용하는 경우 `--character-set-name` 명령과 함께 `create-db-instance` 옵션을 사용합니다. 자세한 내용은 create-db-instance를 참조하세요.
- Amazon RDS API를 사용하는 경우 `CharacterSetName` 작업과 함께 `CreateDBInstance` 파라미터를 사용합니다. 자세한 내용은 CreateDBInstance를 참조하세요.

Microsoft SQL Server의 데이터베이스 수준 콜레이션

새로운 데이터베이스 또는 데이터베이스 객체를 생성하면서 데이터 정렬을 무시하면 데이터베이스, 테이블 또는 열을 기준으로 기본 데이터 정렬을 변경할 수도 있습니다. 예를 들어 기본 서버 콜레이션이 SQL_Latin1_General_CP1_CI_AS인 경우, Mohawk 콜레이션 지원을 위해 이를 Mohawk_100_CI_AS로 변경할 수 있습니다. 모든 쿼리 인수는 필요에 따라 타입캐스트를 통해 다른 데이터 정렬을 사용할 수 있습니다.

예를 들어 다음 쿼리는 AccountName 열의 기본 콜레이션을 Japanese_CI_AS로 변경합니다.

```
CREATE TABLE [dbo].[Account]
 (
     [AccountID] [nvarchar](10) NOT NULL,
     [AccountName] [nvarchar](100) COLLATE Mohawk_100_CI_AS NOT NULL
 ) ON [PRIMARY];
```

Microsoft SQL Server DB 엔진은 기본적인 NCHAR, NVARCHAR 및 NTEXT 데이터 유형에 따라 Unicode를 지원합니다. 이를 테면 CJK 지원이 필요할 경우 문자 스토리지에 이 세 가지 Unicode 데이터 유형을 사용하면 데이터베이스 및 테이블 생성 시 기본 서버 데이터 정렬을 무시할 수 있습니다. SQL Server에 대한 데이터 정렬 및 Unicode 지원을 설명한 Microsoft 링크를 몇 가지 소개합니다.

- Working with Collations
- Collation and International Terminology
- Using SQL Server Collations
- International Considerations for Databases and Database Engine Applications

Amazon RDS for SQL Server용 데이터베이스 사용자 생성

다음 예와 같이 T-SQL 스크립트를 실행하여 Amazon RDS for Microsoft SQL Server DB 인스턴스의 데이터베이스 사용자를 생성할 수 있습니다. SQL 서버 관리 제품군(SSMS)과 같은 애플리케이션을 사용합니다. DB 인스턴스를 생성할 때 생성된 마스터 사용자로 DB 인스턴스에 로그인합니다.

```
--Initially set context to master database
USE [master];
GO
--Create a server-level login named theirname with password theirpassword
CREATE LOGIN [theirname] WITH PASSWORD = 'theirpassword';
GO
--Set context to msdb database
```

```
USE [msdb];
GO
--Create a database user named theirname and link it to server-level login theirname
CREATE USER [theirname] FOR LOGIN [theirname];
GO
```

데이터베이스 사용자를 역할에 추가하는 예는 SQLAgentUser 역할에 사용자 추가 섹션을 참조하세요.

> **Note**
>
> 사용자를 추가할 때 권한 오류가 발생할 경우, DB 인스턴스 마스터 사용자 암호를 수정하여 복원할 수 있습니다. 자세한 내용은 Amazon RDS for SQL Server 마스터 사용자의 db_owner 역할 멤버십 재설정 섹션을 참조하세요.
> 애플리케이션에서 마스터 사용자 권한을 복제하는 것은 모범 사례가 아닙니다. 자세한 내용은 How to clone master user permissions in Amazon RDS for SQL Server를 참조하세요.

Amazon RDS for SQL Server 데이터베이스에 적용할 복구 모델 결정

Amazon RDS에서 복구 모델, 보존 기간 및 데이터베이스 상태는 서로 연결되어 있습니다.

이러한 설정 중 하나를 변경하기 전에 결과를 이해하는 것이 중요합니다. 각 설정은 다른 설정에 영향을 줄 수 있습니다. 예:

- 백업 보존이 활성화되어 있는 상태에서 데이터베이스의 복구 모델을 SIMPLE 또는 BULK_LOGGED로 변경하면 Amazon RDS는 5분 이내에 복구 모델을 FULL로 재설정합니다. 또한 RDS가 DB 인스턴스의 스냅샷을 캡처합니다.
- 백업 보존을 `0`일로 설정하면 RDS가 복구 모드를 SIMPLE로 설정합니다.
- 백업 보존이 `0`일로 설정되어 있는 상태에서 데이터베이스의 복구 모델을 SIMPLE에서 다른 옵션으로 변경하면 RDS가 복구 모델을 SIMPLE로 재설정합니다.

> **Important**
>
> ALTER DATABASE 등을 사용하여—수행할 수 있을 것처럼 보이더라도 다중 AZ 인스턴스에서 복구 모델을 절대로 변경하지 마십시오. 백업 보존, 따라서 전체 복구 모드는 다중 AZ에 필요합니다. 복구 모델을 변경하면 RDS가 즉시 다시 이를 FULL로 변경합니다.
> 이 자동 재설정을 수행하면 RDS가 미러를 완전히 재구축합니다. 이 재구축 중에 미러가 장애 조치에 대비할 때까지 약 30-90분 동안 데이터베이스의 가용성이 저하됩니다. 또한 DB 인스턴

스는 단일 AZ에서 다중 AZ로 변환하는 경우와 같은 방식으로 성능 저하를 경험합니다. 성능이 저하되는 기간은 데이터베이스 스토리지 크기에 따라 달라지는데—저장된 데이터베이스가 클수록 성능 저하가 길어집니다.

SQL Server 복구 모델에 대한 자세한 내용은 Microsoft 설명서의 복구 모델(SQL Server)을 참조하세요.

Amazon RDS for SQL Server의 마지막 장애 조치 시간 결정

마지막 장애 조치 시간을 확인하려면 다음 저장 프로시저를 사용합니다.

```
execute msdb.dbo.rds_failover_time;
```

이 프로시저는 다음 정보를 반환합니다.

출력 파라미터	설명
errorlog_available_from	로그 디렉터리에서 오류 로그를 사용할 수 있게 된 시간을 표시합니다.
recent_failover_time	오류 로그에 장애 조치가 있는 경우 마지막 장애 조치 시간을 표시합니다. 그렇지 않은 경우 null입니다.

> **Note**
>
> 저장 프로시저는 로그 디렉터리에서 사용 가능한 모든 SQL Server 오류 로그를 검색하여 최근의 장애 조치 시간을 확인합니다. SQL Server에서 장애 조치 메시지를 덮어쓴 경우 프로시저는 장애 조치 시간을 확인하지 않습니다.

Example 최근 장애 조치 없음

이 예에서는 오류 로그에 최근 장애 조치가 없는 경우의 출력을 보여 줍니다. 2020-04-29 23:59:00.01 이후로 장애 조치가 발생하지 않았습니다.

errorlog_available_from	recent_failover_time
2020-04-29 23:59:00.0100000	null

Example 최근 장애 조치

이 예에서는 오류 로그에 장애 조치가 있는 경우의 출력을 보여 줍니다. 가장 최근의 장애 조치는 2020-05-05 18:57:51.89에 있었습니다.

errorlog_available_from	recent_failover_time
2020-04-29 23:59:00.0100000	2020-05-05 18:57:51.8900000

로그 시퀀스 번호 차이로 인한 시점 복구 실패 문제 해결

RDS for SQL Server에서 시점 복구(PITR)를 시도할 때 로그 시퀀스 번호(LSN) 차이로 인해 실패가 발생할 수 있습니다. 이러한 차이로 인해 RDS가 데이터베이스를 요청된 시간으로 복원할 수 없으며 RDS는 복원 인스턴스를 `incompatible-restore` 상태로 전환합니다.

이러한 문제가 발생하는 일반적인 원인은 다음과 같습니다.

- 데이터베이스 복구 모델에 대한 수동 변경
- 트랜잭션 로그 백업을 완료하기 위한 리소스가 부족하여 발생하는 RDS의 자동 복구 모델 변경

데이터베이스에서 LSN 차이를 식별하려면 다음 쿼리를 실행합니다.

```
SELECT * FROM msdb.dbo.rds_fn_list_tlog_backup_metadata(database_name)
ORDER BY backup_file_time_utc desc;
```

LSN 차이가 발견되면 다음을 수행할 수 있습니다.

- LSN 차이 이전의 복원 지점을 선택합니다.
- 기다렸다가 다음 인스턴스 백업이 완료된 후 특정 시점으로 복원합니다.

이 문제를 방지하려면 RDS for SQL Server 데이터베이스의 복구 모델을 수동으로 변경하지 않는 것이 좋습니다. 이렇게 하면 인스턴스 내구성이 저하되기 때문입니다. 또한 정기적인 트랜잭션 로그 백업을 보장하기 위해 워크로드에 충분한 리소스가 있는 인스턴스 유형을 선택하는 것이 좋습니다.

트랜잭션 로그 관리에 대한 자세한 내용은 Microsoft SQL Server 설명서의 SQL Server transaction log architecture and management guide를 참조하세요.

Amazon RDS for SQL Server의 데이터베이스 이름 보기 거부 또는 허용

마스터 사용자는 `DENY VIEW ANY DATABASE TO` *`LOGIN`*을 설정하여 사용자에게 데이터베이스를 숨길 수 없습니다. 이 권한을 변경하려면 다음 저장 프로시저를 대신 사용합니다.

- *`LOGIN`*에 대한 데이터베이스 보기 액세스 거부:

```
EXEC msdb.dbo.rds_manage_view_db_permission @permission='DENY',
 @server_principal='LOGIN'

go
```

- *`LOGIN`*에 대한 데이터베이스 보기 액세스 허용:

```
EXEC msdb.dbo.rds_manage_view_db_permission @permission='GRANT',
 @server_principal='LOGIN'

go
```

이 저장 프로시저를 사용할 때는 다음 사항을 고려해야 합니다.

- 데이터베이스 이름은 SSMS 및 내부 동적 관리 보기(DMV)에서 숨겨집니다. 하지만 데이터베이스 이름은 감사, 로그 및 메타데이터 테이블에서 계속 볼 수 있습니다. 이는 보안이 적용되는 `VIEW ANY DATABASE` 서버 권한입니다. 자세한 내용은 DENY 서버 권한을 참조하세요.
- 권한이 `GRANT`(허용)로 되돌려지면 *`LOGIN`*은 모든 데이터베이스를 볼 수 있습니다.
- *`LOGIN`*을 삭제하고 다시 만들면 LOGIN과 관련된 보기 권한이 `ALLOW`로 재설정됩니다.
- 다중 AZ 인스턴스의 경우 기본 호스트의 *`LOGIN`*에만 `DENY` 또는 `GRANT` 권한을 설정합니다. 변경 사항은 보조 호스트에 자동으로 전파됩니다.
- 이 권한은 로그인이 데이터베이스 이름을 볼 수 있는지만 변경합니다. 그러나 데이터베이스와 내부 객체에 대한 액세스는 별도로 관리됩니다.

Amazon RDS for SQL Server에 대한 대량 로드 중 빠른 삽입 비활성화

SQL Server 2016부터는 빠른 삽입이 기본적으로 사용됩니다. 빠른 삽입은 데이터베이스가 삽입 성능을 최적화하기 위해 단순 또는 대량 로그 복구 모델에 있는 동안 발생하는 최소 로깅을 활용합니다. 빠른 삽입을 통해 각 일괄 로드 배치가 새 범위를 획득하여 삽입 성능을 최적화하기 위해 사용 가능한 공간이 있는 기존 범위에 대한 할당 조회를 건너뜁니다.

그러나 배치 크기가 작은 일괄 로드를 빠르게 삽입하면 객체에서 사용되지 않는 공간이 늘어날 수 있습니다. 배치 크기를 늘릴 수 없는 경우 추적 플래그 692를 활성화하면 사용되지 않는 예약된 공간을 줄일 수 있지만 성능이 저하됩니다. 이 추적 플래그를 활성화하면 데이터를 힙 또는 클러스터된 인덱스로 일괄 로드하는 동안 빠른 삽입이 비활성화됩니다.

DB 파라미터 그룹을 사용하여 추적 플래그 692를 시작 파라미터로 활성화합니다. 자세한 내용은 Amazon RDS의 파라미터 그룹 섹션을 참조하세요.

추적 플래그 692는 SQL Server 2016 이상에서 Amazon RDS에 대해 지원됩니다. 추적 플래그에 대한 자세한 내용은 Microsoft 설명서의 DBCC TRACEON - 추적 플래그를 참조하세요.

Amazon RDS for Microsoft SQL Server DB 인스턴스 데이터베이스 삭제

단일 AZ 또는 다중 AZ 배포에서 Microsoft SQL Server를 실행하는 Amazon RDS DB 인스턴스의 데이터베이스를 삭제할 수 있습니다. 데이터베이스를 삭제하려면 다음 명령을 사용합니다.

```
--replace your-database-name with the name of the database you want to drop
EXECUTE msdb.dbo.rds_drop_database  N'your-database-name'
```

> **Note**
>
> 이 명령에서는 직선형 홑따옴표를 사용하세요. 스마트 따옴표를 사용하면 오류가 발생할 수 있습니다.

이 절차를 사용하여 데이터베이스를 삭제하면, Amazon RDS가 데이터베이스와의 기존 연결을 모두 삭제하고 데이터베이스의 백업 기록을 제거합니다.

다른 사용자에게 백업 및 복원 허용량을 부여하려면 다음 프로시저를 따릅니다.

```
USE master
```

```
GO
CREATE LOGIN user1 WITH PASSWORD=N'changeThis', DEFAULT_DATABASE=master,
 CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF
GO
USE msdb
GO
CREATE USER user1 FOR LOGIN user1
GO
use msdb
GO
GRANT EXECUTE ON msdb.dbo.rds_backup_database TO user1
GO
GRANT EXECUTE ON msdb.dbo.rds_restore_database TO user1
GO
```

다중 AZ 배포의 Amazon RDS for Microsoft SQL Server 데이터베이스 이름 변경

다중 AZ를 사용하는 Microsoft SQL Server 데이터베이스 인스턴스의 이름을 변경하려면 다음 절차에 따릅니다.

1. 먼저 해당 DB 인스턴스에 대해 다중 AZ를 해제합니다.
2. `rdsadmin.dbo.rds_modify_db_name`을 실행하여 데이터베이스의 이름을 변경합니다.
3. 그 다음에 해당 DB 인스턴스에 대해 다중 AZ 미러링 또는 상시 작동 가용성 그룹을 활성화하여 원래 상태로 되돌립니다.

자세한 내용은 다중 AZ를 Microsoft SQL Server DB 인스턴스에 추가 섹션을 참조하세요.

> **Note**
>
> 해당 인스턴스에서 다중 AZ를 사용하지 않는 경우 `rdsadmin.dbo.rds_modify_db_name`을 실행하기 전 또는 후에 설정 값을 수정할 필요가 없습니다.
> 읽기 전용 복제본 소스 인스턴스의 데이터베이스 이름은 변경할 수 없습니다.

예시: 다음 예시에서 `rdsadmin.dbo.rds_modify_db_name` 저장 프로시저는 데이터베이스 이름을 **`MOO`**에서 **`ZAR`**로 바꿉니다. 이는 `DDL ALTER DATABASE [MOO] MODIFY NAME = [ZAR]` 문을 실행하는 것과 유사합니다.

```
EXEC rdsadmin.dbo.rds_modify_db_name N'MOO', N'ZAR'
GO
```

Amazon RDS for SQL Server 마스터 사용자의 db_owner 역할 멤버십 재설정

마스터 사용자를 RDS for SQL Server 데이터베이스의 db_owner 역할 멤버십에서 제외하고 다른 데이터베이스 사용자가 멤버십을 부여할 수 없는 경우, DB 인스턴스 마스터 사용자 암호를 수정하여 손실된 멤버십을 복원할 수 있습니다.

RDS는 DB 인스턴스 마스터 사용자 암호를 변경하여 실수로 취소되었을 수 있는 DB 인스턴스의 데이터베이스에 대한 db_owner 멤버십을 부여합니다. Amazon RDS 콘솔, AWS CLI 명령 modify-db-instance 또는 ModifyDBInstance API 작업을 사용하여 DB 인스턴스 암호를 변경할 수 있습니다. DB 인스턴스 변경에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

Amazon RDS for SQL Server에 대한 라이선스 종료 DB 인스턴스 복원

Microsoft는 Microsoft License Mobility 정보를 보고하지 않은 일부 Amazon RDS 고객에게 DB 인스턴스를 종료할 것을 요청했습니다 Amazon RDS는 이러한 DB 인스턴스의 스냅샷을 생성하며, 스냅샷에서 라이선스 포함 모델이 있는 새 DB 인스턴스로 복원할 수 있습니다.

Standard Edition의 스냅샷에서 Standard Edition 또는 Enterprise Edition으로 복원할 수 있습니다.

Enterprise Edition의 스냅샷에서 Standard Edition 또는 Enterprise Edition으로 복원할 수 있습니다.

Amazon RDS가 인스턴스의 최종 스냅샷을 생성한 후 SQL Server 스냅샷에서 복원하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.
3. SQL Server DB 인스턴스의 스냅샷을 선택합니다. Amazon RDS가 DB 인스턴스의 최종 스냅샷을 생성합니다. 종료된 인스턴스 스냅샷의 이름은 *instance_name*-final-snapshot 형식입니다. 예를 들어 DB 인스턴스 이름이 **mytest.cdxgahslksma.us-east-1.rds.com**일 경우, 최종 스냅샷은 **mytest-final-snapshot**이라는 이름으로 원본 DB 인스턴스와 동일한 AWS 리전에 있습니다.
4. 작업에서 스냅샷 복원을 선택합니다.

[Restore DB Instance] 창이 나타납니다.

5. 라이선스 모델에서 라이선스 포함을 선택합니다.
6. 사용할 SQL Server DB 엔진을 선택합니다.
7. DB 인스턴스 식별자에 복원된 DB 인스턴스의 이름을 입력합니다.
8. [Restore DB Instance]를 선택합니다.

스냅샷에서 복원하는 자세한 방법은 DB 인스턴스 복원 단원을 참조하세요.

오프라인에서 온라인으로 Amazon RDS for SQL Server 데이터베이스 전환

`OFFLINE`에서 `ONLINE`으로 Amazon RDS DB 인스턴스의 Microsoft SQL Server 데이터베이스를 전환할 수 있습니다.

SQL Server 메소드	Amazon RDS 메서드
ALTER DATABASE *db_name* SET ONLINE;	EXEC rdsadmin.dbo.rds_set_database_online *db_name*

Amazon RDS for SQL Server에 변경 데이터 캡처 사용

Amazon RDS는 Microsoft SQL Server를 실행하는 DB 인스턴스에 대해 CDC(변경 데이터 캡처)를 지원합니다. 테이블의 데이터에 대한 CDC 캡처 변경입니다. 각 변경 사항에 대한 메타데이터를 저장하고, 이후에 액세스할 수 있습니다. CDC 작동 방식에 대한 자세한 내용은 Microsoft 설명서의 변경 데이터 캡처를 참조하세요. Amazon RDS DB 인스턴스에 CDC를 사용하기 전에 `msdb.dbo.rds_cdc_enable_db`를 실행하여 데이터베이스에서 이를 활성화합니다. CDC가 활성화된 이후 데이터베이스의 `db_owner`인 모든 사용자는 데이터베이스의 테이블에서 CDC를 활성화 또는 비활성화할 수 있습니다.

> **Important**
>
> 복원 도중 CDC가 비활성화됩니다. 모든 관련 메타데이터가 데이터베이스에서 자동으로 제거됩니다. 이는 스냅샷 복원, 특정 시점으로 복원 및 S3로부터의 SQL Server 기본 복원에 적용됩니다. 이러한 유형의 복원 중 하나를 수행한 이후 CDC를 다시 활성화하고 추적할 테이블을 다시 지정할 수 있습니다.

DB 인스턴스에 대해 CDC를 활성화하려면 msdb.dbo.rds_cdc_enable_db 저장 프로시저를 실행합니다.

```
exec msdb.dbo.rds_cdc_enable_db 'database_name'
```

DB 인스턴스에 대해 CDC를 비활성화하려면 msdb.dbo.rds_cdc_disable_db 저장 프로시저를 실행합니다.

```
exec msdb.dbo.rds_cdc_disable_db 'database_name'
```

사용자에게 CDC 권한을 부여하려면 다음 프로시저를 사용합니다.

```
go
  GRANT EXECUTE ON msdb.dbo.rds_cdc_enable_db TO User1
  GRANT EXECUTE ON msdb.dbo.rds_cdc_disable_db TO User1
```

주제

- 변경 데이터 캡처로 테이블 추적
- 변경 데이터 캡처 작업
- 다중 AZ 인스턴스에 대한 변경 데이터 캡처

변경 데이터 캡처로 테이블 추적

데이터베이스에서 CDC가 활성화된 이후 특정 테이블 추적을 시작할 수 있습니다. sys.sp_cdc_enable_table을 실행하여 추적할 테이블을 선택할 수 있습니다.

```
--Begin tracking a table
exec sys.sp_cdc_enable_table
   @source_schema           = N'source_schema'
,  @source_name             = N'source_name'
,  @role_name               = N'role_name'

--The following parameters are optional:

--, @capture_instance       = 'capture_instance'
--, @supports_net_changes   = supports_net_changes
--, @index_name             = 'index_name'
--, @captured_column_list   = 'captured_column_list'
```

```
--, @filegroup_name         = 'filegroup_name'
--, @allow_partition_switch = 'allow_partition_switch'
;
```

테이블에 대한 CDC 구성을 보려면 sys.sp_cdc_help_change_data_capture를 실행합니다.

```
--View CDC configuration
exec sys.sp_cdc_help_change_data_capture

--The following parameters are optional and must be used together.
--  'schema_name', 'table_name'
;
```

SQL Server 설명서의 CDC 테이블, 함수 및 저장된 절차에 대한 자세한 내용은 다음을 참조하세요.

- 변경 데이터 캡처 저장 프로시저(Transact-SQL)
- 변경 데이터 캡처 함수(Transact-SQL)
- 변경 데이터 캡처 테이블(Transact-SQL)

변경 데이터 캡처 작업

CDC를 활성화할 때 SQL Server가 CDC 작업을 생성합니다. 데이터베이스 소유자(db_owner)는 CDC 작업의 보기, 생성, 수정 및 삭제가 가능합니다. 하지만 RDS 시스템 계정은 이들을 소유합니다. 따라서 기본 보기, 절차 또는 SQL Server Management Studio에서 작업이 보이지 않습니다.

데이터베이스의 CDC 동작을 제어하려면 sp_cdc_enable_table 및 sp_cdc_start_job과 같은 기본 SQL Server 절차를 사용합니다. `maxtrans` 및 `maxscans`와 같은 CDC 작업 파라미터를 변경하려면 sp_cdc_change_job을 사용할 수 있습니다.

CDC 작업에 관한 추가 정보를 얻으려면 다음 동적 관리 보기를 쿼리할 수 있습니다.

- sys.dm_cdc_errors
- sys.dm_cdc_log_scan_sessions
- sysjobs
- sysjobhistory

다중 AZ 인스턴스에 대한 변경 데이터 캡처

다중 AZ 인스턴스에서 CDC를 사용하는 경우 미러의 CDC 작업 구성이 보안 주체에 있는 것과 일치해야 합니다. CDC 작업은 `database_id`로 매핑됩니다. 보조의 데이터베이스 ID가 보안 주체의 ID와 다른 경우 작업이 올바른 데이터베이스와 연결되지 않습니다. 장애 조치 이후 오류를 방지하기 위해 RDS는 작업을 취소하고 새 보안 주체에 작업을 다시 생성합니다. 다시 생성된 작업은 장애 조치 이전에 보안 주체가 기록한 파라미터를 사용합니다.

이 프로세스가 빠르게 실행되기는 하지만 CDC 작업은 RDS가 이를 수정하기 전에 실행될 수 있습니다. 기본 복제본과 보조 복제본 사이의 파라미터를 강제로 일관되게 하는 3가지 방법이 있습니다.

- CDC를 활성화한 모든 데이터베이스에 대해 동일한 작업 파라미터를 사용합니다.
- CDC 작업 구성을 변경하기 전에 다중 AZ 인스턴스를 단일 AZ로 변환합니다.
- 보안 주체에서 파라미터를 변경할 때마다 수동으로 전송합니다.

장애 조치 이후 CDC 작업을 다시 생성하는 데 사용되는 CDC 파라미터를 보고 정의하려면 `rds_show_configuration` 및 `rds_set_configuration`을 사용합니다.

다음은 `cdc_capture_maxtrans`의 값을 반환하는 예입니다. `RDS_DEFAULT`로 설정된 모든 파라미터의 경우 RDS가 자동으로 값을 구성합니다.

```
-- Show configuration for each parameter on either primary and secondary replicas.
exec rdsadmin.dbo.rds_show_configuration 'cdc_capture_maxtrans';
```

보조에서 구성을 설정하려면 `rdsadmin.dbo.rds_set_configuration`을 실행합니다. 이 절차는 부 서버의 모든 데이터베이스에 대한 파라미터 값을 설정합니다. 이러한 설정은 장애 조치 이후에만 사용됩니다. 다음 예제는 모든 CDC 캡처 작업에 대한 `maxtrans`를 *1000*으로 설정합니다.

```
--To set values on secondary. These are used after failover.
exec rdsadmin.dbo.rds_set_configuration 'cdc_capture_maxtrans', 1000;
```

보안 주체에서 CDC 작업 파라미터를 설정하려면 sys.sp_cdc_change_job을 대신 사용합니다.

Amazon RDS에 SQL Server Agent 사용

Amazon RDS에서는 Microsoft SQL Server Enterprise Edition, Standard Edition 또는 Web Edition을 실행하는 DB 인스턴스에서 SQL Server 에이전트를 사용할 수 있습니다. SQL Server 에이전트는 예약된 관리 작업을 실행하는 Microsoft Windows 서비스입니다. SQL Server 에이전트를 사용하면 T-SQL

작업을 통해 인덱스를 리빌드하고, 손상 검사를 실행하고, SQL Server DB 인스턴스의 데이터를 집계할 수 있습니다.

SQL Server DB 인스턴스를 생성하면 마스터 사용자가 `SQLAgentUserRole` 역할에 등록됩니다.

SQL Server 에이전트는 특정 이벤트 발생 시, 혹은 필요에 따라 예약 작업을 실행할 수 있습니다. 자세한 내용은 Microsoft 설명서에서 SQL Server 에이전트를 참조하세요.

> **Note**
>
> DB 인스턴스의 유지 관리 및 백업 기간 중에는 작업 실행을 예약하지 마세요. AWS를 통해 시작되는 유지 관리 및 백업 프로세스에 의해 작업이 중단되거나 취소될 수 있습니다.
> 다중 AZ 배포에서 SQL Server 에이전트 작업은 작업 복제 기능이 켜져 있을 때 기본 호스트에서 보조 호스트로 복제됩니다. 자세한 내용은 SQL Server 에이전트 작업 복제 켜기 섹션을 참조하세요.
> 다중 AZ 배포에서는 SQL Server 에이전트 작업이 10,000개로 제한됩니다. 한도를 늘려야 할 경우 지원에 문의하여 할당량 증대를 요청하세요. [지원 센터(AWS Support Center)] 페이지를 열고 필요한 경우, 로그인한 다음 [사례 생성(Create Case)]을 선택합니다. Service Limit increase(서비스 한도 증가)를 선택합니다. 양식을 작성하고 제출합니다.

SQL Server Management Studio(SSMS)에서 SQL Server 에이전트의 개별 작업 기록을 보려면 Object Explorer를 열고 작업을 마우스 오른쪽 버튼으로 클릭한 다음 [기록 보기(View History)]를 선택합니다.

SQL Server 에이전트는 DB 인스턴스의 관리형 호스트에서 실행되므로 일부 작업이 지원되지 않습니다.

- 예를 들어 ActiveX, Windows cmdshell 또는 Windows PowerShell을 사용하여 복제 작업이나 명령줄 스크립트를 실행하는 것은 지원되지 않습니다.
- SQL Server 에이전트를 수동으로 시작, 중지 또는 다시 시작할 수 없습니다.
- SQL Server 에이전트를 통한 이메일 알림은 DB 인스턴스에서 사용할 수 없습니다.
- SQL Server 에이전트 알림 및 연산자는 지원되지 않습니다.
- SQL Server 에이전트를 사용한 백업 생성은 지원되지 않습니다. DB 인스턴스를 백업하려면 Amazon RDS를 사용합니다.
- 현재 RDS for SQL Server는 SQL Server 에이전트 토큰 사용을 지원하지 않습니다.

SQL Server 에이전트 작업 복제 켜기

다음 저장 프로시저를 사용하여 SQL Server 에이전트 작업 복제를 켤 수 있습니다.

```
EXECUTE msdb.dbo.rds_set_system_database_sync_objects @object_types = 'SQLAgentJob';
```

Amazon RDS for SQL Server에서 지원하는 모든 SQL Server 버전에서 저장 프로시저를 실행할 수 있습니다. 다음 범주의 작업이 복제됩니다.

- [분류되지 않음(로컬)]
- [분류되지 않음(멀티 서버)]
- [분류되지 않음]
- 데이터 수집기
- 데이터베이스 엔진 튜닝 관리자
- 데이터베이스 유지 관리
- 전체 텍스트

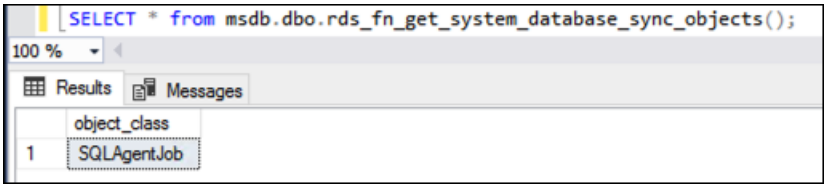
T-SQL 작업 단계를 사용하는 작업만 복제됩니다. SSIS(SQL Server Integration Services), SSRS(SQL Server Reporting Service), 복제 및 PowerShell과 같은 단계 유형이 있는 작업은 복제되지 않습니다. 데이터베이스 메일 및 서버 수준 객체를 사용하는 작업은 복제되지 않습니다.

> **Important**
>
> 프라이머리 호스트는 복제를 위한 신뢰할 수 있는 소스입니다. 작업 복제를 설정하기 전에 SQL Server 에이전트 작업이 프라이머리에 있는지 확인합니다. 새 작업이 보조 호스트에 있을 때 이 기능을 설정하면 SQL Server 에이전트 작업이 삭제될 수 있습니다.

다음 기능을 사용하면 복제가 켜져 있는지 확인할 수 있습니다.

```
SELECT * from msdb.dbo.rds_fn_get_system_database_sync_objects();
```

T-SQL 쿼리는 SQL Server 에이전트 작업이 복제되는 경우 다음을 반환합니다. 복제하지 않으면 `object_class`에 대해 아무 것도 반환하지 않습니다.

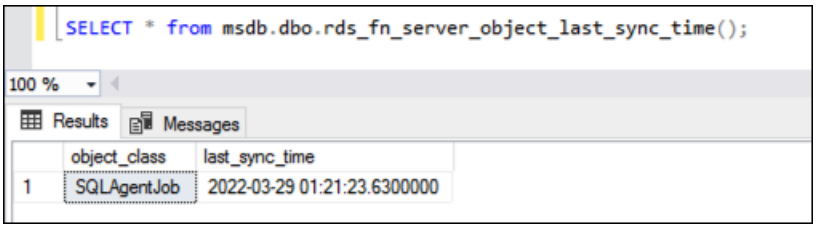


다음 함수를 사용하여 객체가 마지막으로 동기화된 시간(UTC 시간)을 찾을 수 있습니다.

```
SELECT * from msdb.dbo.rds_fn_server_object_last_sync_time();
```

예를 들어 SQL Server 에이전트 작업을 01:00에 수정한다고 가정합니다. 가장 최근에 동기화된 시간은 동기화가 수행되었음을 나타내는 01:00 이후일 것으로 예상합니다.

동기화 후에는 보조 노드에서 반환된 `date_created`와 `date_modified`가 일치할 것으로 예상됩니다.



`tempdb` 복제를 함께 사용하는 경우 `@object_type` 파라미터에 SQL 에이전트 작업과 `tempdb` 구성을 제공하여 둘 모두에 대해 복제를 활성화할 수 있습니다.

```
EXECUTE msdb.dbo.rds_set_system_database_sync_objects @object_types =
 'SQLAgentJob,TempDbFile';
```

`tempdb` 복제에 대한 자세한 내용은 다중 AZ 배포에 대한 TempDB 구성 섹션을 참조하세요.

SQL Server 에이전트 역할

RDS for SQL Server는 작업 관리를 위한 각기 다른 수준의 권한을 가진 다음과 같은 SQL Server 에이전트 역할을 지원합니다.

- SQLAgentUserRole

  권한

  - 자체 작업, 일정 및 연산자 생성 및 관리
  - 자체 작업 및 일정의 속성 보기
  - 다른 사용자가 생성한 작업을 보거나 관리할 수 없음

이 역할은 자신의 작업을 생성하고 관리해야 하지만 다른 사용자가 생성한 작업에는 액세스할 필요가 없는 사용자에게 적합합니다.

- SQLAgentReaderRole

  권한

  - SQLAgentUserRole의 모든 권한
  - 다른 사용자가 생성한 작업 및 일정을 포함하여 모든 작업 및 일정 목록 보기
  - 모든 작업의 속성 보기
  - 작업 기록 검토

  이 역할은 모든 작업의 상태를 모니터링해야 하지만 관리할 필요는 없는 사용자에게 적합합니다.

- SQLAgentOperatorRole

  권한

  - SQLAgentUserRole 및 SQLAgentReaderRole의 모든 권한
  - 작업 실행, 중지 또는 시작
  - 작업 기록 관리
  - 작업 및 일정 활성화/비활성화
  - 연산자 및 프록시 보기

  이 역할은 가장 포괄적인 권한을 제공하며 모든 작업을 완전히 제어해야 하는 사용자에게 적합합니다.

다음 명령을 사용하여 SQL Server 로그인에 역할을 할당합니다.

```
USE msdb;
EXEC sp_addrolemember 'SQLAgentOperatorRole', 'username';
```

RDS for SQL Server에서 SQLAgentOperatorRole 관리

현재 작업을 보려면 SQL Server 로그인에 SQLAgentOperatorRole을 추가하고 데이터베이스 연결을 해제하기 전에 제거해야 합니다.

SQL Server Management Studio에서 SQL Server 에이전트 트리를 시각화하려면 다음 지침을 따르세요.

SQL Server Management Studio(SSMS)에서 SQL Server 에이전트 보기

1. RDS 마스터 자격 증명을 사용하여 RDS SQL Server 인스턴스에 로그인하고 원하는 사용자에게 SQLAgentUserRole을 부여합니다.

```
USE msdb
GO
IF NOT EXISTS(SELECT name FROM sys.database_principals WHERE name = 'UserName')
BEGIN
CREATE USER UserName FROM LOGIN UserName
END
GO
ALTER ROLE SQLAgentUserRole ADD MEMBER UserName
GO
GRANT ALTER ON ROLE::[SQLAgentOperatorRole] to UserName
GO
```

   이러한 명령은 사용자가 존재하지 않는 경우 msdb 데이터베이스에 사용자를 생성합니다. 또한 SSMS의 SQL Server 에이전트 트리가 표시되도록 SQLAgentUserRole에 사용자를 추가합니다. 마지막으로, SQLAgentOperatorRole에 대한 변경 권한을 사용자에게 부여합니다. 이렇게 하면 사용자가 해당 역할에 자신을 추가/제거할 수 있습니다.

2. 위에서 언급한 역할에 자신을 추가하려면 작업을 확인해야 하는 사용자와 함께 RDS SQL Server 인스턴스에 연결하고 다음 스크립트를 실행합니다.

```
use msdb
go
ALTER ROLE SQLAgentOperatorRole ADD MEMBER UserName
GO
```

   그런 다음, 작업 폴더를 마우스 오른쪽 버튼으로 클릭하고 새로 고침을 선택합니다.

3. 이 작업을 수행하면 작업 탭에 +(더하기) 버튼이 표시됩니다. 이 버튼을 클릭하여 SQL Server 에이전트 작업 목록을 확장합니다.

4. **Important**

   RDS SQL Server 인스턴스에서 연결을 해제하기 전에 SQLAgentOperatorRole에서 자신을 제거해야 합니다.

SQLAgentOperatorRole에서 로그인을 제거하려면 Management Studio를 연결 해제하거나 닫기 전에 다음 쿼리를 실행합니다.

```
USE msdb
GO
ALTER ROLE SQLAgentOperatorRole DROP MEMBER UserName
GO
```

자세한 내용은 Leveraging SQLAgentOperatorRole in RDS SQL Server을 참조하세요.

SQLAgentUser 역할에 사용자 추가

추가 로그인 또는 사용자의 SQL Server 에이전트 사용을 허용하려면 마스터 사용자로 로그인한 후 다음을 수행합니다.

1. CREATE LOGIN 명령을 사용하여 다른 서버 수준 로그인을 생성합니다.
2. msdb 명령을 사용하여 CREATE USER 사용자를 생성한 다음 이전 단계에서 생성한 로그인에 이 사용자를 연결합니다.
3. SQLAgentUserRole 시스템 저장 프로시저를 사용하여 sp_addrolemember에 사용자를 추가합니다.

예를 들어 마스터 사용자 이름은 **admin**이며, 이름이 **theirname**이고 암호가 **theirpassword**인 사용자에게 SQL Server 에이전트에 대한 액세스 권한을 부여한다고 가정하겠습니다. 이 경우 다음 절차를 사용할 수 있습니다.

SQLAgentUser 역할에 사용자를 추가하려면

1. 마스터 사용자로 로그인합니다.
2. 다음 명령을 실행합니다:

```
--Initially set context to master database
USE [master];
GO
--Create a server-level login named theirname with password theirpassword
CREATE LOGIN [theirname] WITH PASSWORD = 'theirpassword';
GO
--Set context to msdb database
```

```
USE [msdb];
GO
--Create a database user named theirname and link it to server-level login
 theirname
CREATE USER [theirname] FOR LOGIN [theirname];
GO
--Added database user theirname in msdb to SQLAgentUserRole in msdb
EXEC sp_addrolemember [SQLAgentUserRole], [theirname];
```

SQL Server 에이전트 작업 삭제

`sp_delete_job` 저장 프로시저를 사용하여 Amazon RDS for Microsoft SQL Server의 SQL Server 에이전트 작업을 삭제합니다.

SSMS를 사용하여 SQL Server 에이전트 작업을 삭제할 수는 없습니다. 그렇게 하면 다음과 유사한 오류 메시지가 표시됩니다.

```
The EXECUTE permission was denied on the object 'xp_regread', database
 'mssqlsystemresource', schema 'sys'.
```

RDS는 관리형 서비스이기 때문에 Windows 레지스트리에 액세스하는 프로시저의 실행을 제한합니다. SSMS를 사용하는 경우 SSMS는 RDS에 의해 권한이 부여되지 않은 프로세스(`xp_regread`)를 실행하려고 시도합니다.

> **Note**
>
> RDS for SQL Server에서는 sysadmin 역할의 구성원만 다른 로그인에서 소유한 작업을 업데이트하거나 삭제할 수 있습니다. 자세한 내용은 Leveraging SQLAgentOperatorRole in RDS SQL Server을 참조하세요.

SQL Server 에이전트 작업을 삭제하려면

- 다음 T-SQL 문을 실행합니다.

```
EXEC msdb..sp_delete_job @job_name = 'job_name';
```

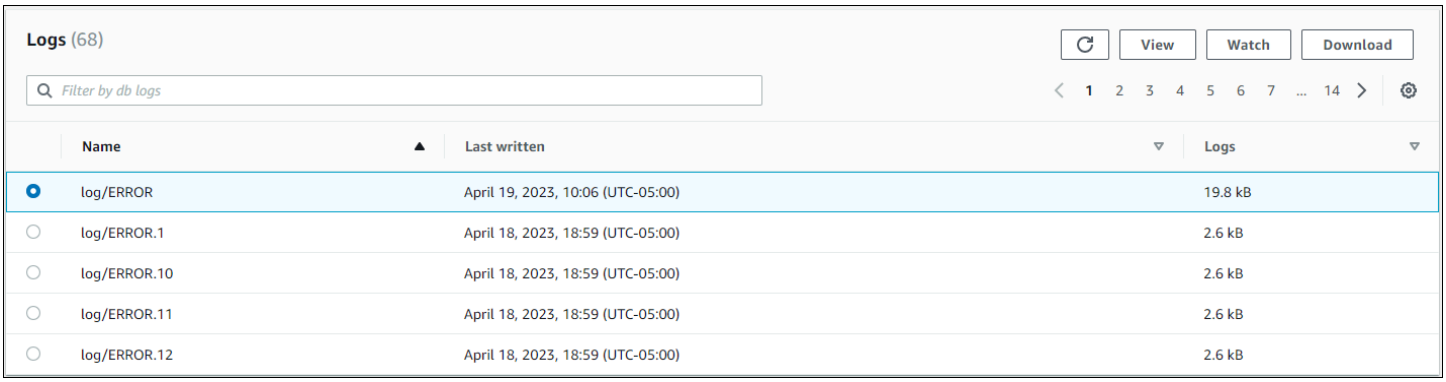
Amazon RDS for Microsoft SQL Server 로그 작업

Amazon RDS 콘솔을 사용하여 SQL Server 에이전트 로그, Microsoft SQL Server 오류 로그 및 SQL Server Reporting Services(SSRS) 로그를 확인하고, 보고, 다운로드할 수 있습니다.

로그 파일 조사

Amazon RDS 콘솔을 통해 로그를 보면 당시에 존재하는 내용을 확인할 수 있습니다. 콘솔에서 로그를 모니터링하면 로그가 동적 상태로 열리기 때문에 거의 실시간으로 업데이트를 볼 수 있습니다.

따라서 최신 상태의 로그만 모니터링됩니다. 예를 들어 다음과 같은 로그가 표시되어 있다고 가정하겠습니다.



가장 최근 로그인 log/ERROR만 업데이트가 활성화되어 있습니다. 다른 로그도 볼 수는 있지만 정적 상태로 열리기 때문에 업데이트되지 않습니다.

로그 파일 보관

Amazon RDS 콘솔은 지난 주부터 당일까지 로그를 표시합니다. 이 로그는 당일 이후 참조를 목적으로 다운로드하여 아카이브(보관)할 수 있습니다. Amazon S3 버킷에 로드하는 것도 로그를 아카이브할 수 있는 한 가지 방법입니다. Amazon S3 버킷 설치 및 파일 업로드 방법에 대한 자세한 내용은 Amazon Simple Storage Service 시작 안내서의 Amazon S3 기본 사항에서 시작하기를 클릭하면 확인할 수 있습니다.

오류 및 에이전트 로그 보기

Microsoft SQL 서버 오류 및 에이전트 로그를 보려면 다음 파라미터와 함께 Amazon RDS 저장 프로시저 `rds_read_error_log`를 사용합니다.

- **`@index`** – 가져올 로그의 버전. 기본값은 0이며, 현재 오류 로그를 가져옵니다. 이전 로그를 가져오려면 1을 지정하고, 그보다 하나 이전의 로그를 가져오려면 2를 지정하는 식입니다.

- **@type** – 가져올 로그의 유형. 오류 로그를 가져오려면 1을 지정합니다. 에이전트 로그를 가져오려면 2를 지정합니다.

Example

다음 예시에서는 현재 오류 로그를 요청합니다.

```
EXEC rdsadmin.dbo.rds_read_error_log @index = 0, @type = 1;
```

SQL Server 오류에 대한 자세한 내용은 Microsoft 설명서의 데이터베이스 엔진 오류를 참조하세요.

Amazon RDS for SQL Server에 대한 추적 및 덤프 파일 작업

이 단원에서는 Microsoft SQL Server를 실행하는 Amazon RDS DB 인스턴스에 대한 추적 파일 및 덤프 파일 작업에 대해 설명합니다.

추적 SQL 쿼리 생성

```
declare @rc int
declare @TraceID int
declare @maxfilesize bigint

set @maxfilesize = 5

exec @rc = sp_trace_create @TraceID output,  0, N'D:\rdsdbdata\log\rdstest',
 @maxfilesize, NULL
```

열린 추적 보기

```
select * from ::fn_trace_getinfo(default)
```

추적 내용 보기

```
select * from ::fn_trace_gettable('D:\rdsdbdata\log\rdstest.trc', default)
```

추적 및 덤프 파일의 보존 기간 설정

추적 및 덤프 파일이 누적되면서 디스크 공간을 낭비할 수 있습니다. Amazon RDS는 7일 이상 지난 추적 및 덤프 파일은 제거하도록 기본 설정되어 있습니다.

다음 예시에서처럼 현재 추적 및 덤프 파일 보존 기간을 보려면 rds_show_configuration 프로시저를 사용합니다.

```
exec rdsadmin..rds_show_configuration;
```

추적 파일의 보존 기간을 수정하려면 rds_set_configuration 프로시저를 사용하여 tracefile retention을 분 단위로 설정합니다. 다음 예시에서는 추적 파일 보존 기간을 24시간으로 설정합니다.

```
exec rdsadmin..rds_set_configuration 'tracefile retention', 1440;
```

덤프 파일의 보존 기간을 수정하려면 rds_set_configuration 프로시저를 사용하여 dumpfile retention을 분 단위로 설정합니다. 다음 예시에서는 덤프 파일 보존 기간을 3일로 설정합니다.

```
exec rdsadmin..rds_set_configuration 'dumpfile retention', 4320;
```

보안상의 이유로 인해 SQL Server DB 인스턴스에서는 특정 추적 또는 덤프 파일을 삭제할 수 없습니다. 사용하지 않는 추적 또는 덤프 파일을 모두 삭제하려면 해당 파일에 대한 보존 기간을 0으로 설정합니다.

Amazon RDS for MySQL

Amazon RDS는 여러 버전의 MySQL for DB 인스턴스를 지원합니다. 지원되는 버전에 대한 전체 정보는 Amazon RDS의 MySQL 버전 섹션을 참조하세요.

Amazon RDS for MySQL DB 인스턴스를 생성하려면 Amazon RDS 관리 도구 또는 인터페이스를 사용하면 됩니다. 그러면 다음 작업을 수행할 수 있습니다.

- DB 인스턴스 크기 조정
- DB 인스턴스로의 연결 승인
- 백업 또는 스냅샷에서 생성 및 복원
- 보조 다중 AZ 생성
- 읽기 전용 복제본 생성
- DB 인스턴스의 성능 모니터링

DB 인스턴스의 데이터를 저장하고 이에 액세스하려면 표준 MySQL 유틸리티 및 애플리케이션을 사용하면 됩니다.

Amazon RDS for MySQL은 다수의 산업 표준을 준수합니다. 예를 들어, RDS for MySQL 데이터베이스를 사용하여 HIPAA 호환 애플리케이션을 개발할 수 있습니다. RDS for MySQL 데이터베이스를 사용하여 AWS와의 비즈니스 제휴 계약(BAA)에 따라 보호 대상 건강 정보(PHI)를 비롯한 의료 관련 정보를 저장할 수 있습니다. Amazon RDS for MySQL은 또한 연방 위험 및 인증 관리 프로그램(FedRAMP) 보안 요건을 충족합니다. 나아가 Amazon RDS for MySQL은 FedRAMP 공동 승인 위원회(JAB)로부터 AWS GovCloud (US) 리전 내에 FedRAMP HIGH Baseline 수준의 잠정적 운영 권한(P-ATO)을 받았습니다. 지원되는 규정 준수 표준에 대한 자세한 내용은 AWS 클라우드 규정 준수를 참조하세요.

MySQL의 각 버전의 기능에 대한 내용은 MySQL 설명서의 MySQL의 기본 기능 단원을 참조하십시오.

DB 인스턴스를 생성하기 전에 Amazon RDS 환경 설정 단계를 완료합니다. DB 인스턴스를 생성하면 RDS 마스터 사용자에 DBA 권한이 부여되나, 몇 가지 제한이 적용됩니다. 추가 데이터베이스 계정 생성과 같은 관리 작업에 이 계정을 사용합니다.

다음을 생성할 수 있습니다.

- DB 인스턴스
- DB 스냅샷
- 특정 시점 복원

- 자동 백업
- 수동 백업

Amazon VPC를 기반으로 Virtual Private Cloud(VPC)에서 MySQL을 실행하는 DB 인스턴스를 사용할 수 있습니다. 또한 다양한 옵션을 활성화하여 MySQL DB 인스턴스에 기능을 추가할 수 있습니다. Amazon RDS는 고가용성 장애 조치 솔루션으로 MySQL에 다중 AZ 배포를 지원합니다.

> **Important**
>
> 관리형 서비스 환경을 제공하기 위해 Amazon RDS는 DB 인스턴스에 대해 shell 액세스를 제공하지 않습니다. 또한 고급 권한이 필요한 특정 시스템 절차와 테이블에 대한 액세스를 제한합니다. mysql 클라이언트와 같은 표준 SQL 클라이언트를 사용하여 데이터베이스에 액세스할 수 있습니다. 그러나 Telnet 또는 SSH(Secure Shell)를 사용하여 호스트에 직접 액세스할 수는 없습니다.

주제

- Amazon RDS에서 지원되는 MySQL 기능
- Amazon RDS의 MySQL 버전
- MySQL DB 인스턴스에 연결
- MySQL DB 인스턴스 연결 보안
- Amazon RDS Optimized Reads로 RDS for MySQL 쿼리 성능 개선
- RDS Optimized Writes for MySQL을 통한 쓰기 성능 개선
- RDS for MySQL DB 엔진 업그레이드
- MySQL DB 스냅샷 엔진 버전 업그레이드
- Amazon RDS for MySQL DB 인스턴스로 데이터 가져오기
- Amazon RDS에서 MySQL 복제 작업
- RDS for MySQL 액티브-액티브 클러스터 구성
- 복제를 사용하여 MySQL DB 인스턴스에서 데이터 내보내기
- MySQL DB 인스턴스 옵션
- MySQL 파라미터
- MySQL DB 인스턴스에 대한 공통 DBA 작업
- MySQL DB 인스턴스의 현지 시간대

- Amazon RDS for MySQL에 대해 알려진 문제 및 제한
- RDS for MySQL 저장 프로시저 참조

Amazon RDS에서 지원되는 MySQL 기능

RDS for MySQL은 MySQL의 특징과 기능을 대부분 지원합니다. 일부 기능에는 제한된 지원 또는 제한된 권한이 있을 수 있습니다.

데이터베이스 관련 새로운 소식 페이지에서 새로운 Amazon RDS 기능을 필터링할 수 있습니다. [제품(Products)]에서 [Amazon RDS]를 선택합니다. 그런 다음 **MySQL 2022**과 같은 키워드를 사용하여 검색합니다.

> **Note**
>
> 다음 목록이 전부는 아닙니다.

주제

- Amazon RDS for MySQL 메이저 버전에 MySQL 기능 지원
- RDS for MySQL에 대해 지원되는 스토리지 엔진
- Amazon RDS의 MySQL에서 memcached 및 기타 옵션 사용
- Amazon RDS의 MySQL용 InnoDB 캐시 워밍
- RDS for MySQL 8.4의 포용적인 언어 변경 사항
- Amazon RDS가 지원하지 않는 MySQL 기능

Amazon RDS for MySQL 메이저 버전에 MySQL 기능 지원

아래 섹션에서 Amazon RDS for MySQL 메이저 버전의 MySQL 기능 지원에 대해 알아볼 수 있습니다.

주제

- Amazon RDS에서 MySQL 8.4 지원

지원되는 Amazon RDS for MySQL 마이너 버전에 대한 자세한 내용은 Amazon RDS에서 지원되는 MySQL 마이너 버전 섹션을 참조하세요.

Amazon RDS에서 MySQL 8.4 지원

Amazon RDS는 MySQL 버전 8.4 이상을 실행하는 DB 인스턴스에서 다음과 같은 새로운 기능을 지원합니다.

- 암호화 라이브러리 - RDS for MySQL이 OpenSSL을 FIPS 140-3 인증을 받은 AWS Libcrypto(AWS-LC)로 대체했습니다. 자세한 내용은 https://github.com/aws/aws-lc에서 AWS-LC GitHub 리포지토리를 참조하세요.
- TLS 변경 사항 - RDS for MySQL은 TLS 1.2 및 TLS 1.3만 지원합니다. 자세한 내용은 the section called “MySQL 사용 시의 SSL/TLS 지원” 섹션을 참조하세요.
- memcached 지원 - MySQL 8.4에서는 memcached 인터페이스를 더 이상 사용할 수 없습니다. 자세한 내용은 MySQL memcached 지원 섹션을 참조하세요.
- 기본 인증 플러그인 - 기본 인증 플러그인은 `caching_sha2_password`입니다. 자세한 내용은 the section called “MySQL 기본 인증 플러그인” 섹션을 참조하세요.
- **`mysqlpump`** 클라이언트 유틸리티 - MySQL 8.4에서 `mysqlpump` 클라이언트 유틸리티를 더 이상 사용할 수 없습니다. 자세한 내용은 AWS 규범적 지침의 RDS for MySQL에 대한 역할 기반 권한 모델 및 mysqldump 및 mysqlpump를 참조하세요.
- 관리형 복제 저장 프로시저 - `caching_sha2_password`로 구성된 복제 사용자로 복제를 관리하기 위해 저장 프로시저를 사용하는 경우, `SOURCE_SSL=1`을 지정하여 TLS를 구성해야 합니다. `caching_sha2_password`는 RDS for MySQL 8.4의 기본 인증 플러그인입니다.
- 파라미터 동작 변경 사항 - MySQL 8.4에서 다음 파라미터가 변경되었습니다.
  - `innodb_dedicated_server` - 이 파라미터가 이제 기본적으로 활성화됩니다. 자세한 내용은 MySQL 8.4에서 버퍼 풀 크기 및 다시 실행 로그 용량 구성 섹션을 참조하세요.
  - `innodb_buffer_pool` - 데이터베이스 엔진이 이제 이 파라미터를 계산하지만 이 설정을 재정의할 수 있습니다. 자세한 내용은 MySQL 8.4에서 버퍼 풀 크기 및 다시 실행 로그 용량 구성 섹션을 참조하세요.
  - `innodb_redo_log_capacity` - 이제 이 파라미터가 다시 실행 로그 파일의 크기를 제어합니다. 데이터베이스 엔진이 이제 이 파라미터를 계산하지만 이 설정을 재정의할 수 있습니다. 자세한 내용은 MySQL 8.4에서 버퍼 풀 크기 및 다시 실행 로그 용량 구성 섹션을 참조하세요.
- 더 이상 사용되지 않거나 제거된 파라미터 - RDS for MySQL은 MySQL 8.4 DB 인스턴스의 파라미터 그룹에서 다음 파라미터를 제거했습니다. 이제 `innodb_redo_log_capacity` 파라미터가 다시 실행 로그 파일의 크기를 제어합니다.
  - `innodb_log_file_size`
  - `innodb_log_files_in_group`
- 파라미터의 새로운 기본값 – 다음 파라미터에는 MySQL 8.4 DB 인스턴스에 대한 새로운 기본값이 포함됩니다.
  - 성능과 관련된 다양한 MySQL 커뮤니티 파라미터가 변경되었습니다. 자세한 내용은 What is New in MySQL 8.4 since MySQL 8.0을 참조하세요.

프로덕션 인스턴스를 마이그레이션하기 전에 RDS for MySQL 8.4에서 애플리케이션의 성능을 테스트하는 것이 좋습니다.

- `innodb_purge_threads` - InnoDB 기록 목록 길이가 너무 크게 증가하는 것을 방지하기 위해 기본값이 공식 `LEAST({DBInstanceVCPU/2},4)`에 설정됩니다.
- `group_replication_exit_state_action` – 기본값은 `OFFLINE_MODE`이며 MySQL 커뮤니티의 기본값과 일치합니다. 자세한 내용은 RDS for MySQL 액티브-액티브 클러스터에 대한 고려 사항 및 모범 사례 섹션을 참조하세요.
- `binlog_format` – 기본값은 `ROW`이며 MySQL 커뮤니티의 기본값과 일치합니다. 단일 AZ DB 인스턴스 또는 다중 AZ DB 인스턴스의 파라미터는 수정할 수 있지만 다중 AZ DB 클러스터의 파라미터는 수정할 수 없습니다. 다중 AZ DB 클러스터는 반동기식 복제를 사용하며 `binlog_format`이 `MIXED` 또는 `STATEMENT`로 설정되면 복제가 실패합니다.

- 포용적인 언어 변경 사항 - RDS for MySQL 8.4에는 포용적인 언어에 대한 키워드 및 시스템 스키마와 관련된 RDS for MySQL 8.0의 변경 사항이 포함되어 있습니다. 자세한 내용은 RDS for MySQL 8.4의 포용적인 언어 변경 사항 섹션을 참조하세요.

모든 MySQL 8.4 기능 및 변경 사항 목록은 MySQL 설명서에서What Is New in MySQL 8.4 since MySQL 8.0을 참조하세요.

지원되지 않는 기능 목록은 the section called “지원되지 않는 기능” 단원을 참조하세요.

RDS for MySQL에 대해 지원되는 스토리지 엔진

MySQL은 다양한 기능으로 여러 스토리지 엔진을 지원하지만, 모든 기능이 복구와 데이터 내구성에 최적화되어 있지는 않습니다. Amazon RDS는 MySQL DB 인스턴스용 InnoDB 스토리지 엔진을 완전히 지원합니다. 특정 시점 복원 및 스냅샷 복원과 같은 Amazon RDS 기능은 복구 가능한 스토리지 엔진이 필요하며 InnoDB 스토리지 엔진에서만 지원됩니다. 자세한 내용은 MySQL memcached 지원을 참조하세요.

외부 스토리지 엔진은 현재 Amazon RDS for MySQL에서 지원되지 않습니다.

사용자가 생성한 스키마의 경우 MyISAM 스토리지 엔진이 안정적인 복구를 지원하지 않으며 복구 후에 MySQL을 다시 시작할 때 데이터가 손실되거나 손상되어 특정 시점으로 복원 또는 스냅샷 복원 기능이 계획대로 작동하지 않을 수 있습니다. 그래도 Amazon RDS에서 MyISAM을 사용하기로 선택한 경우, 일부 조건에서는 스냅샷이 유용할 수 있습니다.

> **Note**
>
> `mysql` 스키마의 시스템 테이블은 MyISAM 스토리지에 있을 수 있습니다.

기존 MyISAM 테이블을 InnoDB 테이블로 변환하려는 경우 `ALTER TABLE`명령(예, `alter table TABLE_NAME engine=innodb;`)을 사용하면 됩니다. MyISAM과 InnoDB는 각기 다른 장점과 단점을 갖고 있으므로 전환하기 전에 전환이 현재 애플리케이션에 미치는 영향을 충분히 평가해야 합니다.

MySQL 5.1, 5.5, 5.6은 Amazon RDS에서 더 이상 지원되지 않습니다. 그러나 기존의 MySQL 5.1, 5.5, 5.6 스냅샷을 복원할 수는 있습니다. MySQL 5.1, 5.5 또는 5.6 스냅샷을 복원하는 경우 DB 인스턴스가 자동으로 MySQL 5.7로 업그레이드됩니다.

Amazon RDS의 MySQL에서 memcached 및 기타 옵션 사용

대부분의 Amazon RDS DB 엔진은 DB 인스턴스 관련 추가 기능을 선택할 수 있도록 옵션 그룹을 지원합니다. RDS for MySQL DB 인스턴스는 단순 키 기반 캐시인 `memcached` 옵션을 지원합니다. `memcached` 및 기타 옵션에 대한 자세한 내용은 MySQL DB 인스턴스 옵션을 참조하세요. 옵션 그룹 작업에 대한 자세한 내용은 옵션 그룹 작업 단원을 참조하세요.

Amazon RDS의 MySQL용 InnoDB 캐시 워밍

InnoDB 캐시 워밍은 DB 인스턴스가 종료될 때 버퍼 풀의 현재 상태를 저장한 다음 DB 인스턴스가 시작될 때 저장된 정보에서 버퍼 풀을 다시 로드하여 MySQL DB 인스턴스의 성능 향상을 제공할 수 있습니다. 이렇게 하면 보통 데이터베이스 사용에서 "준비"까지의 버퍼 풀에 대한 필요를 무시하고, 대신 알려진 공용 쿼리에 대한 페이지와 함께 버퍼 풀을 미리 로드합니다. 저장된 버퍼 풀 정보를 저장하는 파일은 페이지 자체가 아니라 단지 버퍼 풀에 있는 페이지에 대한 메타데이터만 저장합니다. 따라서 파일은 많은 저장 공간을 요구하지 않습니다. 파일 크기는 캐시 크기의 약 0.2%입니다. 예를 들면, 64GiB의 경우 캐시 워밍 파일 크기는 128MiB입니다. InnoDB 캐시 워밍에 대한 자세한 내용은 MySQL 설명서의 버퍼 풀 상태 저장 및 복원을 참조하세요.

RDS for MySQL DB 인스턴스는 InnoDB 캐시 워밍을 지원합니다. InnoDB 캐시 워밍을 활성화하려면 `innodb_buffer_pool_dump_at_shutdown` 및`innodb_buffer_pool_load_at_startup` 파라미터를 DB 인스턴스용 파라미터 그룹에서 1로 설정합니다. 파라미터 그룹에서 이들 파라미터 값을 변경하면 파라미터 그룹을 사용하는 모든 MySQL DB 인스턴스가 영향을 받습니다. 특정 MySQL DB 인스턴스에 대한 InnoDB 캐시 워밍을 활성화하려면, 이들 인스턴스에 대한 새 파라미터 그룹을 생성해야 할 수도 있습니다. 파라미터 그룹에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.

InnoDB 캐시 워밍은 주로 표준 스토리지를 사용하는 DB 인스턴스를 위해 성능 혜택을 제공합니다. PIOPS 스토리지를 사용하는 경우에는 통상적으로 중요한 성능 혜택이 제공되지 않습니다.

> **Important**
>
> MySQL DB 인스턴스가 정상적으로 종료되지 않는 경우(예: 장애 조치 도중), 버퍼 풀 상태가 디스크에 저장되지 않습니다. 이 경우 MySQL은 DB 인스턴스가 다시 시작될 때 이용 가능한 모든 버퍼 풀 파일을 로드합니다. 어떤 손상도 발생하지 않지만, 복원된 버퍼 풀은 대부분의 경우 다시 시작하기 이전의 버퍼 풀 최신 상태를 반영하지 못할 수도 있습니다. 시작 시 InnoDB 캐시를 워밍업하기 위해 버퍼 풀의 최신 상태를 이용할 수 있게 하려면, "요청 시" 버퍼 풀을 주기적으로 덤프하는 것이 좋습니다.
> 이벤트를 생성하여 버퍼 풀을 자동으로 그리고 정기적으로 덤프할 수 있습니다. 예를 들면, 다음 문은 매 시간마다 버퍼 풀을 덤프하는 이름이 `periodic_buffer_pool_dump`인 이벤트를 생성합니다.
>
> ```
> CREATE EVENT periodic_buffer_pool_dump
> ON SCHEDULE EVERY 1 HOUR
> DO CALL mysql.rds_innodb_buffer_pool_dump_now();
> ```
>
> MySQL 이벤트에 대한 자세한 내용은 MySQL 설명서의 이벤트 구문을 참조하십시오.

요청 시 버퍼 풀 덤핑 및 로딩

'요청 시' InnoDB 캐시를 저장하고 로드할 수 있습니다.

- 버퍼 풀의 현재 상태를 디스크에 덤프하려면 mysql.rds_innodb_buffer_pool_dump_now 저장 프로시저를 호출합니다.
- 디스크에서 저장된 버퍼 풀의 상태를 로드하려면 mysql.rds_innodb_buffer_pool_load_now 저장 프로시저를 호출합니다.
- 진행 중인 로드 작업을 취소하려면 mysql.rds_innodb_buffer_pool_load_abort 저장 프로시저를 호출합니다.

RDS for MySQL 8.4의 포용적인 언어 변경 사항

RDS for MySQL 버전 8.4에는 포용적 언어에 대한 키워드 및 시스템 스키마와 관련된 MySQL 8.4의 변경 사항이 포함되어 있습니다. 예를 들어 `SHOW REPLICA STATUS` 명령은 `SHOW SLAVE STATUS`를 대체합니다.

주제

- 구성 파라미터 이름 변경 사항
- 저장 프로시저 이름 변경 사항

구성 파라미터 이름 변경 사항

다음 구성 파라미터는 RDS for MySQL 8.4에서 이름이 변경되었습니다.

호환성을 위해 두 이름 중 하나를 사용하여 `mysql` 클라이언트에서 파라미터 이름을 확인할 수 있습니다. 사용자 정의 MySQL 8.4 파라미터 그룹의 값을 수정할 때는 새 이름만 사용 수 있습니다. 자세한 내용은 기본 및 사용자 지정 파라미터 그룹 섹션을 참조하세요.

제거할 이름	새 이름 또는 기본 이름
`init_slave`	`init_replica`
`log_slave_updates`	`log_replica_updates`
`log_slow_slave_statements`	`log_slow_replica_statements`
`rpl_stop_slave_timeout`	`rpl_stop_replica_timeout`
`skip_slave_start`	`skip_replica_start`
`slave_checkpoint_group`	`replica_checkpoint_group`
`slave_checkpoint_period`	`replica_checkpoint_period`
`slave_compressed_protocol`	`replica_compressed_protocol`
`slave_exec_mode`	`replica_exec_mode`
`slave_load_tmpdir`	`replica_load_tmpdir`

제거할 이름	새 이름 또는 기본 이름
slave_max_allowed_packet	replica_max_allowed_packet
slave_net_timeout	replica_net_timeout
slave_parallel_type	replica_parallel_type
slave_parallel_workers	replica_parallel_workers
slave_pending_jobs_size_max	replica_pending_jobs_size_max
slave_preserve_commit_order	replica_preserve_commit_order
slave_skip_errors	replica_skip_errors
slave_sql_verify_checksum	replica_sql_verify_checksum
slave_transaction_retries	replica_transaction_retries
slave_type_conversions	replica_type_conversions
sql_slave_skip_counter	sql_replica_skip_counter

> **Note**
>
> Amazon RDS는 NDB 클러스터를 지원하지 않으므로 replica_allow_batching 파라미터를 사용할 수 없습니다.

저장 프로시저 이름 변경 사항

다음 저장 프로시저는 RDS for MySQL 8.4에서 이름이 변경되었습니다.

호환성을 위해 초기 RDS for MySQL 8.4 릴리스에서 두 이름 중 하나를 사용할 수 있습니다. 향후 릴리스에서 기존 프로시저 이름이 제거되게 됩니다. 자세한 내용은 이진수 로그(binlog) 복제 구성, 시작 및 중지 섹션을 참조하세요.

제거할 이름	새 이름 또는 기본 이름
`mysql.rds_next_master_log`	`mysql.rds_next_source_log`
`mysql.rds_reset_external_master`	`mysql.rds_reset_external_source`
`mysql.rds_set_external_master`	`mysql.rds_set_external_source`
`mysql.rds_set_external_master_with_auto_position`	`mysql.rds_set_external_source_with_auto_position`
`mysql.rds_set_external_master_with_delay`	`mysql.rds_set_external_source_with_delay`
`mysql.rds_set_master_auto_position`	`mysql.rds_set_source_auto_position`

Amazon RDS가 지원하지 않는 MySQL 기능

Amazon RDS는 현재 다음과 같은 MySQL 기능은 지원하지 않습니다.

- 인증 플러그인
- 시스템 로그에 오류 로깅
- InnoDB Tablespace 암호화
- NDB 클러스터

- 암호 보안 수준 플러그인
- 지속된 시스템 변수
- 리라이터 쿼리 다시 쓰기 플러그인
- 다중 AZ DB 클러스터를 제외한 반동기식 복제
- 전송 가능한 테이블스페이스
- X 플러그인

관리형 서비스 환경을 제공하기 위해 Amazon RDS는 DB 인스턴스에 대해 shell 액세스를 제공하지 않습니다. 또한 고급 권한이 필요한 특정 시스템 절차와 테이블에 대한 액세스를 제한합니다. Amazon RDS는 표준 SQL 클라이언트 애플리케이션을 사용하여 DB 인스턴스의 데이터베이스에 대한 액세스를 지원합니다. Amazon RDS는 Telnet, SSH(보안 셸) 또는 Windows 원격 데스크톱 연결을 사용하여 DB 인스턴스에 대한 직접적인 호스트 액세스를 허용하지 않습니다. DB 인스턴스를 생성할 때 사용자는 해당 인스턴스의 모든 데이터베이스에 대한 db_owner 역할을 할당받게 되며, 백업을 위해 사용된 권한을 제외한 모든 데이터베이스 수준의 권한을 갖게 됩니다. Amazon RDS는 백업을 관리합니다.

Amazon RDS의 MySQL 버전

MySQL의 경우 버전 번호는 버전 = X.Y.Z로 구성됩니다. Amazon RDS 용어에서 X.Y는 메이저 버전을 나타내고 Z는 마이너 버전 번호를 나타냅니다. Amazon RDS 구현을 위해서, 메이저 버전 번호가 변경될 경우—(예: 버전 5.7에서 8.0으로) 이를 메이저 버전 변경으로 간주합니다. 마이너 버전 번호만 변경된 경우(예: 버전 8.0.32에서 8.0.34로 변경)에는 마이너 버전 변경으로 간주합니다.

주제

- Amazon RDS에서 지원되는 MySQL 마이너 버전
- Amazon RDS에서 지원되는 MySQL 메이저 버전
- Amazon RDS for MySQL에 대한 Amazon RDS 추가 지원 버전
- 데이터베이스 미리 보기 환경 작업
- 데이터베이스 미리 보기 환경의 MySQL 버전 8.4
- 데이터베이스 미리 보기 환경의 MySQL 버전 8.3
- 데이터베이스 미리 보기 환경의 MySQL 버전 8.2
- 데이터베이스 미리 보기 환경의 MySQL 버전 8.1
- 더 이상 사용되지 않는 Amazon RDS for MySQL 버전

Amazon RDS에서 지원되는 MySQL 마이너 버전

Amazon RDS는 현재 다음과 같은 MySQL 마이너 버전을 지원합니다.

> **Note**
>
> 월과 연도만 있는 날짜는 대략적인 날짜이며 알 수 있는 정확한 날짜로 업데이트됩니다.
> 마이너 버전에서는 Amazon RDS 확장 지원을 사용할 수 없습니다.

다음 표는 현재 Amazon RDS가 지원하는 MySQL 8.0의 마이너 버전을 보여줍니다.

MySQL 엔진 버전	커뮤니티 릴리스 날짜	RDS 릴리스 날짜	RDS 표준 지원 종료일
8.0.39	2024년 7월 23일	2024년 8월 12일	2025년 9월
8.0.37	2024년 4월 30일	2024년 6월 18일	2025년 9월

MySQL 엔진 버전	커뮤니티 릴리스 날짜	RDS 릴리스 날짜	RDS 표준 지원 종료일
8.0.36	2024년 1월 16일	2024년 2월 12일	2025년 3월
8.0.35	2023년 10월 25일	2023년 11월 9일	2025년 3월
8.0.34	2023년 7월 18일	2023년 8월 9일	2025년 3월
8.0.33	2023년 4월 18일	2023년 6월 15일	2025년 3월
8.0.32	2023년 1월 17일	2023년 2월 7일	2025년 3월

다음 표는 Amazon RDS 추가 지원에서 사용할 수 있는 MySQL 5.7의 마이너 버전을 보여줍니다.

MySQL 엔진 버전	커뮤니티 릴리스 날짜	RDS 릴리스 날짜	RDS 추가 지원 종료일
5.7.44-RDS.20240529*	해당 사항 없음	2024년 6월 25일	2025년 9월
5.7.44-RDS.20240408*	해당 사항 없음	2024년 5월 17일	2025년 9월
5.7.44	2023년 10월 25일	2023년 11월 2일	2025년 3월

* MySQL 커뮤니티는 메이저 버전 5.7을 사용 중지했으며, 새로운 마이너 버전은 릴리스하지 않을 예정입니다. 이 버전은 Amazon RDS에서 RDS 추가 지원이 적용되는 MySQL 5.7 데이터베이스의 중요 보안 패치 및 버그 수정과 함께 출시한 마이너 버전입니다. 마이너 버전에 대한 자세한 내용은 the section called “RDS for MySQL에 대한 Amazon RDS 추가 지원 버전” 섹션을 참조하세요. RDS 추가 지원에 대한 자세한 내용은 Amazon RDS를 사용한 Amazon RDS 추가 지원 섹션을 참조하세요.

마이너 버전은 메이저 버전보다 먼저 표준 지원이 종료될 수 있습니다. 예를 들어 마이너 버전 8.0.28은 2024년 3월 28일에 표준 지원 종료일에 도달한 반면 메이저 버전 8.0은 2026년 7월 31일에 표준 지원 종료일에 도달합니다. RDS는 이 날짜 사이에 MySQL 커뮤니티에서 릴리스하는 8.0.* 마이너 버전을 추가로 지원할 예정입니다.

새 DB 인스턴스를 생성할 때는 현재 지원되는 모든 MySQL 버전을 지정할 수 있습니다. 메이저 버전(예: MySQL 5.7) 및 지정된 메이저 버전에 대해 지원되는 모든 마이너 버전을 지정할 수 있습니다. 버전이 지정되지 않은 경우 Amazon RDS는 지원되는 버전(보통 최신 버전)을 기본값으로 설정합니다.

메이저 버전이 지정되었지만 마이너 버전이 지정되지 않은 경우, Amazon RDS는 고객이 지정한 메이저 버전의 최근 릴리스를 기본값으로 설정합니다. 지원되는 버전 목록과 새로 만든 DB 인스턴스의 기본값을 보려면 `describe-db-engine-versions` AWS CLI 명령을 사용합니다.

예를 들어 RDS for MySQL 버전 목록을 보려면 다음 CLI 명령을 실행합니다.

```
aws rds describe-db-engine-versions --engine mysql --query "*[].
{Engine:Engine,EngineVersion:EngineVersion}" --output text
```

기본 MySQL 버전은 AWS 리전에 따라 다를 수 있습니다. 특정 마이너 버전으로 DB 인스턴스를 생성하려면 DB 인스턴스 생성 중에 마이너 버전을 지정합니다. 다음 AWS CLI 명령을 사용하여 AWS 리전의 기본 마이너 버전을 확인할 수 있습니다.

```
aws rds describe-db-engine-versions --default-only --engine mysql
 --engine-version major-engine-version --region region --query "*[].
{Engine:Engine,EngineVersion:EngineVersion}" --output text
```

*major-engine-version*을 메이저 엔진 버전으로 바꾸고 *##*을 AWS 리전으로 바꿉니다. 예를 들어, 다음 AWS CLI 명령은 5.7 메이저 버전과 미국 서부(오레곤) AWS 리전(us-west-2)에 대한 기본 MySQL 마이너 엔진 버전을 반환합니다.

```
aws rds describe-db-engine-versions --default-only --engine mysql --engine-version 5.7
 --region us-west-2 --query "*[].{Engine:Engine,EngineVersion:EngineVersion}" --output
 text
```

Amazon RDS를 통해 사용자는 MySQL 인스턴스를 Amazon RDS가 지원하는 새 메이저 버전으로 언제 업그레이드할지를 제어합니다. 특정 MySQL 버전과의 호환성을 유지하고, 프로덕션 환경에 배포하기 전에 애플리케이션으로 새 버전을 테스트하고, 가장 원하는 일정에 맞춰 메이저 버전 업그레이드를 수행할 수 있습니다.

자동 마이너 버전 업그레이드가 활성화되는 경우, 새 MySQL 마이너 버전이 Amazon RDS에서 지원되면 DB 인스턴스가 새 MySQL 마이너 버전으로 자동 업그레이드됩니다. 이 패치는 예약 유지보수 중에 발생합니다. DB 인스턴스를 수정하여 자동 마이너 버전 업그레이드를 활성화 또는 비활성화할 수 있습니다.

자동으로 예약된 업그레이드를 사용하지 않기로 선택한 경우, 사용자는 메이저 버전 업데이트를 위해 선택한 것과 동일한 프로시저에 따라 지원되는 마이너 버전 릴리스로 수동으로 업그레이드할 수 있습니다. 자세한 정보는 DB 인스턴스 엔진 버전 업그레이드 섹션을 참조하세요.

Amazon RDS는 현재 MySQL 버전 5.6에서 버전 5.7로, 그리고 MySQL 버전 5.7에서 버전 8.0으로의 메이저 버전 업그레이드를 지원합니다. 메이저 버전 업그레이드에 약간의 호환성 위험이 수반되므로 업그레이드가 자동으로 이루어지지 않습니다. 즉, DB 인스턴스 수정을 요청해야 합니다. 프로덕션 인스턴스를 업그레이드하기 전에 모든 업그레이드를 철저하게 테스트해야 합니다. MySQL DB 인스턴스 업그레이드에 대한 자세한 내용은 RDS for MySQL DB 엔진 업그레이드를 참조하십시오.

기존 DB 인스턴스의 DB 스냅샷을 만들고 DB 스냅샷에서 복구해 새 DB 인스턴스를 만든 다음 새로운 DB 인스턴스에 대한 버전 업그레이드를 시작함으로써, 업그레이드하기 전에 새 버전과 비교하여 DB 인스턴스를 테스트할 수 있습니다. 그런 다음 원래의 DB 인스턴스에 대한 업그레이드 여부를 결정하기 전에 DB 인스턴스의 업그레이드된 복제본에서 안전하게 실험을 진행할 수 있습니다.

Amazon RDS의 MySQL 마이너 버전

마이너 버전

- MySQL 버전 8.0.39
- MySQL 버전 8.0.37

MySQL 버전 8.0.39

MySQL 버전 8.0.39는 이제 Amazon RDS에서 사용할 수 있습니다. 이 릴리스에는 MySQL 커뮤니티와 Amazon RDS에서 추가한 수정 및 개선 사항이 포함되어 있습니다.

새로운 기능 및 개선 사항

- `SESSION_VARIABLES_ADMIN` 권한으로 `sql_log_off`가 제대로 작동하지 않도록 하는 버그를 수정했습니다.
- 마스터 사용자가 다른 데이터베이스 사용자에게 `SESSION_VARIABLE_ADMIN` 권한을 부여할 수 없었던 버그를 수정했습니다.
- RDS 제공 저장 프로시저를 실행하는 동안 데이터 정렬이 잘못 혼합되어 발생하는 버그를 수정했습니다.

MySQL 버전 8.0.37

MySQL 버전 8.0.37은 이제 Amazon RDS에서 사용할 수 있습니다. 이 릴리스에는 MySQL 커뮤니티와 Amazon RDS에서 추가한 수정 및 개선 사항이 포함되어 있습니다.

새로운 기능 및 개선 사항

인스턴트 DDL 문을 실행한 후 업데이트하면 어설션 실패로 이어지는 버그가 수정되었습니다.

Amazon RDS에서 지원되는 MySQL 메이저 버전

RDS for MySQL 메이저 버전은 해당 커뮤니티 버전에 대한 커뮤니티 사용 주기가 끝날 때까지 표준 지원에 따라 사용할 수 있습니다. RDS 표준 지원 종료일이 지난 후에도 유료로 메이저 버전을 계속 실행할 수 있습니다. 자세한 내용은 Amazon RDS를 사용한 Amazon RDS 추가 지원 및 Amazon RDS for MySQL 요금을 참조하세요.

다음 날짜를 사용하여 테스트 및 업그레이드 주기를 계획할 수 있습니다.

> **Note**
>
> 월과 연도만 있는 날짜는 대략적인 날짜이며 알 수 있는 정확한 날짜로 업데이트됩니다.

MySQL 메이저 버전	커뮤니티 릴리스 날짜	RDS 릴리스 날짜	커뮤니티 수명 종료 날짜	RDS 표준 지원 종료일	RDS 추가 지원 요금 부과 시작일(1년)	RDS 추가 지원 요금 부과 시작일(3년)	RDS 추가 지원 종료일
MySQL 8.0	2018년 4월 19일	2018년 10월 23일	2026년 4월	2026년 7월 31일	2026년 8월 1일	2028년 8월 1일	2029년 7월 31일
MySQL 5.7*	2015년 10월 21일	2016년 2월 22일	2023년 10월	2024년 2월 29일	2024년 3월 1일	2026년 3월 1일	2027년 2월 28일

* MySQL 5.7은 이제 RDS 추가 지원하에서만 사용할 수 있습니다. 자세한 내용은 Amazon RDS를 사용한 Amazon RDS 추가 지원 단원을 참조하십시오.

Amazon RDS for MySQL에 대한 Amazon RDS 추가 지원 버전

다음 콘텐츠에는 RDS for MySQL 버전에 대한 모든 RDS 추가 지원 릴리스가 나와 있습니다.

출시

- RDS for MySQL 버전 5.7.44-RDS.20240529에 대한 RDS 추가 지원
- RDS for MySQL 버전 5.7.44-RDS.20240408에 대한 RDS 추가 지원

RDS for MySQL 버전 5.7.44-RDS.20240529에 대한 RDS 추가 지원

RDS for MySQL 버전 5.7.44-RDS.20240529에 대한 RDS 추가 지원을 사용할 수 있습니다.

버그 수정:

- `fix_after_pullout`을 구현하여 `field.cc` 어설션 실패를 수정했습니다.
- 특정 SQL 쿼리에 대해 클라이언트에 메타데이터를 반환할 때 발생하는 널 포인터 오류를 수정했습니다. 이러한 쿼리에는 `SELECT` 절에 동적 파라미터와 하위 쿼리가 포함되어 있었습니다.
- 느슨한 인덱스 스캔 또는 인덱스의 비연속 범위 스캔에 `GROUP BY`를 사용할 때 잘못된 결과가 표시되던 문제를 수정했습니다.
- 지속성 중 MySQL 충돌 시 GTID 정보가 손실되는 문제를 수정했습니다.
- InnoDB 트랜잭션이 무기한 중단될 수 있는 경쟁 조건을 수정했습니다.
- 그룹 복제의 인증 정보 정리에서 경쟁 조건을 수정했습니다.
- 동시 페이지 작업 시 역방향 인덱스 스캔 문제를 수정했습니다.
- 동시 시나리오에서 일관되지 않은 전체 텍스트 검색(FTS) 상태 문제를 수정했습니다.
- 테이블 삭제 시 변경 버퍼와 관련된 어설션 문제를 수정했습니다.
- 모든 플러그인 유형에서 `deinit` 함수 호출에 대한 동작이 통합되었습니다.

CVE 수정됨:

- CVE-2024-20963
- CVE-2024-20963
- CVE-2024-20998
- CVE-2024-21009
- CVE-2024-21054
- CVE-2024-21055
- CVE-2024-21057
- CVE-2024-21062
- CVE-2024-21008
- CVE-2024-21013
- CVE-2024-21047

- CVE-2024-21087
- CVE-2024-21096

RDS for MySQL 버전 5.7.44-RDS.20240408에 대한 RDS 추가 지원

RDS for MySQL 버전 5.7.44-RDS.20240408에 대한 RDS 추가 지원을 사용할 수 있습니다.

이 릴리스에는 다음 CVE에 대한 패치가 포함되어 있습니다.

- CVE-2024-20963

데이터베이스 미리 보기 환경 작업

2023년 7월, Oracle은 MySQL의 새로운 릴리스 모델을 발표했습니다. 이 모델에는 Innovation 릴리스와 LTS 릴리스라는 두 가지 유형의 릴리스가 포함됩니다. Amazon RDS는 MySQL Innovation 릴리스를 RDS 미리 보기 환경에서 사용할 수 있도록 합니다. MySQL Innovation 릴리스에 대한 자세한 내용은 MySQL Innovation 및 장기 지원(LTS) 버전 소개를 참조하세요.

데이터베이스 미리 보기 환경의 RDS for MySQL DB 인스턴스는 다른 RDS for MySQL DB 인스턴스와 기능적으로 유사합니다. 하지만 프로덕션 워크로드에는 데이터베이스 미리 보기 환경을 사용할 수 없습니다.

미리 보기 환경에는 다음과 같은 제한 사항이 적용됩니다.

- Amazon RDS는 생성 시점으로부터 60일이 지난 모든 DB 인스턴스를 백업 및 스냅샷과 함께 삭제합니다.
- 범용 SSD와 프로비저닝된 IOPS SSD 스토리지만 사용할 수 있습니다.
- DB 인스턴스와 관련해서는 지원의 도움을 받을 수 없습니다. 대신 AWS 관리형 Q&A 커뮤니티인 AWS re:Post에 질문을 게시할 수 있습니다.
- DB 인스턴스의 스냅샷을 프로덕션 환경으로 복제할 수 없습니다.

미리 보기에서 지원되는 옵션은 다음과 같습니다.

- db.m6i, db.r6i, db.m6g, db.m5, db.t3, db.r6g, and db.r5 DB 인스턴스 클래스를 사용하여 DB 인스턴스를 생성할 수 있습니다. RDS 인스턴스 클래스에 대한 자세한 내용은 DB 인스턴스 클래스 단원을 참조하세요.
- 단일 AZ 배포와 다중 AZ 배포를 모두 사용할 수 있습니다.

- 표준 MySQL 덤프 및 로드 함수를 사용하여 데이터베이스 미리 보기 환경에서 데이터베이스를 내보내거나 데이터베이스 미리 보기 환경으로 데이터베이스를 가져올 수 있습니다.

데이터베이스 미리 보기 환경에서 지원하지 않는 기능

다음 기능은 데이터베이스 미리 보기 환경에서 사용할 수 없습니다.

- 리전 간 스냅샷 복제
- 리전 간 읽기 전용 복제본
- RDS 프록시

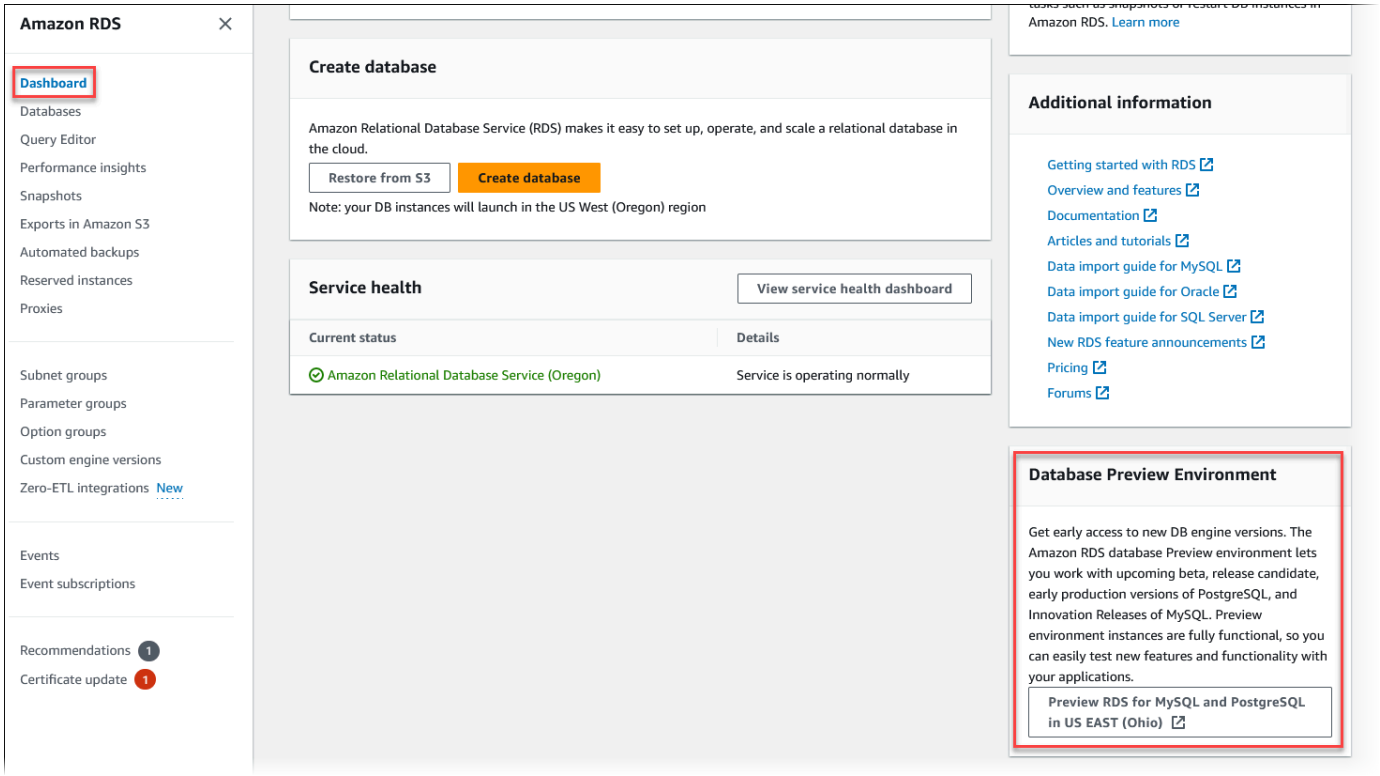
데이터베이스 미리 보기 환경에서 새 DB 인스턴스 생성

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 데이터베이스 미리 보기 환경에서 DB 인스턴스를 생성할 수 있습니다.

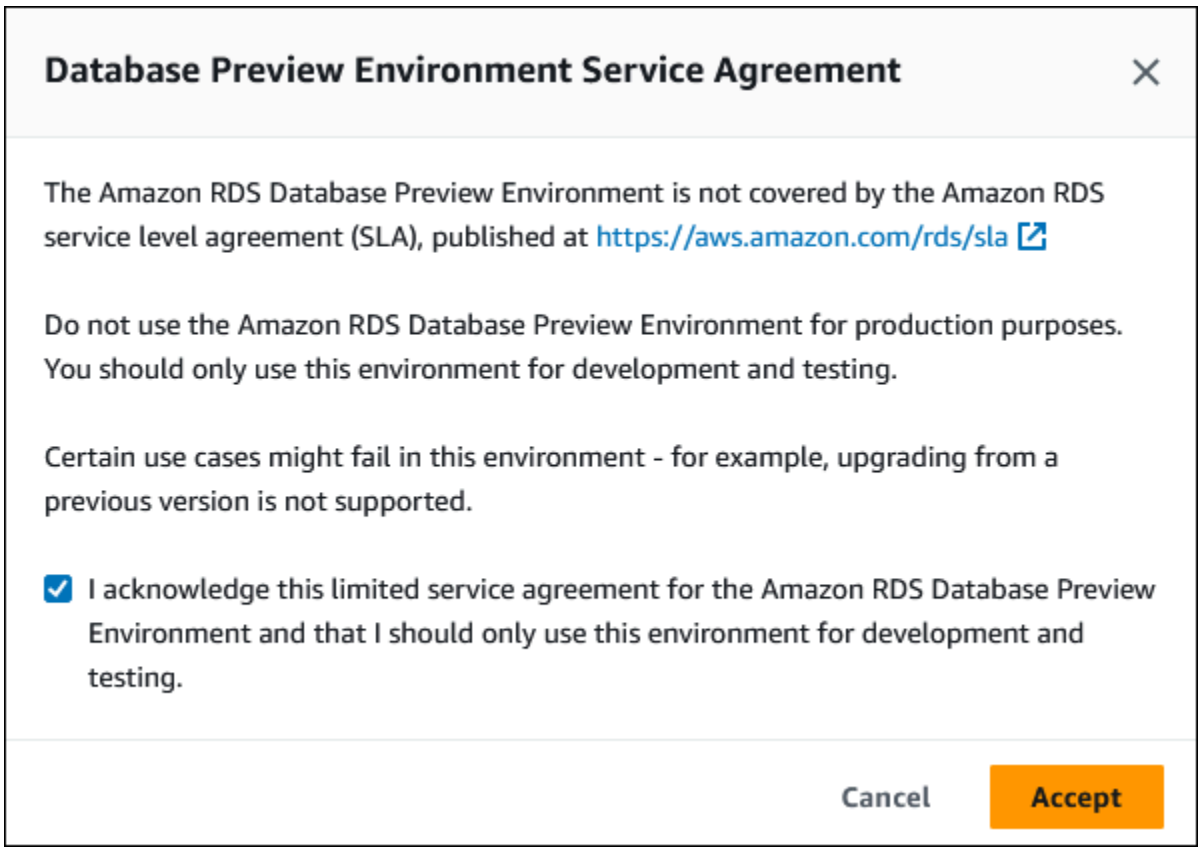
콘솔

데이터베이스 미리 보기 환경에서 DB 인스턴스를 생성하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 대시보드를 선택합니다.
3. 다음 그림과 같이 대시보드 페이지에서 데이터베이스 미리 보기 환경 섹션을 찾습니다.



데이터베이스 미리 보기 환경으로 바로 이동할 수도 있습니다. 계속하려면 먼저 제한 사항을 확인하고 수락해야 합니다.



4. RDS for MySQL DB 인스턴스를 생성하려면 일반적으로 Amazon RDS DB 인스턴스를 생성할 때와 동일한 프로세스를 따릅니다. 자세한 내용은 DB 인스턴스 생성의 콘솔 절차를 참조하세요.

AWS CLI

AWS CLI를 사용하여 데이터베이스 미리 보기 환경에 DB 인스턴스를 생성하려면 다음 엔드포인트를 사용합니다.

```
rds-preview.us-east-2.amazonaws.com
```

RDS for MySQL DB 인스턴스를 생성하려면 일반적으로 Amazon RDS DB 인스턴스를 생성할 때와 동일한 프로세스를 따릅니다. 자세한 내용은 DB 인스턴스 생성의 AWS CLI 절차를 참조하세요.

RDS API

RDS API를 사용하여 데이터베이스 미리 보기 환경에 DB 인스턴스를 생성하려면 다음 엔드포인트를 사용합니다.

```
rds-preview.us-east-2.amazonaws.com
```

RDS for MySQL DB 인스턴스를 생성하려면 일반적으로 Amazon RDS DB 인스턴스를 생성할 때와 동일한 프로세스를 따릅니다. 자세한 내용은 DB 인스턴스 생성의 RDS API 절차를 참조하세요.

데이터베이스 미리 보기 환경의 MySQL 버전 8.4

이제 MySQL 버전 8.4를 Amazon RDS 데이터베이스 미리 보기 환경에서 사용할 수 있습니다. MySQL 버전 8.4는 커뮤니티의 최신 LTS 릴리스이며, MySQL 8.4.0의 변경 사항에 설명된 여러 개선 사항이 포함되어 있습니다. 데이터베이스 미리 보기 환경을 사용하여 모든 AWS 리전 프로덕션 워크로드에 사용할 수 있게 되기 전에 이 릴리스를 기준으로 워크로드를 테스트할 수 있습니다.

데이터베이스 미리 보기 환경의 MySQL 버전 8.4는 Amazon RDS가 모든 AWS 리전에서 프로덕션 워크로드용으로 릴리스하는 것과 다를 수 있습니다. 다음 목록에는 변경될 수 있는 기능이 포함되어 있습니다. 단, 이 목록이 전부는 아닙니다.

- RDS for MySQL 8.4 파라미터 그룹 정의입니다. 예를 들어, Amazon RDS는 파라미터를 추가 또는 제거하거나, 이름을 변경하거나, 파라미터 기본값을 바꿀 수 있습니다.
- 권한 모델입니다.
- TLS 라이브러리입니다.

데이터베이스 미리 보기 환경에 대한 자세한 내용은 the section called “ 데이터베이스 미리 보기 환경” 섹션을 참조하세요. 콘솔에서 미리 보기 환경에 액세스하려면 https://console.aws.amazon.com/rds-preview/를 선택합니다.

데이터베이스 미리 보기 환경의 MySQL 버전 8.3

이제 MySQL 버전 8.3을 Amazon RDS 데이터베이스 미리 보기 환경에서 사용할 수 있습니다. MySQL 버전 8.3에는 Changes in MySQL 8.3.0에 설명된 여러 개선 사항이 포함되어 있습니다.

데이터베이스 미리 보기 환경에 대한 자세한 내용은 the section called “ 데이터베이스 미리 보기 환경” 섹션을 참조하세요. 콘솔에서 미리 보기 환경에 액세스하려면 https://console.aws.amazon.com/rds-preview/를 선택합니다.

데이터베이스 미리 보기 환경의 MySQL 버전 8.2

이제 MySQL 버전 8.2를 Amazon RDS 데이터베이스 미리 보기 환경에서 사용할 수 있습니다. MySQL 버전 8.2에는 MySQL 8.2.0의 변경 사항에 설명된 여러 개선 사항이 포함되어 있습니다.

데이터베이스 미리 보기 환경에 대한 자세한 내용은 the section called “ 데이터베이스 미리 보기 환경” 섹션을 참조하세요. 콘솔에서 미리 보기 환경에 액세스하려면 https://console.aws.amazon.com/rds-preview/를 선택합니다.

데이터베이스 미리 보기 환경의 MySQL 버전 8.1

이제 MySQL 버전 8.1을 Amazon RDS 데이터베이스 미리 보기 환경에서 사용할 수 있습니다. MySQL 버전 8.1에는 MySQL 8.1.0의 변경 사항에 설명된 여러 개선 사항이 포함되어 있습니다.

데이터베이스 미리 보기 환경에 대한 자세한 내용은 the section called “ 데이터베이스 미리 보기 환경” 섹션을 참조하세요. 콘솔에서 미리 보기 환경에 액세스하려면 https://console.aws.amazon.com/rds-preview/를 선택합니다.

더 이상 사용되지 않는 Amazon RDS for MySQL 버전

Amazon RDS for MySQL 버전 5.1, 5.5, 5.6은 더 이상 사용되지 않습니다.

MySQL에 대한 Amazon RDS 사용 중단 정책에 대한 자세한 내용은 Amazon RDS FAQ를 참조하세요.

MySQL DB 인스턴스에 연결

MySQL 데이터베이스 엔진을 실행하는 DB 인스턴스에 연결하려면 우선 DB 인스턴스를 생성해야 합니다. 자세한 정보는 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요. Amazon RDS가 DB 인스턴스를 프로비저닝한 후에는 표준 MySQL 클라이언트 애플리케이션 또는 유틸리티를 사용하여 인스턴스에 연결할 수 있습니다. 연결 문자열에는 호스트 파라미터로 DB 인스턴스 엔드포인트의 DNS 주소와 포트 파라미터로 DB 인스턴스 엔드포인트의 포트 번호를 지정합니다.

RDS DB 인스턴스에 인증하려면, MySQL 및 AWS Identity and Access Management(IAM) 데이터베이스 인증의 인증 방법 중 하나를 사용할 수 있습니다.

- MySQL의 인증 방법 중 하나를 사용하는 MySQL 인증 방법을 확인하려면, MySQL의 인증 방법을 참조하십시오.
- IAM 데이터베이스 인증을 사용하는 MySQL 인증 방법을 확인하려면 MariaDB, MySQL 및 PostgreSQL IAM 데이터베이스 인증 단원을 참조하십시오.

MySQL 명령줄 클라이언트 같은 도구를 사용하여 MySQL DB 인스턴스에 연결할 수 있습니다. MySQL 명령줄 클라이언트 사용에 대한 자세한 내용은 MySQL 문서의 mysql - MySQL 명령줄 클라이언트를 참조하세요. 연결에 사용할 수 있는 GUI 기반 애플리케이션 중 하나는 MySQL Workbench입니다. 자세한 정보는 MySQL Workbench 다운로드 페이지 단원을 참조하십시오. MySQL 설치(MySQL 명령줄 클라이언트 포함)에 대한 정보는 MySQL 설치 및 업그레이드를 참조하세요.

Amazon VPC 외부에서 DB 인스턴스에 연결하려면 DB 인스턴스에 공개적으로 액세스할 수 있어야 하고, DB 인스턴스 보안 그룹의 인바운드 규칙을 사용하여 액세스 권한을 부여해야 하며, 기타 요구 사항을 충족해야 합니다. 자세한 내용은 Amazon RDS DB 인스턴스에 연결할 수 없음 단원을 참조하십시오.

MySQL DB 인스턴스로의 연결에 보안 소켓 계층(SSL) 또는 전송 계층 보안(TLS) 암호화를 사용할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL DB 인스턴스에 대한 SSL/TLS 지원을 참조하세요. AWS Identity and Access Management(IAM) 데이터베이스 인증을 사용하는 경우 SSL/TLS 연결을 사용해야 합니다. 자세한 내용은 MariaDB, MySQL 및 PostgreSQL IAM 데이터베이스 인증을 참조하세요.

웹 서버에서 DB 인스턴스에 연결할 수도 있습니다. 자세한 내용은 자습서: 웹 서버 및 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.

> **Note**
>
> SSL을 사용한 MariaDB DB 인스턴스 연결에 대한 자세한 내용은 MariaDB DB 인스턴스에 연결 단원을 참조하십시오.

RDS for MySQL DB 인스턴스를 찾아 연결하려면 다음 주제를 참조하세요.

주제

- RDS for MySQL DB 인스턴스에 대한 연결 정보 찾기
- MySQL 명령줄 클라이언트 설치
- MySQL 명령줄 클라이언트에서 연결(암호화되지 않음)
- MySQL Workbench에서 연결
- AWS JDBC 드라이버, AWS Python 드라이버 및 MySQL용 AWS ODBC 드라이버를 사용하여 RDS for MySQL에 연결
- MySQL DB 인스턴스에 대한 연결 문제 해결

RDS for MySQL DB 인스턴스에 대한 연결 정보 찾기

DB 인스턴스의 연결 정보에는 엔드포인트, 포트 및 유효한 데이터베이스 사용자(예: 마스터 사용자)가 포함됩니다. 예를 들어 엔드포인트 값이 `mydb.123456789012.us-east-1.rds.amazonaws.com`이라고 가정합니다. 이 경우 포트 값은 `3306`이고 데이터베이스 사용자는 `admin`입니다. 이 정보를 바탕으로 연결 문자열에 다음 값을 지정합니다.

- 호스트 또는 호스트 이름 또는 DNS 이름에 `mydb.123456789012.us-east-1.rds.amazonaws.com`을 지정합니다.
- 포트에 대해 `3306`을 지정합니다.
- 사용자에 `admin`을 지정합니다.

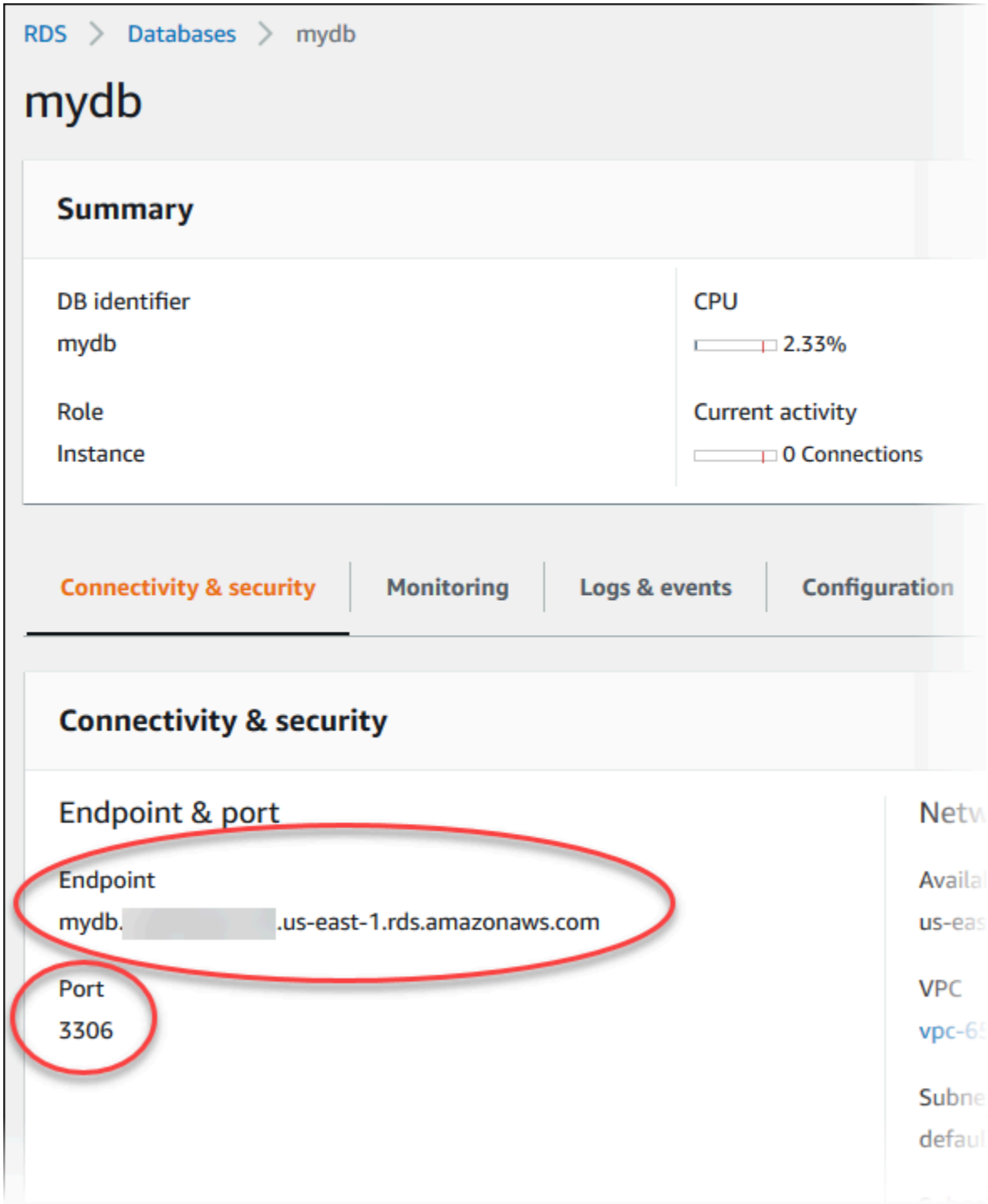
DB 인스턴스에 연결하려면 MySQL DB 엔진에 대해 임의의 클라이언트를 사용합니다. 예를 들어 MySQL 명령줄 클라이언트 또는 MySQL 워크벤치를 사용할 수 있습니다.

DB 인스턴스에 대한 연결 정보를 찾으려면 AWS Management Console, AWS CLI describe-db-instances 명령 또는 Amazon RDS API DescribeDBInstances 작업을 사용하여 세부 정보를 나열하면 됩니다.

콘솔

AWS Management Console에서 DB 인스턴스에 대한 연결 정보를 찾으려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [데이터베이스(Database)] 를 선택하여 DB 인스턴스 목록을 표시합니다.
3. MySQL DB 인스턴스의 이름을 선택하여 세부 정보를 표시합니다.
4. Connectivity & security(연결 및 보안) 탭에서 엔드포인트를 복사합니다. 또한 포트 번호를 적어 둡니다. DB 인스턴스에 연결하려면 엔드포인트와 포트 번호가 모두 필요합니다.



5. 마스터 사용자 이름을 찾아야 하는 경우 [구성(Configuration)] 탭을 선택하고 [마스터 사용자 이름(Master username)] 값을 확인합니다.

AWS CLI

AWS CLI를 사용하여 MySQL DB 인스턴스의 연결 정보를 찾으려면 describe-db-instances 명령을 실행합니다. 이 호출에서 DB 인스턴스 ID, 엔드포인트, 포트 및 마스터 사용자 이름을 쿼리합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-instances \
  --filters "Name=engine,Values=mysql" \
  --query "*[].[DBInstanceIdentifier,Endpoint.Address,Endpoint.Port,MasterUsername]"
```

Windows의 경우:

```
aws rds describe-db-instances ^
  --filters "Name=engine,Values=mysql" ^
  --query "*[].[DBInstanceIdentifier,Endpoint.Address,Endpoint.Port,MasterUsername]"
```

다음과 유사하게 출력되어야 합니다.

```
[
    [
        "mydb1",
        "mydb1.123456789012.us-east-1.rds.amazonaws.com",
        3306,
        "admin"
    ],
    [
        "mydb2",
        "mydb2.123456789012.us-east-1.rds.amazonaws.com",
        3306,
        "admin"
    ]
]
```

RDS API

Amazon RDS API를 사용하여 DB 인스턴스의 연결 정보를 찾으려면 DescribeDBInstances 작업을 호출합니다. 출력에서 엔드포인트 주소, 엔드포인트 포트 및 마스터 사용자 이름의 값을 찾습니다.

MySQL 명령줄 클라이언트 설치

대부분의 Linux 배포에는 Oracle MySQL 클라이언트 대신 MariaDB 클라이언트가 포함됩니다. Amazon Linux 2023에서 MySQL 명령줄 클라이언트를 설치하려면 다음 명령을 실행합니다.

```
sudo dnf install mariadb105
```

Amazon Linux 2에서 MySQL 명령줄 클라이언트를 설치하려면 다음 명령을 실행합니다.

```
sudo yum install mariadb
```

대부분의 DEB 기반 Linux 배포판에 MySQL 명령줄 클라이언트를 설치하려면 다음 명령을 실행합니다.

```
apt-get install mariadb-client
```

MySQL 명령줄 클라이언트의 버전을 확인하려면 다음 명령을 실행합니다.

```
mysql --version
```

현재 클라이언트 버전에 대한 MySQL 문서를 보려면 다음 명령을 실행합니다.

```
man mysql
```

MySQL 명령줄 클라이언트에서 연결(암호화되지 않음)

> **Important**
>
> 클라이언트와 서버가 동일한 VPC에 있고 네트워크를 신뢰할 수 있는 경우에만 암호화되지 않은 MySQL 연결을 사용합니다. 암호화된 연결 사용에 대한 자세한 내용은 MySQL 명령줄 클라이언트(암호화됨)에서 SSL/TLS를 사용하여 Amazon RDS의 MySQL DB 인스턴스에 연결 섹션을 참조하세요.

MySQL 명령줄 클라이언트를 사용하여 DB 인스턴스에 연결하려면 클라이언트 컴퓨터의 명령 프롬프트에서 다음 명령을 입력합니다. -h 파라미터의 경우 해당 DB 인스턴스의 DNS 이름(엔드포인트)으로 대체합니다. -p 파라미터에는 DB 인스턴스의 포트로 대체합니다. -u 파라미터에는 마스터 사용자와 같

이 유효한 데이터베이스 사용자의 사용자 이름으로 대체합니다. 입력 프롬프트가 표시되면 마스터 사용자 암호를 입력합니다.

```
mysql -h mysql-instance1.123456789012.us-east-1.rds.amazonaws.com -P 3306 -
u mymasteruser -p
```

사용자에 대한 암호를 입력하면 다음과 유사한 출력이 나타납니다.

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9738
Server version: 8.0.28 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

MySQL Workbench에서 연결

MySQL Workbench에서 연결하려면

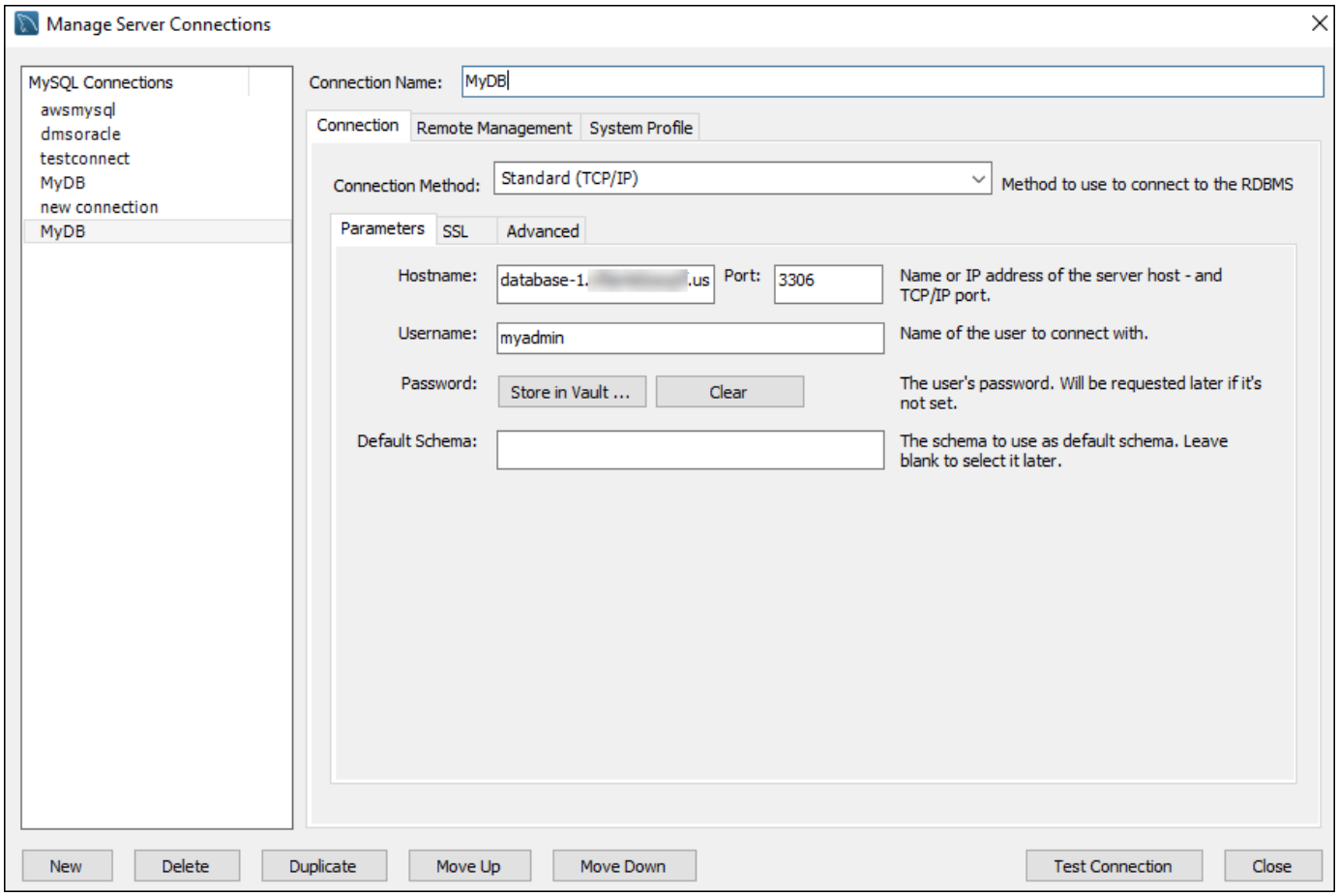
1. MySQL Workbench 다운로드에서 MySQL Workbench를 다운로드하고 설치합니다.
2. MySQL Workbench를 엽니다.



3. 데이터베이스에서 Manage Connections(연결 관리)를 선택합니다.
4. Manage Server Connections(서버 연결 관리) 창에서 새로 생성을 선택합니다.
5. Connect to Database(데이터베이스에 연결) 창에 다음 정보를 입력합니다.

- Stored Connection(저장된 연결) – 연결에 **MyDB**와 같은 이름을 입력합니다.
- Hostname(호스트 이름) – DB 인스턴스 엔드포인트를 입력합니다.
- 포트 – DB 인스턴스에서 사용한 포트를 입력합니다.
- 사용자 이름 – 마스터 사용자와 같이 유효한 데이터이스 사용자의 사용자 이름을 입력합니다.
- 암호 – 선택적으로 Store in Vault(볼트에 저장)를 선택한 후 사용자의 암호를 입력하고 저장합니다.

창은 다음과 비슷하게 표시됩니다.



MySQL Workbench의 기능을 사용하여 연결을 사용자 지정할 수 있습니다. 예를 들어, SSL 탭에서 SSL/TLS 연결을 구성할 수 있습니다. MySQL Workbench 사용에 대한 자세한 내용은 MySQL Workbench 설명서를 참조하십시오. SSL/TLS를 사용하여 MySQL DB 인스턴스로의 클라이언트 연결을 암호화하려면 SSL/TLS를 사용하여 Amazon RDS의 MySQL DB 인스턴스에 대한 클라이언트 연결 암호화 섹션을 참조하세요.

6. 선택적으로 연결 테스트를 선택하여 DB 인스턴스에 대한 연결이 성공적인지 확인합니다.

7. 닫기를 선택합니다.
8. 데이터베이스에서 Connect to Database(데이터베이스에 연결)를 선택합니다.
9. Stored Connection(저장된 연결)에서 연결을 선택합니다.
10. 확인을 선택합니다.

AWS JDBC 드라이버, AWS Python 드라이버 및 MySQL용 AWS ODBC 드라이버를 사용하여 RDS for MySQL에 연결

AWS JDBC 드라이버, AWS Python 드라이버 및 MySQL용 AWS ODBC 드라이버를 사용하여 RDS for MySQL DB 인스턴스에 연결합니다. 자세한 내용은 다음 항목을 참조하십시오.

주제

- Amazon Web Services(AWS) JDBC 드라이버를 사용하여 RDS for MySQL에 연결
- Amazon Web Services(AWS) Python 드라이버를 사용하여 RDS for MySQL에 연결
- MySQL용 Amazon Web Services(AWS) ODBC 드라이버를 사용하여 RDS for MySQL에 연결

Amazon Web Services(AWS) JDBC 드라이버를 사용하여 RDS for MySQL에 연결

Amazon Web Services(AWS) JDBC 드라이버는 고급 JDBC 래퍼로 설계되었습니다. 이 래퍼는 기존 JDBC 드라이버의 기능을 보완하고 확장합니다. 이 드라이버는 커뮤니티 MySQL Connector/J 드라이버 및 커뮤니티 MariaDB Connector/J 드라이버와 드롭인 호환됩니다.

AWS JDBC 드라이버를 설치하려면 CLASSPATH 애플리케이션에 있는 AWS JDBC 드라이버 .jar 파일을 추가하고 해당 커뮤니티 드라이버에 대한 참조를 보관해 두세요. 다음과 같이 해당 연결 URL 접두사를 업데이트하세요.

- `jdbc:mysql://~jdbc:aws-wrapper:mysql://`
- `jdbc:mariadb://~jdbc:aws-wrapper:mariadb://`

AWS JDBC 드라이버에 대한 자세한 내용 및 사용 방법에 대한 전체 지침은 Amazon Web Services (AWS) JDBC Driver GitHub repository를 참조하세요.

Amazon Web Services(AWS) Python 드라이버를 사용하여 RDS for MySQL에 연결

Amazon Web Services(AWS) Python 드라이버는 고급 Python 래퍼로 설계되었습니다. 이 래퍼는 오픈 소스 Psycopg 드라이버의 기능을 보완하고 확장합니다. AWS Python 드라이버는 Python 버전

3.8 이상을 지원합니다. `pip` 명령을 사용하여 `psycopg` 오픈 소스 패키지와 함께 `aws-advanced-python-wrapper` 패키지를 설치할 수 있습니다.

AWS Python 드라이버에 대한 자세한 내용 및 사용 방법에 대한 전체 지침은 Amazon Web Services (AWS) JDBC Python GitHub repository를 참조하세요.

MySQL용 Amazon Web Services(AWS) ODBC 드라이버를 사용하여 RDS for MySQL에 연결

MySQL용 AWS ODBC 드라이버는 RDS for MySQL의 고가용성을 위해 설계된 클라이언트 드라이버입니다. 드라이버는 MySQL Connector/ODBC 드라이버와 함께 존재할 수 있으며 동일한 워크플로와 호환됩니다.

MySQL용 AWS ODBC 드라이버에 대한 자세한 내용 및 설치와 사용에 대한 전체 지침은 MySQL용 Amazon Web Services(AWS) ODBC 드라이버 GitHub 리포지토리를 참조하세요.

MySQL DB 인스턴스에 대한 연결 문제 해결

새로운 DB 인스턴스의 연결 오류가 발생하는 두 가지 공통 원인은 다음과 같습니다.

- 보안 그룹을 사용하여 DB 인스턴스를 생성하였지만 이 보안 그룹이 MySQL 애플리케이션 또는 유틸리티를 실행 중인 디바이스나 Amazon EC2 인스턴스에서 연결을 승인하지 않는 경우. DB 인스턴스에 연결 권한을 부여하는 VPC 보안 그룹이 있어야 합니다. 자세한 내용은 Amazon VPC 및 Amazon RDS 단원을 참조하십시오.

  보안 그룹에서 인바운드 규칙을 추가하거나 편집할 수 있습니다. 소스에서 내 IP를 선택합니다. 이렇게 하면 브라우저에서 감지된 IP 주소에서 DB 인스턴스에 액세스할 수 있습니다.
- 포트 3306을 사용해 DB 인스턴스를 만들었는데 기업 방화벽 규칙에 따라 기업 네트워크의 디바이스에서 해당 포트에 연결하는 것이 차단된 경우. 이 오류를 수정하려면 인스턴스를 다른 포트로 다시 만들어야 합니다.

연결 문제에 대한 자세한 내용은 Amazon RDS DB 인스턴스에 연결할 수 없음 단원을 참조하십시오.

MySQL DB 인스턴스 연결 보안

강력한 보안 조치를 구현하여 무단 액세스 및 잠재적 위협으로부터 MySQL DB 인스턴스를 보호할 수 있습니다. 보안 그룹, SSL/TLS 암호화 및 IAM 데이터베이스 인증이 함께 작동하여 MySQL DB 인스턴스에 대한 여러 연결 보안 계층을 만듭니다. 이러한 보안 제어는 규정 준수 요구 사항을 충족하고, 데이터 침해를 방지하고, 애플리케이션과 데이터베이스 간의 보안 통신 채널을 유지하는 데 도움이 됩니다. 전송 중 데이터를 암호화하고, 특정 IP 범위로 액세스를 제한하고, 데이터베이스 암호가 아닌 IAM 역할을 통해 사용자 인증을 관리하여 MySQL DB 인스턴스를 보호할 수 있습니다.

MySQL DB 인스턴스용 보안은 세 가지 수준에서 관리됩니다.

- AWS Identity and Access Management는 DB 인스턴스에서 Amazon RDS 관리 작업을 수행할 수 있는 사용자를 제어합니다. IAM 자격 증명을 사용하여 AWS에 연결할 때, IAM 계정은 Amazon RDS 관리 작업을 수행하는 데 필요한 권한을 부여하는 IAM 정책을 보유하고 있어야 합니다. 자세한 내용은 Amazon RDS의 자격 증명 및 액세스 관리 섹션을 참조하세요.
- DB 인스턴스를 생성할 때 VPC 보안 그룹을 사용하여 어떤 디바이스와 Amazon EC2 인스턴스가 DB 인스턴스의 엔드포인트 및 포트에 대한 연결을 열 수 있는지를 제어하게 됩니다. 보안 소켓 계층(SSL) 및 전송 계층 보안(TLS)을 사용하여 이러한 연결을 구성할 수 있습니다. 또한 회사의 방화벽 규칙을 통해 회사에서 실행하는 디바이스가 DB 인스턴스에 대한 연결을 열 수 있는지를 제어할 수 있습니다.
- MySQL DB 인스턴스에 대한 로그인 및 권한을 인증하기 위해서는 다음 접근 방식 중 하나를 따르거나 두 방식을 조합할 수 있습니다.
  - 독립형 MySQL 인스턴스와 동일한 접근법을 사용할 수 있습니다. `CREATE USER`, `RENAME USER`, `GRANT`, `REVOKE` 및 `SET PASSWORD` 등의 명령은 온프레미스 데이터베이스에서 작동하는 것과 마찬가지로 작동하며, 데이터베이스 스키마 테이블을 직접 수정할 때도 동일합니다. 하지만 데이터베이스 스키마 테이블을 직접 수정하는 것은 모범 사례가 아니며 RDS for MySQL 버전 8.0.36부터는 지원되지 않습니다. 자세한 내용은 MySQL 설명서의 Access Control and Account Management를 참조하십시오.
  - 또한 IAM 데이터베이스 인증을 사용할 수도 있습니다. IAM 데이터베이스 인증의 경우, IAM 사용자 또는 IAM 역할 및 인증 토큰을 이용해 DB 인스턴스에 인증합니다. 인증 토큰은 서명 버전 4 서명 프로세스를 통해 생성하는 고유 값입니다. IAM 데이터베이스 인증을 사용하면 동일한 자격 증명을 사용해 AWS 리소스 및 데이터베이스에 대한 액세스를 제어할 수 있습니다. 자세한 내용은 MariaDB, MySQL 및 PostgreSQL IAM 데이터베이스 인증 섹션을 참조하세요.
  - 지원되는 또 다른 옵션은 RDS for MySQL용 Kerberos 인증입니다. DB 인스턴스는 AWS Directory Service for Microsoft Active Directory(AWS Managed Microsoft AD)와 함께 작동하여 Kerberos 인증을 활성화합니다. 사용자가 트러스팅 도메인에 가입된 MySQL DB 인스턴스로 인증하면 인증

요청이 전달됩니다. 전달된 요청은 AWS Directory Service에서 생성한 도메인 디렉터리로 이동합니다. 자세한 내용은 Amazon RDS for MySQL에 Kerberos 인증 사용 섹션을 참조하세요.

Amazon RDS DB 인스턴스를 생성할 때 마스터 사용자는 다음과 같은 기본 권한을 갖습니다.

엔진 버전	시스템 권한	데이터베이스 역할
RDS for MySQL 버전 8.4.3 이상	GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, PROCESS, REFERENCES ,INDEX, ALTER, SHOW DATABASES , CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE , REPLICATION CLIENT , CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER, CREATE ROLE, DROP ROLE, APPLICATION_PASSWORD_ADMIN , FLUSH_OPTIMIZER_COSTS , FLUSH_PRIVILEGES , FLUSH_STATUS , FLUSH_TABLES , FLUSH_USER_RESOURCES , ROLE_ADMIN , SENSITIVE_VARIABLES_OBSERVER , SESSION_VARIABLES_ADMIN , SET_ANY_DEFINER , SHOW_ROUTINE , XA_RECOVER_ADMIN	rds_superuser_role rds_superuser_role 에 대한 자세한 정보는 RDS for MySQL에 대한 역할 기반 권한 모델 섹션을 참조하세요.
RDS for MySQL 버전 8.0.36 이상	SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, PROCESS, REFERENCES , INDEX, ALTER, SHOW DATABASES , CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT , CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER, CREATE ROLE, DROP ROLE, APPLICATION_PASSWORD_ADMIN , ROLE_ADMIN , SET_USER_ID , XA_RECOVER_ADMIN	rds_superuser_role rds_superuser_role 에 대한 자세한 정보는 RDS for MySQL에 대한 역할 기반 권한 모델 섹션을 참조하세요.
RDS for MySQL	SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, PROCESS, REFERENCES , INDEX, ALTER,	없음

엔진 버전	시스템 권한	데이터베이스 역할
버전 8.0.36 미만	SHOW DATABASES , CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION CLIENT , CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER, REPLICATION SLAVE	

> **Note**
>
> DB 인스턴스에서 마스터 사용자를 삭제할 수 있지만, 권장하지는 않습니다. 마스터 사용자를 다시 만들려면 ModifyDBInstance RDS API 작업을 사용하거나 modify-db-instance AWS CLI 명령을 실행하고 적절한 파라미터와 함께 새 마스터 사용자 암호를 지정합니다. 마스터 사용자가 인스턴스에 존재하지 않는 경우 마스터 사용자가 지정된 암호와 함께 생성됩니다.

각 DB 인스턴스에 관리 서비스를 제공하기 위해 DB 인스턴스가 생성될 때 `rdsadmin` 사용자가 만들어집니다. `rdsadmin` 계정에 대한 암호를 삭제하거나 이름 바꾸기를 하거나 변경하려고 시도하면 또는 권한을 변경하려고 시도하면 오류가 발생합니다.

DB 인스턴스의 관리를 허용하기 위해 표준 `kill` 및 `kill_query` 명령이 제한되었습니다. DB 인스턴스에서 사용자 세션 또는 쿼리를 종료할 수 있도록 Amazon RDS 명령 `rds_kill` 및 `rds_kill_query`가 제공됩니다.

RDS for MySQL에 대한 암호 검증

MySQL은 향상된 보안을 위한 `validate_password` 플러그인을 제공합니다. 이 플러그인은 MySQL DB 인스턴스의 DB 파라미터 그룹에 있는 파라미터를 사용하여 암호 정책을 적용합니다. 이 플러그인은 MySQL 버전 5.7, 8.0, 8.4를 실행하는 DB 인스턴스에 지원됩니다. `validate_password` 플러그인에 대한 자세한 내용은 MySQL 설명서의 암호 확인 플러그인을 참조하십시오.

MySQL DB 인스턴스에 **`validate_password`** 플러그인을 활성화하는 방법

1. MySQL DB 인스턴스에 연결하고 다음 명령을 실행합니다.

```
INSTALL PLUGIN validate_password SONAME 'validate_password.so';
```

2. DB 인스턴스에서 사용하는 DB 파라미터 그룹에 플러그인에 대한 파라미터를 구성하면 됩니다.

   파라미터에 대한 자세한 내용은 MySQL 설명서의 암호 확인 플러그인 옵션 및 변수를 참조하십시오.

   DB 인스턴스 파라미터 수정에 대한 자세한 정보는 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 단원을 참조하십시오.
3. DB 인스턴스를 다시 시작합니다.

`validate_password` 플러그인을 활성화한 후 새 확인 정책을 준수하도록 기존 암호를 재설정하세요.

Amazon RDS는 암호를 확인하지 않습니다. MySQL DB 인스턴스는 암호 확인을 수행합니다. AWS Management Console, `modify-db-instance` AWS CLI 명령 또는 `ModifyDBInstance` RDS API 작업을 사용하여 사용자 암호를 설정하면 새 암호가 암호 정책에 부합하지 않아도 변경 작업이 완료될 수 있습니다. 그러나 새로운 암호는 암호 정책에 부합하는 경우에만 MySQL DB 인스턴스에 설정됩니다. 이 경우, Amazon RDS는 다음 이벤트를 기록합니다.

```
   "RDS-EVENT-0067" - An attempt to reset the master password for the DB instance has
failed.
```

Amazon RDS 이벤트에 대한 자세한 내용은 Amazon RDS 이벤트 알림 작업 단원을 참조하십시오.

SSL/TLS를 사용하여 Amazon RDS의 MySQL DB 인스턴스에 대한 클라이언트 연결 암호화

SSL(Secure Sockets Layer)은 클라이언트와 서버 간의 네트워크 연결을 보호하는 데 사용되는 업계 표준 프로토콜입니다. SSL 버전 3.0 이후로 이름이 전송 계층 보안(TLS)으로 변경되었습니다. Amazon RDS는 MySQL DB 인스턴스에 SSL/TLS 암호화를 지원합니다. SSL/TLS를 사용하여 애플리케이션 클라이언트와 MySQL DB 인스턴스 간의 연결을 암호화할 수 있습니다. SSL/TLS 지원 기능은 MySQL에 대한 모든 AWS 리전에서 사용할 수 있습니다.

Amazon RDS를 사용하면 SSL/TLS로 MySQL DB 인스턴스에 대한 클라이언트 연결을 암호화하고, MySQL DB 인스턴스에 대한 모든 연결에 SSL/TLS를 요구하고, MySQL 명령줄 클라이언트에서 SSL/

TLS(암호화됨)로 연결하여 전송 중 데이터를 보호할 수 있습니다. 다음 섹션에서는 Amazon RDS의 MySQL DB 인스턴스에 대한 SSL/TLS 암호화를 구성하고 활용하는 방법에 대한 지침을 제공합니다.

주제

- Amazon RDS의 MySQL DB 인스턴스에 대한 SSL/TLS 지원
- Amazon RDS의 MySQL DB 인스턴스에 대한 특정 사용자 계정에 SSL/TLS 요구
- Amazon RDS에서 MySQL DB 인스턴스에 대한 모든 연결에 SSL/TLS 요구
- MySQL 명령줄 클라이언트(암호화됨)에서 SSL/TLS를 사용하여 Amazon RDS의 MySQL DB 인스턴스에 연결

Amazon RDS의 MySQL DB 인스턴스에 대한 SSL/TLS 지원

Amazon RDS가 SSL/TLS 인증서를 생성한 후 Amazon RDS가 인스턴스를 프로비저닝할 때 DB 인스턴스에 인증서를 설치합니다. 인증 기관이 서명하는 SSL 인증서에는 SSL/TLS 인증서에는 스푸핑 공격으로부터 보호해주는 SSL/TLS 인증서를 위한 일반 이름(CN)으로 DB 인스턴스 엔드포인트가 포함되어 있습니다.

Amazon RDS에서 생성하는 SSL/TLS 인증서는 신뢰할 수 있는 루트 개체이므로 대부분의 경우에 작동하지만, 애플리케이션에서 인증서 체인을 수락하지 않을 경우 사용하지 못할 수 있습니다. 애플리케이션에서 인증서 체인을 허용하지 않는 경우 중간 인증서로 사용 중인 AWS 리전에 연결해 보세요. 예를 들어, SSL/TLS를 사용하여 AWS GovCloud (US) 리전에 연결하려면 중간 인증서를 사용해야 합니다.

인증서 다운로드에 대한 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 단원을 참조하세요. MySQL에서 SSL/TLS를 사용하는 방법에 대한 자세한 내용은 새 SSL/TLS 인증서를 사용해 MySQL DB 인스턴스에 연결할 애플리케이션 업데이트 섹션을 참조하세요.

MySQL 버전 8.0 이하의 경우 Amazon RDS for MySQL은 보안 연결에 OpenSSL을 사용합니다. MySQL 버전 8.4 이상의 경우 Amazon RDS for MySQL은 AWS-LC를 사용합니다. TLS 지원은 MySQL 버전에 따라 달라집니다. 다음 표는 MySQL 버전에 대한 TLS 지원을 보여줍니다.

MySQL 버전	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3
MySQL 8.4	지원되지 않음	지원되지 않음	지원	지원
MySQL 8.0	지원되지 않음	지원되지 않음	지원	지원

MySQL 버전	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3
MySQL 5.7	지원	지원	지원	지원되지 않음

Amazon RDS의 MySQL DB 인스턴스에 대한 특정 사용자 계정에 SSL/TLS 요구

Amazon RDS의 MySQL DB 인스턴스에 대한 지정된 사용자 계정 연결에 SSL/TLS 암호화가 필요할 수 있습니다. 데이터 기밀성이 우려되는 보안 정책을 적용하려면 민감한 정보를 무단 액세스 또는 가로채기로부터 보호하는 것이 중요합니다.

특정 사용자의 계정에 대해 SSL/TLS 연결을 요구하려면 MySQL 버전에 따라 다음 문 중 하나를 사용하여 사용자 계정 `encrypted_user`에서 SSL/TLS 연결을 요구하세요.

이렇게 하려면 다음 문을 사용하면 됩니다.

```
ALTER USER 'encrypted_user'@'%' REQUIRE SSL;
```

MySQL과의 SSL/TLS 연결에 대한 자세한 내용은 MySQL 설명서의 암호화된 연결 사용을 참조하세요.

Amazon RDS에서 MySQL DB 인스턴스에 대한 모든 연결에 SSL/TLS 요구

`require_secure_transport` 파라미터를 사용하여 MySQL DB 인스턴스에 대한 모든 사용자 연결이 SSL/TLS를 사용하도록 요구합니다. 기본적으로 `require_secure_transport` 파라미터는 `OFF`로 설정됩니다. `require_secure_transport` 파라미터를 `ON`으로 설정하면 해당 DB 인스턴스에 대한 연결 시 SSL/TLS를 요구합니다.

`require_secure_transport` 파라미터 값은 DB 인스턴스의 DB 파라미터 그룹을 업데이트하여 설정할 수 있습니다. 변경 사항을 적용하기 위해 DB 인스턴스를 재부팅할 필요가 없습니다.

DB 클러스터에 대해 `require_secure_transport` 파라미터를 `ON`으로 설정하면 암호화된 연결을 설정할 수 있는 경우 데이터베이스 클라이언트가 인스턴스에 연결할 수 있습니다. 그렇지 않으면 다음과 유사한 오류 메시지가 클라이언트에 반환됩니다.

```
MySQL Error 3159 (HY000): Connections using insecure transport are prohibited while --
require_secure_transport=ON.
```

파라미터 설정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정을 참조하세요.

`require_secure_transport` 파라미터에 대한 자세한 내용은 MySQL 설명서를 참조하세요.

MySQL 명령줄 클라이언트(암호화됨)에서 SSL/TLS를 사용하여 Amazon RDS의 MySQL DB 인스턴스에 연결

mysql 클라이언트 프로그램 파라미터는 사용 중인 MySQL 또는 MariaDB 버전에 따라 약간 다릅니다.

사용 중인 버전을 확인하려면 --version 옵션을 사용하여 mysql 명령을 실행합니다. 다음 예에서는 출력은 클라이언트 프로그램이 MariaDB의 프로그램임을 나타냅니다.

```
$ mysql --version
mysql  Ver 15.1 Distrib 10.5.15-MariaDB, for osx10.15 (x86_64) using readline 5.1
```

Amazon Linux, CentOS, SUSE 및 Debian과 같은 대부분의 Linux 배포판은 MySQL을 MariaDB로 대체했으며 mysql 버전은 MariaDB에서 가져온 것입니다.

다음 단계에 따라 SSL/TLS를 사용하여 DB 인스턴스에 연결합니다.

MySQL 명령줄 클라이언트를 사용하여 SSL/TLS를 통해 DB 인스턴스에 연결하려면

1. 모든 AWS 리전에 적용되는 루트 인증서를 다운로드할 수 있습니다.

   인증서 다운로드에 대한 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 단원을 참조하세요.
2. MySQL 명령줄 클라이언트를 사용하여 SSL/TLS를 통해 DB 인스턴스에 연결합니다. -h 파라미터의 경우 해당 DB 인스턴스의 DNS 이름(엔드포인트)로 대체합니다. --ssl-ca 파라미터는 해당하는 SSL/TLS 인증서 파일 이름으로 대체합니다. -P 파라미터에는 DB 인스턴스의 포트로 대체합니다. -u 파라미터에는 마스터 사용자와 같이 유효한 데이터베이스 사용자의 사용자 이름으로 대체합니다. 입력 프롬프트가 표시되면 마스터 사용자 암호를 입력합니다.

   다음 예제는 MySQL 5.7 클라이언트 이후 버전의 경우 --ssl-ca 파라미터를 사용하여 클라이언트를 시작하는 방법을 보여줍니다.

   ```
   mysql -h mysql-instance1.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=global-bundle.pem --ssl-mode=REQUIRED -P 3306 -u myadmin -p
   ```

   SSL/TLS 연결에서 SSL/TLS 인증서의 엔드포인트와 비교하여 DB 인스턴스 엔드포인트를 확인하도록 요구할 수 있습니다.

   ```
   mysql -h mysql-instance1.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=global-bundle.pem --ssl-mode=VERIFY_IDENTITY -P 3306 -u myadmin -p
   ```

다음 예제는 MariaDB 클라이언트를 사용하여 `--ssl-ca` 파라미터를 통해 클라이언트를 시작하는 방법을 보여줍니다.

```
mysql -h mysql-instance1.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=global-bundle.pem --ssl -P 3306 -u myadmin -p
```

3. 입력 프롬프트가 표시되면 마스터 사용자 암호를 입력합니다.

출력은 다음과 비슷합니다.

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9738
Server version: 8.0.28 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

새 SSL/TLS 인증서를 사용해 MySQL DB 인스턴스에 연결할 애플리케이션 업데이트

2023년 1월 13일부터 Amazon RDS는 보안 소켓 계층(SSL) 또는 전송 계층 보안(TLS)을 사용해 RDS DB 인스턴스에 연결하기 위한 용도의 새 인증 기관(CA) 인증서를 게시하였습니다. 아래에서 새 인증서를 사용하기 위해 애플리케이션을 업데이트하는 방법에 관한 정보를 찾으실 수 있습니다.

이 주제는 클라이언트 애플리케이션에서 SSL/TLS를 사용해 DB 인스턴스에 연결하는지 여부를 판단하는 데 도움이 됩니다. SSL/TLS를 사용해 연결한다면 이 애플리케이션에서 연결 시 인증서 확인이 필요한지 여부를 추가로 확인할 수 있습니다.

> **Note**
>
> 어떤 애플리케이션은 서버에서 인증서를 성공적으로 확인할 수 있는 경우에만 MySQL DB 인스턴스에 연결하도록 구성되어 있습니다. 이러한 애플리케이션의 경우 클라이언트 애플리케이션 트러스트 스토어를 업데이트하여 새 CA 인증서를 포함해야 합니다.
>
> `disabled`, `preferred` 및 `required`의 SSL 모드를 지정할 수 있습니다. `preferred` SSL 모드를 사용할 때 CA 인증서가 없거나 최신 버전이 아닌 경우 연결이 SSL을 사용하지 않는 것으로 폴백되고 암호화 없이 연결됩니다.

`preferred` 모드는 피하는 것이 좋습니다. `preferred` 모드에서 연결이 잘못된 인증서가 발견되면 모드에서 암호화 사용을 중지하고 암호화되지 않은 상태로 진행합니다.

클라이언트 애플리케이션 트러스트 스토어에서 CA 인증서를 업데이트한 후에는 DB 인스턴스에서 인증서를 교환할 수 있습니다. 이 절차를 프로덕션 환경에서 구현하기 전에 개발 또는 스테이징 환경에서 테스트해볼 것을 적극 권장합니다.

인증서 교환에 대한 자세한 내용은 SSL/TLS 인증서 교체 단원을 참조하십시오. 인증서 다운로드에 대한 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 단원을 참조하십시오. MySQL DB 인스턴스에서 SSL/TLS를 사용하는 방법에 관한 자세한 내용은 Amazon RDS의 MySQL DB 인스턴스에 대한 SSL/TLS 지원 단원을 참조하십시오.

주제

- 애플리케이션에서 SSL을 사용해 MySQL DB 인스턴스에 연결하는지 여부 확인
- 클라이언트에서 연결 시 인증서 확인이 필요한지 여부 확인
- 애플리케이션 트러스트 스토어 업데이트
- SSL 연결 설정을 위한 Java 코드 예시

애플리케이션에서 SSL을 사용해 MySQL DB 인스턴스에 연결하는지 여부 확인

Amazon RDS for MySQL 버전 5.7, 8.0 또는 8.4를 사용 중이고 성능 스키마가 활성화되어 있는 경우 다음 쿼리를 실행하여 연결 시 SSL/TLS를 사용하는지 여부를 확인하세요. 성능 스키마 활성화에 대한 자세한 내용은 MySQL 설명서의 성능 스키마 빠른 시작을 참조하십시오.

```
mysql> SELECT id, user, host, connection_type
       FROM performance_schema.threads pst
       INNER JOIN information_schema.processlist isp
       ON pst.processlist_id = isp.id;
```

이 샘플 출력에서는 고객님 자신의 세션(`admin`)과 `webapp1`로 로그인된 애플리케이션에서 모두 SSL을 사용 중임을 알 수 있습니다.

```
+----+-----------------+-----------------+-----------------+
```

```
| id | user            | host             | connection_type |
+----+-----------------+------------------+-----------------+
|  8 | admin           | 10.0.4.249:42590 | SSL/TLS         |
|  4 | event_scheduler | localhost        | NULL            |
| 10 | webapp1         | 159.28.1.1:42189 | SSL/TLS         |
+----+-----------------+------------------+-----------------+
3 rows in set (0.00 sec)
```

클라이언트에서 연결 시 인증서 확인이 필요한지 여부 확인

JDBC 클라이언트 및 MySQL 클라이언트에서 연결 시 인증서 확인이 필요한지 여부를 확인할 수 있습니다.

JDBC

아래의 MySQL 커넥터/J 8.0 관련 예시에서는 애플리케이션의 JDBC 연결 속성을 확인하여 성공적인 연결을 위해 유효한 인증서가 필요한지 여부를 판단하는 한 가지 방법을 보여줍니다. MySQL에 대한 모든 JDBC 연결 옵션에 대한 자세한 내용은 MySQL 문서의 구성 속성을 참조하십시오.

MySQL 커넥터/J 8.0을 사용 중인 경우 다음 예시와 같이 연결 속성에서 `sslMode`가 `VERIFY_CA` 또는 `VERIFY_IDENTITY`로 설정되어 있다면 SSL 연결 시 DB 서버 인증서에 대한 확인이 필요합니다.

```
Properties properties = new Properties();
properties.setProperty("sslMode", "VERIFY_IDENTITY");
properties.put("user", DB_USER);
properties.put("password", DB_PASSWORD);
```

> **Note**
>
> 데이터베이스에 연결할 때 SSL/TLS를 사용하도록 애플리케이션을 명시적으로 구성하지 않았더라도 MySQL Java Connector v5.1.38 이상 또는 MySQL Java Connector v8.0.9 이상을 사용하여 데이터베이스에 연결하는 경우, 이러한 클라이언트 드라이버는 기본적으로 SSL/TLS를 사용합니다. 또한 SSL/TLS 사용 시 부분 인증서 확인을 수행하고 데이터베이스 서버 인증서가 만료되면 연결에 실패합니다.

MySQL

MySQL 클라이언트에 관한 다음 예시에서는 스크립트의 MySQL 연결을 확인하여 성공적인 연결을 위해 유효한 인증서가 필요한지 여부를 판단하는 두 가지 방법을 보여줍니다. MySQL 클라이언트의 모든 연결 옵션에 대한 자세한 내용은 MySQL 문서의 암호화된 연결을 위한 클라이언트 측 구성을 참조하십시오.

MySQL Client 버전 5.7 이상을 사용 중인 경우 다음 예시와 같이 `--ssl-mode` 옵션에 대해 `VERIFY_CA` 또는 `VERIFY_IDENTITY`를 지정했다면 SSL 연결 시 서버 CA 인증서에 대한 확인이 필요합니다.

```
mysql -h mysql-database.rds.amazonaws.com -uadmin -ppassword --ssl-ca=/tmp/ssl-cert.pem
 --ssl-mode=VERIFY_CA
```

애플리케이션 트러스트 스토어 업데이트

MySQL 애플리케이션을 위한 트러스트 스토어 업데이트에 관한 자세한 내용은 MySQL 문서의 SSL 인증서 설치 단원을 참조하십시오.

루트 인증서 다운로드에 대한 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 단원을 참조하십시오.

인증서를 가져오는 샘플 스크립트는 트러스트 스토어로 인증서를 가져오기 위한 샘플 스크립트 섹션을 참조하세요.

> **Note**
>
> 트러스트 스토어를 업데이트할 때 새 인증서를 추가할 뿐 아니라 이전 인증서를 유지할 수도 있습니다.

애플리케이션에서 mysql JDBC를 사용 중인 경우 애플리케이션에서 다음 속성을 설정하십시오.

```
System.setProperty("javax.net.ssl.trustStore", certs);
System.setProperty("javax.net.ssl.trustStorePassword", "password");
```

애플리케이션을 시작할 때 다음 속성을 설정하십시오.

```
java -Djavax.net.ssl.trustStore=/path_to_trust_store/MyTruststore.jks -
Djavax.net.ssl.trustStorePassword=my_trust_store_password com.companyName.MyApplication
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

SSL 연결 설정을 위한 Java 코드 예시

다음 코드 예시에서는 JDBC를 사용하여 서버 인증서를 확인하는 SSL 연결을 설정하는 방법을 보여줍니다.

```
public class MySQLSSLTest {

        private static final String DB_USER = "username";
        private static final String DB_PASSWORD = "password";
        // This trust store has only the prod root ca.
        private static final String TRUST_STORE_FILE_PATH = "file-path-to-trust-store";
        private static final String TRUST_STORE_PASS = "trust-store-password";

        public static void test(String[] args) throws Exception {
            Class.forName("com.mysql.jdbc.Driver");

            System.setProperty("javax.net.ssl.trustStore", TRUST_STORE_FILE_PATH);
            System.setProperty("javax.net.ssl.trustStorePassword", TRUST_STORE_PASS);

            Properties properties = new Properties();
            properties.setProperty("sslMode", "VERIFY_IDENTITY");
            properties.put("user", DB_USER);
            properties.put("password", DB_PASSWORD);


            Connection connection = null;
            Statement stmt = null;
            ResultSet rs = null;
            try {
```

```
                connection =
 DriverManager.getConnection("jdbc:mysql://mydatabase.123456789012.us-
east-1.rds.amazonaws.com:3306",properties);
                stmt = connection.createStatement();
                rs=stmt.executeQuery("SELECT 1 from dual");
            } finally {
                if (rs != null) {
                    try {
                        rs.close();
                    } catch (SQLException e) {
                    }
                }
                if (stmt != null) {
                   try {
                        stmt.close();
                    } catch (SQLException e) {
                   }
                }
                if (connection != null) {
                    try {
                        connection.close();
                    } catch (SQLException e) {
                        e.printStackTrace();
                    }
                }
            }
            return;
        }
    }
```

Important

데이터베이스 연결에서 SSL/TLS를 사용함을 확인하고 애플리케이션 트러스트 스토어를 업데이트한 후에는 데이터베이스에서 rds-ca-rsa2048-g1 인증서를 사용하도록 업데이트할 수 있습니다. 지침은 DB 인스턴스 또는 클러스터를 수정하여 CA 인증서 업데이트의 3단계를 참조하십시오.

보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

Amazon RDS for MySQL에 Kerberos 인증 사용

사용자가 MySQL DB 인스턴스에 접속하려고 할 때 Kerberos 인증을 사용하여 사용자를 인증할 수 있습니다. DB 인스턴스는 AWS Directory Service for Microsoft Active Directory(AWS Managed Microsoft AD)와 함께 작동하여 Kerberos 인증을 활성화합니다. 사용자가 트러스팅 도메인에 가입된 MySQL DB 인스턴스로 인증하면 인증 요청이 전달됩니다. 전달된 요청은 AWS Directory Service에서 생성한 도메인 디렉터리로 이동합니다.

모든 자격 증명을 동일한 디렉터리에 보관하면 시간과 노력을 절약할 수 있습니다. 이 접근 방식의 경우, 여러 DB 인스턴스에 대한 자격 증명을 보관하고 관리할 수 있는 중앙 집중식 공간이 있습니다. 디렉터리를 사용하면 전체 보안 프로필을 향상할 수도 있습니다.

리전 및 버전 사용 가능 여부

기능 가용성 및 해당 지원은 각 데이터베이스 엔진의 특정 버전 및 AWS 리전에 따라 다릅니다. Kerberos 인증을 사용하는 Amazon RDS의 버전 및 리전 가용성에 대한 자세한 내용은 Amazon RDS에서 Kerberos 인증을 지원하는 리전 및 DB 엔진 단원을 참조하세요.

MySQL DB 인스턴스에 대해 Kerberos 인증 설정 개요

MySQL DB 인스턴스에 대해 Kerberos 인증을 설정하려면 다음과 같은 일반적인 단계를 완료하십시오(이후에 자세히 설명).

1. AWS Managed Microsoft AD를 사용하여 AWS Managed Microsoft AD 디렉터리를 생성합니다. AWS Management Console, AWS CLI 또는 AWS Directory Service를 사용하여 디렉터리를 생성할 수 있습니다. 이에 대한 자세한 내용은 AWS Directory Service 관리 가이드의 AWS Managed Microsoft AD 디렉터리 생성을 참조하세요.
2. 관리형 IAM 정책 `AmazonRDSDirectoryServiceAccess`를 사용하는 AWS Identity and Access Management(IAM) 역할을 생성합니다. 이 역할을 사용하여 Amazon RDS에서 디렉터리를 호출할 수 있습니다.

   역할이 액세스를 허용하려면 AWS 리전에서 AWS 계정의 AWS Security Token Service(AWS STS) 엔드포인트를 활성화해야 합니다. AWS STS 엔드포인트는 기본적으로 모든 AWS 리전에서 활성화되어 있으므로 별도의 조치 없이 사용할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 리전에서 AWS STS 활성화 및 비활성화를 참조하세요.
3. Microsoft Active Directory 도구를 사용하여 AWS Managed Microsoft AD 디렉터리에서 사용자를 만들고 구성합니다. Active Directory에서 사용자를 생성하는 방법에 대한 자세한 내용은 AWS Directory Service 관리 안내서의 AWS 관리형 Microsoft AD에서 사용자 및 그룹 관리를 참조하세요.

4. MySQL DB 인스턴스를 생성 또는 수정합니다. 생성 요청에 CLI 또는 RDS API를 사용하는 경우 `Domain` 파라미터를 사용해 도메인 식별자를 지정합니다. 디렉터리를 만들 때 생성된 `d-*` 식별자와 생성한 역할의 이름을 사용합니다.

   Kerberos 인증을 사용하도록 기존 MySQL DB 인스턴스를 수정하는 경우 DB 인스턴스에 대해 도메인 및 IAM 역할 파라미터를 설정합니다. 도메인 디렉터리와 동일한 VPC에서 DB 인스턴스를 찾습니다.

5. Amazon RDS 마스터 사용자 자격 증명을 사용하여 MySQL DB 인스턴스에 연결합니다. `CREATE USER` 절 `IDENTIFIED WITH 'auth_pam'`을 사용하여 MySQL에서 사용자를 생성합니다. 이 방법으로 생성한 사용자는 Kerberos 인증을 사용하여 MySQL DB 인스턴스에 로그인할 수 있습니다.

MySQL DB 인스턴스에 대해 Kerberos 인증 설정

AWS Managed Microsoft AD를 사용하여 MySQL DB 인스턴스에 대해 Kerberos 인증을 설정합니다. Kerberos 인증을 설정하려면 다음 단계를 수행하십시오.

1단계: AWS Managed Microsoft AD를 사용하여 디렉터리 생성

AWS Directory Service는 AWS 클라우드에서 완전 관리형 Microsoft Active Directory를 생성합니다. AWS Managed Microsoft AD 디렉터리를 생성할 때 AWS Directory Service에서 두 개의 도메인 컨트롤러 및 Domain Name System(DNS) 서버가 자동으로 생성됩니다. 디렉터리 서버는 VPC 내 다른 서브넷에서 생성됩니다. 이러한 중복으로 인해 장애가 발생해도 디렉터리에 액세스할 수 있습니다.

AWS Managed Microsoft AD 디렉터리를 생성하는 경우 AWS Directory Service에서 다음 작업이 자동으로 수행됩니다.

- VPC 내에서 Active Directory를 설정합니다.
- 사용자 이름 Admin과 지정된 암호를 사용하여 디렉터리 관리자 계정을 생성합니다. 이 계정을 사용하여 디렉터리를 관리할 수 있습니다.

> **Note**
>
> 이 암호를 저장하십시오. AWS Directory Service에서는 저장되지 않습니다. 재설정은 가능하지만 검색은 불가능합니다.

- 디렉터리 컨트롤러에 대한 보안 그룹을 만듭니다.

AWS Managed Microsoft AD를 시작하면 AWS에서 모든 디렉터리의 객체를 포함하는 OU(조직 단위)를 생성합니다. 이 OU는 디렉터리를 생성할 때 입력한 NetBIOS 이름을 가지고 있으며, 도메인 루트에 위치합니다. 도메인 루트는 AWS에서 소유하고 관리합니다.

AWS Managed Microsoft AD 디렉터리를 사용해 생성한 관리자 계정은 OU의 가장 일반적인 관리 활동에 대한 권한을 갖습니다.

- 사용자 생성, 업데이트 또는 삭제
- 도메인(예: 파일 또는 인쇄 서버)에 리소스를 추가한 다음 OU 내의 사용자에 해당 리소스에 대한 권한 할당
- 추가 OU 및 컨테이너 생성
- 권한 위임
- Active Directory 휴지통에서 삭제된 객체 복원
- Active Directory 웹 서비스에서 AD 및 DNS Windows PowerShell 모듈 실행

또한 admin 계정은 다음과 같은 도메인 차원 활동을 수행할 권한이 있습니다.

- DNS 구성 관리(레코드, 영역 및 전달자 추가, 제거 또는 업데이트)
- DNS 이벤트 로그 보기
- 보안 이벤트 로그 보기

AWS Managed Microsoft AD으로 디렉터리를 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/directoryservicev2/에서 AWS Directory Service 콘솔을 엽니다.
2. 탐색 창에서 디렉터리를 선택한 후 디렉터리 설정을 선택합니다.
3. AWS Managed Microsoft AD를 선택합니다. AWS Managed Microsoft AD는 현재 Amazon RDS와 함께 사용할 수 있는 유일한 옵션입니다.
4. 다음 정보를 입력합니다.

   디렉터리 DNS 이름

   디렉터리를 위한 정규화된 이름(예: **`corp.example.com`**)입니다.

   디렉터리 NetBIOS 이름

   디렉터리의 짧은 이름(예: **`CORP`**)입니다.

디렉터리 설명

(선택 사항) 디렉터리에 대한 설명입니다.

관리자 암호

디렉터리 관리자의 암호입니다. 디렉터리 생성 프로세스에서는 사용자 이름 Admin과 이 암호를 사용하여 관리자 계정을 생성합니다.

디렉터리 관리자 암호는 "admin"이라는 단어를 포함할 수 없습니다. 암호는 대소문자를 구분하며 길이가 8~64자 사이여야 합니다. 또한 다음 네 범주 중 세 개에 해당하는 문자를 1자 이상 포함해야 합니다.

- 소문자(a–z)
- 대문자(A–Z)
- 숫자(0–9)
- 영숫자 외의 특수 문자(~!@#$%^&*_-+=`|\(){}[]:;"'<>,.?/)

[Confirm Password]

관리자 암호를 다시 입력했습니다.

5. 다음을 선택합니다.
6. 네트워킹 섹션에 다음 정보를 입력하고 다음을 선택합니다.

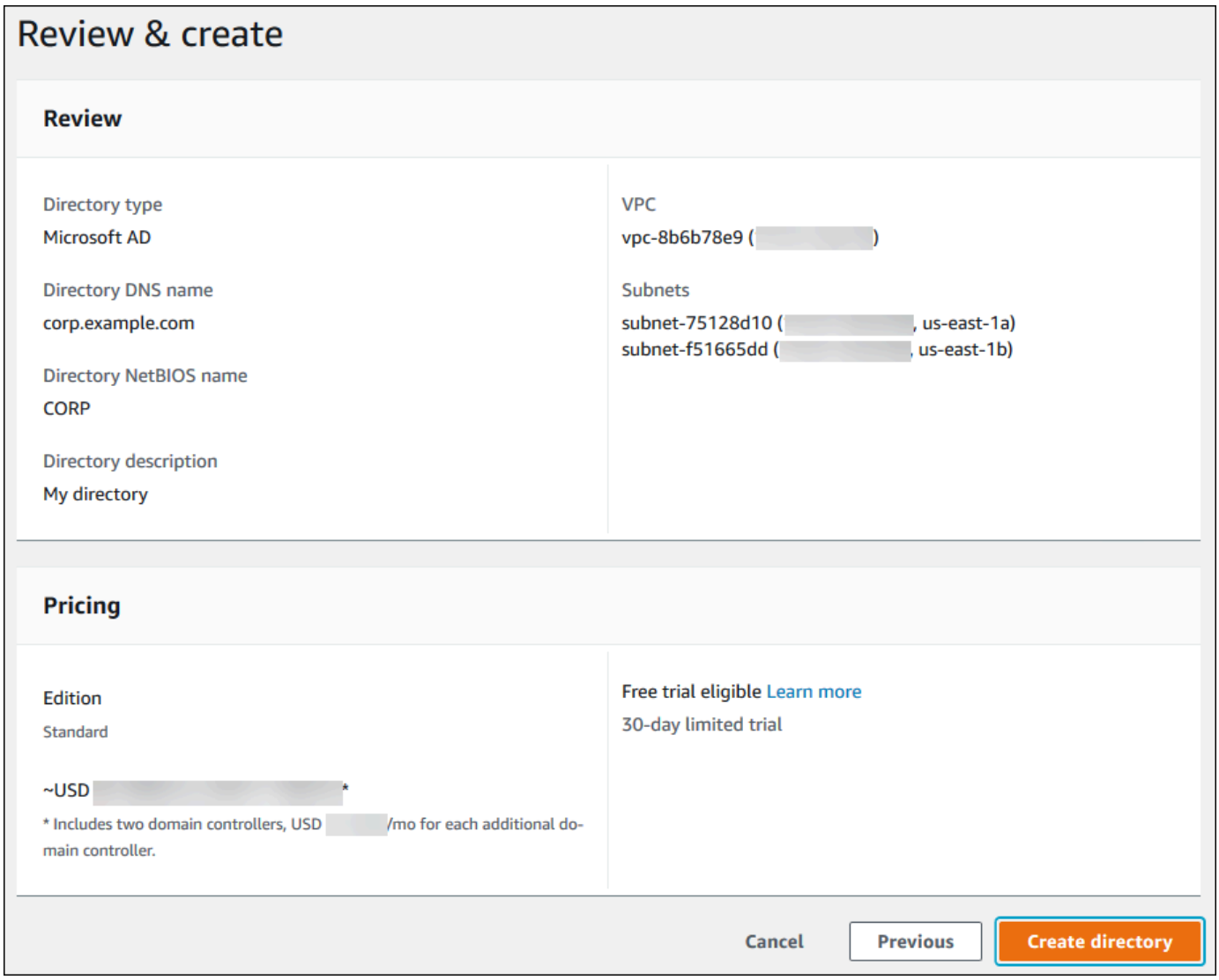
VPC

디렉터리에 대한 VPC입니다. 동일한 VPC에서 MySQL DB 인스턴스를 생성합니다.

서브넷

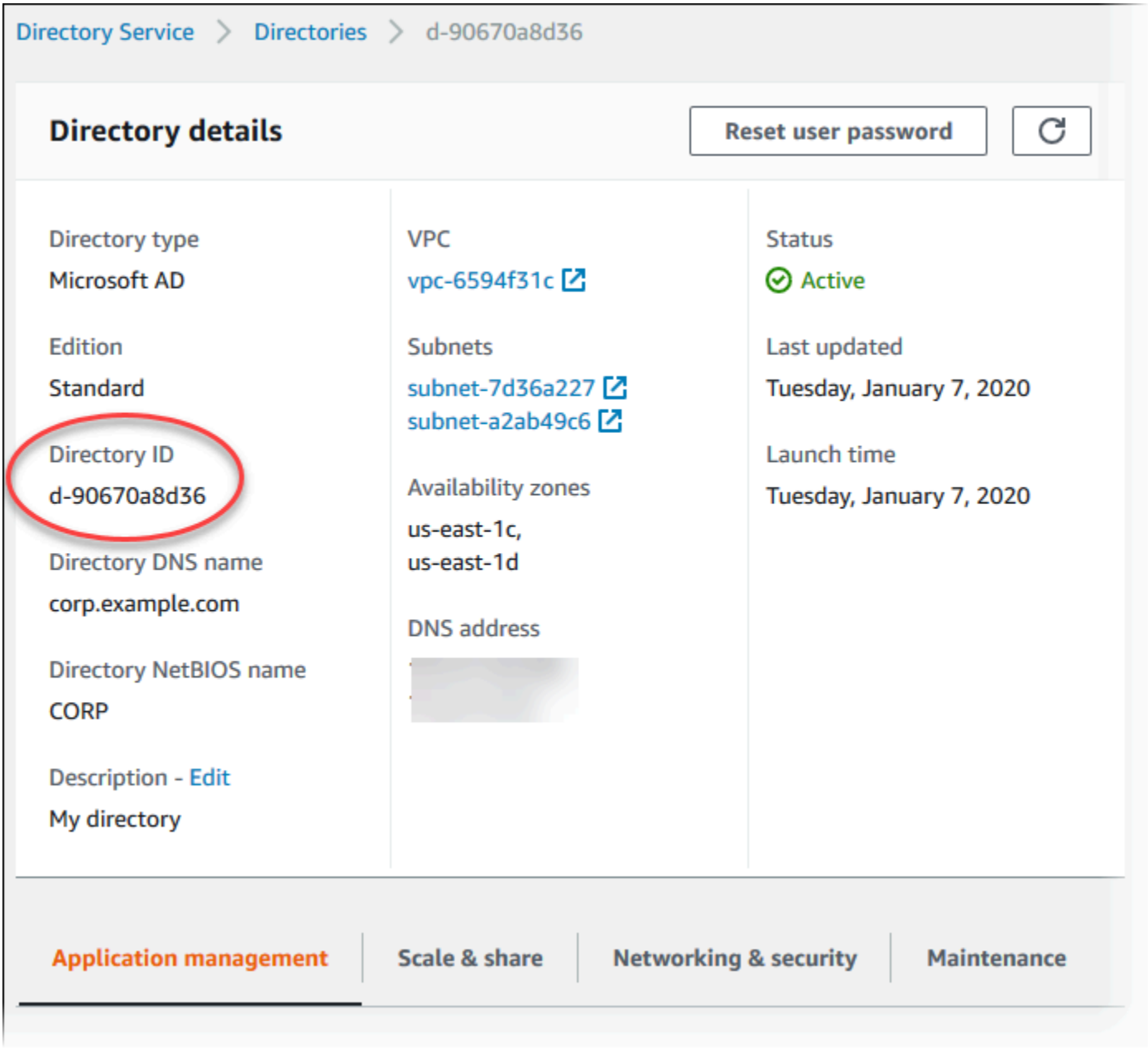
디렉터리 서버에 대한 서브넷입니다. 두 서브넷이 서로 다른 가용 영역에 있어야 합니다.

7. 디렉터리 정보를 검토하고 필요한 사항을 변경합니다. 정보가 올바르면 디렉터리 생성을 선택합니다.



디렉터리를 생성하는 데 몇 분 정도 걸립니다. 디렉터리가 성공적으로 생성되면 상태 값이 활성으로 변경됩니다.

디렉터리에 대한 정보를 보려면 디렉터리 목록에서 해당 디렉터리 이름을 선택합니다. MySQL DB 인스턴스를 생성하거나 수정할 때 이 값이 필요하므로 디렉터리 ID 값을 기록해 두십시오.



2단계: Amazon RDS에 사용할 IAM 역할 생성

Amazon RDS에서 AWS Directory Service를 호출하려면 관리형 IAM 정책 `AmazonRDSDirectoryServiceAccess`를 사용하는 IAM 역할이 필요합니다. 이 역할을 사용하여 Amazon RDS에서 AWS Directory Service를 호출할 수 있습니다.

AWS Management Console을 사용하여 DB 인스턴스를 생성할 때 콘솔 사용자에게 `iam:CreateRole` 권한이 있으면 콘솔에서 이 역할을 자동으로 생성합니다. 이 경우 역할 이름은 `rds-directoryservice-kerberos-access-role`입니다. 그렇지 않으면 IAM 역할을 수동으로

생성해야 합니다. 이 IAM 역할을 생성할 때 `Directory Service`를 선택하고 여기에 AWS 관리형 정책인 `AmazonRDSDirectoryServiceAccess`를 연결합니다.

서비스에 대한 IAM 역할 생성에 대한 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.

> **Note**
>
> RDS for SQL Server에 대한 Windows 인증에 사용되는 IAM 역할은 RDS for MySQL에 사용할 수 없습니다.

선택 사항으로 관리형 IAM 정책인 `AmazonRDSDirectoryServiceAccess`를 사용하는 대신 필요한 권한으로 정책을 생성할 수 있습니다. 이 경우 IAM 역할에 다음과 같은 IAM 신뢰 정책이 있어야 합니다.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.amazonaws.com",
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

또한 역할에는 다음과 같은 IAM 역할 정책도 있어야 합니다.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ds:DescribeDirectories",
        "ds:AuthorizeApplication",
        "ds:UnauthorizeApplication",
        "ds:GetAuthorizedApplicationDetails"
      ],
    "Effect": "Allow",
    "Resource": "*"
    }
  ]
}
```

3단계: 사용자 생성 및 구성

Active Directory Users and Computers 도구를 사용하여 사용자를 생성할 수 있습니다. 이 도구는 Active Directory Domain Services 및 Active Directory Lightweight Directory Services 도구에 포함되어 있습니다. 사용자는 디렉터리에 액세스할 수 있는 개별 사용자 또는 개체를 나타냅니다.

AWS Directory Service 디렉터리에서 사용자를 생성하려면 Microsoft Windows를 기반으로 Amazon EC2 인스턴스에 연결해야 합니다. 이 인스턴스는 AWS Directory Service 디렉터리의 멤버여야 하며 사용자를 생성할 수 있는 권한을 가진 사용자로 로그인해야 합니다. 자세한 내용은 AWS Managed Microsoft AD Directory Service 관리 가이드에서 AWS의 사용자 및 그룹 관리를 참조하세요.

4단계: MySQL DB 인스턴스 생성 또는 수정

디렉터리에서 사용할 MySQL DB 인스턴스를 생성하거나 수정합니다. 콘솔, CLI 또는 RDS API를 사용하여 DB 인스턴스를 디렉터리에 연결할 수 있습니다. 이 작업을 다음 중 한 가지 방법으로 수행할 수 있습니다.

- 콘솔, create-db-instance CLI 명령 또는 CreateDBInstance RDS API 작업을 사용하여 새 MySQL DB 인스턴스를 생성합니다.

  지침은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.

- 콘솔, modify-db-instance CLI 명령 또는 ModifyDBInstance RDS API 작업을 사용하여 기존 MySQL DB 인스턴스를 수정합니다.

  지침은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

- 콘솔, restore-db-instance-from-db-snapshot CLI 명령 또는 RestoreDBInstanceFromDBSnapshot RDS API 작업을 사용하여 DB 스냅샷에서 MySQL DB 인스턴스를 복원합니다.

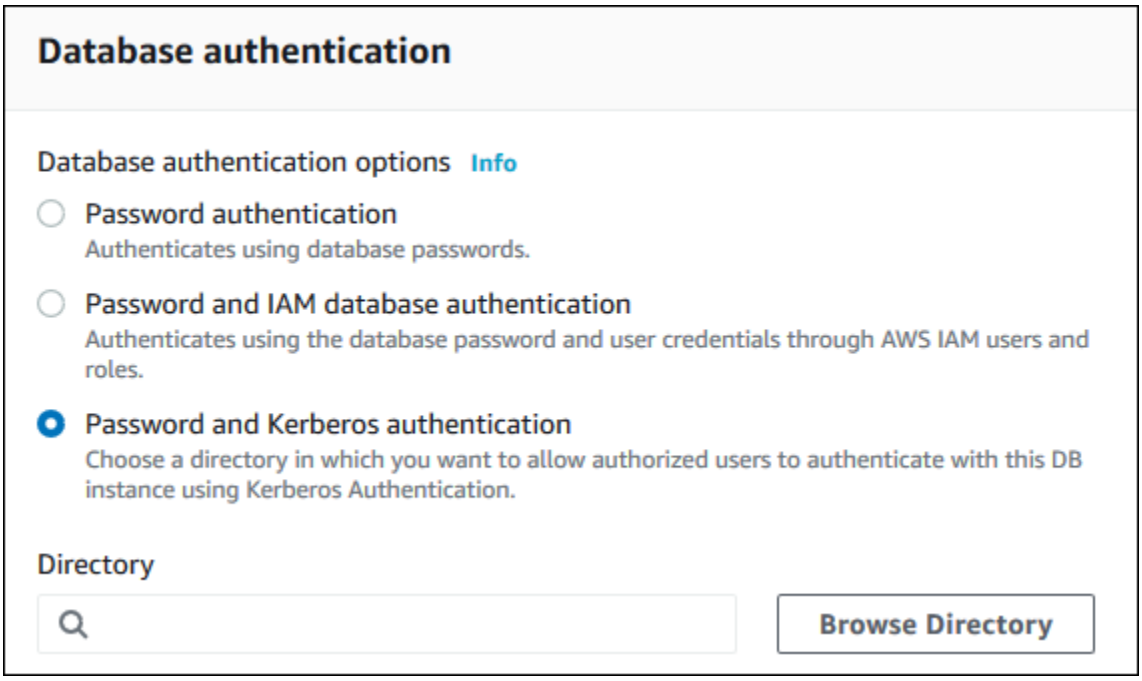
  지침은 DB 인스턴스 복원 섹션을 참조하세요.

- 콘솔, restore-db-instance-to-point-in-time CLI 명령 또는 RestoreDBInstanceToPointInTime RDS API 작업을 사용하여 MySQL DB 인스턴스를 특정 시점으로 복원합니다.

  지침은 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 섹션을 참조하세요.

Kerberos 인증은 VPC의 MySQL DB 인스턴스에 대해서만 지원됩니다. DB 인스턴스는 디렉터리와 동일한 VPC 또는 다른 VPC에 있을 수 있습니다. DB 인스턴스가 디렉터리와 통신할 수 있도록 DB 인스턴스는 디렉터리의 VPC 내 송신을 허용하는 보안 그룹을 사용해야 합니다.

콘솔을 사용하여 DB 인스턴스를 생성, 수정 또는 복원하는 경우 데이터베이스 인증 섹션에서 암호 및 Kerberos 인증을 선택합니다. 디렉터리 찾아보기를 선택한 다음 디렉터리를 선택하거나 새 디렉터리 생성을 선택합니다.



AWS CLI 또는 RDS API를 사용할 경우 DB 인스턴스를 디렉터리에 연결합니다. DB 인스턴스에서 생성된 도메인 디렉터리를 사용하려면 다음과 같은 파라미터가 필요합니다.

- `--domain` 파라미터의 경우 디렉터리를 만들 때 생성된 도메인 식별자("d-*" 식별자)를 사용하세요.
- `--domain-iam-role-name` 파라미터의 경우 귀하가 생성한, 관리형 IAM 정책 `AmazonRDSDirectoryServiceAccess`를 사용하는 역할을 사용하십시오.

예를 들어 다음 CLI 명령에서는 디렉터리를 사용하도록 DB 인스턴스를 수정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --domain d-ID \
    --domain-iam-role-name role-name
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --domain d-ID ^
    --domain-iam-role-name role-name
```

> **Important**
>
> DB 인스턴스를 수정하여 Kerberos 인증을 활성화하는 경우에는 변경 후 DB 인스턴스를 재부팅하세요.

5단계: Kerberos 인증 MySQL 로그인 생성

다른 DB 인스턴스의 경우와 마찬가지로 Amazon RDS 마스터 사용자 자격 증명을 사용하여 MySQL DB 인스턴스에 연결합니다. DB 인스턴스는 AWS Managed Microsoft AD 도메인에 조인됩니다. 따라서 도메인 내 Microsoft Active Directory 사용자에서 MySQL 로그인 및 사용자를 프로비저닝할 수 있습니다. 데이터베이스 권한은 이러한 로그인에서 부여 및 취소되는 표준 MySQL 권한을 통해 관리됩니다.

Active Directory 사용자가 MySQL에서 인증을 하도록 허용할 수 있습니다. 이렇게 하려면 먼저 다른 DB 인스턴스의 경우와 마찬가지로 Amazon RDS 마스터 사용자 자격 증명을 사용하여 MySQL DB 인

스턴스에 연결합니다. 로그인한 후에는 다음 명령을 실행하여 MySQL에서 플러그형 인증 모듈(PAM)을 사용하여 외부에서 인증된 사용자를 생성합니다. *testuser*를 사용자 이름으로 바꿉니다.

```
CREATE USER 'testuser'@'%' IDENTIFIED WITH 'auth_pam';
```

이제 도메인의 사용자(사람 및 애플리케이션)는 Kerberos 인증을 사용하여 도메인이 조인된 클라이언트 머신에서 DB 인스턴스에 연결할 수 있습니다.

Important

PAM 인증을 사용할 때는 클라이언트가 SSL/TLS 연결을 사용하는 것이 좋습니다. SSL/TLS 연결을 사용하지 않는 경우 암호가 일반 텍스트로 전송될 수도 있습니다. AD 사용자의 SSL/TLS 암호화된 연결을 사용하려면 다음 명령을 실행하고 *testuser*를 사용자 이름으로 교체합니다.

```
ALTER USER 'testuser'@'%' REQUIRE SSL;
```

자세한 내용은 Amazon RDS의 MySQL DB 인스턴스에 대한 SSL/TLS 지원 섹션을 참조하세요.

도메인에서 DB 인스턴스 관리

CLI 또는 RDS API를 사용하여 DB 인스턴스 및 DB 인스턴스와 관리형 Active Directory의 관계를 관리할 수 있습니다. 예를 들어 Active Directory를 연결하여 Kerberos 인증을 활성화하고 Active Directory의 연결을 해제하여 Kerberos 인증을 비활성화할 수 있습니다. 또한 한 Active Directory에서 외부 인증할 DB 인스턴스를 다른 도메인으로 이동시킬 수 있습니다.

예를 들어 Amazon RDS API를 사용하여 다음을 수행할 수 있습니다.

- 실패한 멤버십에 대해 Kerberos 인증 활성화를 다시 시도하려면 `ModifyDBInstance` API 작업을 사용하여 현재 멤버십의 디렉터리 ID를 지정합니다.
- 멤버십에 대한 IAM 역할 이름을 업데이트하려면 `ModifyDBInstance` API 작업을 사용하고 현재 멤버십의 디렉터리 ID 및 새 IAM 역할을 지정합니다.
- DB 인스턴스에서 Kerberos 인증을 비활성화하려면 `ModifyDBInstance` API 작업을 사용하여 `none`을 도메인 파라미터로 지정합니다.
- 한 도메인에서 다른 도메인으로 DB 인스턴스를 이동하려면 `ModifyDBInstance` API 작업을 사용하여 새 도메인의 도메인 식별자를 도메인 파라미터로 지정합니다.

- 각 DB 인스턴스의 멤버십을 나열하려면 `DescribeDBInstances` API 작업을 사용합니다.

도메인 멤버십 이해

DB 인스턴스를 생성하거나 수정하고 나면 해당 인스턴스는 도메인의 멤버가 됩니다. describe-db-instances CLI 명령을 실행하여 DB 인스턴스에 대한 도메인 멤버십의 상태를 확인할 수 있습니다. DB 인스턴스의 상태는 다음 중 한 가지가 될 수 있습니다.

- `kerberos-enabled` - DB 인스턴스에 Kerberos 인증이 활성화되어 있습니다.
- `enabling-kerberos` - AWS에서 이 DB 인스턴스에 대한 Kerberos 인증 활성화를 진행 중입니다.
- `pending-enable-kerberos` – 이 DB 인스턴스에 대한 Kerberos 인증 활성화가 보류 중입니다.
- `pending-maintenance-enable-kerberos` - AWS에서 예약된 다음 유지 관리 기간에 DB 인스턴스에 대한 Kerberos 인증을 활성화하려 합니다.
- `pending-disable-kerberos` – 이 DB 인스턴스에 대한 Kerberos 인증 비활성화가 보류 중입니다.
- `pending-maintenance-disable-kerberos` - AWS에서 예약된 다음 유지 관리 기간에 DB 인스턴스에 대한 Kerberos 인증을 비활성화하려 합니다.
- `enable-kerberos-failed` - 구성 문제로 인해 AWS에서 DB 인스턴스에 대해 Kerberos 인증을 활성화하지 못했습니다. DB 인스턴스 modify 명령을 다시 발행하기 전에 구성을 확인하고 수정합니다.
- `disabling-kerberos` - AWS에서 이 DB 인스턴스에 대한 Kerberos 인증 비활성화를 진행 중입니다.

네트워크 연결 문제 또는 잘못된 IAM 역할로 인해 Kerberos 인증 활성화 요청이 실패할 수 있습니다. 예를 들어 DB 인스턴스를 생성하거나 기존 DB 인스턴스를 수정하는데 Kerberos 인증을 활성화하려는 시도가 실패한다고 가정합니다. 이 경우 modify 명령을 다시 발행하거나 새로 생성된 DB 인스턴스를 수정하여 도메인에 조인합니다.

Kerberos 인증을 사용하여 MySQL에 연결

Kerberos 인증을 사용하여 MySQL에 연결하려면 Kerberos 인증 유형을 사용하여 로그인을 해야 합니다.

Kerberos 인증을 사용하여 연결할 수 있는 데이터베이스 사용자를 생성하려면 `IDENTIFIED WITH` 문에서 `CREATE USER` 절을 사용합니다. 지침은 5단계: Kerberos 인증 MySQL 로그인 생성 섹션을 참조하세요.

오류를 방지하려면 MariaDB `mysql` 클라이언트를 사용합니다. MariaDB 소프트웨어는 https://downloads.mariadb.org/에서 다운로드할 수 있습니다.

명령 프롬프트에서 MySQL DB 인스턴스와 연결된 엔드포인트 중 하나에 연결합니다. MySQL DB 인스턴스에 연결의 일반 절차를 따르십시오. 암호를 입력하라는 메시지가 표시되면 해당 사용자 이름과 연결된 Kerberos 암호를 입력합니다.

MySQL DB 인스턴스 복원 및 도메인에 추가

MySQL DB 인스턴스에서 DB 스냅샷을 복원하거나 특정 시점으로 복원을 수행한 후 이를 도메인에 추가할 수 있습니다. DB 인스턴스가 복원된 후 4단계: MySQL DB 인스턴스 생성 또는 수정에 설명된 프로세스를 사용하여 DB 인스턴스를 도메인에 추가하도록 인스턴스를 수정합니다.

Kerberos 인증 MySQL 제한 사항

MySQL용 Kerberos 인증에는 다음과 같은 제한이 적용됩니다.

- AWS Managed Microsoft AD만 지원됩니다. 하지만 RDS for MySQL DB 인스턴스에 가입하여 동일한 AWS 리전의 서로 다른 계정이 소유한 Managed Microsoft AD 도메인을 공유할 수 있습니다.
- 기능을 활성화한 후에는 DB 인스턴스를 재부팅해야 합니다.
- 도메인 이름 길이는 61자를 초과할 수 없습니다.
- Kerberos 인증과 IAM 인증을 동시에 활성화할 수 없습니다. MySQL DB 인스턴스에 대해 하나의 인증 방법을 선택하거나 다른 방법을 선택합니다.
- 기능을 활성화한 후에는 DB 인스턴스 포트를 수정하지 마십시오.
- 읽기 전용 복제본에는 Kerberos 인증을 사용하지 마십시오.
- Kerberos 인증을 사용하는 MySQL DB 인스턴스에 대해 자동 마이너 버전 업그레이드를 설정한 경우 Kerberos 인증을 해제했다가 자동 업그레이드 후 다시 설정해야 합니다. 자동 마이너 버전 업그레이드에 대한 자세한 내용은 RDS for MySQL 마이너 버전 자동 업그레이드 섹션을 참조하세요.
- 이 기능이 활성화된 DB 인스턴스를 삭제하려면 먼저 이 기능을 비활성화합니다. 이렇게 하려면 DB 인스턴스에 대해 `modify-db-instance` CLI 명령을 사용하고 `--domain` 파라미터에 대해 `none`을 지정합니다.

  CLI 또는 RDS API를 사용하여 이 기능이 활성화된 DB 인스턴스를 삭제하는 경우 지연이 발생할 수 있습니다.
- RDS for MySQL은 온프레미스 또는 자체 호스팅 AD와 AWS Managed Microsoft AD 간의 포리스트 신뢰 관계에서 Kerberos 인증을 지원하지 않습니다.

Amazon RDS Optimized Reads로 RDS for MySQL 쿼리 성능 개선

Amazon RDS Optimized Reads로 RDS for MySQL의 쿼리 처리 속도를 높일 수 있습니다. RDS Optimized Reads를 사용하는 RDS for MySQL DB 인스턴스 또는 다중 AZ DB 클러스터는 이를 사용하지 않는 DB 인스턴스 또는 클러스터에 비해 쿼리 처리 속도가 최대 2배 더 빠릅니다.

주제

- RDS Optimized Reads 개요
- RDS Optimized Reads 사용 사례
- RDS Optimized Reads 모범 사례
- RDS Optimized Reads 사용
- RDS Optimized Reads를 사용하는 DB 인스턴스 모니터링
- RDS Optimized Reads 제한 사항

RDS Optimized Reads 개요

RDS Optimized Reads가 켜져 있는 RDS for MySQL DB 인스턴스 또는 다중 AZ DB 클러스터를 사용하면 인스턴스 스토어를 사용하여 더 빠른 쿼리 성능을 얻을 수 있습니다. 인스턴스 스토어는 DB 인스턴스 또는 다중 AZ DB 클러스터에 블록 수준의 임시 스토리지를 제공합니다. 스토리지는 호스트 서버에 물리적으로 연결된 NVMe(Non-Volatile Memory Express) 솔리드 스테이트 드라이브(SSD)에 있습니다. 이 스토리지는 짧은 지연 시간, 높은 임의 I/O 성능, 높은 순차 읽기 처리량(throughput)에 최적화되어 있습니다.

RDS Optimized Reads는 DB 인스턴스 또는 다중 AZ DB 클러스터가 db.m5d 또는 db.m6gd 같은 DB 인스턴스 클래스를 인스턴스 스토어와 함께 사용할 때 기본적으로 켜집니다. RDS Optimized Reads를 사용하면 일부 임시 객체가 인스턴스 스토어에 저장됩니다. 이러한 임시 객체에는 내부 임시 파일, 내부 온디스크 임시 테이블, 메모리 맵 파일, 이진 로그(binlog) 캐시 파일이 포함됩니다. 인스턴스 스토어 유형에 대한 자세한 내용은 Linux 인스턴스용 Amazon Elastic Compute Cloud 사용 설명서의 Amazon EC2 인스턴스 스토어 섹션을 참조하세요.

쿼리 처리를 위해 MySQL에서 임시 객체를 생성하는 워크로드는 인스턴스 스토어를 활용하여 쿼리를 더 빠르게 처리할 수 있습니다. 이러한 유형의 워크로드에는 정렬, 해시 집계, 고부하 조인, CTE(Common Table Expression) 및 인덱싱되지 않은 열에 대한 쿼리가 포함됩니다. 이러한 인스턴스 스토어 볼륨은 영구 Amazon EBS 스토리지에 사용되는 스토리지 구성과 관계없이 더 높은 IOPS와 성능을 제공합니다. RDS Optimized Reads는 임시 객체에 대한 작업을 인스턴스 스토어로 오프로드하므

로 이제 영구 객체에 대한 작업에 영구 스토리지(Amazon EBS)의 초당 입출력 작업 처리량(IOPS) 또는 처리량(throughput)을 사용할 수 있습니다. 이러한 작업에는 일반 데이터 파일 읽기 및 쓰기와 플러싱 및 버퍼 병합 삽입과 같은 백그라운드 엔진 작업이 포함됩니다.

> **Note**
>
> 수동 및 자동 RDS 스냅샷에는 영구 객체를 위한 엔진 파일만 포함됩니다. 인스턴스 스토어에서 생성된 임시 객체는 RDS 스냅샷에 포함되지 않습니다.

RDS Optimized Reads 사용 사례

쿼리 실행을 위해 내부 테이블이나 파일 같은 임시 객체에 크게 의존하는 워크로드가 있는 경우 RDS Optimized Reads를 켜는 것이 좋습니다. RDS Optimized Reads의 사용 사례는 다음과 같습니다.

- 복잡한 CTE(Common Table Expression), 파생된 테이블, 그룹화 작업을 사용하여 분석 쿼리를 실행하는 애플리케이션
- 최적화되지 않은 쿼리로 많은 읽기 트래픽을 처리하는 읽기 전용 복제본
- `GROUP BY` 및 `ORDER BY` 절이 있는 쿼리와 같이 복잡한 작업이 필요한 온디맨드 또는 동적 보고 쿼리를 실행하는 애플리케이션
- 쿼리 처리를 위해 내부 임시 테이블을 사용하는 워크로드

  엔진 상태 변수 `created_tmp_disk_tables`를 모니터링하여 DB 인스턴스에 생성된 디스크 기반 임시 테이블의 수를 확인할 수 있습니다.
- 직접 또는 절차에 따라 대형 임시 테이블을 생성하여 중간 결과를 저장하는 애플리케이션
- 인덱싱되지 않은 열에서 그룹화 또는 정렬 작업을 수행하는 데이터베이스 쿼리

RDS Optimized Reads 모범 사례

RDS Optimized Reads에 대한 다음 모범 사례를 따릅니다.

- 실행 중 인스턴스 스토어가 가득 차서 쿼리가 실패하는 경우에 대비하여 읽기 전용 쿼리의 재시도 로직을 추가합니다.
- CloudWatch 지표 `FreeLocalStorage`를 사용하여 인스턴스 스토어에서 사용 가능한 스토리지 공간을 모니터링합니다. DB 인스턴스의 워크로드로 인해 인스턴스 스토어가 한도에 도달하면 더 큰 DB 인스턴스 클래스를 사용하도록 DB 인스턴스를 수정하세요.

- DB 인스턴스 또는 다중 AZ DB 클러스터의 메모리가 충분하지만 여전히 인스턴스 스토어의 스토리지 한도에 도달하면 `binlog_cache_size` 값을 늘려 세션별 binlog 항목을 메모리에 유지합니다. 이 구성은 binlog 항목을 디스크의 임시 binlog 캐시 파일에 쓰는 것을 방지합니다.

  `binlog_cache_size` 파라미터는 세션별로 다릅니다. 새 세션마다 값을 변경할 수 있습니다. 이 파라미터를 설정하면 피크 워크로드 중에 DB 인스턴스의 메모리 사용률을 높일 수 있습니다. 따라서 애플리케이션의 워크로드 패턴과 DB 인스턴스의 가용 메모리를 기반으로 파라미터 값을 높이는 것을 고려해 보세요.
- MySQL 8.0 버전 이하에서는 `binlog_format` 파라미터에 기본값인 `MIXED`를 사용합니다. 트랜잭션 크기에 따라 `binlog_format`을 `ROW`로 설정하면 인스턴스 스토어에 큰 binlog 캐시 파일이 생성될 수 있습니다. MySQL 8.4 이상의 경우 `binlog_format` 파라미터에 기본값인 `ROW`를 사용합니다.
- internal_tmp_mem_storage_engine 파라미터를 `TempTable`로 설정하고, temptable_max_mmap 파라미터를 인스턴스 스토어의 사용 가능한 스토리지 크기에 맞게 설정합니다.
- 단일 트랜잭션에서 대량 변경을 수행하지 마세요. 이러한 유형의 트랜잭션은 인스턴스 스토어에 대용량 binlog 캐시 파일을 생성할 수 있으며, 인스턴스 스토어가 가득 차면 문제가 발생할 수 있습니다. binlog 캐시 파일의 스토리지 사용을 최소화하려면 쓰기를 여러 개의 작은 트랜잭션으로 분할하는 것이 좋습니다.
- `ABORT_SERVER` 파라미터에 기본값 `binlog_error_action`를 사용합니다. 이렇게 하면 백업이 활성화된 DB 인스턴스에서 이진 로깅 문제를 방지할 수 있습니다.

RDS Optimized Reads 사용

단일 AZ DB 인스턴스 배포, 다중 AZ DB 인스턴스 배포 또는 다중 AZ DB 클러스터 배포에서 다음 DB 인스턴스 클래스 중 하나를 사용하여 RDS for MySQL DB 인스턴스를 프로비저닝하면 DB 인스턴스는 자동으로 RDS Optimized Reads를 사용합니다.

RDS Optimized Read를 켜려면 다음 중 하나를 수행합니다.

- 이러한 DB 인스턴스 클래스 중 하나를 사용하여 RDS for MySQL DB 인스턴스 또는 다중 AZ DB 클러스터를 생성합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 단원을 참조하십시오.
- 이러한 DB 인스턴스 클래스 중 하나를 사용하여 기존 RDS for MySQL DB 인스턴스 또는 다중 AZ DB 클러스터를 수정합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

RDS Optimized Reads는 로컬 NVMe SSD 스토리지가 있는 DB 인스턴스 클래스 중 하나 이상이 지원되는 모든 AWS 리전 RDS에서 사용 가능합니다. DB 인스턴스 클래스에 대한 자세한 내용은 the section called “DB 인스턴스 클래스” 섹션을 참조하세요.

DB 인스턴스 클래스 가용성은 AWS 리전에 따라 다릅니다. 특정 AWS 리전 클래스에서 DB 인스턴스 클래스가 지원되는지 여부를 확인하려면 the section called “AWS 리전에서 DB 인스턴스 클래스 지원 확인”를 참조하세요.

RDS Optimized Reads를 사용하지 않으려면 기능을 지원하는 DB 인스턴스 클래스를 사용하지 않도록 DB 인스턴스 또는 다중 AZ DB 클러스터를 수정하세요.

RDS Optimized Reads를 사용하는 DB 인스턴스 모니터링

다음 CloudWatch 지표를 사용하여 RDS Optimized Reads를 사용하는 DB 인스턴스를 모니터링할 수 있습니다.

- `FreeLocalStorage`
- `ReadIOPSLocalStorage`
- `ReadLatencyLocalStorage`
- `ReadThroughputLocalStorage`
- `WriteIOPSLocalStorage`
- `WriteLatencyLocalStorage`
- `WriteThroughputLocalStorage`

이러한 지표는 사용 가능한 인스턴스 스토어 스토리지, IOPS 및 처리량(throughput)에 대한 데이터를 제공합니다. 이러한 지표에 대한 자세한 내용은 Amazon RDS에 대한 Amazon CloudWatch 지표 섹션을 참조하세요.

RDS Optimized Reads 제한 사항

RDS Optimized Reads에는 다음과 같은 제한 사항이 적용됩니다.

- RDS Optimized Reads는 다음 버전에서 지원됩니다.
  - RDS for MySQL 8.0.28 이상 메이저 및 마이너 버전

  RDS for MySQL 버전에 대한 자세한 내용은 Amazon RDS의 MySQL 버전 섹션을 참조하세요.
- RDS Optimized Reads를 지원하는 DB 인스턴스 클래스에서는 임시 객체의 위치를 영구 스토리지(Amazon EBS)로 변경할 수 없습니다.
- DB 인스턴스에서 이진 로깅이 활성화된 경우 최대 트랜잭션 크기는 인스턴스 스토어의 크기에 의해 제한됩니다. MySQL에서 `binlog_cache_size` 값보다 더 많은 스토리지가 필요한 세션은 인스턴스 스토어에 생성된 임시 binlog 캐시 파일에 트랜잭션 변경 사항을 기록합니다.

- 인스턴스 스토어가 가득 차면 트랜잭션이 실패할 수 있습니다.

RDS Optimized Writes for MySQL을 통한 쓰기 성능 개선

RDS Optimized Writes for MySQL을 사용하여 쓰기 트랜잭션의 성능을 개선할 수 있습니다. RDS for MySQL 데이터베이스에서 RDS Optimized Writes를 사용하는 경우 쓰기 트랜잭션 처리량(throughput)을 최대 2배까지 높일 수 있습니다.

주제

- RDS Optimized Writes 개요
- RDS Optimized Writes 사용
- 기존 데이터베이스에서 RDS 최적화된 쓰기 활성화
- RDS Optimized Writes 제한 사항

RDS Optimized Writes 개요

RDS 최적화된 쓰기를 켜면 이중 쓰기 버퍼를 사용할 필요 없이 내구성이 뛰어난 스토리지로 데이터를 플러싱할 때 RDS for MySQL 데이터베이스가 데이터를 한 번만 씁니다. 데이터베이스는 향상된 성능과 함께 안정적인 데이터베이스 트랜잭션을 위한 ACID 속성 보호를 지속적으로 제공합니다.

MySQL 같은 관계형 데이터베이스는 신뢰할 수 있는 데이터베이스 트랜잭션을 위한 ACID(원자성, 일관성, 격리, 내구성) 속성을 제공합니다. 이러한 속성을 제공하기 위해 MySQL은 이중 쓰기 버퍼라는 데이터 스토리지 영역을 사용하여 부분 페이지 쓰기 오류를 방지합니다. 이러한 오류는 정전의 경우처럼 데이터베이스에서 페이지를 업데이트하는 동안 하드웨어 장애가 있을 때 발생합니다. MySQL 데이터베이스는 부분 페이지 쓰기를 감지하고 이중 쓰기 버퍼에 있는 페이지 복사본으로 복구할 수 있습니다. 이 기술은 보호 기능을 제공하지만 추가 쓰기 작업도 초래합니다. MySQL 이중 쓰기 버퍼에 대한 자세한 내용은 MySQL 설명서의 Doublewrite Buffer를 참조하세요.

Amazon RDS Optimized Writes를 켜면 이중 쓰기 버퍼를 사용하지 않고 내구성이 뛰어난 스토리지로 데이터를 플러싱할 때 RDS for MySQL 데이터베이스가 데이터를 한 번만 씁니다. RDS Optimized Writes는 RDS for MySQL 데이터베이스에서 쓰기가 많은 워크로드를 실행하는 경우에 유용합니다. 쓰기 작업이 많은 데이터베이스의 예로는 디지털 결제, 금융 거래, 게임 애플리케이션을 지원하는 데이터베이스가 있습니다.

이러한 데이터베이스는 AWS Nitro System을 사용하는 DB 인스턴스 클래스에서 실행됩니다. 이러한 시스템의 하드웨어 구성으로 인해 데이터베이스는 한 단계로 안정적이고 내구성 있게 16KiB 페이지를 데이터 파일에 직접 쓸 수 있습니다. AWS Nitro 시스템은 RDS Optimized Writes를 가능하게 합니다.

새 데이터베이스 매개 변수 `rds.optimized_writes`를 설정하여 RDS for MySQL 데이터베이스의 RDS Optimized Writes 기능을 제어할 수 있습니다. RDS for MySQL 버전 8.0 및 RDS for MySQL 버전

8.4의 DB 파라미터 그룹에서 이 파라미터에 액세스하세요. 다음 값을 사용하여 파라미터를 설정합니다.

- `AUTO` - 데이터베이스에서 지원하는 경우 RDS Optimized Writes를 켭니다. 데이터베이스에서 지원하지 않는 경우 RDS Optimized Writes를 해제합니다. 이 설정이 기본값입니다.
- `OFF` - 데이터베이스에서 지원하더라도 RDS Optimized Writes를 해제합니다.

엔진 버전, DB 인스턴스 클래스 및/또는 파일 시스템 형식이 RDS 최적화된 쓰기를 지원하지 않는 기존 데이터베이스가 있는 경우 블루/그린 배포를 생성하여 이 기능을 활성화할 수 있습니다. 자세한 내용은 the section called “기존 데이터베이스에서 활성화” 단원을 참조하십시오.

RDS Optimized Writes를 사용하도록 구성된 RDS for MySQL 데이터베이스를 해당 기능을 지원하지 않는 DB 인스턴스 클래스로 마이그레이션하는 경우, RDS는 데이터베이스의 RDS Optimized Writes를 자동으로 해제합니다.

RDS Optimized Writes가 해제된 경우 데이터베이스는 MySQL 이중 쓰기 버퍼를 사용합니다.

RDS for MySQL 데이터베이스에서 RDS Optimized Writes를 사용하고 있는지 확인하려면 데이터베이스 `innodb_doublewrite` 파라미터의 현재 값을 확인하세요. 데이터베이스에서 RDS Optimized Writes를 사용 중인 경우 이 파라미터는 `FALSE(0)`로 설정됩니다.

RDS Optimized Writes 사용

RDS 콘솔, AWS CLI 또는 RDS API를 사용하여 RDS for MySQL 데이터베이스를 생성할 때 RDS Optimized Writes를 켤 수 있습니다. 데이터베이스 생성 중에 다음 두 가지 조건이 모두 적용되는 경우 RDS Optimized Writes가 자동으로 켜집니다.

- RDS Optimized Writes를 지원하는 DB 엔진 버전과 DB 인스턴스 클래스를 지정합니다.
  - RDS Optimized Writes는 RDS for MySQL 버전 8.0.30 이상에서 지원됩니다. RDS for MySQL 버전에 대한 자세한 내용은 Amazon RDS의 MySQL 버전 섹션을 참조하세요.
  - RDS Optimized Writes는 다음 DB 인스턴스 클래스를 사용하는 RDS for MySQL 데이터베이스에서 지원됩니다.
    - db.m7i
    - db.m7g
    - db.m6g
    - db.m6gd
    - db.m6i

- db.m5
- db.m5d
- db.r7i
- db.r7g
- db.r6g
- db.r6gd
- db.r6i
- db.r5
- db.r5b
- db.r5d
- db.x2idn
- db.x2iedn

DB 인스턴스 클래스에 대한 자세한 내용은 the section called “DB 인스턴스 클래스” 섹션을 참조하세요.

DB 인스턴스 클래스 가용성은 AWS 리전에 따라 다릅니다. 특정 AWS 리전 클래스에서 DB 인스턴스 클래스가 지원되는지 여부를 확인하려면 the section called “AWS 리전에서 DB 인스턴스 클래스 지원 확인”를 참조하세요.

RDS 최적화된 쓰기를 지원하는 DB 인스턴스 클래스로 데이터베이스를 업그레이드하려면 블루/그린 배포를 생성하면 됩니다. 자세한 내용은 the section called “기존 데이터베이스에서 활성화” 단원을 참조하십시오.

- 데이터베이스에 연결된 파라미터 그룹에서는 `rds.optimized_writes` 파라미터가 `AUTO`로 설정됩니다. 기본 파라미터 그룹에서는 이 파라미터가 항상 `AUTO`로 설정됩니다.

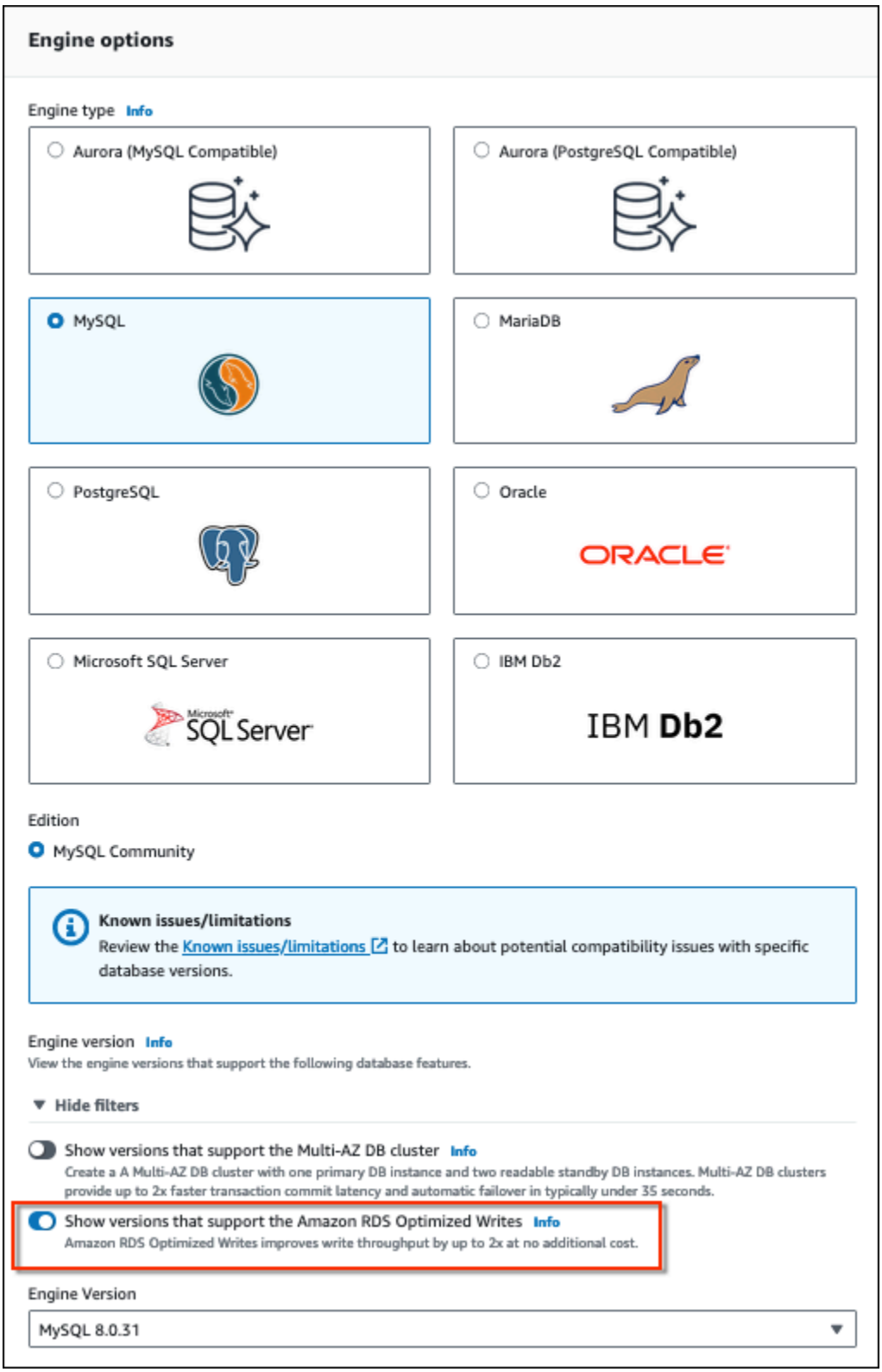
RDS Optimized Writes를 지원하는 DB 엔진 버전과 DB 인스턴스 클래스를 사용하고 싶지만 이 기능은 사용하고 싶지 않을 경우, 데이터베이스를 생성할 때 사용자 지정 파라미터 그룹을 지정하세요. 이 파라미터 그룹에서 `rds.optimized_writes` 파라미터를 `OFF`로 설정합니다. 나중에 데이터베이스에서 RDS Optimized Writes를 사용하도록 하려면 파라미터를 `AUTO`로 설정하여 켤 수 있습니다. 사용자 지정 파라미터 그룹 및 파라미터 설정에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 섹션을 참조하세요.

DB 인스턴스 생성에 대한 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.

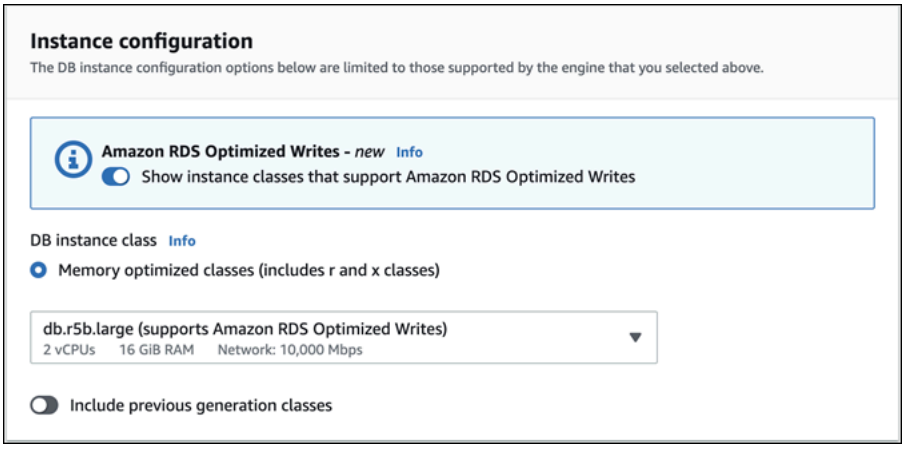
콘솔

RDS 콘솔을 사용하여 RDS for MySQL 데이터베이스를 생성할 때 RDS Optimized Writes를 지원하는 DB 엔진 버전 및 DB 인스턴스 클래스를 필터링할 수 있습니다. 필터를 켜면 사용 가능한 DB 엔진 버전 및 DB 인스턴스 클래스 중에서 선택할 수 있습니다.

RDS Optimized Writes를 지원하는 DB 엔진 버전을 선택하려면 엔진 버전에서 이를 지원하는 RDS for MySQL DB 엔진 버전을 필터링한 다음 버전을 선택합니다.



인스턴스 구성 섹션에서 RDS Optimized Writes를 지원하는 DB 인스턴스 클래스를 필터링한 다음 DB 인스턴스 클래스를 선택합니다.



이러한 선택을 완료한 후 요구 사항에 맞는 다른 설정을 선택하고 콘솔을 사용하여 RDS for MySQL 데이터베이스 생성을 완료할 수 있습니다.

AWS CLI

AWS CLI를 사용하여 DB 인스턴스를 만들려면 create-db-instance 명령을 실행합니다. `--engine-version` 및 `--db-instance-class` 값이 RDS Optimized Writes를 지원하는지 확인하세요. 또한 DB 인스턴스에 연결된 파라미터 그룹에서 `rds.optimized_writes` 파라미터가 `AUTO`로 설정되어 있는지 확인합니다. 다음 예제에서는 기본 파라미터 그룹을 DB 인스턴스와 연결합니다.

Example RDS Optimized Writes를 사용하는 DB 인스턴스 생성

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --engine mysql \
    --engine-version 8.0.30 \
    --db-instance-class db.r5b.large \
    --manage-master-user-password \
    --master-username admin \
    --allocated-storage 200
```

Windows의 경우:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --engine mysql ^
    --engine-version 8.0.30 ^
    --db-instance-class db.r5b.large ^
    --manage-master-user-password ^
```

```
    --master-username admin ^
    --allocated-storage 200
```

RDS API

CreateDBInstance 작업을 사용하여 DB 인스턴스를 생성할 수 있습니다. 이 작업을 사용할 경우 `EngineVersion` 및 `DBInstanceClass` 값이 RDS Optimized Writes를 지원하는지 확인하세요. 또한 DB 인스턴스에 연결된 파라미터 그룹에서 `rds.optimized_writes` 파라미터가 `AUTO`로 설정되어 있는지 확인합니다.

기존 데이터베이스에서 RDS 최적화된 쓰기 활성화

RDS 최적화된 쓰기를 켜도록 기존 RDS for MySQL 데이터베이스를 수정하려면 지원되는 DB 엔진 버전 및 DB 인스턴스 클래스로 데이터베이스가 생성된 상태여야 합니다. 또한 RDS 최적화된 쓰기가 출시된 2022년 11월 27일 이후에 생성한 데이터베이스여야 합니다. 필요한 기본 파일 시스템 구성이 릴리스 전에 생성된 데이터베이스의 구성과 호환되지 않기 때문입니다. 이러한 조건이 충족되면 `rds.optimized_writes` 파라미터를 `AUTO`로 설정하여 RDS 최적화된 쓰기를 활성화할 수 있습니다.

지원되는 엔진 버전, 인스턴스 클래스 또는 파일 시스템 구성으로 데이터베이스를 생성하지 않은 경우 RDS 블루/그린 배포를 사용하여 지원되는 구성으로 마이그레이션할 수 있습니다. 블루/그린 배포를 생성하는 동안 다음을 수행하세요.

- 그린 데이터베이스에서 Optimized Writes 활성화를 선택하고 RDS 최적화된 쓰기를 지원하는 엔진 버전과 DB 인스턴스 클래스를 지정합니다. 지원되는 엔진 버전과 인스턴스 클래스의 목록은 RDS Optimized Writes 사용 섹션을 참조하세요.
- 스토리지에서 스토리지 파일 시스템 구성 업그레이드를 선택합니다. 이 옵션은 데이터베이스를 호환 가능한 기본 파일 시스템 구성으로 업그레이드합니다.

블루/그린 배포를 생성하는 도중 `rds.optimized_writes` 파라미터가 `AUTO`로 설정된 경우 그린 환경에서 RDS 최적화된 쓰기가 자동으로 활성화됩니다. 그런 다음 블루/그린 배포를 전환하여 그린 환경을 새로운 프로덕션 환경으로 승격합니다.

자세한 내용은 the section called “블루/그린 배포 생성” 단원을 참조하십시오.

RDS Optimized Writes 제한 사항

스냅샷에서 RDS for MySQL 데이터베이스를 복원할 때는 다음 조건이 모두 적용되는 경우에만 데이터베이스의 RDS 최적화된 쓰기를 켤 수 있습니다.

- RDS Optimized Writes를 지원하는 데이터베이스에서 스냅샷이 생성되었습니다.
- RDS Optimized Writes가 릴리스된 이후에 생성한 데이터베이스에서 스냅샷이 생성되었습니다.
- RDS Optimized Writes를 지원하는 데이터베이스로 스냅샷이 복원됩니다.
- 복원된 데이터베이스는 `rds.optimized_writes` 파라미터가 `AUTO`로 설정된 파라미터 그룹에 연결됩니다.

RDS for MySQL DB 엔진 업그레이드

Amazon RDS에서 새 데이터베이스 엔진 버전을 지원하는 경우, DB 인스턴스를 새 버전으로 업그레이드할 수 있습니다. MySQL 데이터베이스의 업그레이드에는 메이저 버전 업그레이드와 마이너 버전 업그레이드라는 2가지 종류가 있습니다.

메이저 버전 업그레이드

메이저 버전 업그레이드에는 기존 애플리케이션과 호환되지 않는 데이터베이스 변경 사항이 포함될 수 있습니다. 따라서 DB 인스턴스의 메이저 버전 업그레이드를 수동으로 수행해야 합니다. DB 인스턴스를 수정하여 메이저 버전 업그레이드를 시작할 수 있습니다. 메이저 버전 업그레이드를 수행하기 전에 RDS for MySQL 메이저 버전 업그레이드의 지침을 따르는 것이 좋습니다.

다중 AZ DB 인스턴스 배포의 메이저 버전 업그레이드인 경우 Amazon RDS는 기본 복제본과 대기 복제본을 동시에 업그레이드합니다. 업그레이드가 완료될 때까지 DB 인스턴스를 사용할 수 없습니다. 다중 AZ DB 클러스터 배포의 메이저 버전 업그레이드인 경우 Amazon RDS는 클러스터 멤버 인스턴스를 한 번에 하나씩 업그레이드합니다.

> **Tip**
>
> 블루/그린 배포를 사용하면 메이저 버전 업그레이드에 필요한 다운타임을 최소화할 수 있습니다. 자세한 내용은 데이터베이스 업데이트에 Amazon RDS 블루/그린 배포 사용 단원을 참조하십시오.

마이너 버전 업그레이드

마이너 버전 업그레이드에는 기존 애플리케이션과 호환되는 변경 사항만 포함됩니다. DB 인스턴스를 수정하여 마이너 버전 업그레이드를 수동으로 시작할 수 있습니다. 또는 DB 인스턴스를 생성하거나 수정할 때 마이너 버전 자동 업그레이드 옵션을 활성화할 수 있습니다. 이렇게 하면 Amazon RDS에서 새 버전을 테스트 및 승인한 후 DB 인스턴스가 자동으로 업그레이드됩니다. 업그레이드 수행에 대한 자세한 내용은 DB 인스턴스 엔진 버전 업그레이드 단원을 참조하십시오.

다중 AZ DB 클러스터의 마이너 버전 업그레이드를 수행하면 Amazon RDS는 리더 DB 인스턴스를 한 번에 하나씩 업그레이드합니다. 그러면 리더 DB 인스턴스가 새 라이터 DB 인스턴스로 전환됩니다. 그러면 Amazon RDS가 이전 라이터 인스턴스(현재는 리더 인스턴스)를 업그레이드합니다.

> **Note**
>
> 다중 AZ DB 인스턴스 배포의 마이너 버전 업그레이드로 인한 다운타임은 몇 분 동안 지속될 수 있습니다. 다중 AZ DB 클러스터는 일반적으로 마이너 버전 업그레이드의 가동 중지 시간을 약 35초로 줄입니다. RDS 프록시와 함께 사용하면 다운타임을 1초 이하로 더 줄일 수 있습니다. 자세한 내용은 Amazon RDS Proxy 단원을 참조하십시오. 또는 ProxySQL, PgBouncer 또는 AWS 고급 JDBC 래퍼 드라이버와 같은 오픈 소스 데이터베이스 프록시를 사용할 수 있습니다.

MySQL DB 인스턴스가 읽기 복제본을 사용하는 경우 소스 인스턴스를 업그레이드하기 전에 읽기 복제본을 모두 업그레이드해야 합니다.

주제

- MySQL 업그레이드 고려 사항
- 유효한 업그레이드 대상 찾기
- MySQL 버전 번호
- RDS for MySQL의 RDS 버전 번호
- RDS for MySQL 메이저 버전 업그레이드
- RDS for MySQL 업그레이드 테스트
- MySQL DB 인스턴스 업그레이드
- RDS for MySQL 마이너 버전 자동 업그레이드
- RDS for MySQL 데이터베이스 업그레이드 시 읽기 전용 복제본을 사용하여 가동 중지 시간 단축

MySQL 업그레이드 고려 사항

Amazon RDS는 업그레이드 프로세스 중에 DB 스냅샷을 2개 이상 캡처합니다. Amazon RDS는 업그레이드를 변경하기 전에 DB 인스턴스의 스냅샷을 최대 2개까지 캡처합니다. 업그레이드가 데이터베이스에 맞지 않는 경우에는 이 스냅샷을 복구하여 이전 버전을 실행하는 DB 인스턴스를 생성할 수 있습니다. Amazon RDS는 업그레이드가 완료되면 DB 인스턴스의 또 다른 스냅샷을 캡처합니다. Amazon RDS는 AWS Backup에서 DB 인스턴스의 백업을 관리하는지 여부에 관계없이 이러한 스냅샷을 생성합니다.

> **Note**
>
> DB 인스턴스에 대한 백업 보존 기간을 0보다 큰 수로 설정하면 Amazon RDS는 DB 스냅샷만 캡처합니다. 백업 보존 기간을 변경하려면 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

업그레이드가 완료되면 이전 버전의 데이터베이스 엔진으로 되돌릴 수 없습니다. 이때 이전 버전으로 되돌리려면 첫 번째로 캡처한 DB 스냅샷을 복구하여 새로운 DB 인스턴스를 생성해야 합니다.

DB 인스턴스를 Amazon RDS가 지원하는 새 버전으로 업그레이드하는 시기는 사용자가 직접 관리합니다. 이러한 관리 수준은 특정 데이터베이스 버전과 호환성을 유지하거나 프로덕션 환경에 배포하기 전에 애플리케이션을 이용해 새 버전을 테스트하는 데 효과적입니다. 모든 준비를 마치면 일정에 가장 적합한 시기에 버전 업그레이드를 실행할 수 있습니다.

DB 인스턴스가 읽기 복제본을 사용하는 경우 소스 인스턴스를 업그레이드하기 전에 읽기 복제본부터 모두 업그레이드해야 합니다.

유효한 업그레이드 대상 찾기

AWS Management Console을 사용하여 DB 인스턴스를 업그레이드하면 DB 인스턴스의 유효한 업그레이드 대상이 표시됩니다. 다음 AWS CLI 명령을 실행하여 DB 인스턴스의 유효한 업그레이드 대상을 식별할 수도 있습니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-engine-versions \
  --engine mysql \
  --engine-version version_number \
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

Windows의 경우:

```
aws rds describe-db-engine-versions ^
  --engine mysql ^
  --engine-version version_number ^
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

예를 들어 MySQL 버전 8.0.28 DB 인스턴스의 유효한 업그레이드 대상을 식별하려면 다음 AWS CLI 명령을 실행합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-engine-versions \
  --engine mysql \
  --engine-version 8.0.28 \
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

Windows의 경우:

```
aws rds describe-db-engine-versions ^
  --engine mysql ^
  --engine-version 8.0.28 ^
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

MySQL 버전 번호

RDS for MySQL용 데이터베이스 엔진의 버전 번호 지정 시퀀스는 major.minor.patch.YYYYMMDD 또는 major.minor.patch 형식(예: 8.0.33.R2.20231201 또는 5.7.44)입니다. 사용된 형식은 MySQL 엔진 버전에 따라 다릅니다. RDS 추가 지원 버전 번호 지정에 대한 자세한 내용은 Amazon RDS 추가 지원 버전 명명 규칙 섹션을 참조하세요.

메이저

버전 번호의 정수 부분과 첫 번째 소수 부분 모두가 메이저 버전 번호입니다(예: 8.0). 메이저 버전 업그레이드는 버전 번호의 메이저 부분이 증가합니다. 예를 들어 5.7.44에서 8.0.33으로 업그레이드하는 것은 메이저 버전 업그레이드입니다. 여기서 5.7과 8.0이 메이저 버전 번호입니다.

마이너

버전 번호의 세 번째 부분이 마이너 버전 번호입니다(예: 8.0.33의 33).

패치

패치는 버전 번호의 네 번째 부분입니다(예: 8.0.33.R2의 R2). RDS 패치 버전에는 릴리스 후 마이너 버전에 추가된 중요한 버그 수정이 포함되어 있습니다.

YYYYMMDD(날짜)

날짜는 버전 번호의 다섯 번째 부분입니다(예: 8.0.33.R2.20231201의 20231201). RDS 날짜 버전은 보안 패치로 릴리스 후 마이너 버전에 추가된 중요한 보안 수정이 포함되어 있습니다. 엔진 동작을 변경할 수 있는 수정 사항은 포함되지 않습니다.

다음 표에서는 RDS for MySQL 버전 8.4의 이름 지정 체계를 설명합니다.

8.4 마이너 버전	이름 지정 체계
≥ 3	새 DB 인스턴스는 major.minor.patch.YYMMDD를 사용합니다(예: 8.4.3.R2.20241201). 기존 DB 인스턴스는 다음 메이저 또는 마이너 버전 업그레이드 전까지 major.minor.patch(예: 8.4.3.R2)를 사용할 수 있습니다.

다음 표에서는 RDS for MariaDB 버전 8.0의 이름 지정 체계를 설명합니다.

8.0 마이너 버전	이름 지정 체계
33 이상	새 DB 인스턴스는 major.minor.patch.YYMMDD를 사용합니다(예: 8.0.33.R2.20231201). 기존 DB 인스턴스는 다음 메이저 또는 마이너 버전 업그레이드 전까지 major.minor.patch(예: 8.0.33.R2)를 사용할 수 있습니다.
33 미만	기존 DB 인스턴스는 major.minor.patch를 사용합니다(예: 8.0.32.R2).

다음 표에서는 RDS for MariaDB 버전 5.7의 이름 지정 체계를 설명합니다.

5.7 마이너 버전	이름 지정 체계
42 이상	새 DB 인스턴스는 major.minor.patch.YYMMDD를 사용합니다(예: 5.7.42.R2.20231201).

5.7 마이너 버전	이름 지정 체계
	기존 DB 인스턴스는 다음 메이저 또는 마이너 버전 업그레이드 전까지 major.minor.patch(예: 5.7.42.R2)를 사용할 수 있습니다.

RDS for MySQL의 RDS 버전 번호

RDS 버전 번호는 *major*.*minor*.*patch* 또는 *major*.*minor*.*patch*.*YYYYMMDD* 명명 체계를 사용합니다. Amazon RDS 확장 지원 버전은 *minor*-*RDS*.*YYYYMMDD* 마이너 버전 이름 지정 체계를 사용합니다.

RDS 패치 버전에는 릴리스 후 마이너 버전에 추가된 중요한 버그 수정이 포함되어 있습니다. RDS 날짜 버전(*YYYYMMDD*)은 보안 패치입니다. 보안 패치에는 엔진 동작을 변경할 수 있는 수정 사항은 포함되지 않습니다. RDS 추가 지원 버전 번호 지정에 대한 자세한 내용은 Amazon RDS 추가 지원 버전 명명 규칙 섹션을 참조하세요.

다음 SQL 쿼리를 사용하여 RDS for MySQL 데이터베이스의 RDS 버전 번호를 확인할 수 있습니다.

```
mysql> select mysql.rds_version();
```

예를 들어 RDS for MySQL 8.0.34 데이터베이스를 쿼리하면 다음 아웃풋이 반환됩니다.

```
+---------------------+
| mysql.rds_version() |
+---------------------+
| 8.0.34.R2.20231201  |
+---------------------+
1 row in set (0.01 sec)
```

RDS for MySQL 메이저 버전 업그레이드

Amazon RDS는 MySQL 데이터베이스 엔진의 다음과 같은 메이저 버전 업그레이드를 지원합니다.

- MySQL 5.7에서 MySQL 8.0으로
- MySQL 8.0에서 MySQL 8.4로

> **Note**
>
> 최신 세대 및 현재 세대의 DB 인스턴스 클래스로만 MySQL 버전 5.7, 8.0, 8.4 DB 인스턴스를 만들 수 있습니다.
> 경우에 따라, 이전 세대의 DB 인스턴스 클래스에서 실행되는 DB 인스턴스를 더 높은 버전의 MySQL 엔진 버전을 사용하는 DB 인스턴스로 업그레이드하고 싶을 수 있습니다. 이러한 경우 먼저 DB 인스턴스를 수정하여 최신 세대 또는 현재 세대 DB 인스턴스 클래스를 사용합니다. 이렇게 한 다음 DB 인스턴스를 수정하여 더 높은 MySQL 데이터베이스 엔진 버전을 사용하도록 할 수 있습니다. Amazon RDS DB 인스턴스 클래스에 대한 자세한 내용은 DB 인스턴스 클래스 단원을 참조하십시오.

주제

- MySQL 메이저 버전 업그레이드 개요
- 업그레이드 사전 점검
- 업그레이드 실패 후 롤백

MySQL 메이저 버전 업그레이드 개요

메이저 버전 업그레이드에는 기존 애플리케이션과 호환되지 않는 데이터베이스 변경 사항이 포함될 수 있습니다. 결과적으로 Amazon RDS에서는 메이저 버전 업그레이드가 자동으로 적용되지 않으므로 DB 인스턴스를 수동으로 변경해야 합니다. 모든 업그레이드는 프로덕션 환경의 인스턴스에 적용하기 전에 반드시 철저하게 테스트하는 것이 좋습니다.

메이저 버전 업그레이드를 수행하려면 먼저 사용 가능한 OS 업데이트를 수행합니다. OS 업데이트가 완료된 후 각 메이저 버전으로 업그레이드합니다. 예를 들어, 5.7을 8.0로 업그레이드한 후 8.0을 8.4로 업그레이드합니다. RDS for MySQL 다중 AZ DB 클러스터 업그레이드에 대한 자세한 내용은 Amazon RDS에서 다중 AZ DB 클러스터의 엔진 버전 업그레이드 섹션을 참조하세요. 2014년 4월 24일 이전에 생성한 MySQL DB 인스턴스에는 업데이트가 적용될 때까지 사용 가능한 OS 업데이트가 표시됩니다. OS 업데이트에 대한 자세한 내용은 DB 인스턴스에 업데이트 적용 단원을 참조하십시오.

MySQL의 메이저 버전 업그레이드 중에는 필요하다면 Amazon RDS가 MySQL 바이너리 `mysql_upgrade`를 실행하여 테이블을 업그레이드합니다. 또한 Amazon RDS는 메이저 버전 업그레이드 도중 `slow_log` 및 `general_log` 테이블을 비웁니다. 로그 정보를 보존하려면 메이저 버전 업그레이드에 앞서 로그 내용을 저장하십시오.

MySQL 메이저 버전 업그레이드는 일반적으로 약 10분 정도 걸립니다. DB 인스턴스 클래스 크기 때문에 또는 인스턴스가 Amazon RDS의 모범 사례의 특정 작업 지침에 따르지 않는 탓에 일부 업그레이드는 시간이 더 걸릴 수도 있습니다. Amazon RDS 콘솔에서 DB 인스턴스를 업그레이드하는 경우, DB 인스턴스 상태를 보고 업그레이드 완료 시간을 알 수 있습니다. AWS Command Line Interface(AWS CLI)를 사용하여 업그레이드하는 경우, describe-db-instances 명령을 사용하여 `Status` 값을 확인합니다.

업그레이드 사전 점검

Amazon RDS는 비호환성을 확인하기 위해 업그레이드하기 전에 사전 점검을 실행합니다. 이러한 비호환성은 업그레이드한 후의 MySQL 버전에 따라 다릅니다.

사전 점검에는 MySQL에 포함된 내용과 Amazon RDS 팀에서 생성한 내용이 포함됩니다. MySQL에서 제공하는 사전 점검에 대한 자세한 내용은 업그레이드 확인 프로그램 유틸리티를 참조하십시오.

사전 점검은 업그레이드를 위해 DB 인스턴스가 중지되기 전에 실행됩니다. 즉, 점검을 실행해도 가동 중지를 일으키지 않습니다. 사전 점검에서 비호환성이 발견되면 Amazon RDS는 DB 인스턴스가 중지되기 전에 자동으로 업그레이드를 취소합니다. 또한 Amazon RDS는 비호환성에 대한 이벤트를 생성합니다. Amazon RDS 이벤트에 대한 자세한 내용은 Amazon RDS 이벤트 알림 작업 단원을 참조하십시오.

Amazon RDS는 각 비호환성에 대한 자세한 정보를 로그 파일 `PrePatchCompatibility.log`에 기록합니다. 대부분의 경우 로그 항목에는 비호환성 문제를 해결하기 위한 MySQL 설명서 링크가 포함되어 있습니다. 로그 파일 보기에 대한 자세한 내용은 데이터베이스 로그 파일 보기 및 나열 단원을 참조하십시오.

사전 점검의 특성으로 인해 데이터베이스의 객체를 분석합니다. 이 분석은 리소스를 소비하고 업그레이드가 완료되는 시간을 늘립니다.

주제

- MySQL 8.0에서 8.4로 업그레이드하기 위한 사전 점검
- MySQL 5.7에서 8.0으로 업그레이드하기 위한 사전 점검

MySQL 8.0에서 8.4로 업그레이드하기 위한 사전 점검

MySQL 8.4에는 MySQL 8.0과 상당한 비호환성이 포함되어 있습니다. 이러한 비호환성으로 인해 MySQL 8.0에서 MySQL 8.4로 업그레이드하는 동안 문제가 발생할 수 있습니다. 따라서 업그레이드가 성공하려면 데이터베이스에 몇 가지 준비가 필요할 수 있습니다. 다음은 이러한 비호환성을 나열한 전체 목록입니다.

- 사용되지 않는 데이터 형식이나 함수를 사용하는 테이블이 없어야 합니다.
- 트리거에는 누락되었거나 빈 definer 또는 잘못된 생성 컨텍스트가 없어야 합니다.
- 키워드 또는 예약된 단어 위반이 없어야 합니다. 이전에 예약되지 않은 일부 키워드는 MySQL 8.4에서 예약할 수 있습니다.

  자세한 내용은 MySQL 설명서의 키워드 및 예약어를 참조하십시오.
- MySQL 8.0 `mysql` 시스템 데이터베이스에는 MySQL 8.4 데이터 딕셔너리가 사용하는 테이블과 동일한 이름의 테이블이 없어야 합니다.
- `sql_mode` 시스템 변수 설정에 정의된 사용되지 않는 SQL 모드가 없어야 합니다.
- 255자 또는 1020바이트 길이를 초과하는 개별 `ENUM` 또는 `SET` 열 요소가 있는 테이블이나 저장 프로시저가 없어야 합니다.
- MySQL 8.0 설치는 MySQL 8.4에 지원되지 않는 기능을 사용해서는 안 됩니다.

  자세한 내용은 MySQL 설명서의 Features removed in MySQL 8.4를 참조하세요.
- 64자를 초과하는 외래 키 제약 조건 이름이 없어야 합니다.
- 향상된 유니코드 지원을 위해 다음 정보를 검토하세요.
  - `utf8mb3` charset을 사용하는 객체가 `utf8mb4` charset을 사용하도록 변환하는 것을 고려하세요. `utf8mb3` 문자 집합은 사용되지 않습니다.
  - 현재 `utf8`은 `utf8mb3` charset의 별칭이므로 `utf8` 대신 문자 집합 참조를 위한 `utf8mb4` 사용을 고려하세요. 가능하면 먼저 `utf8`을 `utf8mb4`로 변경한 다음 데이터베이스를 업그레이드합니다.
  - 이전 클라이언트는 `utf8mb3`에 대해 알 수 없는 문자 집합 오류를 수신할 수 있으므로 데이터베이스를 업그레이드하기 전에 데이터베이스 클라이언트를 업그레이드하세요.

  자세한 내용은 MySQL 설명서의 utf8mb3 문자 집합(3바이트 UTF-8 유니코드 인코딩)을 참조하십시오.

  문자 집합을 변경하려는 경우 데이터베이스의 백업, 복원 및 복제를 수동으로 수행할 수 있습니다. 또는 Amazon RDS 블루/그린 배포를 사용할 수 있습니다. 자세한 내용은 데이터베이스 업데이트에 Amazon RDS 블루/그린 배포 사용 단원을 참조하십시오.

MySQL 8.0에서 8.4로 업그레이드를 시작하면 Amazon RDS가 자동으로 사전 점검을 실행하여 이러한 비호환성을 찾아냅니다. MySQL 8.4로 업그레이드하는 방법은 MySQL 설명서에서 Upgrading MySQL을 참조하세요.

이러한 사전 점검은 필수입니다. 건너뛸 수 없습니다. 사전 점검은 다음과 같은 이점을 제공합니다.

- 이를 통해 업그레이드 중 예기치 않은 가동 중단을 피할 수 있습니다.
- 비호환성이 있는 경우 Amazon RDS가 업그레이드를 차단하고 이에 대해 알 수 있는 로그를 제공합니다. 그러면 로그를 사용해 비호환성을 제거함으로써 MySQL 8.4로 업그레이드하기 위한 데이터베이스 준비를 마칠 수 있습니다. 비호환성 문제를 제거하는 방법에 대한 자세한 내용은 MySQL 설명서의 Preparing your installation for upgrade를 참조하세요.

MySQL 5.7에서 8.0으로 업그레이드하기 위한 사전 점검

MySQL 8.0에는 MySQL 5.7과 상당한 비호환성이 포함되어 있습니다. 이러한 비호환성으로 인해 MySQL 5.7에서 MySQL 8.0으로 업그레이드하는 동안 문제가 발생할 수 있습니다. 따라서 업그레이드가 성공하려면 데이터베이스에 몇 가지 준비가 필요할 수 있습니다. 다음은 이러한 비호환성을 나열한 전체 목록입니다.

- 사용되지 않는 데이터 형식이나 함수를 사용하는 테이블이 없어야 합니다.
- orphan *.frm 파일이 없어야 합니다.
- 트리거에는 누락되었거나 빈 definer 또는 잘못된 생성 컨텍스트가 없어야 합니다.
- 기본 파티셔닝 지원이 없는 스토리지 엔진을 사용하는 분할된 테이블이 없어야 합니다.
- 키워드 또는 예약된 단어 위반이 없어야 합니다. 이전에 예약되지 않은 일부 키워드는 MySQL 8.0에서 예약할 수 있습니다.

  자세한 내용은 MySQL 설명서의 키워드 및 예약어를 참조하십시오.
- MySQL 5.7 `mysql` 시스템 데이터베이스에는 MySQL 8.0 데이터 딕셔너리가 사용하는 테이블과 동일한 이름의 테이블이 없어야 합니다.
- `sql_mode` 시스템 변수 설정에 정의된 사용되지 않는 SQL 모드가 없어야 합니다.
- 255자 또는 1020바이트 길이를 초과하는 개별 `ENUM` 또는 `SET` 열 요소가 있는 테이블이나 저장 프로시저가 없어야 합니다.
- MySQL 8.0.13 이상으로 업그레이드하기 전에 공유 InnoDB 테이블스페이스에 있는 테이블 파티션이 없어야 합니다.
- `ASC` 절에 사용하는 `DESC` 또는 `GROUP BY` 한정자를 사용하는 MySQL 8.0.12 이하의 쿼리 및 저장 프로그램 정의가 없어야 합니다.
- MySQL 5.7 설치는 MySQL 8.0에 지원되지 않는 기능을 사용해서는 안 됩니다.

  자세한 내용은 MySQL 설명서의 MySQL 8.0에서 제거된 기능을 참조하십시오.
- 64자를 초과하는 외래 키 제약 조건 이름이 없어야 합니다.

- 향상된 유니코드 지원을 위해 다음 정보를 검토하세요.
  - utf8mb3 charset을 사용하는 객체가 utf8mb4 charset을 사용하도록 변환하는 것을 고려하세요. utf8mb3 문자 집합은 사용되지 않습니다.
  - 현재 utf8은 utf8mb3 charset의 별칭이므로 utf8 대신 문자 집합 참조를 위한 utf8mb4 사용을 고려하세요. 가능하면 먼저 utf8을 utf8mb4로 변경한 다음 데이터베이스를 업그레이드합니다.
  - 이전 클라이언트는 utf8mb3에 대해 알 수 없는 문자 집합 오류를 수신할 수 있으므로 데이터베이스를 업그레이드하기 전에 데이터베이스 클라이언트를 업그레이드하세요.

  자세한 내용은 MySQL 설명서의 utf8mb3 문자 집합(3바이트 UTF-8 유니코드 인코딩)을 참조하십시오.

  문자 집합을 변경하려는 경우 데이터베이스의 백업, 복원 및 복제를 수동으로 수행할 수 있습니다. 또는 Amazon RDS 블루/그린 배포를 사용할 수 있습니다. 자세한 내용은 데이터베이스 업데이트에 Amazon RDS 블루/그린 배포 사용 단원을 참조하십시오.

MySQL 5.7에서 8.0으로 업그레이드를 시작하면 Amazon RDS가 자동으로 사전 점검을 실행하여 이러한 비호환성을 찾아냅니다. MySQL 8.0으로 업그레이드하는 방법은 MySQL 설명서에서 MySQL 업그레이드를 참조하십시오.

이러한 사전 점검은 필수입니다. 건너뛸 수 없습니다. 사전 점검은 다음과 같은 이점을 제공합니다.

- 이를 통해 업그레이드 중 예기치 않은 가동 중단을 피할 수 있습니다.
- 비호환성이 있는 경우 Amazon RDS가 업그레이드를 차단하고 이에 대해 알 수 있는 로그를 제공합니다. 그러면 로그를 사용해 비호환성을 제거함으로써 MySQL 8.0으로 업그레이드하기 위한 데이터베이스 준비를 마칠 수 있습니다. 비호환성 문제를 제거하는 방법에 대한 자세한 내용은 MySQL 설명서의 업그레이드를 위한 설치 준비 및 MySQL Server 블로그의 MySQL 8.0으로 업그레이드할 때 알아야 할 내용을 참조하세요.

업그레이드 실패 후 롤백

DB 인스턴스를 MySQL 버전 5.7에서 MySQL 버전 8.0으로 또는 MySQL 8.0에서 8.4로 업그레이드하면 업그레이드가 실패할 수 있습니다. 특히 데이터 딕셔너리에 사전 검사에서 캡처되지 않은 비호환성 문제가 포함되어 있는 경우 실패할 수 있습니다. 이 경우 데이터베이스가 새 MySQL 8.0 또는 8.4 버전에서 성공적으로 시작되지 않습니다. 이때 Amazon RDS는 업그레이드를 위해 수행된 변경 사항을 롤백합니다. 롤백 후 MySQL DB 인스턴스는 원래 버전을 실행합니다.

- MySQL 버전 8.0(MySQL 8.4에서 롤백하는 경우)

- MySQL 버전 5.7(MySQL 8.0에서 롤백하는 경우)

업그레이드가 실패하고 롤백되면 Amazon RDS는 이벤트 ID가 ID RDS-EVENT-0188인 이벤트를 생성합니다.

일반적으로, DB 인스턴스의 데이터베이스와 대상 MySQL 버전 간에 메타데이터가 호환되지 않기 때문에 업그레이드가 실패합니다. 업그레이드가 실패한 경우 이러한 비호환성에 대한 세부 정보를 `upgradeFailure.log` 파일에서 확인할 수 있습니다. 업그레이드를 다시 시도하기 전에 비호환성을 해결하세요.

업그레이드 시도 및 롤백이 실패하는 과정에서 DB 인스턴스가 다시 시작됩니다. 보류 중인 파라미터 변경 사항은 재시작 중에 적용되고 롤백 후에도 유지됩니다.

MySQL 8.0으로 업그레이드하는 방법에 대한 자세한 내용은 MySQL 설명서의 다음 항목을 참조하세요.

- 업그레이드를 위한 설치 준비
- MySQL 8.0으로 업그레이드 하시겠습니까? 알아야 할 내용...

MySQL 8.4로 업그레이드하는 방법에 대한 자세한 내용은 MySQL 설명서에서 Preparing Your Installation for Upgrade를 참조하세요.

RDS for MySQL 업그레이드 테스트

DB 인스턴스에 대한 메이저 버전 업그레이드를 수행하기 전에 데이터베이스가 새 버전과 호환되는지 여부를 철저하게 테스트합니다. 또한 새 버전과의 호환성을 위해 데이터베이스에 액세스하는 모든 애플리케이션을 철저히 테스트합니다. 다음 절차를 참조하는 것이 좋습니다.

메이저 버전 업그레이드를 테스트하려면

1. 다음과 같이 새 버전의 데이터베이스 엔진에 대한 업그레이드 문서를 검토하여 데이터베이스나 애플리케이션에 영향을 끼칠 수도 있는 호환성 문제가 있는지 살펴봅니다.
   - MySQL 5.7의 변경 사항
   - MySQL 8.0의 변경 사항
   - MySQL 8.4의 변경 사항
2. DB 인스턴스가 사용자 정의 DB 파라미터 그룹의 구성원인 경우에는 기존 설정을 이용해 새로운 메이저 버전과 호환되는 새로운 DB 파라미터 그룹을 생성합니다. 테스트 인스턴스를 업그레이드

할 때는 새로운 DB 파라미터 그룹을 지정해야만 업그레이드 테스트가 올바로 진행될 수 있습니다. DB 파라미터 그룹을 생성하는 것에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.

3. 업그레이드할 DB 인스턴스의 DB 스냅샷을 생성합니다. 자세한 내용은 Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성 섹션을 참조하세요.
4. DB 스냅샷을 복구하여 새로운 테스트 DB 인스턴스를 생성합니다. 자세한 내용은 DB 인스턴스 복원 섹션을 참조하세요.
5. 이후 이어지는 세부적인 방법 중 한 가지를 사용하여 이 새로운 테스트 DB 인스턴스를 변경하고 새로운 버전으로 업그레이드합니다. 2단계에서 새로운 파라미터 그룹을 생성하였다면 이 파라미터 그룹을 지정합니다.
6. 업그레이드한 인스턴스에서 사용할 스토리지를 평가하여 업그레이드 시 추가 스토리지의 필요 여부를 결정합니다.
7. 업그레이드한 DB 인스턴스와 관련하여 데이터베이스 및 애플리케이션과 새로운 버전의 호환성을 보장하는 데 필요하다면 최대한 많은 수의 품질 보증 테스트를 실행합니다. 또한 1단계에서 발견된 호환성 문제의 영향을 평가하는 데 필요한 새로운 테스트도 모두 실행합니다. 저장된 프로시저와 함수를 모두 테스트합니다. 업그레이드한 DB 인스턴스에 대해 애플리케이션의 테스트 버전을 실행합니다.
8. 모든 테스트가 통과되면 프로덕션 환경의 DB 인스턴스에도 업그레이드를 실행합니다. 단, 모든 기능이 정상 작동하는 것을 확인할 때까지 쓰기 작업은 DB 인스턴스에 실행하지 않는 것이 좋습니다.

MySQL DB 인스턴스 업그레이드

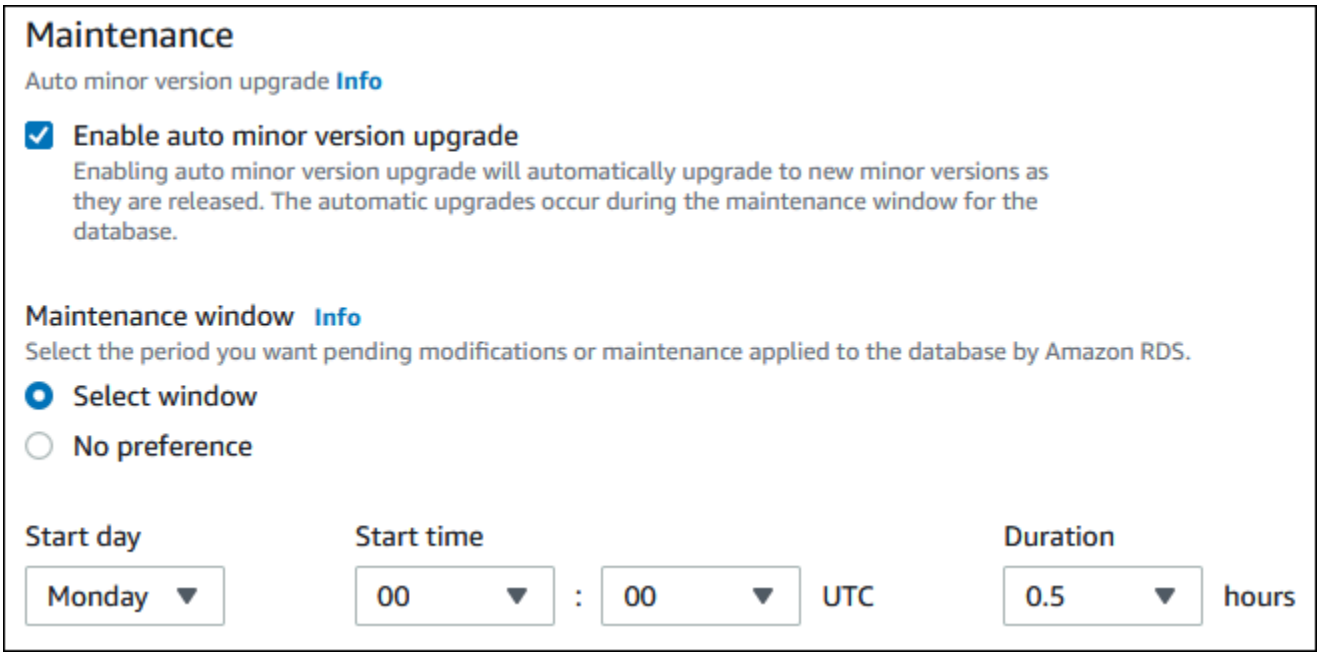
MySQL DB 인스턴스의 수동 또는 자동 업그레이드에 대한 자세한 내용은 DB 인스턴스 엔진 버전 업그레이드 단원을 참조하십시오.

RDS for MySQL 마이너 버전 자동 업그레이드

DB 인스턴스를 생성하거나 수정할 때 다음 설정을 지정하면 DB 인스턴스가 자동으로 업그레이드되도록 할 수 있습니다.

- 마이너 버전 자동 업그레이드(Auto minor version upgrade) 설정을 활성화되어 있습니다.
- 백업 보존 기간(Backup retention period) 설정이 0보다 큽니다.

AWS Management Console에서 이 설정은 추가 구성(Additional configuration)에 있습니다. 다음 이미지는 자동 마이너 버전 업그레이드(Auto minor version upgrade) 설정을 보여줍니다.



이러한 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 섹션을 참조하세요.

일부 AWS 리전의 특정 RDS for MySQL 메이저 버전의 경우 RDS에서 하나의 마이너 버전을 자동 업그레이드 버전으로 지정합니다. Amazon RDS가 마이너 버전을 테스트하고 승인하면 유지 관리 기간 중에 자동으로 마이너 버전 업그레이드가 실행됩니다. RDS는 자동으로 새로 릴리스된 마이너 버전을 자동 업그레이드 버전으로 설정하지 않습니다. RDS가 더 새로운 자동 업그레이드 버전을 지정하기 전에 다음과 같은 여러 기준이 고려됩니다.

- 알려진 보안 문제
- MySQL 커뮤니티 버전의 버그
- 마이너 버전 릴리스 이후 전반적인 플릿 안정성

다음 AWS CLI 명령을 실행하여 특정 AWS 리전의 지정된 MySQL 마이너 버전에 대한 현재의 자동 마이너 업그레이드 대상 버전을 확인할 수 있습니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-engine-versions \
--engine mysql \
--engine-version minor_version \
--region region \
```

```
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" \
--output text
```

Windows의 경우:

```
aws rds describe-db-engine-versions ^
--engine mysql ^
--engine-version minor_version ^
--region region ^
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" ^
--output text
```

예를 들어, 다음 AWS CLI 명령은 미국 동부(오하이오) AWS 리전(us-east-2)의 MySQL 마이너 버전 8.0.11에 대한 자동 마이너 업그레이드 대상을 안내합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-engine-versions \
--engine mysql \
--engine-version 8.0.11 \
--region us-east-2 \
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" \
--output table
```

Windows의 경우:

```
aws rds describe-db-engine-versions ^
--engine mysql ^
--engine-version 8.0.11 ^
--region us-east-2 ^
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" ^
--output table
```

다음과 같은 출력이 표시됩니다.

```
---------------------------------
```

```
|    DescribeDBEngineVersions    |
+--------------+-----------------+
|  AutoUpgrade |  EngineVersion  |
+--------------+-----------------+
|  False       |  8.0.15         |
|  False       |  8.0.16         |
|  False       |  8.0.17         |
|  False       |  8.0.19         |
|  False       |  8.0.20         |
|  False       |  8.0.21         |
|  True        |  8.0.23         |
|  False       |  8.0.25         |
+--------------+-----------------+
```

이 예제에서 AutoUpgrade 값은 MySQL 버전 8.0.23의 경우 True입니다. 따라서 자동 마이너 업그레이드 대상은 출력에서 강조 표시된 MySQL 버전 8.0.23입니다.

MySQL DB 인스턴스는 다음 기준이 충족되면 유지 관리 기간 중에 자동으로 업그레이드됩니다.

- 마이너 버전 자동 업그레이드(Auto minor version upgrade) 설정을 활성화되어 있습니다.
- 백업 보존 기간(Backup retention period) 설정이 0보다 큽니다.
- DB 인스턴스가 현재 자동 업그레이드 마이너 버전보다 낮은 DB 엔진 버전을 실행 중입니다.

자세한 내용은 마이너 엔진 버전 자동 업그레이드 섹션을 참조하세요.

RDS for MySQL 데이터베이스 업그레이드 시 읽기 전용 복제본을 사용하여 가동 중지 시간 단축

대부분의 경우 블루/그린 배포는 MySQL DB 인스턴스를 업그레이드할 때 다운타임을 줄이는 가장 좋은 방법입니다. 자세한 내용은 데이터베이스 업데이트에 Amazon RDS 블루/그린 배포 사용 단원을 참조하십시오.

블루/그린 배포를 사용할 수 없으며 현재 프로덕션 애플리케이션에 MySQL 데이터베이스 인스턴스를 사용 중인 경우, 다음 절차를 사용하여 DB 인스턴스의 데이터베이스 버전을 업그레이드할 수 있습니다. 이 절차는 애플리케이션의 가동 중지 시간을 줄일 수 있습니다.

읽기 전용 복제본을 사용하면 대부분의 유지 관리 단계를 미리 수행하고 실제 운영 중단 중에 필요한 변경 사항을 최소화할 수 있습니다. 이 기법을 사용하면 기존 DB 인스턴스를 변경하지 않으면서 새 DB 인스턴스를 테스트하고 준비할 수 있습니다.

이 절차는 MySQL 버전 5.5를 MySQL 버전 7.6으로 업그레이드하는 예를 보여줍니다. 동일한 일반 절차를 사용하여 다른 메이저 버전으로 업그레이드할 수 있습니다. 동일한 일반 절차를 사용하여 다른 메이저 버전으로 업그레이드할 수 있습니다.

> **Note**
>
> MySQL 버전 5.7에서 MySQL 버전 8.0으로 업그레이드하거나 MySQL 버전 8.0에서 MySQL 버전 8.4로 업그레이드하는 경우 업그레이드를 수행하기 전에 사전 점검을 완료합니다. 자세한 내용은 MySQL 5.7에서 8.0으로 업그레이드하기 위한 사전 점검 및 MySQL 8.0에서 8.4로 업그레이드하기 위한 사전 점검 단원을 참조하세요.

DB 인스턴스를 사용하면서 MySQL 데이터베이스를 업그레이드하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. MySQL 5.7 DB 인스턴스의 읽기 전용 복제본을 생성합니다. 이 프로세스에서 업그레이드 가능한 데이터베이스 사본이 만들어집니다. DB 인스턴스의 다른 읽기 전용 복제본도 존재할 수 있습니다.
   a. 콘솔에서 데이터베이스(Databases)와 업그레이드하려는 DB 인스턴스를 차례로 선택합니다.
   b. 작업에서 읽기 전용 복제본 만들기를 선택합니다.
   c. 읽기 전용 복제본의 DB 인스턴스 식별자(DB instance identifier) 값을 제공하고 DB 인스턴스 클래스(DB instance class) 및 기타 설정이 MySQL 5.7 DB 인스턴스와 일치하는지 확인합니다.
   d. [Create read replica]를 선택합니다.
3. (선택 사항) 읽기 전용 복제본이 생성되고 상태(Status)가 사용 가능(Available)으로 표시되면 읽기 전용 복제본을 다중 AZ 배포로 변환하고 백업을 활성화합니다.

   기본적으로 읽기 전용 복제본은 백업이 비활성화된 상태로 생성됩니다. 읽기 전용 복제본은 궁극적으로 프로덕션 DB 인스턴스가 되기 때문에 다중 AZ 배포를 구성하고 백업을 활성화하는 것이 가장 좋습니다.

   a. 콘솔에서 데이터베이스(Databases)와 방금 생성한 읽기 전용 복제본을 차례로 선택합니다.
   b. 수정을 선택합니다.
   c. 다중 AZ 배포(Multi-AZ deployment)에서 대기 인스턴스 생성(Create a standby instance)을 선택합니다.

d. Backup Retention Period(백업 보존 기간)로 0이 아닌 양수 값(예: 3일)을 선택한 후 Continue(계속)를 선택합니다.

e. 수정 사항 예약에 대해 즉시 적용을 선택합니다.

f. DB 인스턴스 수정을 선택합니다.

4. 읽기 전용 복제본 상태(Status)가 사용 가능(Available)으로 표시되면 읽기 전용 복제본을 MySQL 8.0으로 업그레이드합니다.

a. 콘솔에서 데이터베이스(Databases)와 방금 생성한 읽기 전용 복제본을 차례로 선택합니다.

b. 수정을 선택합니다.

c. DB 엔진 버전(DB engine version)에서 업그레이드할 MySQL 8.0 버전을 선택한 후 계속(Continue)을 선택합니다.

d. 수정 사항 예약에 대해 즉시 적용을 선택합니다.

e. [Modify DB instance]를 선택하여 업그레이드를 시작합니다.

5. 업그레이드가 완료되고 상태(Status)가 사용 가능(Available)으로 표시되면 업그레이드한 읽기 전용 복제본이 소스 MySQL 5.7 DB 인스턴스로 업데이트되는지 확인합니다. 확인하려면 읽기 전용 복제본에 연결하고 `SHOW REPLICA STATUS` 명령을 실행합니다. `Seconds_Behind_Master` 필드가 `0`이면 복제본이 최신 상태입니다.

> **Note**
>
> 이전 버전의 MySQL에는 `SHOW SLAVE STATUS` 대신 `SHOW REPLICA STATUS`가 사용되었습니다. 8.0.23 이전 MySQL 버전을 사용하는 경우 `SHOW SLAVE STATUS`를 사용합니다.

6. (선택 사항) 읽기 전용 복제본의 읽기 전용 복제본을 생성합니다.

DB 인스턴스가 독립형 DB 인스턴스로 승격된 후 읽기 전용 복제본을 갖도록 하려면 지금 읽기 전용 복제본을 생성하면 됩니다.

a. 콘솔에서 데이터베이스(Databases)와 방금 업그레이드한 읽기 전용 복제본을 차례로 선택합니다.

b. 작업에서 읽기 전용 복제본 만들기를 선택합니다.

c. 읽기 전용 복제본의 DB 인스턴스 식별자(DB instance identifier) 값을 제공하고 DB 인스턴스 클래스(DB instance class) 및 기타 설정이 MySQL 5.7 DB 인스턴스와 일치하는지 확인합니다.

d. [Create read replica]를 선택합니다.

7. (선택 사항) 읽기 전용 복제본에 대한 사용자 지정 DB 파라미터 그룹을 구성합니다.

   DB 인스턴스가 독립형 DB 인스턴스로 승격된 후 사용자 지정 파라미터 그룹을 사용하도록 하려면 지금 DB 파라미터 그룹을 생성하여 읽기 전용 복제본과 연결하면 됩니다.

   a. MySQL 8.0에 대한 사용자 지정 DB 파라미터 그룹을 생성합니다. 지침은 Amazon RDS에서 DB 파라미터 그룹 생성 섹션을 참조하세요.

   b. 방금 생성한 DB 파라미터 그룹에서 변경하려는 파라미터를 수정합니다. 지침은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요.

   c. 콘솔에서 데이터베이스(Databases)와 읽기 전용 복제본을 차례로 선택합니다.

   d. 수정을 선택합니다.

   e. DB 파라미터 그룹(DB parameter group)에서 방금 생성한 MySQL 8.0 DB 파라미터 그룹을 선택한 후 계속(Continue)을 선택합니다.

   f. 수정 사항 예약에 대해 즉시 적용을 선택합니다.

   g. [Modify DB instance]를 선택하여 업그레이드를 시작합니다.

8. MySQL 8.0 읽기 전용 복제본을 독립형 DB 인스턴스로 만듭니다.

   > **Important**
   >
   > MySQL 8.0 읽기 전용 복제본을 독립형 DB 인스턴스로 승격하면 더 이상 MySQL 5.7 DB 인스턴스의 복제본이 아닙니다. 원본 MySQL 5.7 DB 인스턴스가 읽기 전용 모드이고, 모든 쓰기 작업이 일시 중단되는 유지 관리 기간 동안 MySQL 8.0 읽기 전용 복제본을 승격하는 것이 좋습니다. 승격이 완료되면 쓰기 작업을 MySQL 8.0 DB 인스턴스에서 실행하여 쓰기 작업이 손실되는 것을 막을 수 있습니다.
   > 그 밖에도 MySQL 8.0 읽기 전용 복제본을 승격하기 전에 MySQL 8.0 읽기 전용 복제본에서 필요한 모든 DDL(데이터 정의 언어) 작업을 수행하는 것이 좋습니다. 인덱스 생성을 예로 들 수 있습니다. 그러면 승격 후에도 MySQL 8.0 읽기 전용 복제본의 성능에 미치는 부정적인 영향을 방지할 수 있습니다. 읽기 전용 복제본을 승격하려면 다음 절차를 사용하십시오.

   a. 콘솔에서 데이터베이스(Databases)와 방금 업그레이드한 읽기 전용 복제본을 차례로 선택합니다.

   b. 작업에서 Promote(승격)를 선택합니다.

   c. 읽기 전용 복제본 인스턴스에 대해 자동 백업을 활성화하려면 예를 선택합니다. 자세한 내용은 백업 소개 섹션을 참조하세요.

d. [Continue]를 선택합니다.

e. [Promote Read Replica]를 선택합니다.

9. 이제 MySQL 데이터베이스 버전이 업그레이드되었습니다. 이제 애플리케이션을 새로운 MySQL 8.0 DB 인스턴스로 리디렉션할 수 있습니다.

MySQL DB 스냅샷 엔진 버전 업그레이드

Amazon RDS를 사용하여 MySQL DB 인스턴스의 스토리지 볼륨 DB 스냅샷을 생성할 수 있습니다. DB 스냅샷 생성 시, 생성되는 스냅샷은 DB 인스턴스에서 사용하는 엔진 버전에 기반합니다. DB 스냅샷의 엔진 버전을 업그레이드할 수 있습니다.

RDS for MySQL 버전의 경우 버전 5.7 스냅샷을 버전 8.0으로 업그레이드하거나 버전 8.0 스냅샷을 버전 8.4로 업그레이드할 수 있습니다. 암호화되거나 암호화되지 않은 DB 스냅샷을 업그레이드할 수 있습니다.

RDS for MySQL DB 스냅샷에 사용 가능한 엔진 버전을 보려면 다음 AWS CLI 예제를 사용합니다.

```
aws rds describe-db-engine-versions --engine mysql --include-all --engine-
version example-engine-version --query "DBEngineVersions[*].ValidUpgradeTarget[*].
{EngineVersion:EngineVersion}" --output text
```

스냅샷에 대한 결과가 표시되지 않으면 엔진 버전이 더 이상 사용되지 않을 수 있습니다. 엔진 버전이 더 이상 지원되지 않는 경우 최신 메이저 버전 업그레이드 대상 또는 해당 버전에 대해 사용 가능한 다른 업그레이드 대상 중 하나로 업그레이드하는 것이 좋습니다. 자세한 내용은 RDS for MySQL에서 지원되지 않는 엔진 버전이 있는 DB 스냅샷에 대한 업그레이드 옵션 섹션을 참조하세요.

새 엔진 버전으로 업그레이드된 DB 스냅샷을 복원한 후에는 업그레이드가 성공적이었는지 테스트해야 합니다. 메이저 버전 업그레이드에 대한 자세한 내용은 the section called “MySQL DB 엔진 업그레이드” 단원을 참조하십시오. DB 스냅샷을 복원하는 방법은 the section called “DB 인스턴스 복원” 단원을 참조하십시오.

> **Note**
>
> 자동 백업 과정에서 생성된 자동 DB 스냅샷은 업그레이드할 수 없습니다.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB 스냅샷을 업그레이드할 수 있습니다.

Console

AWS Management Console을 사용하여 DB 스냅샷 엔진 버전을 업그레이드하려면 다음 절차를 사용합니다.

DB 스냅샷을 업그레이드하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.
3. 업그레이드할 스냅샷을 선택합니다.
4. 작업에서 Upgrade snapshot(스냅샷 업그레이드)을 선택합니다. Upgrade snapshot(스냅샷 업그레이드) 페이지가 표시됩니다.
5. 업그레이드할 New engine version(새 엔진 버전)을 선택합니다.
6. 스냅샷을 업그레이드하려면 변경 내용 저장을 선택합니다.

   업그레이드 중에는 이 DB 스냅샷의 모든 스냅샷 작업이 비활성화됩니다. 또한 DB 스냅샷 상태가 사용 가능에서 업그레이드 중으로 바뀐 다음 완료되면 활성으로 바뀝니다 스냅샷 손상 문제로 인해 DB 스냅샷을 업그레이드할 수 없는 경우, 상태가 사용할 수 없음으로 바뀝니다. 이 상태로부터 스냅샷을 복구할 수는 없습니다.

   > **Note**
   >
   > DB 스냅샷 업그레이드에 실패하면 스냅샷이 원래 버전의 원래 상태로 롤백됩니다.

AWS CLI

DB 스냅샷을 새 데이터베이스 엔진 버전으로 업그레이드하려면 AWS CLI modify-db-snapshot 명령을 실행합니다.

옵션

- `--db-snapshot-identifier` – 업그레이드할 DB 스냅샷의 식별자입니다. 식별자는 고유의 Amazon 리소스 이름(ARN)이어야 합니다. 자세한 내용은 Amazon RDS의 Amazon 리소스 이름(ARN) 섹션을 참조하세요.
- `--engine-version` – DB 스냅샷을 업그레이드할 엔진 버전입니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-snapshot \
```

```
    --db-snapshot-identifier my_db_snapshot \
    --engine-version new_version
```

Windows의 경우:

```
aws rds modify-db-snapshot ^
    --db-snapshot-identifier my_db_snapshot ^
    --engine-version new_version
```

Amazon RDS API

DB 스냅샷을 새 데이터베이스 엔진 버전으로 업그레이드하려면 RDS API ModifyDBSnapshot 작업을 직접 호출하세요.

파라미터

- `DBSnapshotIdentifier` – 업그레이드할 DB 스냅샷의 식별자입니다. 식별자는 고유의 Amazon 리소스 이름(ARN)이어야 합니다. 자세한 내용은 Amazon RDS의 Amazon 리소스 이름(ARN) 섹션을 참조하세요.
- `EngineVersion` – DB 스냅샷을 업그레이드할 엔진 버전입니다.

RDS for MySQL에서 지원되지 않는 엔진 버전이 있는 DB 스냅샷에 대한 업그레이드 옵션

다음 표에는 RDS for MySQL DB 스냅샷에 대해 지원되지 않는 엔진 버전에서 업그레이드할 수 있는 엔진 버전이 나와 있습니다.

> **Note**
>
> 선택한 엔진 버전으로 업그레이드하려면 DB 스냅샷을 두 번 이상 업그레이드해야 할 수 있습니다.

DB 스냅샷 엔진 버전	업그레이드 가능한 엔진 버전
5.5.8	5.5.62, 5.6.51
5.5.12	5.5.62, 5.6.51

DB 스냅샷 엔진 버전	업그레이드 가능한 엔진 버전
5.5.20	5.5.62, 5.6.51
5.5.23	5.5.62, 5.6.51
5.5.25a	5.5.62, 5.6.51
5.5.27	5.5.62, 5.6.51
5.5.31	5.5.62, 5.6.51
5.5.33	5.5.62, 5.6.51
5.5.37	5.5.62, 5.6.51
5.5.38	5.5.62, 5.6.51
5.5.40	5.5.62, 5.6.51
5.5.40a	5.5.62, 5.6.51
5.5.40b	5.5.62, 5.6.51
5.5.41	5.5.62, 5.6.51
5.5.42	5.5.62, 5.6.51
5.5.59	5.5.62, 5.6.51
5.6.12	5.6.51, 5.7.44
5.6.13	5.6.51, 5.7.44
5.6.17	5.6.51, 5.7.44
5.6.19	5.6.51, 5.7.44
5.6.19a	5.6.51, 5.7.44
5.6.19b	5.6.51, 5.7.44

DB 스냅샷 엔진 버전	업그레이드 가능한 엔진 버전
5.6.21	5.6.51, 5.7.44
5.6.21b	5.6.51, 5.7.44
5.6.22	5.6.51, 5.7.44
5.6.23	5.6.51, 5.7.44
5.6.27	5.6.51, 5.7.44
5.6.27a	5.6.51, 5.7.44
5.7.10	5.7.44, 5.7.44-rds.20240408, 5.7.44-rds.20240529, 5.7.44-rds.20250103, 5.7.44-rds.20250213, 8.0.32, 8.0.33, 8.0.34, 8.0.35, 8.0.36, 8.0.37, 8.0.39, 8.0.40, 8.0.41
5.7.11	5.7.44, 5.7.44-rds.20240408, 5.7.44-rds.20240529, 5.7.44-rds.20250103, 5.7.44-rds.20250213, 8.0.32, 8.0.33, 8.0.34, 8.0.35, 8.0.36, 8.0.37, 8.0.39, 8.0.40, 8.0.41
5.7.12	5.7.44, 5.7.44-rds.20240408, 5.7.44-rds.20240529, 5.7.44-rds.20250103, 5.7.44-rds.20250213, 8.0.32, 8.0.33, 8.0.34, 8.0.35, 8.0.36, 8.0.37, 8.0.39, 8.0.40, 8.0.41

Amazon RDS for MySQL DB 인스턴스로 데이터 가져오기

MySQL DB 인스턴스용 RDS로 데이터를 가져오는 기법에는 몇 가지가 있습니다. 가장 좋은 방법은 다음과 같은 여러 요인에 따라 달라집니다.

- 데이터 원본
- 데이터 분량
- 일회성 혹은 지속적
- 가동 중지 시간 길이

데이터와 함께 애플리케이션을 마이그레이션하는 경우라면 감당할 수 있는 가동 중지 시간도 고려해야 합니다.

다음 표에는 RDS for MySQL DB 인스턴스로 데이터를 가져오는 기법이 나와 있습니다.

소스	데이터 분량	일회성 혹은 지속적	애플리케이션 가동 중지	기술	추가 정보
온프레미스 또는 Amazon EC2에 있는 기존 MySQL 데이터베이스	모두 선택	한 번만	약간	온프레미스 데이터베이스의 백업을 만들어서 Amazon S3에 저장한 다음 MySQL을 실행하여 새로운 Amazon RDS DB 인스턴스에 백업 파일을 복원하십시오.	MySQL DB 인스턴스로 백업 복원
온프레미스 또는 Amazon EC2에 있	모두 선택	지속적	최소화	기존 MySQL 데이터베이스가 복제 소스가 되도록 복제본을 구성합니다.	외부 소스 인스턴스를 사용하여 이진 로그 파

소스	데이터 분량	일회성 혹은 지속적	애플리케이션 가동 중지	기술	추가 정보
는 기존 MySQL 데이터베이스					일 위치 복제 구성 가동 중지 시간을 줄이면서 Amazon RDS for MySQL 데이터베이스로 데이터 가져오기

소스	데이터 분량	일회성 혹은 지속적	애플리케이션 가동 중지	기술	추가 정보
기존의 모든 데이터베이스	모두 선택	일회성 혹은 지속적	최소화	AWS Database Migration Service을 사용하면 가동 중지 시간을 최소화하면서 데이터베이스를 마이그레이션할 수 있으며 대부분의 DB 엔진에서는 지속적으로 복제를 계속할 수 있습니다.	AWS Database Migration Service란? 및 AWS Database Migration Service 사용 설명서의 AWS DMS에서 MySQL 호환 데이터베이스를 대상으로 사용
기존 MySQL DB 인스턴스	모두 선택	일회성 혹은 지속적	최소화	지속적인 복제를 위한 읽기 전용 복제본을 생성합니다. 새 DB 인스턴스를 한 번만 생성하도록 읽기 전용 복제본을 승격시킵니다.	DB 인스턴스 읽기 전용 복제본 작업

소스	데이터 분량	일회성 혹은 지속적	애플리케이션 가동 중지	기술	추가 정보
기존 MySQL 데이터베이스	스몰	한 번만	약간	명령줄 유틸리티를 사용하여 MySQL DB 인스턴스에 바로 데이터를 복제합니다.	외부 MySQL 데이터베이스에서 Amazon RDS for MySQL DB 인스턴스로 데이터 가져오기
기존 데이터베이스에 저장되지 않은 데이터	Medium	한 번만	약간	플랫 파일을 만들고 MySQL `LOAD DATA LOCAL INFILE` 문을 이용하여 가져옵니다.	원하는 소스에서 Amazon RDS for MySQL DB 인스턴스로 데이터 가져오기

> **Note**
>
> `mysql` 시스템 데이터베이스에는 DB 인스턴스에 로그인하고 데이터에 액세스하는 데 필요한 인증 및 권한 부여 정보가 포함되어 있습니다. DB 인스턴스에 있는 `mysql` 데이터베이스의 각종 테이블, 데이터 또는 기타 콘텐츠를 삭제하거나 변경하거나 이름을 바꾸거나 자르면 오류

가 발생하여 DB 인스턴스와 데이터에 액세스할 수 없게 될 수 있습니다. 이 문제가 발생할 경우 AWS CLI restore-db-instance-from-db-snapshot 명령을 사용하여 DB 인스턴스를 스냅샷에서 복원할 수 있습니다. AWS CLI restore-db-instance-to-point-in-time 명령을 사용하여 DB 인스턴스를 복원할 수 있습니다.

MySQL에 대한 데이터 가져오기 고려 사항

다음 내용에는 MySQL로 데이터를 로드하는 것과 관련된 기술 정보가 포함되어 있습니다. 이 콘텐츠는 MySQL 서버 아키텍처에 익숙한 사용자를 대상으로 합니다.

이진 로깅

이진 로깅을 활성화하면 로깅을 비활성화했을 때와 비교해 데이터 로드 성능이 저하되고 최대 4배의 추가 디스크 공간이 필요합니다. 데이터를 로드하는 데 사용되는 트랜잭션 크기는 시스템 성능 및 디스크 공간 요구 사항에 직접적인 영향을 미칩니다. 트랜잭션이 클수록 더 많은 리소스가 필요합니다.

트랜잭션 크기

트랜잭션 크기는 MySQL 데이터 로드의 다음 측면에 영향을 미칩니다.

- 리소스 소비
- 디스크 공간 사용률
- 프로세스 재개
- 복구 시간
- 입력 형식(플랫 파일 또는 SQL)

이 섹션에서는 트랜잭션 크기가 이진 로깅에 미치는 영향을 설명하고 큰 데이터를 로드하는 중에 이진 로깅을 비활성화하는 이유를 논증합니다. Amazon RDS 자동 백업 보존 기간을 설정하여 이진 로깅을 활성화 및 비활성화할 수 있습니다. 0이 아닌 값으로 설정하면 이진 로깅이 활성화되고 0으로 설정하면 비활성화됩니다. 자세한 내용은 백업 보관 기간 섹션을 참조하세요.

이 섹션에서는 대규모 트랜잭션이 InnoDB에 미치는 영향과 트랜잭션 크기를 작게 유지하는 것이 중요한 이유도 설명합니다.

작은 트랜잭션

작은 트랜잭션의 경우, 이진 로깅을 사용하면 데이터 로드에 필요한 디스크 쓰기 작업 수가 배가됩니다. 이 결과 다른 데이터베이스 세션의 성능이 심각하게 저하되고 데이터 로딩 시간이 증가할 수 있습니다. 발생하는 성능 저하는 부분적으로 다음 요인에 따라 달라집니다.

- 업로드 속도
- 로드 중에 발생하는 기타 데이터베이스 활동
- Amazon RDS DB 인스턴스의 용량

또한, 이진 로그는 로그가 백업 및 제거될 때까지 로드된 데이터의 양과 대략적으로 같은 양의 디스크 공간을 사용합니다. Amazon RDS는 이진 로그를 자주 백업하고 제거하는 방법으로 이 문제를 최소화합니다.

큰 트랜잭션

대규모 트랜잭션의 경우 다음과 같은 이유로 이진 로깅이 사용하는 IOPS 및 디스크가 3배로 늘어납니다.

- 이진 로그 캐시는 트랜잭션 데이터를 일시적으로 디스크에 저장합니다.
- 이 캐시는 디스크 공간을 소비하는 트랜잭션 크기에 따라 증가합니다.
- 트랜잭션(커밋 또는 롤백)이 완료되면 시스템은 캐시를 이진 로그에 복사합니다.

이 프로세스는 다음과 같은 데이터 복사본 세 개를 만듭니다.

- 원본 데이터
- 디스크의 캐시
- 최종 이진 로그 항목

각 쓰기 작업에는 추가 IO가 발생하여 성능에 더욱 영향을 미칩니다.

따라서 이진 로깅에는 로깅을 비활성화했을 때에 비해 디스크 공간이 3배 필요합니다. 예를 들어 10GiB의 데이터를 단일 트랜잭션으로 로드하면 다음과 같은 세 개의 복사본이 만들어집니다.

- 테이블 데이터 10GiB
- 이진 로그 캐시 10GiB
- 이진 로그 파일 10GiB

필요한 총 임시 디스크 공간은 30GiB입니다.

중요한 디스크 공간 고려 사항:

- 캐시 파일은 세션이 종료되거나 새 트랜잭션이 다른 캐시를 만들 때까지 유지됩니다.
- 이진 로그는 백업될 때까지 유지되며, 잠재적으로 20GiB(캐시 및 로그)를 장기간 차지할 수 있습니다.

`LOAD DATA LOCAL INFILE`을 사용하여 데이터를 로드하는 경우 데이터 복구는 로드 전에 수행된 백업에서 데이터베이스를 복구해야 하는 경우를 대비하여 네 번째 복사본을 만듭니다. 복원 중에 MySQL이 이진 로그에서 플랫 파일로 데이터를 추출합니다. 그런 다음 MySQL은 `LOAD DATA LOCAL INFILE`을 실행합니다. 이전 예시의 내용을 바탕으로 봤을 때, 이 복구를 수행하려면 테이블, 캐시, 로그 및 로컬 파일에 대해 각각 10GiB씩 총 40GiB의 임시 디스크 공간이 필요합니다. 최소 40GiB의 여유 디스크 공간이 없으면 복구가 실패합니다.

대규모 데이터 로드 최적화

대규모 데이터 로드의 경우 이진 로깅을 비활성화하여 오버헤드 및 디스크 공간 요구 사항을 줄입니다. 백업 보존 기간을 0으로 설정하여 이진 로깅을 비활성화할 수 있습니다. 로드가 완료되면 백업 보존 기간을 0이 아닌 적절한 값으로 복원합니다. 자세한 정보는 Amazon RDS DB 인스턴스 수정 및 설정 표의 백업 보존 기간을 참조하세요.

> **Note**
>
> DB 인스턴스가 읽기 전용 복제본의 원본 DB 인스턴스인 경우에는 백업 보존 기간을 0으로 설정할 수 없습니다.

데이터를 로드하기 전에 DB 스냅샷을 만드는 것이 좋습니다. 자세한 내용은 수동 백업 관리 섹션을 참조하세요.

InnoDB

실행 취소 로깅 및 복구 옵션에 대한 다음 정보는 데이터베이스 성능을 최적화하기 위해 InnoDB 트랜잭션을 작게 유지하는 데 도움이 됩니다.

InnoDB 실행 취소 로깅 이해

실행 취소는 트랜잭션 롤백을 활성화하고 다중 버전 동시성 제어(MVCC)를 지원하는 로깅 메커니즘입니다.

MySQL 5.7 이하 버전의 경우 실행 취소 로그는 InnoDB 시스템 테이블스페이스(일반적으로 ibdata1)에 저장되며 제거 스레드가 로그를 제거할 때까지 유지됩니다. 따라서 대규모 데이터 로드 트랜잭션은 시스템 테이블스페이스가 상당히 커지는 원인이 될 수 있고, 이때 사용되는 디스크 공간은 데이터베이스를 다시 만들어야 회수할 수 있습니다.

모든 MySQL 버전에서 제거 스레드는 가장 오래된 활성 트랜잭션이 커밋되거나 롤백될 때까지 실행 취소 로그를 제거할 수 없습니다. 데이터베이스가 로드 중에 다른 트랜잭션을 처리 중인 경우, 이들 트랜잭션의 실행 취소 로그 역시 누적되며 트랜잭션이 커밋되고 MVCC에 대해 실행 취소 로그가 필요한 다른 트랜잭션이 없더라도 실행 취소 로그를 제거할 수 없습니다. 이 경우 읽기 전용 트랜잭션을 포함한 모든 트랜잭션이 느려집니다. 이 속도 저하는 로드 트랜잭션뿐만 아니라 모든 트랜잭션이 변경하는 모든 행에 모든 트랜잭션이 액세스하기 때문에 발생합니다. 실제로 트랜잭션은 장기 실행 로드 트랜잭션으로 인해 실행 취소 로그 정리 중에 제거되지 않는 실행 취소 로그를 검사해야 합니다. 이는 수정된 행에 액세스하는 모든 작업의 성능에 영향을 줍니다.

InnoDB 트랜잭션 복구 옵션

InnoDB는 커밋 작업을 최적화하지만 대규모 트랜잭션 롤백은 느립니다. 더 빠른 복구를 위해 시점 복구를 수행하거나 DB 스냅샷을 복원합니다. 자세한 내용은 시점 복구 및 DB 인스턴스 복원(을)를 참조하세요.

데이터 가져오기 형식

MySQL은 플랫 파일과 SQL이라는 두 가지 데이터 가져오기 형식을 지원합니다. 각 형식에 대한 정보를 검토하여 요구 사항에 가장 적합한 옵션을 결정합니다.

플랫 파일

소규모 트랜잭션의 경우 `LOAD DATA LOCAL INFILE`을 사용하여 플랫 파일을 로드합니다. 이 데이터 가져오기 형식은 SQL 사용에 비해 다음과 같은 이점을 제공할 수 있습니다.

- 더 낮은 네트워크 트래픽
- 데이터 전송 비용 절감
- 데이터베이스 처리 오버헤드 감소
- 더 빠른 처리

`LOAD DATA LOCAL INFILE`은 전체 플랫 파일을 하나의 트랜잭션으로 로드합니다. 다음과 같은 이점을 위해 개별 파일의 크기를 작게 유지합니다.

- 재개 기능 – 로드된 파일을 추적할 수 있습니다. 로드 중에 문제가 발생하면 중단된 부분부터 다시 시작할 수 있습니다. 일부 데이터를 Amazon RDS로 다시 전송해야 할 수도 있지만, 파일이 작으면 재전송되는 양이 최소화됩니다.
- 병렬 데이터 로드 - 단일 파일 로드에 충분한 IOPS와 네트워크 대역폭이 있는 경우 병렬로 로드하면 시간이 절약될 수 있습니다.
- 로드 속도 제어 - 데이터 로드가 다른 프로세스에 부정적인 영향을 미치는 경우 파일 간 간격을 늘려 로드 속도를 제어할 수 있습니다.

대규모 트랜잭션은 `LOAD DATA LOCAL INFILE`을 사용하여 데이터를 가져올 때의 이점을 줄입니다. 대량의 데이터를 더 작은 파일로 나눌 수 없는 경우 SQL을 사용하는 것을 고려하세요.

SQL

SQL은 플랫 파일에 비해 한 가지 중요한 장점이 있는데, 그것은 바로 트랜잭션 크기를 작게 유지하기 쉽다는 점입니다. 그러나 SQL은 플랫 파일보다 로드하는 데 훨씬 더 오래 걸릴 수 있습니다. 또한 장애 발생 후 재개 위치를 결정하기 어려울 수 있으며, mysqldump 파일을 다시 시작할 수 없습니다. mysqldump 파일을 로드하는 동안 오류가 발생하는 경우 이 파일을 수정하거나 바꾸어야 로드를 재개할 수 있습니다. 또는 장애의 원인을 수정한 후 파일을 로드하고 재전송하기 전의 특정 시점으로 복원할 수 있습니다. 자세한 내용은 시점 복구 섹션을 참조하세요.

데이터베이스 체크포인트에 Amazon RDS DB 스냅샷 사용

이진 로깅 없이 몇 시간 또는 며칠 등의 긴 기간 동안 데이터를 로드하는 경우 DB 스냅샷을 사용하여 데이터 안전을 위한 주기적 체크포인트를 제공하세요. 각 DB 스냅샷은 시스템 장애 또는 데이터 손상 이벤트 중에 복구 시점 역할을 하는 데이터베이스 인스턴스의 일관된 사본을 만듭니다. DB 스냅샷은 빠르기 때문에 체크포인트를 자주 제공해도 로드 성능에 미치는 영향이 최소화됩니다. 데이터베이스 내구성이나 복구 기능에 영향을 주지 않고 이전 DB 스냅샷을 삭제할 수 있습니다. DB 스냅샷에 대한 자세한 내용은 수동 백업 관리 섹션을 참조하세요.

데이터베이스 로드 시간 단축

다음 항목은 로드 시간을 줄이기 위한 추가 팁입니다.

- MySQL 데이터베이스로 데이터를 로드하기 전에 모든 보조 인덱스를 만듭니다. 다른 데이터베이스 시스템과 달리 MySQL은 보조 인덱스를 추가하거나 수정할 때 전체 테이블을 다시 빌드합니다. 이 프로세스는 인덱스 변경 사항이 있는 새 테이블을 만들고, 모든 데이터를 복사하고, 원본 테이블을 삭제합니다.

- 프라이머리 키 순서로 데이터를 로드합니다. InnoDB 테이블의 경우 이렇게 하면 로드 시간을 75%~80% 줄이고 데이터 파일 크기를 50% 줄일 수 있습니다.
- `foreign_key_checks`를 `0`으로 설정하여 외래 키 제약 조건을 비활성화합니다. `LOAD DATA LOCAL INFILE`을 통해 로드되는 플랫 파일의 경우 이 단계는 많은 경우에 필수입니다. 모든 로드에서 외래 키 검사를 비활성화하면 데이터 로드가 가속화됩니다. 로드가 완료된 후 `foreign_key_checks`를 `1`로 설정하고 데이터를 확인하여 제약 조건을 다시 활성화합니다.
- 리소스 한도가 가까워지지 않은 한 데이터를 병렬로 로드합니다. 여러 테이블 세그먼트에서 동시 로드를 활성화하기 위해 적절한 경우 파티셔닝된 테이블을 사용합니다.
- SQL 실행 오버헤드를 줄이려면 여러 `INSERT` 문을 단일 다중 값 `INSERT` 작업으로 결합합니다.`mysqldump`는 이 최적화를 자동으로 구현합니다.
- `innodb_flush_log_at_trx_commit`을 `0`으로 설정하여 InnoDB 로그 IO 작업을 줄입니다. 로드가 완료되면 `innodb_flush_log_at_trx_commit`을 `1`로 복원합니다.

> **Warning**
>
> `innodb_flush_log_at_trx_commit`을 `0`으로 설정하면 InnoDB가 커밋이 발생할 때마다가 아닌 1초마다 로그를 플러시합니다. 이 설정은 성능을 높이지만 시스템 장애 발생 시 트랜잭션이 손실될 위험이 있습니다.

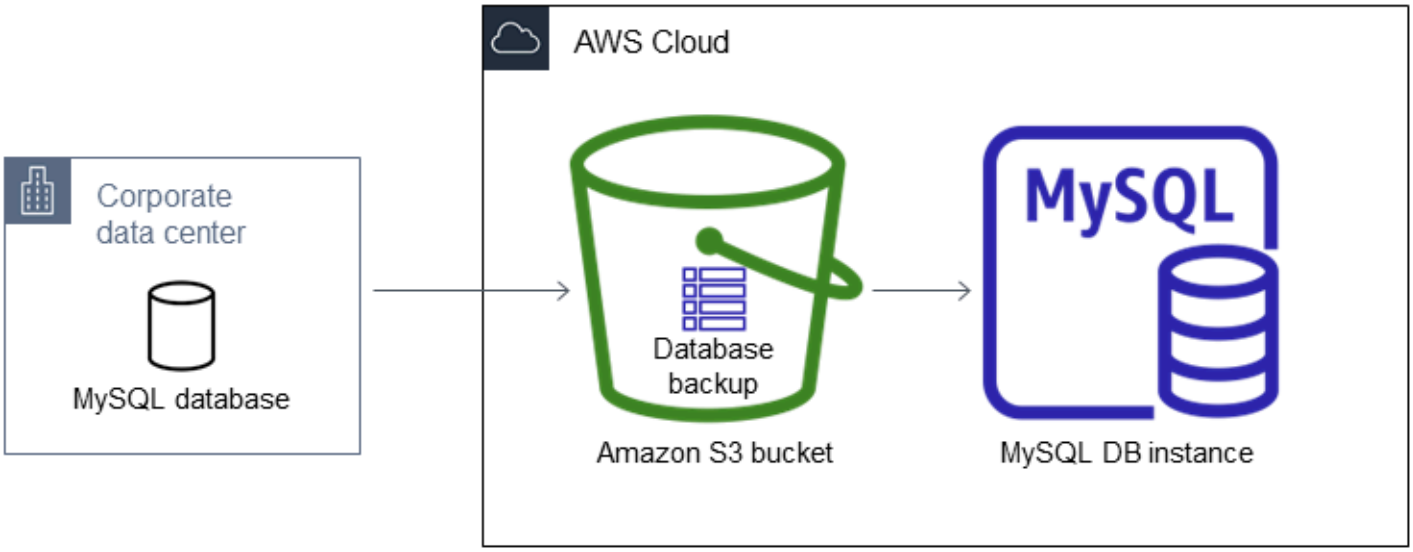
- 읽기 전용 복제본이 없는 DB 인스턴스로 데이터를 로드하는 경우 `sync_binlog`를 `0`으로 설정합니다. 로드가 완료되면 `sync_binlog parameter`를 `1`로 복원합니다.
- DB 인스턴스를 다중 AZ 배포로 변환하기 전에 단일 AZ DB 인스턴스로 데이터를 로드합니다. DB 인스턴스가 이미 다중 AZ 배포를 사용하는 경우 데이터 로드를 위해 단일 AZ 배포로 전환하지 않는 것이 좋습니다. 이렇게 해도 아주 작은 부분만이 개선됩니다.

MySQL DB 인스턴스로 백업 복원

백업 파일을 사용하여 Amazon RDS에서 MySQL 데이터베이스 가져오기를 지원합니다. 데이터베이스의 백업을 생성하여 Amazon S3에 저장한 다음 MySQL을 실행하는 새로운 Amazon RDS DB 인스턴스에 백업 파일을 복원할 수 있습니다.

이 단원에서 설명하는 시나리오는 온프레미스 데이터베이스의 백업을 복원합니다. 데이터베이스에 액세스할 수 있는 한, 이 기술을 다른 위치(예: Amazon EC2 또는 AWS 이외의 클라우드 서비스)의 데이터베이스에 사용할 수 있습니다.

다음 다이어그램에서 지원되는 시나리오를 찾을 수 있습니다.



Amazon S3에서 백업 파일을 가져오는 작업은 모든 AWS 리전에서 MySQL에 대해 지원됩니다.

백업 파일을 만들고, 복사하고, 복원하는 동안 온프레미스 데이터베이스를 오프라인 상태로 둘 수 있다면 백업 파일을 사용하여 데이터베이스를 Amazon RDS로 가져오는 것이 좋습니다. 데이터베이스를 오프라인 상태로 둘 수 없는 경우 이 주제에서 설명한 대로 Amazon S3를 통해 Amazon RDS로 마이그레이션한 후 데이터베이스를 업데이트하기 위해 바이너리 로그(binlog) 복제를 사용할 수 있습니다. 자세한 내용은 외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성 섹션을 참조하세요. 또한 AWS Database Migration Service를 사용하여 데이터베이스를 Amazon RDS로 마이그레이션할 수도 있습니다. 자세한 내용은 AWS Database Migration Service란 무엇입니까?를 참조하십시오.

Amazon S3에서 Amazon RDS로 백업 파일 가져오기에 대한 제한 및 권장 사항

다음은 Amazon S3에서 백업 파일을 가져올 때의 몇 가지 제한 및 권장 사항입니다.

- 기존 DB 인스턴스가 아닌 새 DB 인스턴스로만 데이터를 가져올 수 있습니다.
- 온프레미스 데이터베이스 백업을 생성할 때 Percona XtraBackup을 사용해야 합니다.
- DB 스냅샷 내보내기에서 Amazon S3로 데이터를 가져올 수 없습니다.
- 기본 MySQL 데이터 디렉터리 외부에서 정의된 테이블이 있는 원본 데이터베이스에서 마이그레이션할 수 없습니다.
- Percona Server for MySQL은 `mysql` 스키마에 `compression_dictionary*` 테이블을 포함할 수 있으므로 소스 데이터베이스로 지원되지 않습니다.
- AWS 리전에 있는 MySQL 메이저 버전의 기본 마이너 버전으로 데이터를 가져와야 합니다. 예를 들어, 메이저 버전이 MySQL 8.0이고 AWS 리전의 기본 마이너 버전이 8.0.28인 경우 데이터를 MySQL

버전 8.0.28 DB 인스턴스로 가져와야 합니다. 가져온 후 DB 인스턴스를 업그레이드할 수 있습니다. 기본 마이너 버전 결정에 대한 자세한 내용은 Amazon RDS의 MySQL 버전 단원을 참조하십시오.

- 메이저 버전과 마이너 버전 모두 역방향 마이그레이션은 지원하지 않습니다. 예를 들어 버전 8.0에서 버전 5.7로, 버전 8.0.32에서 버전 8.0.31로 마이그레이션할 수 없습니다.
- MySQL 5.5 또는 5.6 데이터베이스는 가져올 수 없습니다.
- 한 메이저 버전에서 다른 메이저 버전으로 온프레미스 MySQL 데이터베이스를 가져올 수 없습니다. 예를 들어 MySQL 5.7 데이터베이스를 RDS for MySQL 8.0 데이터베이스로 가져올 수 없습니다. 가져오기가 완료되면 DB 인스턴스를 업그레이드할 수 있습니다.
- 암호화된 원본 데이터베이스에서 복원할 수 없지만 암호화된 Amazon RDS DB 인스턴스로 복원할 수 있습니다.
- Amazon S3 버킷에서는 암호화된 버킷에서 복원할 수 없습니다.
- Amazon RDS DB 인스턴스와 다른 AWS 리전에서는 Amazon S3 버킷으로 복원할 수 없습니다.
- Amazon S3에서 가져오기는 db.t2.micro DB 인스턴스 클래스에서 지원되지 않습니다. 그러나 다른 DB 인스턴스 클래스로 복원한 다음 나중에 DB 인스턴스 클래스를 변경할 수 있습니다. 인스턴스 클래스에 대한 자세한 내용은 에 대한 DB 인스턴스 클래스의 하드웨어 사양 단원을 참조하십시오.
- Amazon S3는 Amazon S3 버킷에 업로드되는 파일 크기를 5TB로 제한합니다. 백업 파일이 5TB를 초과하면 해당 백업 파일을 더 작은 크기의 파일들로 나누어야 합니다.
- 데이터베이스를 복원하면 백업이 복사된 다음 DB 인스턴스에 추출됩니다. 따라서 백업 크기의 합계와 디스크의 원본 데이터베이스 크기를 더한 값보다 크거나 같은 DB 인스턴스에 대한 스토리지 공간을 프로비저닝합니다.
- Amazon RDS는 Amazon S3 버킷에 업로드되는 파일을 100만 개로 제한합니다. 모든 전체 및 증분 백업을 포함하여 데이터베이스에 대한 백업 데이터가 100만 개의 파일을 초과하는 경우 Gzip(.gz), tar(.tar.gz) 또는 Percona xbstream(.xbstream) 파일을 사용하여 전체 및 증분 백업 파일을 Amazon S3 버킷에 저장합니다. Percona Xtrabackup 8.0은 압축에 Percona xbstream만 지원합니다.
- 사용자 계정을 자동으로 가져오지 않습니다. 원본 데이터베이스에서 사용자 계정을 저장하고 나중에 새 DB 인스턴스에 추가하십시오.
- 함수를 자동으로 가져오지 않습니다. 원본 데이터베이스에서 함수를 저장하고 나중에 새 DB 인스턴스에 추가하십시오.
- 저장 프로시저를 자동으로 가져오지 않습니다. 원본 데이터베이스에서 저장 프로시저를 저장하고 나중에 새 DB 인스턴스에 추가하십시오.
- 시간대 정보를 자동으로 가져오지 않습니다. 원본 데이터베이스에서 시간대 정보를 저장하고 나중에 새 DB 인스턴스의 시간대를 설정하십시오. 자세한 내용은 MySQL DB 인스턴스의 현지 시간대 섹션을 참조하세요.

- `innodb_data_file_path` 파라미터는 기본 데이터 파일 이름 `"ibdata1:12M:autoextend"`를 사용하는 데이터 파일 하나만 사용하여 구성해야 합니다. 두 개의 데이터 파일이 있거나 다른 이름의 데이터 파일이 있는 데이터베이스는 이 방법을 사용하여 마이그레이션할 수 없습니다.

  다음은 허용되지 않는 파일 이름의 예입니다. `"innodb_data_file_path=ibdata1:50M; ibdata2:50M:autoextend"` 및 `"innodb_data_file_path=ibdata01:50M:autoextend"`.
- 복원된 데이터베이스의 최대 크기는 지원되는 최대 데이터베이스 크기에서 백업 크기를 뺀 값입니다. 따라서 지원되는 최대 데이터베이스 크기가 64TiB이고 백업 크기가 30TiB이면 복원된 데이터베이스의 최대 크기는 다음 예와 같이 34TiB가 됩니다.

  ```
  64 TiB - 30 TiB = 34 TiB
  ```

  Amazon RDS for MySQL에서 지원하는 최대 데이터베이스 크기에 대한 자세한 내용은 범용 SSD 스토리지 및 프로비저닝된 IOPS SSD 스토리지 단원을 참조하십시오.

Amazon S3에서 Amazon RDS로 백업 파일을 가져오는 설정 개요

Amazon S3에서 Amazon RDS로 백업 파일을 가져오는 설정에 필요한 구성 요소가 있습니다.

- 백업 파일을 저장할 Amazon S3 버킷.
- Percona XtraBackup이 생성한 온프레미스 데이터베이스 백업.
- Amazon RDS가 버킷에 액세스할 수 있도록 허용하는 AWS Identity and Access Management(IAM) 역할입니다.

이미 Amazon S3 버킷이 있으면 그 버킷을 사용하면 됩니다. Amazon S3 버킷이 없는 경우에는 새 버킷을 만들 수 있습니다. 새 버킷을 만드는 방법은 버킷 생성을 참조하십시오.

백업을 생성하려면 Percona XtraBackup 도구를 사용하십시오. 자세한 내용은 데이터베이스 백업 생성 섹션을 참조하세요.

이미 IAM 역할이 있으면 그 역할을 사용하면 됩니다. IAM 역할이 없는 경우에는 수동으로 새 역할을 만들 수 있습니다. 또는 AWS Management Console을 사용하여 데이터베이스를 복원할 때 마법사로 계정에서 새 IAM 역할이 생성되도록 선택할 수 있습니다. 수동으로 새 IAM 역할을 만들거나 기존 IAM 역할에 신뢰 및 권한 정책을 연결하려면 수동으로 IAM 역할 만들기 단원을 참조하십시오. 새 IAM 역할을 생성하려면 콘솔의 프로시저를 따르십시오.

데이터베이스 백업 생성

백업을 생성하려면 Percona XtraBackup 소프트웨어를 사용하십시오. 최신 버전의 Percona XtraBackup을 사용하는 것이 좋습니다. Percona XtraBackup은 Download Percona XtraBackup에서 설치할 수 있습니다.

> **Warning**
>
> 데이터베이스 백업을 만들 때 Xtrabackup은 xtrabackup_info 파일에 자격 증명을 저장할 수 있습니다. 해당 파일을 검사하여 `tool_command` 설정에는 민감한 정보가 포함되어 있지 않아야 합니다.

> **Note**
>
> MySQL 8.0 마이그레이션을 위해서는 Percona XtraBackup 8.0을 사용해야 합니다. Percona Xtrabackup 8.0.12 이상 버전은 모든 MySQL 버전의 마이그레이션을 지원합니다. RDS for MySQL 8.0.20 이상으로 마이그레이션하는 경우, 당신은 Percona XtraBackup 8.0.12 이상을 사용해야 합니다.
> MySQL 5.7 마이그레이션을 위해 Percona XtraBackup 2.4도 사용할 수 있습니다. 또한 이전 MySQL 버전의 마이그레이션을 위해 Percona XtraBackup 2.3 또는 2.4를 사용할 수 있습니다.

Percona XtraBackup을 사용하여 MySQL 데이터베이스 파일의 전체 백업을 생성할 수 있습니다. 또는 이미 Percona XtraBackup을 사용하여 MySQL 데이터베이스 파일을 백업 중인 경우 기존의 전체 및 증분 백업 디렉터리 및 파일을 업로드할 수 있습니다.

Percona XtraBackup을 사용한 데이터베이스 백업에 대한 자세한 내용은 Percona 웹 사이트의 Percona XtraBackup - Documentation 및 The xtrabackup Binary를 참조하십시오.

Percona XtraBackup을 사용한 전체 백업 생성

Amazon S3에서 복원할 수 있는 MySQL 데이터베이스 파일의 전체 백업을 생성하려면 Percona XtraBackup 유틸리티(`xtrabackup`)를 사용하여 데이터베이스를 백업합니다.

예를 들어, 다음 명령을 실행하면 MySQL 데이터베이스 백업을 만들고 `/on-premises/s3-restore/backup` 폴더에 파일을 저장합니다.

```
xtrabackup --backup --user=<myuser> --password=<password> --target-dir=</on-premises/
s3-restore/backup>
```

백업을 파일 하나로 압축하려면(필요하면 나중에 분할 가능) 다음 형식 중 하나로 백업을 저장하면 됩니다.

- Gzip(.gz)
- tar(.tar)
- Percona xbstream(.xbstream)

> **Note**
>
> Percona Xtrabackup 8.0은 압축에 Percona xbstream만 지원합니다.

다음 명령을 실행하면 Gzip 파일 여러 개로 된 MySQL 데이터베이스 백업이 만들어집니다.

```
xtrabackup --backup --user=<myuser> --password=<password> --stream=tar \
   --target-dir=</on-premises/s3-restore/backup> | gzip - | split -d --bytes=500MB \
   - </on-premises/s3-restore/backup/backup>.tar.gz
```

다음 명령을 실행하면 tar 파일 여러 개로 된 MySQL 데이터베이스 백업이 만들어집니다.

```
xtrabackup --backup --user=<myuser> --password=<password> --stream=tar \
   --target-dir=</on-premises/s3-restore/backup> | split -d --bytes=500MB \
   - </on-premises/s3-restore/backup/backup>.tar
```

다음 명령을 실행하면 xbstream 파일 여러 개로 된 MySQL 데이터베이스 백업이 만들어집니다.

```
xtrabackup --backup --user=<myuser> --password=<password> --stream=xbstream \
   --target-dir=</on-premises/s3-restore/backup> | split -d --bytes=500MB \
   - </on-premises/s3-restore/backup/backup>.xbstream
```

> **Note**
>
> 다음 오류가 표시되는 경우 명령에서 파일 형식이 혼합된 것이 원인일 수 있습니다.

```
ERROR:/bin/tar: This does not look like a tar archive
```

Percona XtraBackup을 사용한 증분 백업 사용

이미 Percona XtraBackup을 사용하여 MySQL 데이터베이스 파일의 전체 및 증분 백업을 수행 중인 경우 전체 백업을 만들고 백업 파일을 Amazon S3로 업로드할 필요가 없습니다. 대신, 기존 백업 디렉터리 및 파일을 Amazon S3 버킷으로 복사하여 시간을 크게 절약할 수 있습니다. Percona XtraBackup을 사용한 증분 백업 생성에 대한 자세한 내용은 Incremental Backup을 참조하십시오.

기존의 전체 및 증분 백업 파일을 Amazon S3 버킷으로 복사할 때 기본 디렉터리의 콘텐츠를 반복적으로 복사해야 합니다. 이러한 콘텐츠에는 전체 백업이 포함되며 모든 증분 백업 디렉터리 및 파일도 포함됩니다. 이 복사본은 Amazon S3 버킷의 디렉터리 구조를 보존해야 합니다. Amazon RDS는 모든 파일과 디렉터리를 반복합니다. Amazon RDS는 각 증분 백업에 포함된 `xtrabackup-checkpoints` 파일을 사용하여 기본 디렉터리를 식별하고 LSN(로그 시퀀스 번호) 범위별로 증분 백업을 정렬합니다.

Percona XtraBackup의 백업 고려 사항

Amazon RDS은 파일 이름을 기준으로 백업 파일을 사용합니다. 파일 형식에 따라 백업 파일에 적절한 파일 확장명을 지정합니다. 예를 들어, Percona xbstream 형식을 사용하여 저장한 파일에는 `.xbstream`을 지정합니다.

Amazon RDS는 알파벳 순서뿐 아니라 자연수 순서로도 백업 파일을 사용합니다. `split` 명령을 실행할 때는 `xtrabackup` 옵션을 사용하여 적절한 순서로 백업 파일을 작성하고 이름을 붙여야 합니다.

Amazon RDS는 Percona XtraBackup을 사용하여 생성되는 부분 백업을 지원하지 않습니다. 데이터베이스의 소스 파일을 백업할 때 `--tables`, `--tables-exclude`, `--tables-file`, `--databases`, `--databases-exclude` 또는 `--databases-file` 옵션을 사용하여 부분 백업을 만들 수 없습니다.

Amazon RDS에서는 Percona XtraBackup을 사용한 증분 백업을 지원합니다. Percona XtraBackup을 사용한 증분 백업 생성에 대한 자세한 내용은 Incremental Backup을 참조하십시오.

수동으로 IAM 역할 만들기

IAM 역할이 없는 경우에는 수동으로 새 역할을 만들 수 있습니다. 그러나 AWS Management Console를 사용하여 데이터베이스를 복원하는 경우에는 콘솔의 프로시저를 따르고 RDS가 이 새 IAM 역할을 만들도록 선택하는 것이 좋습니다.

Amazon S3에서 데이터베이스를 가져오기 위해 새 IAM 역할을 수동으로 생성하려면 Amazon RDS에서 Amazon S3 버킷으로 권한을 위임하기 위한 역할을 생성합니다. IAM 역할을 만들 때 신뢰 및 권한

정책을 연결합니다. Amazon S3에서 백업 파일을 가져오려는 경우 다음 예제와 비슷한 신뢰 및 권한 정책을 사용합니다. 역할을 만드는 방법에 대한 자세한 내용은 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.

신뢰 및 권한 정책에 따라 ARN(Amazon 리소스 이름)을 제공해야 합니다. ARN 형식에 대한 자세한 내용은 ARN(Amazon 리소스 이름) 및 AWS 서비스 네임스페이스를 참조하세요.

Example Amazon S3에서 가져오기 위한 신뢰 정책

```
{
    "Version": "2012-10-17",
    "Statement":
    [{
        "Effect": "Allow",
        "Principal": {"Service":  "rds.amazonaws.com"},
        "Action": "sts:AssumeRole"
    }]
}
```

Example Amazon S3에서 가져오기 위한 권한 정책 - IAM 사용자 권한

다음 예제에서는 *iam_user_id*를 자신의 값으로 바꿉니다.

```
{
    "Version":"2012-10-17",
    "Statement":
    [
        {
            "Sid":"AllowS3AccessRole",
            "Effect":"Allow",
            "Action":"iam:PassRole",
            "Resource":"arn:aws:iam::iam_user_id:role/S3Access"
        }
    ]
}
```

Example Amazon S3에서 가져오기 위한 권한 정책 - 역할 권한

다음 예제에서는 *amzn-s3-demo-bucket*과 *###*를 고유한 값으로 바꿉니다.

```
{
    "Version": "2012-10-17",
    "Statement":
```

```
    [
        {
        "Effect": "Allow",
        "Action":
            [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
        },
        {
        "Effect": "Allow",
        "Action":
            [
                "s3:GetObject"
            ],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/prefix*"
        },
        { // If your bucket is encrypted, include the following permission. This
 permission allows decryption of your AWS KMS key.
        "Effect": "Allow",
        "Action":
            [
                "kms:Decrypt"
            ],
        "Resource": [
            "arn:aws:kms:region:customer_id:key/key_id*"
            ]
        }
    ]
}
```

> **Note**
>
> 파일 이름 접두사를 포함하는 경우 접두사 다음에 별표(*)를 포함합니다. 접두사를 지정하지 않으려면 별표만 지정하면 됩니다.

Amazon S3에서 새 MySQL DB 인스턴스로 데이터 가져오기

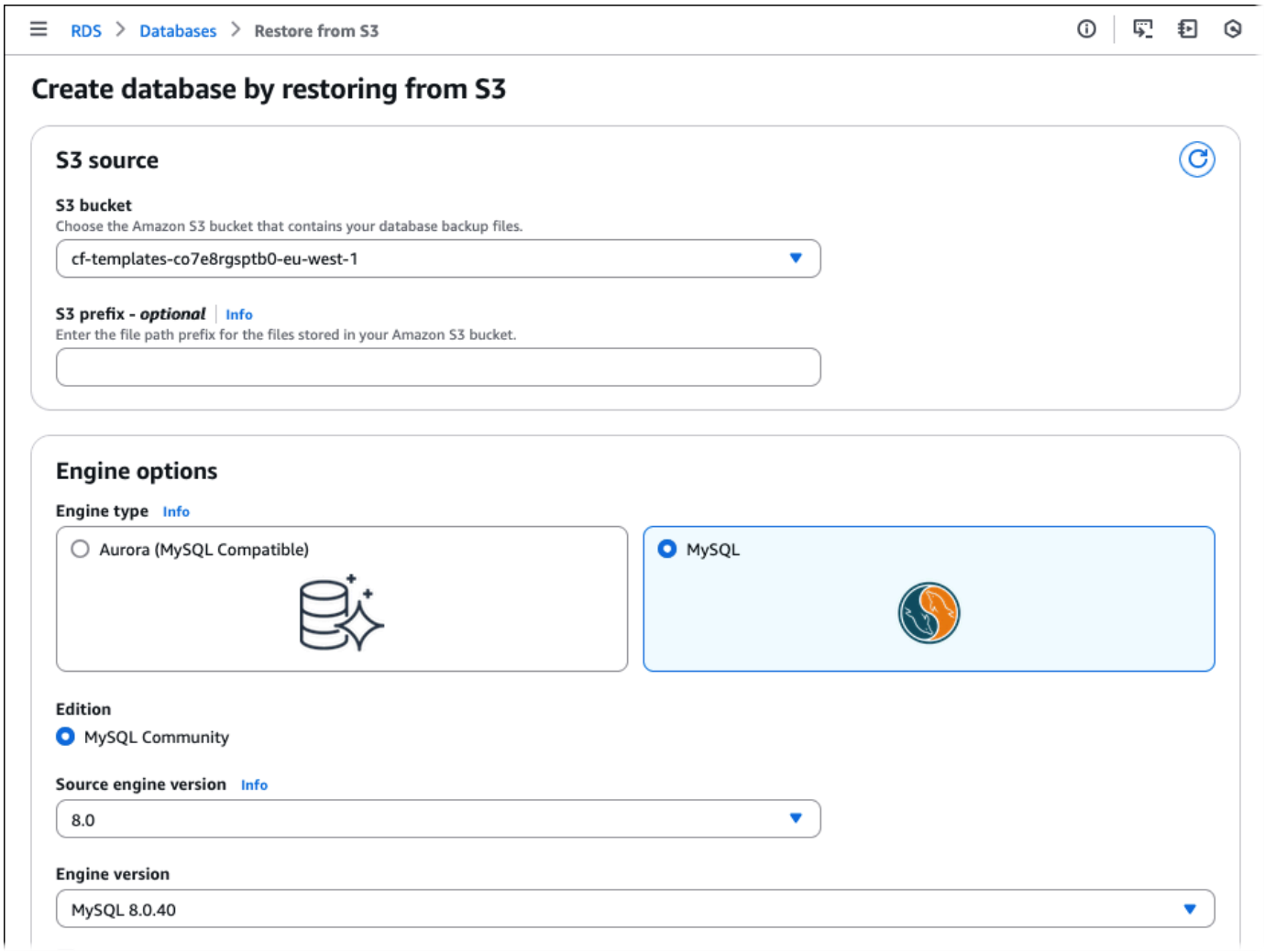
AWS Management Console, AWS CLI 또는 RDS API를 사용하여 새로운 MySQL DB 인스턴스로 Amazon S3의 데이터를 가져올 수 있습니다.

콘솔

Amazon S3에서 새 MySQL DB 인스턴스로 데이터를 가져오려면

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 콘솔의 오른쪽 상단에서 DB 인스턴스를 생성할 AWS 리전을 선택합니다. 데이터베이스 백업이 포함된 Amazon S3 버킷과 동일한 AWS 리전을 선택합니다.
3. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
4. S3에서 복원(Restore from S3)을 선택합니다.

   S3에서 복원하여 데이터베이스 생성(Create database by restoring from S3) 페이지가 나타납니다.



5. S3 대상(S3 destination)에서 다음을 수행합니다.

a. 백업이 포함된 S3 버킷을 선택합니다.

b. (선택 사항) S3 접두사에 Amazon S3 버킷에 저장되는 파일의 파일 경로 접두사를 입력합니다.

   접두사를 지정하지 않는 경우, RDS는 S3 버킷의 루트 폴더에 있는 모든 파일과 폴더를 사용하여 DB 인스턴스를 만듭니다. 접두사를 지정하는 경우 RDS는 파일의 경로가 지정된 접두사로 시작하는 S3 버킷의 파일과 폴더를 사용하여 DB 인스턴스를 만듭니다.

   예를 들어 이름이 backups인 하위 폴더의 S3에 백업 파일을 저장했고 여러 세트의 백업 파일이 각각 자체 디렉터리(gzip_backup1, gzip_backup2 등)에 들어 있다고 가정해봅시다. 이 경우 gzip_backup1 폴더의 파일에서 복원하려면 backups/gzip_backup1의 접두사를 지정합니다.

6. 엔진 옵션(Engine options)에서 다음을 수행합니다.

   a. 엔진 유형(Engine type)에서 MySQL을 선택합니다.

   b. 소스 엔진 버전(Source engine version)에서 소스 데이터베이스의 MySQL 메이저 버전을 선택합니다.

   c. 엔진 버전의 경우 AWS 리전에서 MySQL 메이저 버전의 기본 마이너 버전을 선택합니다.

      AWS Management Console에서 기본 마이너 버전만 사용할 수 있습니다. 가져온 후 DB 인스턴스를 업그레이드할 수 있습니다.

7. IAM 역할의 경우, Amazon RDS가 Amazon S3 버킷에 액세스할 수 있도록 허용하는 필수 신뢰 정책 및 권한 정책으로 IAM 역할을 만들거나 선택합니다. 다음 조치 중 하나를 취하세요.

   - (권장) 새 역할 생성을 선택하고 IAM 역할 이름을 입력합니다. 이 옵션을 사용하면 RDS가 자동으로 신뢰 정책 및 사용 권한 정책을 사용하여 역할을 만듭니다.
   - 기존 IAM 역할을 선택합니다. 이 역할이 the section called “수동으로 IAM 역할 만들기”의 모든 기준을 충족하는지 확인하세요.

8. DB 인스턴스 정보를 지정합니다. 각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하십시오.

> **Note**
>
> 새 DB 인스턴스에 충분한 메모리를 할당해야 복원 작업이 성공할 수 있습니다.
> 스토리지 Auto Scaling 활성화(Enable storage autoscaling)를 선택하여 향후 자동 확장을 허용할 수도 있습니다.

9. 필요에 따라 추가 설정을 선택합니다.
10. 데이터베이스 생성을 선택합니다.

AWS CLI

AWS CLI를 사용하여 Amazon S3에서 데이터를 새 MySQL DB 인스턴스로 가져오려면 다음 파라미터를 사용하여 restore-db-instance-from-s3 명령을 호출합니다. 각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하십시오.

> **Note**
>
> 새 DB 인스턴스에 충분한 메모리를 할당해야 복원 작업이 성공할 수 있습니다.
> 또한 `--max-allocated-storage` 파라미터를 사용하여 스토리지 Auto Scaling을 활성화하고 향후 자동 확장을 허용할 수도 있습니다.

- `--allocated-storage`
- `--db-instance-identifier`
- `--db-instance-class`
- `--engine`
- `--master-username`
- `--manage-master-user-password`
- `--s3-bucket-name`
- `--s3-ingestion-role-arn`
- `--s3-prefix`
- `--source-engine`
- `--source-engine-version`

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds restore-db-instance-from-s3 \
    --allocated-storage 250 \
    --db-instance-identifier myidentifier \
```

```
    --db-instance-class db.m5.large \
    --engine mysql \
    --master-username admin \
    --manage-master-user-password \
    --s3-bucket-name amzn-s3-demo-bucket \
    --s3-ingestion-role-arn arn:aws:iam::account-number:role/rolename \
    --s3-prefix bucketprefix \
    --source-engine mysql \
    --source-engine-version 8.0.32 \
    --max-allocated-storage 1000
```

Windows의 경우:

```
aws rds restore-db-instance-from-s3 ^
    --allocated-storage 250 ^
    --db-instance-identifier myidentifier ^
    --db-instance-class db.m5.large ^
    --engine mysql ^
    --master-username admin ^
    --manage-master-user-password ^
    --s3-bucket-name amzn-s3-demo-bucket ^
    --s3-ingestion-role-arn arn:aws:iam::account-number:role/rolename ^
    --s3-prefix bucketprefix ^
    --source-engine mysql ^
    --source-engine-version 8.0.32 ^
    --max-allocated-storage 1000
```

RDS API

Amazon RDS API를 사용하여 Amazon S3에서 새 MySQL DB 인스턴스로 데이터를 가져오려면 RestoreDBInstanceFromS3 작업을 호출하세요.

외부 MySQL 데이터베이스에서 Amazon RDS for MySQL DB 인스턴스로 데이터 가져오기

기존 MariaDB 또는 MySQL 데이터베이스에서 RDS for MariaDB 또는 RDS for MySQL DB 인스턴스로 데이터를 가져올 수 있습니다. 이렇게 하려면 mysqldump를 사용하여 데이터베이스를 복사하고 RDS for MariaDB 또는 RDS for MySQL DB 인스턴스에 바로 파이프합니다. `mysqldump` 명령줄 유틸리티는 한 MySQL 또는 MariaDB 서버에서 다른 MySQL 또는 MariaDB 서버로 데이터를 전송하고 백업본을 만드는 데 흔히 사용됩니다. 이 유틸리티는 MySQL 및 MariaDB 클라이언트 소프트웨어와 함께 포함되어 있습니다.

MariaDB 10.5에서는 이 클라이언트를 mariadb-dump라고 합니다. MariaDB 11.0.1부터 `mysqldump` 대신 `mariadb-dump`를 사용해야 합니다.

> **Note**
>
> MySQL DB 인스턴스에 많은 양의 데이터를 가져와서 내보내는 경우 `xtrabackup` 백업 파일 및 Amazon S3를 사용하여 Amazon RDS 내부 및 외부로 데이터를 더 빠르고 안정적으로 이동할 수 있습니다. 자세한 내용은 MySQL DB 인스턴스로 백업 복원 섹션을 참조하세요.
> Amazon RDS는 `xtrabackup` for MariaDB 또는 `mariabackup`을 지원하지 않습니다.
> Amazon RDS는 Amazon S3 for MySQL에서만 가져오기를 지원합니다.

외부 데이터베이스에서 Amazon RDS DB 인스턴스로 데이터를 이동하는 일반적인 `mysqldump` 명령은 다음 예와 같습니다. 값을 실제 정보로 바꿉니다. MariaDB 11.0.1 이상 버전의 경우 `mysqldump`를 `mariadb-dump`로 교체하고 `mysql`을 `mariadb`로 교체하세요.

```
mysqldump -u local_user \
    --databases database_name \
    --single-transaction \
    --compress \
    --order-by-primary  \
    --routines=0 \
    --triggers=0 \
    --events=0 \
    -plocal_password | mysql -u RDS_user \
        --port=port_number \
        --host=host_name \
        -pRDS_password
```

> **Important**
>
> `-p` 옵션과 입력한 암호 사이에 공백이 없어야 합니다.
> 보안 모범 사례로 예시에 표시된 프롬프트 이외의 자격 증명을 지정하는 것이 좋습니다.

다음 권장 사항과 고려 사항을 잘 파악하고 있어야 합니다.

- 덤프 파일에서 다음 스키마를 제외합니다.
  - `sys`

  - performance_schema
  - information_schema

  mysqldump 및 mariadb-dump 유틸리티는 기본적으로 이러한 스키마를 제외합니다.
- 사용자 및 권한을 마이그레이션해야 하는 경우 이를 다시 생성하는 데이터 제어 언어(DCL)를 생성하는 도구 사용을 고려합니다. 예를 들어 pt-show-grants 유틸리티가 있습니다.
- 가져오기를 수행하려면 사용자가 DB 인스턴스에 액세스할 수 있어야 합니다. 자세한 내용은 보안 그룹을 통한 액세스 제어 섹션을 참조하세요.

사용되는 파라미터는 다음과 같습니다.

- -u *local_user* – 사용자 이름을 지정하기 위해 사용합니다. 이 파라미터를 처음 사용할 때, 로컬 MySQL 또는 MariaDB 데이터베이스에서 --databases 파라미터로 식별되는 사용자 계정 이름을 지정합니다.
- --databases *database_name* – 로컬 MySQL 또는 MariaDB 인스턴스에서 Amazon RDS로 가져오려는 데이터베이스 이름을 지정합니다.
- --single-transaction – 로컬 데이터베이스에서 로드한 모든 데이터가 단일 시점에서 일치하는지 확인하기 위해 사용합니다. mysqldump가 데이터를 읽는 동안 데이터를 변경하는 다른 프로세스가 있는 경우, 이 파라미터를 사용하여 데이터 무결성을 유지합니다.
- --compress – 데이터를 Amazon RDS로 전송하기 전에 로컬 데이터베이스에서 데이터를 압축하여 네트워크 대역폭 사용을 줄이기 위해 사용합니다.
- --order-by-primary – 기본 키를 기준으로 각 테이블의 데이터를 정렬하여 로드 시간을 줄이기 위해 사용합니다.
- --routines - 복사하려는 데이터베이스에 저장 프로시저 또는 함수와 같은 루틴이 있는 경우 사용합니다. 파라미터를 0으로 설정하면 가져오기 프로세스 중에 루틴이 제외됩니다. 그런 다음 나중에 Amazon RDS 데이터베이스에서 루틴을 수동으로 다시 만듭니다.
- --triggers - 복사 중인 데이터베이스에 트리거가 있는 경우 사용합니다. 파라미터를 0으로 설정하면 가져오기 프로세스 중에 트리거가 제외됩니다. 그런 다음 Amazon RDS 데이터베이스에서 트리거를 수동으로 다시 만듭니다.
- --events - 복사하려는 데이터베이스에 이벤트가 있는 경우 사용합니다. 파라미터를 0으로 설정하면 가져오기 프로세스 중에 이벤트가 제외됩니다. 그런 다음 Amazon RDS 데이터베이스에서 이벤트를 수동으로 다시 만듭니다.
- -p*local_password* – 암호를 지정하기 위해 사용합니다. 이 파라미터를 처음 사용할 때, 첫 번째 -u 파라미터로 식별되는 사용자 계정 암호를 지정합니다.

- `-u` *`RDS_user`* – 사용자 이름을 지정하기 위해 사용합니다. 이 파라미터를 두 번째로 사용할 때, `--host` 파라미터로 식별되는 MariaDB 또는 MySQL DB 인스턴스에 대한 기본 데이터베이스의 사용자 계정 이름을 지정합니다.
- `--port` *`port_number`* - MySQL 또는 MariaDB DB 인스턴스의 포트를 지정하기 위해 사용합니다. DB 인스턴스를 만들 때 값을 변경하지 않는 한, 기본값은 3306입니다.
- `--host` *`host_name`* - Amazon RDS DB 인스턴스 엔드포인트의 도메인 이름 시스템(DNS) 이름을 지정하기 위해 사용합니다(예:`myinstance.123456789012.us-east-1.rds.amazonaws.com`). Amazon RDS 콘솔의 인스턴스 세부 정보에서 엔드포인트 값을 찾을 수 있습니다.
- `-p`*`RDS_password`* – 암호를 지정하기 위해 사용합니다. 이 파라미터를 두 번째로 사용할 때, 두 번째 `-u` 파라미터로 식별되는 사용자 계정 암호를 지정합니다.

Amazon RDS 데이터베이스에서 저장 프로시저, 트리거, 함수 또는 이벤트를 수동으로 만들어야 합니다. 복사 중인 데이터베이스에 이런 객체가 하나라도 있는 경우에는 `mysqldump` 또는 `mariadb-dump`를 실행할 때 이런 객체를 제외합니다. 이렇게 하려면 `mysqldump` 또는 `mariadb-dump` 명령에 다음 파라미터를 포함합니다.

- `--routines=0`
- `--triggers=0`
- `--events=0`

예제

다음 예제에서는 로컬 호스트에 있는 `world` 샘플 데이터베이스를 RDS for MySQL DB 인스턴스로 복사합니다. 값을 실제 정보로 바꿉니다. 샘플 데이터베이스를 RDS for MariaDB DB 인스턴스에 복사하려면 `mysqldump`를 `mariadb-dump`로 교체하고 `mysql`을 `mariadb`로 교체합니다.

대상 LinuxmacOS, 또는Unix:

```
sudo mysqldump -u local_user \
    --databases world \
    --single-transaction \
    --compress \
    --order-by-primary  \
    --routines=0 \
    --triggers=0 \
    --events=0 \
    -plocal_password | mysql -u rds_user \
```

```
        --port=3306 \
        --host=my_instance.123456789012.us-east-1.rds.amazonaws.com \
        -pRDS_password
```

Windows의 경우:

다음 명령은 Windows 프로그램 메뉴에서 명령 프롬프트를 마우스 오른쪽 버튼으로 클릭하고 관리자 권한으로 실행을 선택하여 열린 명령 프롬프트에서 실행해야 합니다. 값을 실제 정보로 바꿉니다. 샘플 데이터베이스를 RDS for MariaDB DB 인스턴스에 복사하려면 `mysqldump`를 `mariadb-dump`로 교체하고 `mysql`을 `mariadb`로 교체합니다.

```
mysqldump -u local_user ^
    --databases world ^
    --single-transaction ^
    --compress ^
    --order-by-primary  ^
    --routines=0 ^
    --triggers=0 ^
    --events=0 ^
    -plocal_password | mysql -u RDS_user ^
        --port=3306 ^
        --host=my_instance.123456789012.us-east-1.rds.amazonaws.com ^
        -pRDS_password
```

> **Note**
>
> 보안 모범 사례로 예시에 표시된 프롬프트 이외의 자격 증명을 지정하는 것이 좋습니다.

가동 중지 시간을 줄이면서 Amazon RDS for MySQL 데이터베이스로 데이터 가져오기

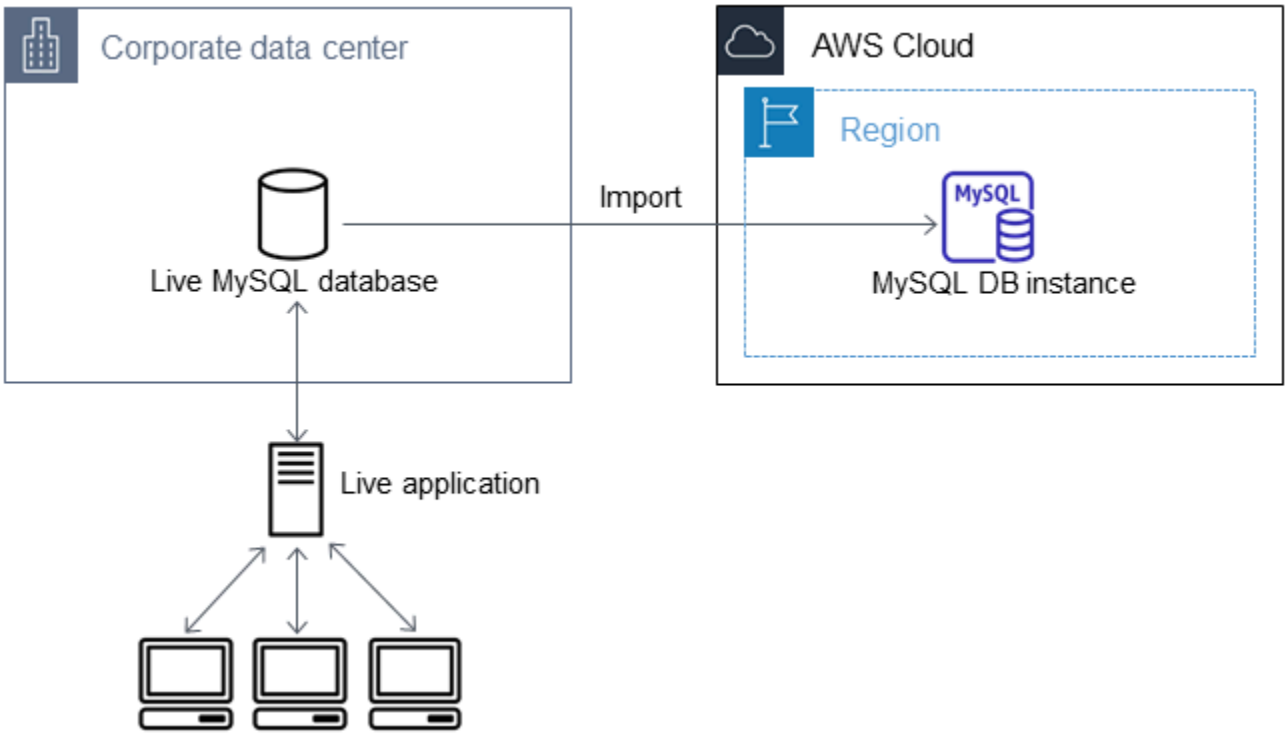
라이브 애플리케이션을 지원하는 외부 MariaDB 또는 MySQL 데이터베이스에서 RDS for MariaDB 인스턴스, RDS for MySQL 인스턴스 또는 RDS for MySQL 다중 AZ DB 클러스터로 데이터를 가져와야 할 경우가 있습니다. 다음 절차를 통해 애플리케이션 가용성에 미치는 영향을 최소화하세요. 이 절차는 대규모 데이터베이스로 작업하는 경우에도 유용합니다. 이 절차를 통해 네트워크를 거쳐 AWS로 전달되는 데이터의 양을 줄여 가져오기 비용을 줄일 수 있습니다.

이 절차에서는 데이터베이스 데이터의 복사본을 Amazon EC2 인스턴스로 전송하고 데이터를 새 Amazon RDS 데이터베이스로 가져옵니다. 그런 다음, 애플리케이션을 Amazon RDS 데이터베이스

로 리디렉션하기 전에 복제를 사용하여 Amazon RDS 데이터베이스를 라이브 외부 인스턴스에 맞춰 최신 상태로 업데이트합니다. 외부 인스턴스가 MariaDB 10.0.24 이상이고 대상 인스턴스가 RDS for MariaDB일 경우에는 글로벌 트랜잭션 식별자(GTID)를 기반으로 MariaDB 복제를 구성합니다. 그렇지 않으면 이진 로그 좌표를 기반으로 복제를 구성합니다. GTID 기반 복제가 더욱 신뢰성이 높은 방법이므로 외부 데이터베이스에서 지원된다면 GTID 기반 복제를 사용하는 것이 좋습니다. 자세한 내용은 MariaDB 설명서에서 Global Transaction ID 단원을 참조하세요.

> **Note**
>
> RDS for MySQL DB 인스턴스에 데이터를 가져오고자 하며 시나리오가 이 방법을 지원하는 경우 백업 파일 및 Amazon S3를 사용하여 Amazon RDS 내부 및 외부로 데이터를 이동하는 것이 좋습니다. 자세한 내용은 MySQL DB 인스턴스로 백업 복원 섹션을 참조하세요.

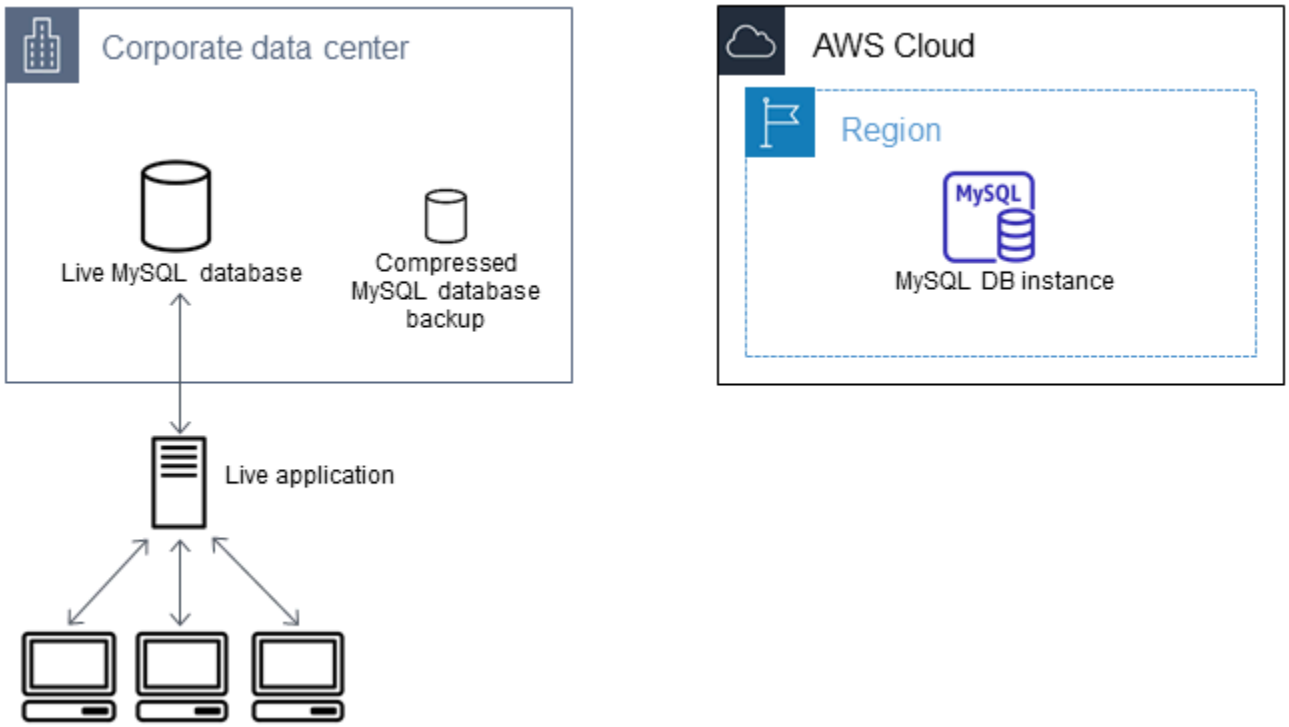
다음 다이어그램은 외부 MySQL 데이터베이스를 Amazon RDS의 MySQL 데이터베이스로 가져오는 것을 보여 줍니다.



작업 1: 기존 데이터베이스의 복사본 만들기

최소한의 가동 중지 시간으로 대량의 데이터를 RDS for MariaDB 또는 RDS for MySQL 데이터베이스로 마이그레이션하는 프로세스에서 첫 번째 단계는 원본 데이터의 복사본을 만드는 것입니다.

다음 다이어그램은 MySQL 데이터베이스의 백업을 만드는 방법을 보여 줍니다.



mysqldump 유틸리티를 사용하여 SQL 또는 구분 기호로 분리된 텍스트 형식으로 데이터베이스 백업본을 만들 수 있습니다. MariaDB 10.5에서는 이 클라이언트를 mariadb-dump라고 합니다. MariaDB 11.0.1부터 mysqldump 대신 mariadb-dump를 사용해야 합니다. 비프로덕션 환경에서 각각의 형식으로 테스트 실행을 통해 어떤 방법을 사용해야 mysqldump 실행 시간이 최소화되는지 확인하는 것을 권장합니다.

또한, 로드를 위해 구분 기호로 분리된 텍스트 형식을 사용할 때의 이점에 대해 mysqldump 성능상의 이점을 비교 검토하는 것을 권장합니다. 구분 기호로 분리된 텍스트 형식을 사용하는 백업에서는 덤프되는 각 테이블에 대해 탭으로 구분된 텍스트 파일이 생성됩니다. 데이터베이스를 가져오는 데 필요한 시간을 줄이기 위해 LOAD DATA LOCAL INFILE 명령을 사용하여 이런 파일을 병렬로 로드할 수 있습니다. 자세한 내용은 Load the data for MariaDB 또는 Load the data for MySQL을 참조하세요.

백업 작업을 시작하기 전에 Amazon RDS로 복사할 MySQL 또는 MariaDB 데이터베이스에서 복제 옵션을 설정해야 합니다. 복제 옵션에는 이진 로깅 활성화와 고유한 서버 ID 설정이 포함됩니다. 이러한 옵션을 설정하면 서버가 데이터베이스 트랜잭션 로깅을 시작하고 이 프로세스의 후반부에 소스 복제 인스턴스가 되도록 서버를 준비시킵니다.

다음 권장 사항과 고려 사항을 잘 파악하고 있어야 합니다.

- 데이터베이스의 일관된 상태를 덤프하기 때문에 `--single-transaction`와 함께 `mysqldump` 옵션을 사용하세요. 덤프 파일이 올바른지 확인하려면 `mysqldump`가 실행되는 동안 데이터 정의 언어(DDL) 문을 실행하지 마시기 바랍니다. 이러한 작업에 대한 유지 관리 기간을 예약할 수 있습니다.
- 덤프 파일에서 다음 스키마를 제외합니다.
  - `sys`
  - `performance_schema`
  - `information_schema`

  `mysqldump` 및 `mariadb-dump` 유틸리티는 기본적으로 이러한 스키마를 제외합니다.
- 사용자 및 권한을 마이그레이션해야 하는 경우 이를 다시 생성하는 데이터 제어 언어(DCL)를 생성하는 도구 사용을 고려합니다. 예를 들어 pt-show-grants 유틸리티가 있습니다.

복제 옵션을 설정하려면

1. `my.cnf` 파일을 편집합니다. 이 파일은 보통 `/etc` 아래에 있습니다.

   ```
   sudo vi /etc/my.cnf
   ```

   `log_bin` 및 `server_id` 옵션을 `[mysqld]` 섹션에 추가합니다. `log_bin` 옵션은 이진 로그 파일에 대한 파일 이름 식별자를 제공합니다. `server_id` 옵션은 소스-복제본 관계에서 서버의 고유 식별자를 제공합니다.

   다음 예제에서는 `my.cnf` 파일의 업데이트된 `[mysqld]` 섹션을 보여줍니다.

   ```
   [mysqld]
   log-bin=mysql-bin
   server-id=1
   ```

   자세한 내용은 MySQL 설명서의 Setting the Replication Source Configuration을 참조하세요.
2. 다중 AZ DB 클러스터를 사용한 복제의 경우 `ENFORCE_GTID_CONSISTENCY` 및 `GTID_MODE` 파라미터를 `ON`으로 설정합니다.

   ```
   mysql> SET @@GLOBAL.ENFORCE_GTID_CONSISTENCY = ON;
   ```

   ```
   mysql> SET @@GLOBAL.GTID_MODE = ON;
   ```

DB 인스턴스를 사용한 복제에는 이러한 설정이 필요하지 않습니다.

3. mysql 서비스를 다시 시작합니다.

```
sudo service mysqld restart
```

기존 데이터베이스의 백업 복사본 만들기

1. SQL 또는 구분 기호로 분리된 텍스트 형식을 지정하는 mysqldump 유틸리티 또는 mariadb-dump 유틸리티를 사용하여 데이터 백업을 만듭니다.

   MySQL 8.0.25 이하 버전에서는 서버 간 복제를 시작할 때 사용할 수 있는 백업 파일을 만들려면 --master-data=2를 지정합니다. MySQL 8.0.26 이상 버전에서는 서버 간 복제를 시작할 때 사용할 수 있는 백업 파일을 만들려면 --source-data=2를 지정합니다. 자세한 내용은 MySQL 설명서의 mysqldump — A Database Backup Program을 참조하세요.

   성능을 개선하고 데이터 무결성을 보장하려면 mysqldump에 --order-by-primary 및 --single-transaction 옵션을 사용합니다.

   백업에 MySQL 시스템 데이터베이스가 포함되지 않도록 하려면 mysqldump에 --all-databases 옵션을 사용하지 마세요. 자세한 내용은 MySQL 설명서의 Creating a Data Snapshot Using mysqldump를 참조하세요.

   필요한 경우 chmod를 사용하여 백업 파일이 만들어질 디렉터리가 쓰기 가능한 디렉터리가 되도록 하세요.

   > **Important**
   >
   > Windows에서는 명령 창을 관리자 권한으로 실행하십시오.

   - SQL 출력을 표시하려면 다음 명령을 사용합니다.

     대상 LinuxmacOS, 또는Unix:

     ```
     sudo mysqldump \
         --databases database_name \
         --master-data=2  \
         --single-transaction \
     ```

```
    --order-by-primary \
    -r backup.sql \
    -u local_user \
    -ppassword
```

> **Note**
>
> 보안 모범 사례로 예시에 표시된 프롬프트 이외의 자격 증명을 지정하는 것이 좋습니다.

Windows의 경우:

```
mysqldump ^
    --databases database_name ^
    --master-data=2  ^
    --single-transaction ^
    --order-by-primary ^
    -r backup.sql ^
    -u local_user ^
    -ppassword
```

> **Note**
>
> 보안 모범 사례로 예시에 표시된 프롬프트 이외의 자격 증명을 지정하는 것이 좋습니다.

- 구분 기호로 분리된 텍스트 출력을 표시하려면 다음 명령을 사용합니다.

  대상 LinuxmacOS, 또는Unix:

```
sudo mysqldump \
    --tab=target_directory \
    --fields-terminated-by ',' \
    --fields-enclosed-by '"' \
    --lines-terminated-by 0x0d0a \
    database_name \
    --master-data=2 \
    --single-transaction \
    --order-by-primary \
    -ppassword
```

Windows의 경우:

```
mysqldump ^
    --tab=target_directory ^
    --fields-terminated-by "," ^
    --fields-enclosed-by """ ^
    --lines-terminated-by 0x0d0a ^
    database_name ^
    --master-data=2 ^
    --single-transaction ^
    --order-by-primary ^
    -ppassword
```

> **Note**
>
> 보안 모범 사례로 예시에 표시된 프롬프트 이외의 자격 증명을 지정하는 것이 좋습니다. Amazon RDS 데이터베이스에서 저장 프로시저, 트리거, 함수 또는 이벤트를 수동으로 만들어야 합니다. 복사 중인 데이터베이스에 이런 객체가 하나라도 있는 경우에는 mysqldump를 실행할 때 이런 객체를 제외합니다. 이렇게 하려면 mysqldump 명령에 다음 인수를 포함하세요.
>
> - `--routines=0`
> - `--triggers=0`
> - `--events=0`

MySQL 8.0.22 이하 버전 및 MariaDB 버전의 경우 mysqldump를 실행하고 구분된 텍스트 형식을 지정하면 `CHANGE MASTER TO` 설명이 반환됩니다. 이 설명에는 마스터 로그 파일 이름과 위치가 포함됩니다. MySQL 8.0.23 이상 버전의 경우 구분된 텍스트 형식을 사용하여 mysqldump를 실행하면 `CHANGE REPLICATION SOURCE TO` 설명이 반환됩니다. 이 설명에는 소스 로그 파일 이름과 위치가 포함됩니다. 외부 인스턴스가 MariaDB 10.0.24 이상 버전 또는 MySQL 8.0.23 이상 버전이 아닌 경우 `MASTER_LOG_FILE` 및 `MASTER_LOG_POS`의 값을 확인합니다. 복제를 설정할 때 이러한 값이 필요합니다.

MariaDB 버전과 MySQL 8.0.22 이하 버전에 대해 다음 출력이 반환됩니다.

```
-- Position to start replication or point-in-time recovery from
--
```

```
-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin-changelog.000031',
 MASTER_LOG_POS=107;
```

MySQL 8.0.23 이상 버전에 대해 다음 출력이 반환됩니다.

```
-- Position to start replication or point-in-time recovery from
--
-- CHANGE SOURCE TO SOURCE_LOG_FILE='mysql-bin-changelog.000031',
 SOURCE_LOG_POS=107;
```

MySQL 8.0.22 이하 버전에서 SQL 형식을 사용하는 경우 백업 파일의 CHANGE MASTER TO 설명에서 마스터 로그 파일 이름 및 위치를 가져올 수 있습니다. MySQL 8.0.23 이상 버전에서 SQL 형식을 사용하는 경우 백업 파일의 CHANGE REPLICATION SOURCE TO 설명에서 소스 로그 파일 이름 및 위치를 가져올 수 있습니다. 외부 인스턴스가 MariaDB 버전 10.0.24 이상일 경우에는 GTID를 다음 단계에서 가져올 수 있습니다.

2. 사용하는 외부 인스턴스가 MariaDB 버전 10.0.24 이상일 경우 GTID 기반 복제를 사용합니다. 외부 MariaDB 인스턴스에서 SHOW MASTER STATUS를 실행하여 이진 로그 파일 이름 및 위치를 가져온 다음, 외부 MariaDB 인스턴스에서 BINLOG_GTID_POS를 실행하여 GTID로 변환합니다.

```
SELECT BINLOG_GTID_POS('binary_log_file_name', binary_log_file_position);
```

반환된 GTID를 기록해 둡니다. 복제를 구성하는 데 GTID가 필요합니다.

3. 복사된 데이터를 압축하여 데이터를 Amazon RDS 데이터베이스로 복사하는 데 필요한 네트워크 리소스의 양을 줄입니다. 백업 파일의 크기를 기록해 둡니다. Amazon EC2 인스턴스를 얼마나 크게 만들지 결정할 때 이 정보가 필요합니다. 모두 완료되었으면, GZIP 또는 선호하는 압축 유틸리티를 사용하여 백업 파일을 압축합니다.

   - SQL 출력을 압축하려면 다음 명령을 사용합니다.

   ```
   gzip backup.sql
   ```

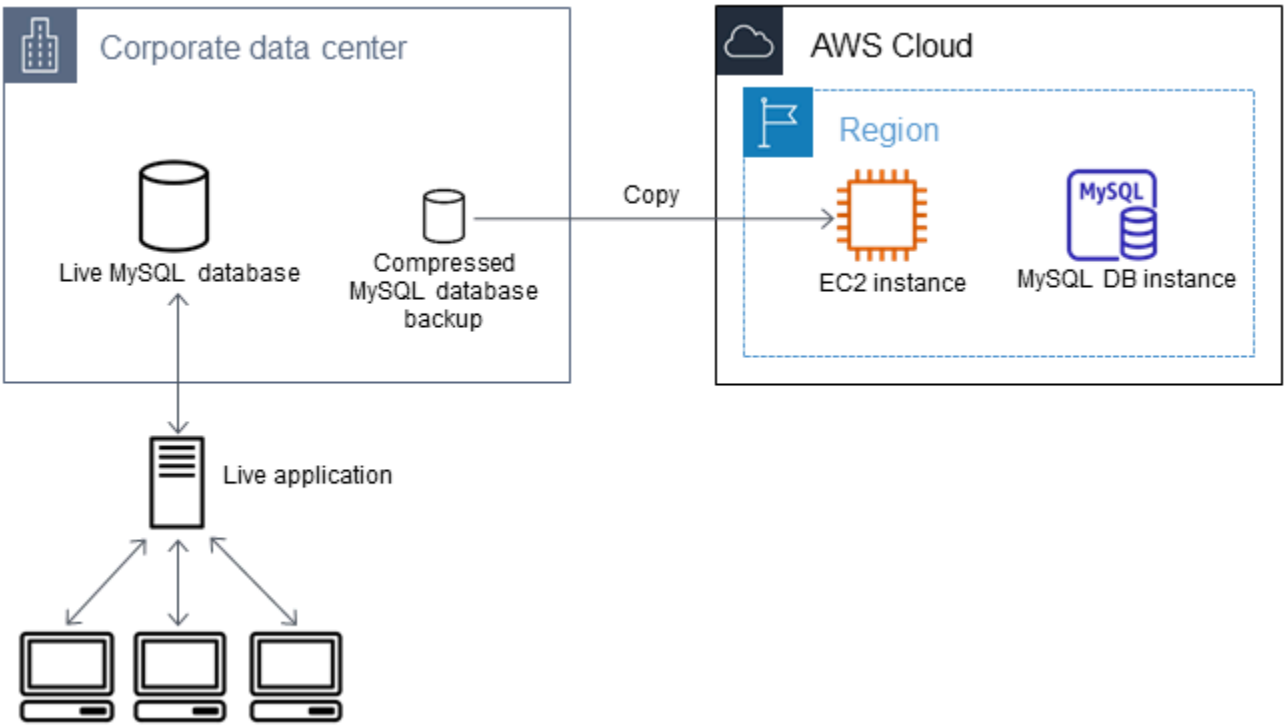
   - 구분 기호로 분리된 텍스트 출력을 압축하려면 다음 명령을 사용합니다.

   ```
   tar -zcvf backup.tar.gz target_directory
   ```

작업 2: Amazon EC2 인스턴스 만들기 및 압축된 데이터베이스 복사

압축된 데이터베이스 백업 파일을 Amazon EC2 인스턴스로 복사할 때는 데이터베이스 인스턴스 사이에서 압축되지 않은 데이터를 직접 복사할 때보다 네트워크 리소스를 덜 사용합니다. 데이터가 Amazon EC2에 있으면 MySQL 또는 MariaDB 데이터베이스로 바로 데이터를 복사할 수 있습니다. 네트워크 리소스의 비용을 절약하려면 Amazon EC2 인스턴스가 Amazon RDS DB 인스턴스와 같은 AWS 리전에 있어야 합니다. Amazon EC2 인스턴스를 Amazon RDS 데이터베이스와 같은 AWS 리전에 두면 가져오기 도중의 네트워크 대기 시간도 줄어듭니다.

다음 다이어그램은 데이터베이스 백업을 Amazon EC2 인스턴스로 복사하는 것을 보여줍니다.



Amazon EC2 인스턴스를 만들고 데이터를 복사하려면

1. Amazon RDS 데이터베이스를 만들려는 AWS 리전에서 가상 프라이빗 클라우드(VPC), VPC 보안 그룹 및 VPC 서브넷을 만듭니다. VPC 보안 그룹의 인바운드 규칙에서 애플리케이션에 필요한 IP 주소가 AWS에 연결하도록 허용하는지 확인합니다. IP 주소의 범위(예: `203.0.113.0/24`) 또는 다른 VPC 보안 그룹을 지정할 수 있습니다. Amazon VPC 콘솔을 사용하여 VPC, 서브넷 및 보안 그룹을 만들고 관리할 수 있습니다. 자세한 내용은 Amazon Virtual Private Cloud 사용 설명서의 Getting started with Amazon VPC를 참조하세요.
2. Amazon EC2 콘솔을 열고 Amazon EC2 인스턴스와 Amazon RDS 데이터베이스를 모두 포함하는 AWS 리전을 선택합니다. 1단계에서 만든 VPC, 서브넷 및 보안 그룹을 사용하여 Amazon EC2 인스턴스를 시작합니다. 데이터베이스 백업 파일이 압축되어 있지 않을 때는 이 파일을 위한 충분한 스

토리지 공간을 가진 인스턴스 유형을 선택해야 합니다. Amazon EC2 인스턴스에 관한 세부 정보는 Amazon Elastic Compute Cloud 사용 설명서의 Getting started with Amazon EC2를 참조하세요.

3. Amazon EC2 인스턴스에서 Amazon RDS 데이터베이스로 연결하려면 VPC 보안 그룹을 편집하세요. EC2 인스턴스의 프라이빗 IP 주소를 지정하는 인바운드 규칙을 추가합니다. EC2 콘솔 창에 있는 인스턴스 창의 세부 정보 탭에서 프라이빗 IP 주소를 찾을 수 있습니다. VPC 보안 그룹을 편집하고 인바운드 규칙을 추가하려면 EC2 콘솔 탐색 창에서 보안 그룹(Security Groups)를 선택하고 보안 그룹을 선택한 다음 EC2 인스턴스의 프라이빗 IP 주소를 지정하는 MySQL 또는 Aurora에 대한 인바운드 규칙을 추가합니다. 인바운드 규칙을 VPC 보안 그룹에 추가하는 방법을 알아보려면 Amazon Virtual Private Cloud 사용 설명서의 Security group rules를 참조하세요.
4. 로컬 시스템에서 Amazon EC2 인스턴스로 압축된 데이터베이스 백업 파일을 복사합니다. 필요한 경우 chmod를 사용하여 Amazon EC2 인스턴스의 대상 디렉터리에 대해 쓰기 권한이 있는지 확인하세요. scp 또는 Secure Shell(SSH) 클라이언트를 사용하여 파일을 복사할 수 있습니다. 다음은 예제 scp 명령입니다.

```
scp -r -i key pair.pem backup.sql.gz ec2-user@EC2 DNS:/target_directory/backup.sql.gz
```

> **Important**
>
> 민감한 데이터는 반드시 보안 네트워크 전송 프로토콜을 사용하여 복사해야 합니다.

5. Amazon EC2 인스턴스에 연결하고 다음 명령을 사용하여 최신 업데이트와 MySQL 클라이언트 도구를 설치합니다.

```
sudo yum update -y
sudo yum install mysql -y
```

자세한 내용은 Amazon Elastic Compute Cloud 사용 설명서의 Connect to your instance를 참조하세요.

> **Important**
>
> 이 예제에서는 Amazon Linux AMI 배포의 Amazon Machine Image(AMI)에 MySQL 클라이언트를 설치합니다. 이 예제에서는 Ubuntu 또는 Red Hat Enterprise Linux 등 다른 배포에 MySQL 클라이언트를 설치하지 않습니다. MySQL 설치에 대한 자세한 내용은 MySQL 설명서의 Installing MySQL을 참조하세요.

6. Amazon EC2 인스턴스에 연결되어 있는 상태에서 데이터베이스 백업 파일의 압축을 풉니다. 다음 명령은 예시입니다.

   - SQL 출력을 압축 해제하려면 다음 명령을 사용합니다.

     ```
     gzip backup.sql.gz -d
     ```

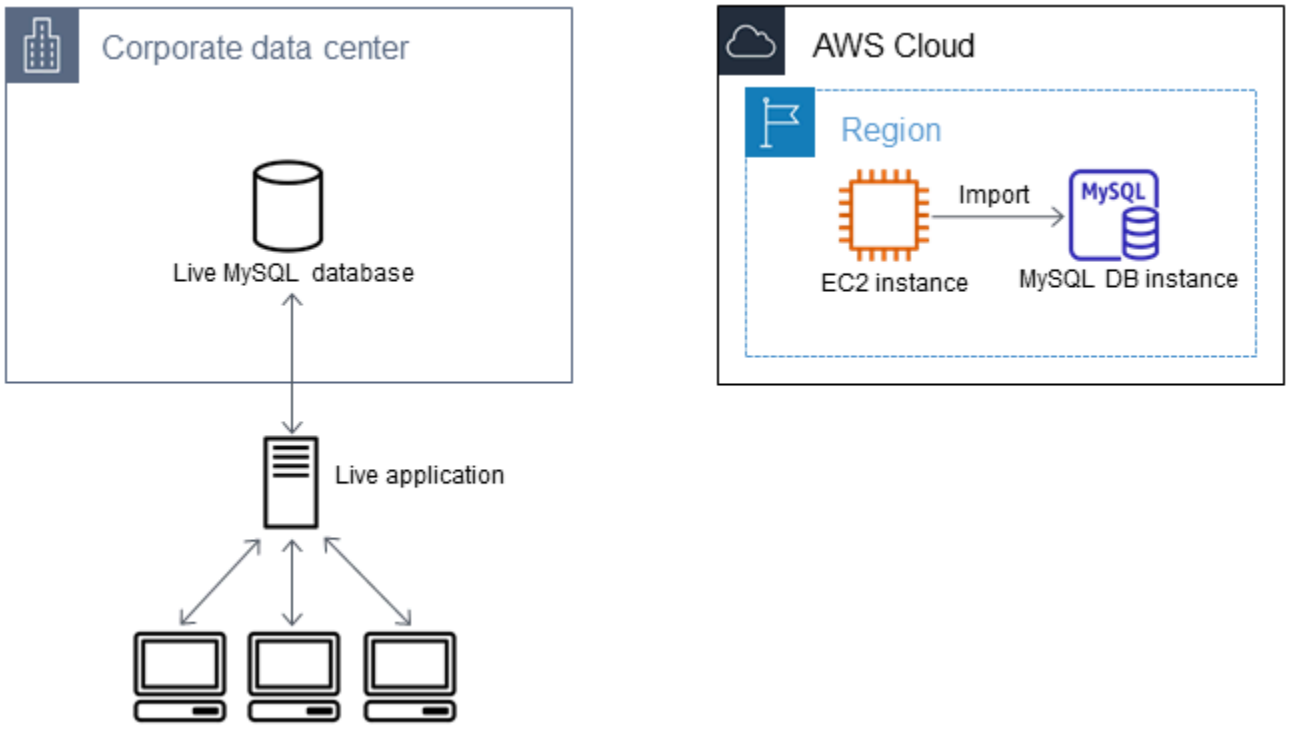
   - 구분 기호로 분리된 텍스트 출력을 압축 해제하려면 다음 명령을 사용합니다.

     ```
     tar xzvf backup.tar.gz
     ```

작업 3: MySQL 또는 MariaDB 데이터베이스 만들기 및 Amazon EC2 인스턴스에서 데이터 가져오기

Amazon EC2 인스턴스와 같은 AWS 리전에 RDS for MariaDB DB 인스턴스, RDS for MySQL DB 인스턴스 또는 RDS for MySQL 다중 AZ DB 클러스터를 만들면 인터넷에서 가져오는 것보다 빠른 속도로 Amazon EC2에서 데이터베이스 백업 파일을 가져올 수 있습니다.

다음 다이어그램은 Amazon EC2 인스턴스에서 MySQL 데이터베이스로 백업을 가져오는 것을 보여줍니다.



MariaDB 또는 MySQL 데이터베이스를 만들고 데이터 가져오기

1. 이 Amazon RDS 데이터베이스에 대해 예상되는 워크로드를 지원하기 위해 필요한 DB 인스턴스 클래스와 스토리지 공간의 양을 결정합니다. 이 프로세스의 일환으로 데이터 로드 절차를 위해 충분한 공간과 처리 용량이 어느 정도인지 결정합니다. 또한 프로덕션 워크로드를 처리하는 데 필요한 사항을 결정합니다. 원본 MySQL 또는 MariaDB 데이터베이스의 크기와 리소스를 바탕으로 평가할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요.
2. Amazon EC2 인스턴스가 포함된 AWS 리전에서 DB 인스턴스 또는 다중 AZ DB 클러스터를 만듭니다.

   RDS for MySQL 다중 AZ DB 클러스터를 만들려면 Amazon RDS용 다중 AZ DB 클러스터 생성 섹션의 지침을 따르세요.

   RDS for MariaDB 또는 RDS for MySQL DB 인스턴스를 생성하려면 Amazon RDS DB 인스턴스 생성의 지침과 다음 지침을 따르세요.

   - 소스 DB 인스턴스와 호환되는 DB 엔진 버전을 지정합니다.
     - 소스 인스턴스가 MySQL 5.5.x인 경우 Amazon RDS DB 인스턴스는 MySQL이어야 합니다.
     - 소스 인스턴스가 MySQL 5.6.x 또는 5.7.x인 경우 Amazon RDS DB 인스턴스는 MySQL 또는 MariaDB여야 합니다.
     - 원본 인스턴스가 MySQL 8.0.x인 경우 Amazon RDS DB 인스턴스는 MySQL 8.0.x여야 합니다.
     - 원본 인스턴스가 MySQL 8.4.x인 경우 Amazon RDS DB 인스턴스는 MySQL 8.4.x여야 합니다.
     - 소스 인스턴스가 MariaDB 5.5 이상인 경우 Amazon RDS DB 인스턴스는 MariaDB여야 합니다.
   - Amazon EC2 인스턴스에서와 동일한 Virtual Private Cloud(VPC) 및 VPC 보안 그룹을 지정합니다. 이 접근 방식에서는 Amazon EC2 인스턴스와 Amazon RDS 인스턴스가 네트워크를 통해 서로를 볼 수 있습니다. DB 인스턴스가 공개적으로 액세스 가능한지 확인합니다. 다음 섹션에 설명한 대로 소스 데이터베이스를 사용하여 복제를 설정하려면 DB 인스턴스에 공개적으로 액세스할 수 있어야 합니다.
   - 데이터베이스 백업을 가져온 후에만 여러 가용 영역, 백업 보존 또는 읽기 전용 복제본을 구성해야 합니다 가져오기가 완료되면 프로덕션 인스턴스에 대해 다중 AZ 및 백업 보존을 설정할 수 있습니다.
3. Amazon RDS 데이터베이스에 대한 기본 구성 옵션을 검토합니다. 데이터베이스에 대한 기본 파라미터 그룹에 원하는 구성 옵션이 없는 경우, 원하는 구성 옵션이 있는 다른 파라미터 그룹

을 찾거나 새 파라미터 그룹을 생성합니다. 파라미터 그룹을 생성하는 것에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 섹션을 참조하세요.

4. 마스터 사용자로 Amazon RDS 데이터베이스에 연결합니다. DB 인스턴스에 액세스해야 하는 서비스, 애플리케이션 및 관리자를 지원하는 데 필요한 사용자를 만듭니다. Amazon RDS 데이터베이스의 호스트 이름은 포트 번호를 포함하지 않은 DB 인스턴스의 엔드포인트 값입니다(예: `mysampledb.123456789012.us-west-2.rds.amazonaws.com`). Amazon RDS 콘솔의 데이터베이스 세부 정보에서 엔드포인트 값을 찾을 수 있습니다.
5. Amazon EC2 인스턴스에 연결합니다. 자세한 내용은 Amazon Elastic Compute Cloud 사용 설명서의 Connect to your instance를 참조하세요.
6. `mysql` 명령을 사용하여 Amazon EC2 인스턴스에서 원격 호스트로 Amazon RDS 데이터베이스에 연결합니다. 다음 명령은 예제입니다.

   ```
   mysql -h host_name -P 3306 -u db_master_user -p
   ```

   *host_name*은 Amazon RDS 데이터베이스 엔드포인트입니다.
7. `mysql` 프롬프트에서 `source` 명령을 실행하고 데이터베이스 덤프 파일의 이름을 전달합니다. 이 명령은 데이터를 Amazon RDS DB 인스턴스로 로드합니다.

   - SQL 형식인 경우 다음 명령을 사용합니다.

     ```
     mysql> source backup.sql;
     ```

   - 구분 기호로 분리된 텍스트 형식의 경우 Amazon RDS 데이터베이스를 설정할 때 만든 기본 데이터베이스가 아니라면 먼저 데이터베이스를 만듭니다.

     ```
     mysql> create database database_name;
     mysql> use database_name;
     ```

     그런 다음, 테이블을 생성합니다.

     ```
     mysql> source table1.sql
     mysql> source table2.sql
     etc...
     ```

     그런 다음, 데이터를 가져옵니다.

```
mysql> LOAD DATA LOCAL INFILE 'table1.txt' INTO TABLE table1 FIELDS TERMINATED BY
 ',' ENCLOSED BY '"' LINES TERMINATED BY '0x0d0a';
mysql> LOAD DATA LOCAL INFILE 'table2.txt' INTO TABLE table2 FIELDS TERMINATED BY
 ',' ENCLOSED BY '"' LINES TERMINATED BY '0x0d0a';
etc...
```

성능을 개선하려면 여러 연결에서 이들 작업을 병렬로 수행하여 모든 테이블이 만들어진 다음에 동시에 로드되도록 할 수 있습니다.

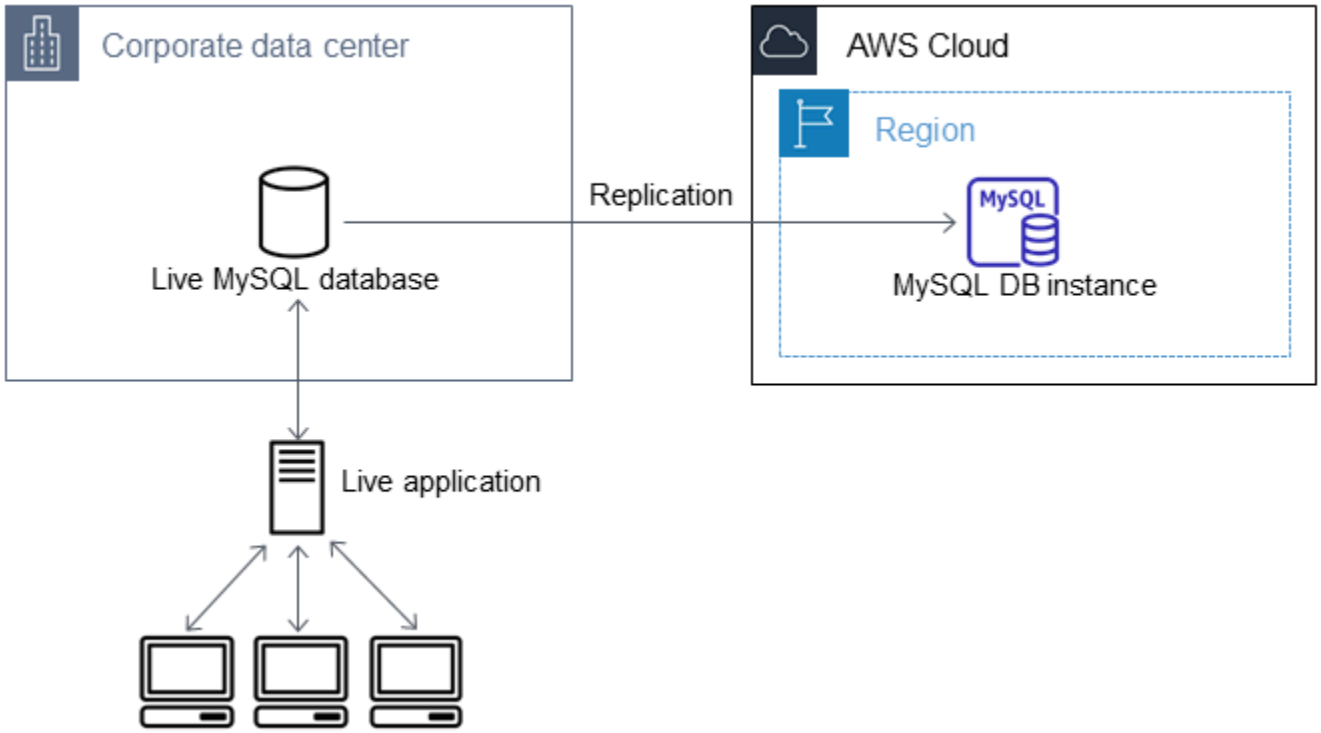
> **Note**
>
> 처음에 테이블을 덤프할 때 `mysqldump`와 함께 데이터 서식 옵션을 사용했다면, `LOAD DATA LOCAL INFILE`과 같은 옵션을 사용하여 데이터 파일 내용을 올바로 해석해야 합니다.

8. 가져온 데이터베이스에 있는 테이블 중 1개 또는 2개에 대해 간단한 `SELECT` 쿼리를 실행하여 가져오기에 성공했는지 확인합니다.

이 절차에 사용되는 Amazon EC2 인스턴스가 더 이상 필요하지 않은 경우 EC2 인스턴스를 종료하여 AWS 리소스 사용량을 줄여야 합니다. EC2 인스턴스를 종료하려면 Amazon Elastic Compute Cloud 사용 설명서의 Terminate an instance를 참조하세요.

작업 4: 외부 데이터베이스에서 새 Amazon RDS 데이터베이스로 데이터 복제

MariaDB 또는 MySQL 데이터베이스로 데이터를 복사하고 전송하는 데 걸린 시간 중에 원본 데이터베이스가 업데이트되었을 것입니다. 따라서 복제를 사용하여 복사된 데이터베이스를 원본 데이터베이스에 맞춰 최신 상태로 업데이트합니다.



Amazon RDS 데이터베이스에서 복제를 시작하는 데 필요한 권한은 제한되고 Amazon RDS 마스터 사용자는 사용할 수 없습니다. 따라서 메이저 엔진 버전에 적합한 다음의 Amazon RDS 저장 프로시저를 사용합니다.

- mysql.rds_set_external_master(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하)
- mysql.rds_set_external_source(RDS for MySQL 메이저 버전 8.4 이상)
- 복제를 구성하려면 mysql.rds_set_external_master_gtid, 복제를 시작하려면 mysql.rds_start_replication

복제를 시작하려면

작업 1에서 복제 옵션을 설정할 때 이진 로깅을 활성화하고 소스 데이터베이스의 고유한 서버 ID를 설정합니다. 이제 라이브 데이터베이스를 소스 복제 인스턴스로 포함하고 있는 복제본으로서 Amazon RDS 데이터베이스를 설정할 수 있습니다.

1. Amazon RDS 콘솔에서 Amazon RDS 데이터베이스에 대한 VPC 보안 그룹에 원본 데이터베이스를 호스팅하는 서버의 IP 주소를 추가합니다. VPC 보안 그룹 구성에 관한 자세한 내용은 Amazon Virtual Private Cloud 사용 설명서의 Configure security group rules를 참조하세요.

원본 인스턴스와 통신할 수 있도록, Amazon RDS 데이터베이스 IP 주소에서의 연결을 허용하도록 로컬 네트워크를 구성해야 할 수도 있습니다. Amazon RDS 데이터베이스의 IP 주소를 확인하려면 host 명령을 사용합니다.

```
host host_name
```

*host_name*은 Amazon RDS 데이터베이스 엔드포인트의 DNS 이름입니다(예: myinstance.123456789012.us-east-1.rds.amazonaws.com). Amazon RDS 콘솔의 인스턴스 세부 정보에서 엔드포인트 값을 찾을 수 있습니다.

2. 선택한 클라이언트를 사용하여 원본 인스턴스에 연결하고 복제에 사용될 사용자를 만듭니다. 이 계정은 오직 복제용으로만 사용되며 보안 향상을 위해 사용자의 도메인으로 제한되어야 합니다. 다음 명령은 예제입니다.

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 보안 인증 정보를 지정하는 것이 좋습니다.

3. 원본 인스턴스의 경우 복제 사용자에게 REPLICATION CLIENT 및 REPLICATION SLAVE 권한을 부여합니다. 예를 들어 도메인의 'repl_user' 사용자를 위해 모든 데이터베이스에서 REPLICATION CLIENT 및 REPLICATION SLAVE 권한을 부여하려면 다음 명령을 실행합니다.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
```

4. SQL 형식을 사용하여 백업 파일을 만들었고 외부 인스턴스가 MariaDB 10.0.24 이상이 아닌 경우 다음 명령을 실행하여 해당 파일의 내용을 살펴봅니다.

```
cat backup.sql
```

이 파일에는 마스터 로그 파일 이름과 위치를 포함한 CHANGE MASTER TO 설명이 포함됩니다. 이 설명은 --master-data와 함께 mysqldump 옵션을 사용할 때 백업 파일에 포함됩니다. MASTER_LOG_FILE 및 MASTER_LOG_POS에 대한 값을 확인합니다.

```
--
-- Position to start replication or point-in-time recovery from
```

```
--

-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin-changelog.000031', MASTER_LOG_POS=107;
```

구분 기호로 분리된 텍스트 형식을 사용하여 백업 파일을 만들었고 외부 인스턴스가 MariaDB 10.0.24 이상이 아닌 경우, 작업 1의 '존재하는 데이터베이스의 백업 복사본 만들기' 절차 1단계에서 확보한 바이너리 로그 좌표가 이미 있을 것입니다.

외부 인스턴스가 MariaDB 10.0.24 이상인 경우 복제를 시작할 수 있는 GTID를 작업 1의 '존재하는 데이터베이스의 백업 복사본 만들기' 절차 2단계에서 이미 확보했을 것입니다.

5. Amazon RDS 데이터베이스를 복제본으로 만듭니다. 외부 인스턴스가 MariaDB 10.0.24 이상이 아닌 경우 마스터 사용자로 Amazon RDS 데이터베이스에 연결하고 적절한 Amazon RDS 저장 프로시저를 사용하여 소스 데이터베이스를 소스 복제 인스턴스로 식별합니다.

   - mysql.rds_set_external_master(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하)
   - mysql.rds_set_external_source(RDS for MySQL 메이저 버전 8.4 이상)

   SQL 형식 백업 파일이 있는 경우 4단계에서 확인한 마스터 로그 파일 이름과 마스터 로그 위치를 사용합니다. 구분 기호로 분리된 텍스트 형식을 사용한 경우 백업 파일을 만들 때 확인한 이름과 위치를 사용합니다. 다음 명령은 예시입니다.

   MySQL 8.4 이상 버전

   ```
   CALL mysql.rds_set_external_source ('myserver.mydomain.com', 3306,
       'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 1);
   ```

   MariaDB 및 MySQL 8.0 이하 버전

   ```
   CALL mysql.rds_set_external_master ('myserver.mydomain.com', 3306,
       'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 1);
   ```

   > **Note**
   >
   > 보안 모범 사례로 여기에 표시된 프롬프트 이외의 보안 인증 정보를 지정하는 것이 좋습니다.

   외부 인스턴스가 MariaDB 10.0.24 이상인 경우 마스터 사용자로 Amazon RDS 데이터베이스에 연결하고 mysql.rds_set_external_master_gtid 명령을 사용하여 소스 데이터베이스를 소스 복제 인

스턴스로 식별합니다. 작업 1의 '기존 데이터베이스의 백업 복사본 만들기' 절차 2단계에서 확인된 GTID를 사용합니다. 다음 명령은 예제입니다.

```
CALL mysql.rds_set_external_master_gtid ('source_server_ip_address', 3306,
 'ReplicationUser', 'password', 'GTID', 1);
```

*source_server_ip_address*는 소스 복제 인스턴스의 IP 주소입니다. EC2 프라이빗 DNS 주소는 현재 지원되지 않습니다.

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 보안 인증 정보를 지정하는 것이 좋습니다.

6. Amazon RDS 데이터베이스에서 다음 mysql.rds_start_replication 명령을 실행하여 복제를 시작합니다.

```
CALL mysql.rds_start_replication;
```

7. Amazon RDS 데이터베이스에서 SHOW REPLICA STATUS 명령을 실행하여 복제본이 소스 복제 인스턴스에 맞게 최신 상태가 되는 시점을 파악합니다. `SHOW REPLICA STATUS` 명령의 결과에는 `Seconds_Behind_Master` 필드가 포함됩니다. `Seconds_Behind_Master` 필드가 0을 반환하면 복제본이 소스 복제 인스턴스에 맞게 최신 상태인 것입니다.

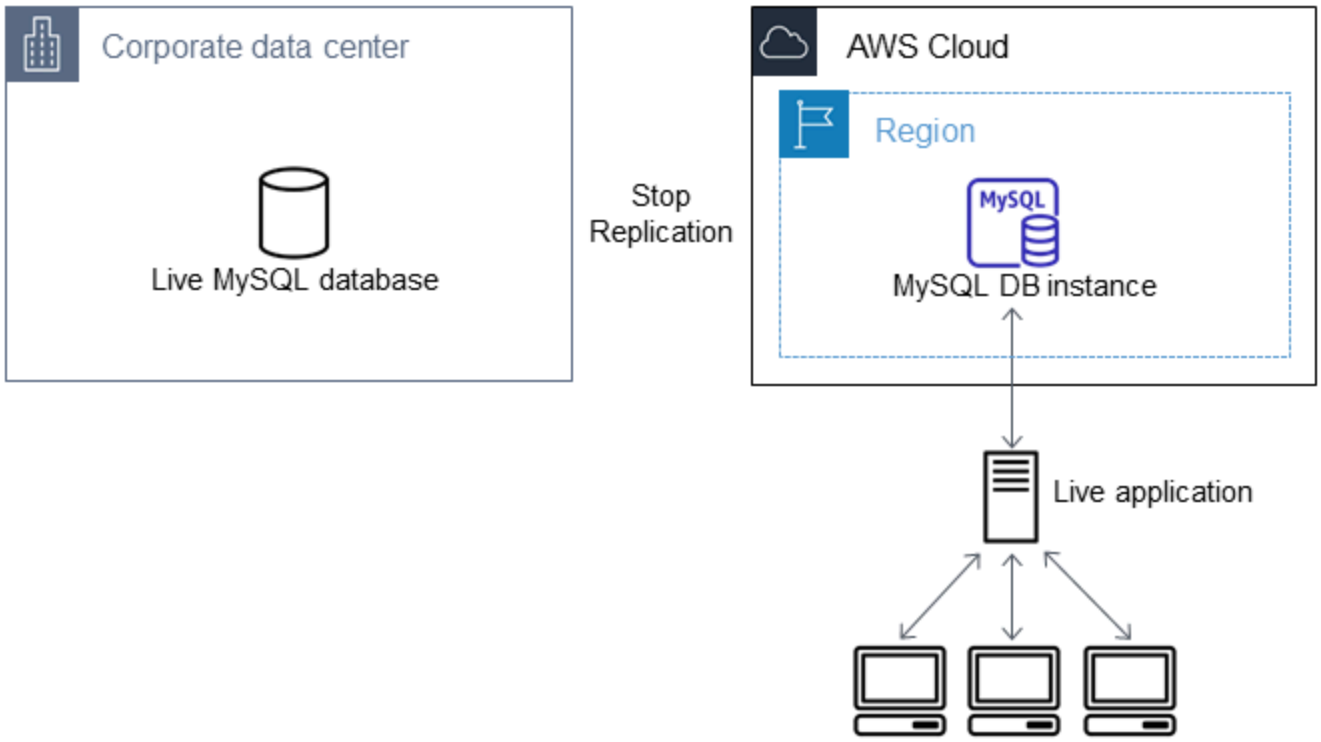
> **Note**
>
> 이전 버전의 MySQL에는 `SHOW SLAVE STATUS` 대신 `SHOW REPLICA STATUS`가 사용되었습니다. 8.0.23 이전 MySQL 버전을 사용하는 경우 `SHOW SLAVE STATUS`를 사용합니다.

MariaDB 10.5, 10.6 또는 10.11 DB 인스턴스의 경우 MySQL 명령 대신 mysql.rds_replica_status 저장 프로시저를 실행합니다.

8. Amazon RDS 데이터베이스가 최신 상태로 업데이트된 후, 필요할 경우 그 데이터베이스를 복원할 수 있도록 자동 백업을 켭니다. Amazon RDS 콘솔을 사용하여 Amazon RDS 데이터베이스에 대한 자동 백업을 켜거나 수정할 수 있습니다. 자세한 내용은 백업 소개 섹션을 참조하세요.

작업 5: 라이브 애플리케이션을 Amazon RDS 인스턴스로 리디렉션

MariaDB 또는 MySQL 데이터베이스가 소스 복제 인스턴스에 맞게 최신 상태로 업데이트되면 라이브 애플리케이션을 업데이트하여 Amazon RDS 인스턴스를 사용할 수 있습니다.



라이브 애플리케이션을 MariaDB 또는 MySQL 데이터베이스로 리디렉션하고 복제 중지

1. Amazon RDS 데이터베이스에 대한 VPC 보안 그룹을 추가하려면 애플리케이션을 호스팅하는 서버의 IP 주소를 추가합니다. VPC 보안 그룹 수정에 대한 자세한 내용은 Amazon Virtual Private Cloud 사용 설명서의 Configure security group rules를 참조하세요.
2. SHOW REPLICA STATUS 명령 결과에서 `Seconds_Behind_Master` 필드가 0인지 확인합니다. 이는 복제본이 소스 복제 인스턴스에 맞게 최신 상태라는 의미입니다.

```
SHOW REPLICA STATUS;
```

> **Note**
>
> 이전 버전의 MySQL에는 `SHOW SLAVE STATUS` 대신 `SHOW REPLICA STATUS`가 사용되었습니다. 8.0.23 이전 MySQL 버전을 사용하는 경우 `SHOW SLAVE STATUS`를 사용합니다.

MariaDB 10.5, 10.6, 10.11 DB 인스턴스의 경우 MySQL 대신 mysql.rds_replica_status 프로시저를 실행합니다.

3. 트랜잭션이 완료되면 소스에 대한 모든 연결을 닫습니다.
4. Amazon RDS 데이터베이스를 사용할 수 있도록 애플리케이션을 업데이트합니다. 이 업데이트에는 일반적으로 Amazon RDS 데이터베이스의 호스트 이름과 포트, 연결할 사용자 계정과 암호, 사용할 데이터베이스를 식별하기 위해 연결 설정을 변경하는 과정이 포함됩니다.
5. DB 인스턴스에 연결합니다.

   다중 AZ DB 클러스터의 경우에는 라이터 DB 인스턴스에 연결합니다.
6. 다음 mysql.rds_stop_replication 명령을 사용하여 Amazon RDS 인스턴스에 대한 복제를 중지합니다.

```
CALL mysql.rds_stop_replication;
```

7. 이 인스턴스가 더 이상 복제본으로 식별되지 않도록 Amazon RDS 데이터베이스에서 적절한 Amazon RDS 저장 프로시저를 실행하여 복제 구성을 재설정합니다.
   - mysql.rds_reset_external_master(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하)
   - mysql.rds_reset_external_source(RDS for MySQL 메이저 버전 8.4 이상)

   MySQL 8.4 이상 버전

```
CALL mysql.rds_reset_external_source;
```

   MariaDB 및 MySQL 8.0 이하 버전

```
CALL mysql.rds_reset_external_master;
```

8. 다중 AZ 지원 및 읽기 전용 복제본과 같은 Amazon RDS 기능을 추가로 활성화합니다. 자세한 내용은 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 및 DB 인스턴스 읽기 전용 복제본 작업 단원을 참조하세요.

원하는 소스에서 Amazon RDS for MySQL DB 인스턴스로 데이터 가져오기

다음 단계에서는 Amazon RDS DB 인스턴스로 데이터를 가져오는 방법을 안내합니다.

1단계: 로드할 데이터를 포함한 플랫 파일 만들기

쉼표로 구분된 값(CSV)과 같은 일반적인 형식을 사용하여 로드할 데이터를 저장합니다. 각 테이블에는 자체 파일이 있어야 하며, 여러 테이블에 대한 데이터를 같은 파일에 결합할 수 없습니다. 각 파일의 이름은 그에 상응하는 테이블과 같은 이름으로 지정합니다. 파일 확장명은 원하는 대로 정할 수 있습니다. 예를 들어, 테이블 이름이 `sales`인 경우 파일 이름은 `sales.csv` 또는 `sales.txt`일 수 있습니다.

가능하면 로드되는 테이블의 프라이머리 키를 기준으로 데이터를 정렬하세요. 그러면 로드 시간이 대폭 단축되고 데이터 스토리지 요구 사항이 최소화됩니다.

이 절차의 수행 속도와 효율성은 파일의 크기를 작게 유지하느냐에 좌우됩니다. 개별 파일의 압축되지 않은 크기가 1GiB보다 큰 경우에는 파일을 여러 개의 파일로 분할하고 각각 하나씩 따로 로드합니다.

Unix와 같은 시스템(Linux 포함)에서는 `split` 명령을 사용합니다. 예를 들어 다음 명령을 실행하면 `sales.csv` 파일이 1GiB 미만의 여러 파일로 분할되며, 줄 바꿈에서만 분할됩니다(`-C 1024m`). 새 파일의 이름에는 오름차순 숫자 접미사가 포함됩니다. 다음 명령은 `sales.part_00` 및 `sales.part_01`과 같은 이름의 파일을 생성합니다.

```
split -C 1024m -d sales.csv sales.part_
```

다른 운영 체제에서는 이와 유사한 유틸리티를 사용할 수 있습니다.

플랫 파일은 어디에나 저장할 수 있습니다. 그러나 5단계에서 데이터를 로드할 때는 파일이 있는 위치와 동일한 위치에서 `mysql` 쉘을 간접적으로 호출하거나 `LOAD DATA LOCAL INFILE`을 실행할 때 파일의 절대 경로를 사용해야 합니다.

2단계: 대상 DB 인스턴스에 액세스 중인 애플리케이션을 모두 중지

대량 로드를 시작하기 전에 로드하려는 대상 DB 인스턴스에 액세스하는 모든 애플리케이션의 활동을 중단합니다. 로드 중인 테이블 혹은 참조 테이블을 다른 세션이 수정하는 경우에는 더욱 더 중단해야 합니다. 그러면 로드 중에 발생하는 제약 조건 위반의 위험이 줄어들고 로드 성능이 향상됩니다. 또한 로드에 관여하지 않는 프로세스에 의한 변경 내용이 손실되지 않고 로드 직전의 시점으로 DB 인스턴스를 복원할 수 있습니다.

물론, 이것이 가능하지 않거나 유용하지 않을 수도 있습니다. 로드 전에 애플리케이션이 DB 인스턴스에 액세스하지 못하게 막을 수 없다면, 데이터의 가용성과 무결성을 보장하기 위한 일련의 단계를 수행하세요. 필요한 구체적인 단계는 특정 사용 사례와 사이트 요구 사항에 따라 크게 달라집니다.

3단계: DB 스냅샷 만들기

아무런 데이터도 없는 새 DB 인스턴스로 데이터를 로드할 경우에는 이 단계를 건너뛰어도 됩니다. 아니면 데이터 로드 전후에 대상 Amazon RDS DB 인스턴스의 DB 스냅샷을 만드는 것이 좋습니다. Amazon RDS DB 스냅샷은 DB 인스턴스를 알려진 상태로 복원하는 데 사용할 수 있는 DB 인스턴스의 완전한 백업입니다. DB 스냅샷을 시작하면 데이터베이스가 백업되는 동안 DB 인스턴스에 대한 I/O 작업이 일시 중단됩니다.

필요한 경우 로드 직전에 DB 스냅샷을 만들면 데이터베이스를 로드 이전의 상태로 복원할 수 있습니다. 로드 직후에 생성된 DB 스냅샷은 사고 발생 시 데이터를 다시 로드하지 않아도 됩니다. 또한 로드 후 DB 스냅샷을 사용하여 데이터를 새 데이터베이스 인스턴스로 가져올 수 있습니다.

아래 예시에서는 AWS CLI create-db-snapshot 명령을 실행하여 `AcmeRDS` 인스턴스의 DB 스냅샷을 만들고 DB 스냅샷에 `"preload"` 식별자를 지정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-snapshot \
    --db-instance-identifier AcmeRDS \
    --db-snapshot-identifier preload
```

Windows의 경우:

```
aws rds create-db-snapshot ^
    --db-instance-identifier AcmeRDS ^
    --db-snapshot-identifier preload
```

DB 스냅샷 기능에서 복원하는 기능을 사용하여 테스트 실행을 위한 테스트 DB 인스턴스를 만들거나 로드 중의 변경 사항을 실행 취소할 수도 있습니다.

DB 스냅샷에서 데이터베이스를 복원하면 모든 DB 인스턴스와 마찬가지로 고유 식별자와 엔드포인트가 있는 새 DB 인스턴스가 생성된다는 점을 꼭 명심해야 합니다. 엔드포인트를 변경하지 않고 DB 인스턴스를 복원해야 한다면, 우선 엔드포인트를 다시 사용할 수 있도록 DB 인스턴스를 삭제해야 합니다.

예를 들어 테스트 실행 또는 다른 테스트를 위한 DB 인스턴스를 만들려면 DB 인스턴스에 고유 식별자를 지정합니다. 예제에서 `AcmeRDS-2"`는 식별자입니다. 이 예제는 `AcmeRDS-2`와 연결된 엔드포인트를 사용하여 DB 인스턴스에 연결합니다. 자세한 정보는 restore-db-instance-from-db-snapshot을 참조하세요.

대상 LinuxmacOS, 또는Unix:

```
aws rds restore-db-instance-from-db-snapshot \
    --db-instance-identifier AcmeRDS-2 \
    --db-snapshot-identifier preload
```

Windows의 경우:

```
aws rds restore-db-instance-from-db-snapshot ^
    --db-instance-identifier AcmeRDS-2 ^
    --db-snapshot-identifier preload
```

기존 엔드포인트를 재사용하려면 우선 DB 인스턴스를 삭제한 다음, 복원된 데이터베이스에 같은 식별자를 지정해야 합니다. 자세한 내용은 delete-db-instance를 참조하세요.

다음 예제에서는 또한 DB 인스턴스의 최종 DB 스냅샷을 생성한 후에 삭제합니다. 이 단계는 선택 사항이지만 권장됩니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds delete-db-instance \
    --db-instance-identifier AcmeRDS \
    --final-db-snapshot-identifier AcmeRDS-Final

aws rds restore-db-instance-from-db-snapshot \
    --db-instance-identifier AcmeRDS \
    --db-snapshot-identifier preload
```

Windows의 경우:

```
aws rds delete-db-instance ^
    --db-instance-identifier AcmeRDS ^
    --final-db-snapshot-identifier AcmeRDS-Final

aws rds restore-db-instance-from-db-snapshot ^
    --db-instance-identifier AcmeRDS ^
    --db-snapshot-identifier preload
```

4단계(선택 사항): Amazon RDS 자동 백업 끄기

> **Warning**
>
> 시점 복구를 수행할 필요가 있는 경우에는 자동 백업을 끄지 마세요.

자동 백업을 끄는 것은 성능 최적화 수단이며 데이터 로드에 필수적이지는 않습니다. 자동 백업을 끄면 기존 백업이 모두 지워집니다. 따라서 자동 백업을 끈 후에는 시점 복구를 수행할 수 없습니다. 자동 백업을 비활성화해도 수동 DB 스냅샷에는 영향을 주지 않습니다. 기존의 모든 수동 DB 스냅샷을 계속 복원 작업에 사용할 수 있습니다.

자동 백업을 비활성화하면 로드 시간이 약 25% 단축되고 로드 중에 필요한 스토리지 공간의 양이 줄어듭니다. 아무런 데이터도 없는 새 DB 인스턴스로 데이터를 로드할 경우에는 백업을 비활성화하는 것이 로드 속도를 높이고 백업에 필요한 추가 스토리지의 사용을 피하는 손쉬운 방법입니다. 하지만 이미 데이터가 있는 DB 인스턴스로 로드할 경우도 있습니다. 그렇다면 특정 시점으로 복구를 수행할 능력을 상실하는 데 따른 영향에 비해 백업을 해제할 때의 이점을 비교합니다.

DB 인스턴스에서는 기본적으로 자동 백업이 활성화되어 있습니다(보존 기간은 하루). 자동 백업을 비활성화하기 위해 백업 보존 기간을 0으로 설정합니다. 로드 후에는 백업 보존 기간을 0이 아닌 값으로 설정하여 백업을 다시 활성화할 수 있습니다. 백업을 켜거나 끄기 위해 Amazon RDS가 DB 인스턴스를 종료했다가 다시 시작하여 MariaDB 또는 MySQL 로깅을 켜거나 끕니다.

AWS CLI `modify-db-instance` 명령을 실행하여 백업 보존 기간을 0으로 설정하고 변경 사항을 즉시 적용합니다. 보존 기간을 0으로 설정하려면 DB 인스턴스를 다시 시작해야 하므로, 다시 시작 프로세스가 완료될 때까지 기다린 후 계속 진행합니다. 자세한 내용은 modify-db-instance를 참조하세요.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier AcmeRDS \
    --apply-immediately \
    --backup-retention-period 0
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier AcmeRDS ^
    --apply-immediately ^
```

```
    --backup-retention-period 0
```

AWS CLI describe-db-instances 명령으로 DB 인스턴스의 상태를 확인할 수 있습니다. 다음 예제에서는 AcmeRDS DB 인스턴스의 DB 인스턴스 상태를 표시합니다.

```
aws rds describe-db-instances --db-instance-identifier AcmeRDS --query "*[].
{DBInstanceStatus:DBInstanceStatus}"
```

DB 인스턴스 상태가 available이면 다음 단계로 계속할 준비가 된 것입니다.

5단계: 데이터 로드

플랫 파일의 행을 데이터베이스 테이블로 읽어들이려면 MySQL LOAD DATA LOCAL INFILE 문을 사용합니다.

> **Note**
>
> 플랫 파일이 있는 위치와 동일한 위치에서 mysql 쉘을 호출하거나 LOAD DATA LOCAL INFILE을 실행할 때 파일의 절대 경로를 사용해야 합니다.

다음 예시는 이름이 sales.txt인 파일에서 데이터베이스의 Sales 테이블로 데이터를 로드하는 방법을 보여줍니다.

```
mysql> LOAD DATA LOCAL INFILE 'sales.txt' INTO TABLE Sales FIELDS TERMINATED BY ' '
 ENCLOSED BY '' ESCAPED BY '\\';
Query OK, 1 row affected (0.01 sec)
Records: 1  Deleted: 0  Skipped: 0  Warnings: 0
```

LOAD DATA 문에 대한 자세한 내용은 MySQL 설명서의 LOAD DATA Statement를 참조하세요.

6단계: Amazon RDS 자동 백업 다시 켜기

4단계에서 Amazon RDS 자동 백업을 껐다면 로드가 완료된 후 백업 보존 기간을 로드 이전의 값으로 다시 설정하여 자동 백업을 켭니다. 4단계에서 언급한 것처럼, Amazon RDS가 DB 인스턴스를 다시 시작하므로 잠시 작동이 중단되는 상황에 대비하세요.

다음 예시에서는 AWS CLI modify-db-instance 명령을 실행하여 AcmeRDS DB 인스턴스에서 자동 백업을 켜고 보존 기간을 1일로 설정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier AcmeRDS \
    --backup-retention-period 1 \
    --apply-immediately
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier AcmeRDS ^
    --backup-retention-period 1 ^
    --apply-immediately
```

Amazon RDS에서 MySQL 복제 작업

일반적으로 읽기 전용 복제본을 사용하여 Amazon RDS DB 인스턴스 간 복제를 구성합니다. 읽기 전용 복제본에 대한 일반적인 정보는 DB 인스턴스 읽기 전용 복제본 작업 단원을 참조하십시오. Amazon RDS for MySQL의 읽기 복제본 작업에 대한 자세한 내용은 MySQL 읽기 전용 복제본 작업 단원을 참조하세요.

RDS for MySQL를 통한 복제에 전역 트랜잭션 ID(GTID)를 사용할 수 있습니다. 자세한 내용은 GTID 기반 복제 사용 단원을 참조하십시오.

RDS for MySQL DB 인스턴스와 Amazon RDS 외부에 있는 MySQL 또는 MariaDB 인스턴스 간의 복제를 설정할 수도 있습니다. 외부 소스를 사용하여 복제를 구성하는 방법에 대한 자세한 내용은 외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성 단원을 참조하십시오.

이러한 복제 옵션의 경우 행 기반 복제, 문 기반 복제 또는 혼합 복제를 사용할 수 있습니다. 행 기반 복제는 SQL 문으로 인해 변경된 행만 복제합니다. 문 기반 복제는 전체 SQL 문을 복제합니다. 혼합 복제는 가능한 경우 문 기반 복제를 사용하지만, 문 기반 복제에 안전하지 않은 SQL 문이 실행될 경우 행 기반 복제로 전환합니다. 대부분의 경우 혼합 복제가 권장됩니다. DB 인스턴스의 이진 로그 형식은 복제가 행 기반인지, 문 기반인지, 혼합인지 결정합니다. 이진 로그 형식 설정에 대한 자세한 내용은 단일 AZ 데이터베이스의 RDS for MySQL 이진 로깅 구성 단원을 참조하십시오.

> **Note**
>
> Amazon RDS 외부에 있는 MySQL 또는 MariaDB 인스턴스에서 데이터베이스를 가져오거나 그런 인스턴스로 데이터베이스를 내보내도록 복제를 구성할 수 있습니다. 자세한 내용은 가동 중지 시간을 줄이면서 Amazon RDS for MySQL 데이터베이스로 데이터 가져오기 및 복제를 사용하여 MySQL DB 인스턴스에서 데이터 내보내기 단원을 참조하세요.

스냅샷에서 DB 인스턴스를 복원하거나 시점 복구를 수행한 후 RDS 콘솔의 소스 데이터베이스에서 마지막으로 복구된 binlog 위치를 볼 수 있습니다. 로그 및 이벤트에서 binlog를 입력합니다. binlog 위치가 시스템 메모 아래에 나타납니다.

주제

- MySQL 읽기 전용 복제본 작업
- GTID 기반 복제 사용
- 외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성

- Amazon RDS for MySQL용 다중 소스 복제 구성

MySQL 읽기 전용 복제본 작업

다음으로, RDS for MySQL에서의 읽기 전용 복제본 작업에 대한 특정 정보를 찾을 수 있습니다. 읽기 전용 복제본에 대한 일반적인 정보와 사용 지침은 DB 인스턴스 읽기 전용 복제본 작업 단원을 참조하십시오.

MySQL 읽기 전용 복제본에 대한 자세한 내용은 다음 주제를 참조하세요.

- MySQL을 사용한 복제 필터 구성
- MySQL을 사용한 지연 복제 구성
- MySQL을 사용한 읽기 전용 복제본 업로드
- MySQL을 사용한 다중 AZ 읽기 전용 복제본 배포 작업
- RDS for MySQL에서의 계단식 읽기 전용 복제본 사용
- MySQL 읽기 전용 복제본의 복제 지연 모니터링
- MySQL 읽기 전용 복제본을 사용한 복제 시작 및 중지
- MySQL 읽기 전용 복제본의 문제 해결

MySQL을 사용한 읽기 전용 복제본 구성

MySQL DB 인스턴스를 복제 원본으로 사용하려면 먼저 원본 DB 인스턴스에서 자동 백업을 활성화해야 합니다. 이렇게 하려면 백업 보존 기간을 0 이외의 값으로 설정합니다. 이 요구 사항은 다른 읽기 전용 복제본의 원본 DB 인스턴스인 읽기 전용 복제본에도 적용됩니다. 자동 백업은 모든 버전의 MySQL을 실행 중인 읽기 전용 복제본에서 지원됩니다. MySQL DB 인스턴스에 대해 바이너리 로그 좌표를 기반으로 복제를 구성할 수 있습니다.

다음 버전에서 전역 트랜잭션 식별자(GTID)를 사용하여 복제를 구성할 수 있습니다.

- RDS for MySQL 버전 5.7.44 이상 5.7 버전
- RDS for MySQL 버전 8.0.28 이상 8.0 버전
- RDS for MySQL 버전 8.4.3 이상 8.4 버전

자세한 내용은 GTID 기반 복제 사용 단원을 참조하십시오.

동일 리전 내의 DB 인스턴스 하나에서 최대 15개까지 읽기 전용 복제본을 생성할 수 있습니다. 효과적인 복제를 위해서는 읽기 전용 복제본도 각각 원본 DB 인스턴스와 동일한 양의 컴퓨팅 및 스토리지 리소스를 가져야 합니다. 원본 DB 인스턴스를 확장하는 경우 읽기 전용 복제본도 확장합니다.

RDS for MySQL은 계단식 읽기 전용 복제본을 지원합니다. 읽기 전용 복제본을 계단식으로 구성하는 방법을 알아보려면 RDS for MySQL에서의 계단식 읽기 전용 복제본 사용 단원을 참조하세요.

동일한 원본 DB 인스턴스를 참조하는 여러 읽기 전용 복제본 생성 및 삭제 작업을 동시에 실행할 수 있습니다. 이러한 작업을 수행할 때 각 원본 인스턴스의 읽기 전용 복제본 한도 15개를 넘지 않아야 합니다.

MySQL DB 인스턴스의 읽기 전용 복제본은 소스 DB 인스턴스보다 낮은 DB 엔진 버전을 사용할 수 없습니다.

MyISAM을 사용하는 MySQL DB 인스턴스 준비

MySQL DB 인스턴스가 MyISAM 같은 비트랜잭션 엔진을 사용할 경우 다음 단계를 수행하여 읽기 전용 복제본을 성공적으로 설정해야 합니다. 이러한 단계는 읽기 전용 복제본에 데이터의 일관성 있는 복사본이 포함되도록 하기 위해 필요합니다. 그러나 모든 테이블이 InnoDB와 같은 트랜잭션 엔진을 사용하는 경우에는 이 단계가 필요 없습니다.

1. 원본 DB 인스턴스에서 비트랜잭션 테이블의 모든 데이터 조작 언어(DML) 및 데이터 정의 언어(DDL) 작업을 중지하고 완료될 때까지 기다립니다. SELECT 문은 계속해서 실행할 수 있습니다.
2. 원본 DB 인스턴스의 테이블을 플러시한 후 잠급니다.
3. 다음 단원의 방법 중 하나를 사용하여 읽기 전용 복제본을 생성합니다.
4. 예를 들어, `DescribeDBInstances` API 작업 등을 사용하여 읽기 전용 복제본 생성 진행률을 확인합니다. 읽기 전용 복제본을 사용할 수 있게 되면 원본 DB 인스턴스의 테이블 잠금을 해제하고 정상적인 데이터베이스 작업을 재개합니다.

MySQL을 사용한 복제 필터 구성

복제 필터를 사용하여 읽기 복제본과 함께 복제할 데이터베이스와 테이블을 지정할 수 있습니다. 복제 필터는 데이터베이스와 테이블을 복제에 포함하거나 복제에서 제외할 수 있습니다.

다음은 복제 필터의 몇 가지 사용 사례입니다.

- 읽기 복제본의 크기를 줄이는 방법. 복제 필터링을 사용하면 읽기 복제본에 필요하지 않은 데이터베이스와 테이블을 제외할 수 있습니다.

- 보안상의 이유로 읽기 복제본에서 데이터베이스와 테이블을 제외하는 방법.
- 다른 읽기 복제본에서 특정 사용 사례에 대해 서로 다른 데이터베이스와 테이블을 복제하는 방법. 예를 들어 분석 또는 샤딩에 특정 읽기 복제본을 사용할 수 있습니다.
- 여러 AWS 리전에 읽기 전용 복제본이 있는 DB 인스턴스의 경우 서로 다른 AWS 리전에 있는 다른 데이터베이스 또는 테이블을 복제합니다.

> **Note**
>
> 복제 필터를 사용하여 인바운드 복제 토폴로지에서 복제본으로 구성된 기본 MySQL DB 인스턴스에서 복제할 데이터베이스와 테이블을 지정할 수도 있습니다. 이 구성에 대한 자세한 내용은 외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성 단원을 참조하십시오.

주제

- RDS for MySQL에 대한 복제 필터링 파라미터 설정
- RDS for MySQL에 대한 복제 필터링 제한 사항
- RDS for MySQL에 대한 복제 필터링 예제
- 읽기 복제본에 대한 복제 필터 보기

RDS for MySQL에 대한 복제 필터링 파라미터 설정

복제 필터를 구성하려면 읽기 복제본에서 다음 복제 필터링 파라미터를 설정합니다.

- `replicate-do-db` – 지정된 데이터베이스에 변경 내용을 복제합니다. 읽기 복제본에 대해 이 파라미터를 설정하면 파라미터에 지정된 데이터베이스만 복제됩니다.
- `replicate-ignore-db` – 지정된 데이터베이스에 변경 내용을 복제하지 마세요. 읽기 복제본에 대해 `replicate-do-db` 파라미터가 설정되면 이 파라미터는 평가되지 않습니다.
- `replicate-do-table` – 지정된 테이블에 변경 사항을 복제합니다. 읽기 복제본에 대해 이 파라미터를 설정하면 파라미터에 지정된 테이블만 복제됩니다. 또한, `replicate-do-db` 또는 `replicate-ignore-db` 파라미터가 설정되면 복제에 지정된 테이블이 포함된 데이터베이스를 읽기 복제본과 함께 포함해야 합니다.
- `replicate-ignore-table` – 지정된 테이블에 변경 내용을 복제하지 마세요. 읽기 복제본에 대해 `replicate-do-table` 파라미터가 설정되면 이 파라미터는 평가되지 않습니다.
- `replicate-wild-do-table` – 지정된 데이터베이스와 테이블 이름 패턴을 기반으로 테이블을 복제합니다. `%` 및 `_` 와일드카드 문자가 지원됩니다. `replicate-do-db` 또는 `replicate-ignore-`

db 파라미터가 설정되면 복제에 지정된 테이블이 포함된 데이터베이스를 읽기 복제본과 함께 포함해야 합니다.

- `replicate-wild-ignore-table` – 지정된 데이터베이스와 테이블 이름 패턴을 기반으로 테이블을 복제하지 마세요 `%` 및 `_` 와일드카드 문자가 지원됩니다. 읽기 복제본에 대해 `replicate-do-table` 또는 `replicate-wild-do-table` 파라미터가 설정되면 이 파라미터는 평가되지 않습니다.

파라미터는 나열된 순서대로 평가됩니다. 이러한 파라미터의 작동 방식에 대한 자세한 내용은 MySQL 설명서를 참조하세요.

- 일반적인 내용은 복제 서버 옵션 및 변수를 참조하세요.
- 데이터베이스 복제 필터링 파라미터를 평가하는 방법에 대한 자세한 내용은 데이터베이스 수준의 복제 및 이진 로깅 옵션 평가를 참조하세요.
- 테이블 복제 필터링 파라미터가 평가되는 방법에 대한 자세한 내용은 테이블 수준의 복제 옵션 평가를 참조하세요.

기본적으로 이러한 각 파라미터에는 빈 값이 있습니다. 각 읽기 복제본에서 이러한 파라미터를 사용하여 복제 필터를 설정, 변경 및 삭제할 수 있습니다. 이러한 파라미터 중 하나를 설정할 때 각 필터를 쉼표로 구분합니다.

`%` 및 `_` 파라미터에 `replicate-wild-do-table` 및 `replicate-wild-ignore-table` 와일드카드 문자를 사용할 수 있습니다. `%` 와일드카드는 원하는 수의 문자를 찾으며 `_` 와일드카드는 한 문자만 찾습니다.

원본 DB 인스턴스의 이진 로깅 형식은 데이터 변경 기록을 결정하므로 복제에 중요합니다. `binlog_format` 파라미터 설정에 따라 복제가 행 기반인지 또는 문 기반인지가 결정됩니다. 자세한 내용은 단일 AZ 데이터베이스의 RDS for MySQL 이진 로깅 구성 섹션을 참조하세요.

> **Note**
>
> 원본 DB 인스턴스의 `binlog_format` 설정과 관계없이, 모든 DDL(데이터 정의어) 문은 문으로 복제됩니다.

RDS for MySQL에 대한 복제 필터링 제한 사항

RDS for MySQL에 대한 복제 필터링에 다음과 같은 제한 사항이 적용됩니다.

- 각 복제 필터링 파라미터에는 2,000자 제한이 있습니다.
- 파리미터 값의 복제 필터에서는 쉼표가 지원되지 않습니다. 파라미터 목록에서 쉼표는 값 구분자로만 사용할 수 있습니다. 예를 들어, `ParameterValue='a,b'`는 지원되지만 `ParameterValue='`a,b`'`는 지원되지 않습니다.
- 이진 로그 필터링에 대해서는 MySQL `--binlog-do-db` 및 `--binlog-ignore-db` 옵션이 지원되지 않습니다.
- 복제 필터링은 XA 트랜잭션을 지원하지 않습니다.

  자세한 내용은 MySQL 설명서에서 XA 트랜잭션에 대한 제한 사항을 참조하세요.

RDS for MySQL에 대한 복제 필터링 예제

읽기 복제본에 대한 복제 필터링을 구성하려면 읽기 복제본과 연결된 파라미터 그룹에서 복제 필터링 파라미터를 수정합니다.

> **Note**
>
> 기본 파라미터 그룹을 수정할 수 없습니다. 읽기 복제본에서 기본 파라미터 그룹을 사용 중인 경우 새 파라미터 그룹을 생성하여 읽기 복제본과 연결합니다. DB 파라미터 그룹에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 파라미터 그룹에서 파라미터를 설정할 수 있습니다. 파라미터 설정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요. 파라미터 그룹에서 파라미터를 설정하면 파라미터 그룹과 연결된 모든 DB 인스턴스가 파라미터 설정을 사용합니다. 파라미터 그룹에서 복제 필터링 파라미터를 설정하는 경우, 파라미터 그룹이 읽기 복제본에만 연결되어 있는지 확인합니다. 원본 DB 인스턴스에 대해 복제 필터링 파라미터를 비워 둡니다.

다음 예제에서는 AWS CLI를 사용하여 파라미터를 설정합니다. 이 예제는 CLI 명령이 완료된 직후에 파라미터가 변경되도록 `ApplyMethod`을(를) `immediate`(으)로 설정합니다. 읽기 복제본이 재부팅된 후 보류 중인 변경 사항을 적용하려면 `ApplyMethod`을(를) `pending-reboot`(으)로 설정합니다.

다음 예제에서는 복제 필터를 설정합니다.

- Including databases in replication
- Including tables in replication
- Including tables in replication with wildcard characters

- Excluding databases from replication
- Excluding tables from replication
- Excluding tables from replication using wildcard characters

Example 복제에 데이터베이스 포함

다음 예제에서는 복제에 mydb1 및 mydb2 데이터베이스가 포함되어 있습니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-do-
db,ParameterValue='mydb1,mydb2',ApplyMethod=immediate"
```

Windows의 경우:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-do-
db,ParameterValue='mydb1,mydb2',ApplyMethod=immediate"
```

Example 복제에 테이블 포함

다음 예제에서는 복제의 table1 데이터베이스에 table2 및 mydb1 테이블이 포함되어 있습니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-do-
table,ParameterValue='mydb1.table1,mydb1.table2',ApplyMethod=immediate"
```

Windows의 경우:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
```

```
  --parameters "ParameterName=replicate-do-
table,ParameterValue='mydb1.table1,mydb1.table2',ApplyMethod=immediate"
```

Example 와일드카드 문자를 사용하여 복제에 테이블 포함

다음 예제에서는 복제의 데이터베이스 `order`에 `return` 및 `mydb`(으)로 시작하는 이름을 가진 테이블이 포함되어 있습니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-wild-do-table,ParameterValue='mydb.order
%,mydb.return%',ApplyMethod=immediate"
```

Windows의 경우:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-wild-do-table,ParameterValue='mydb.order
%,mydb.return%',ApplyMethod=immediate"
```

Example 복제에서 데이터베이스 제외

다음 예제에서는 `mydb5` 및 `mydb6` 데이터베이스를 복제에서 제외합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-ignore-
db,ParameterValue='mydb5,mydb6',ApplyMethod=immediate"
```

Windows의 경우:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-ignore-
db,ParameterValue='mydb5,mydb6',ApplyMethod=immediate"
```

Example 복제에서 테이블 제외

다음 예시에서는 데이터베이스 mydb5의 table1 테이블과 데이터베이스 mydb6의 table2 테이블을 복제에서 제외합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-ignore-
table,ParameterValue='mydb5.table1,mydb6.table2',ApplyMethod=immediate"
```

Windows의 경우:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-ignore-
table,ParameterValue='mydb5.table1,mydb6.table2',ApplyMethod=immediate"
```

Example 와일드카드 문자를 사용하여 복제에서 테이블 제외

다음 예제에서는 데이터베이스 order에서 return 및 mydb7(으)로 시작하는 이름을 가진 테이블을 복제에서 제외합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-wild-ignore-table,ParameterValue='mydb7.order
%,mydb7.return%',ApplyMethod=immediate"
```

Windows의 경우:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-wild-ignore-table,ParameterValue='mydb7.order
%,mydb7.return%',ApplyMethod=immediate"
```

읽기 복제본에 대한 복제 필터 보기

다음과 같은 방법으로 읽기 복제본에 대한 복제 필터를 볼 수 있습니다.

- 읽기 복제본과 연결된 파라미터 그룹에서 복제 필터링 파라미터 설정을 확인합니다.

  지침은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 값 보기 섹션을 참조하세요.
- MySQL 클라이언트에서 읽기 전용 복제본에 연결하고 `SHOW REPLICA STATUS` 문을 실행합니다.

  출력 시, 다음 필드에서 읽기 복제본에 대한 복제 필터를 보여줍니다.

  - `Replicate_Do_DB`
  - `Replicate_Ignore_DB`
  - `Replicate_Do_Table`
  - `Replicate_Ignore_Table`
  - `Replicate_Wild_Do_Table`
  - `Replicate_Wild_Ignore_Table`

  이러한 필드에 대한 자세한 내용은 MySQL 설명서의 복제 상태 확인을 참조하세요.

MySQL을 사용한 지연 복제 구성

지연 복제를 재해 복구를 위한 전략으로 사용할 수 있습니다. 지연된 복제를 사용하여 원본에서 읽기 전용 복제본으로의 복제를 지연할 최소 시간(초)을 지정합니다. 재해 발생 시(예: 실수로 테이블 삭제) 다음 단계를 완료하여 재해로부터 빠르게 복구할 수 있습니다.

- 재해를 일으킨 변경 사항이 읽기 전용 복제본으로 전송되기 이전에 읽기 전용 복제본에 대한 복제를 중지합니다.

  mysql.rds_stop_replication 저장 프로시저를 사용하여 복제를 중지합니다.
- 복제를 시작하고 로그 파일 위치에서 복제가 자동으로 중지되도록 지정합니다.

  mysql.rds_start_replication_until 저장 프로시저를 사용하여 재해 직전 위치를 지정합니다.
- 읽기 전용 복제본을 독립 DB 인스턴스로 승격의 지침에 따라 읽기 전용 복제본을 새 원본 DB 인스턴스로 승격합니다.

> **Note**
>
> - RDS for MySQL 8.4에서는 지연 복제가 MySQL 8.4.3 이상에 대해 지원됩니다. RDS for MySQL 8.0에서는 지연 복제가 MySQL 8.0.28 이상에 대해 지원됩니다. RDS for MySQL 5.7에서는 지연 복제가 MySQL 5.7.44 이상에 대해 지원됩니다.

- 저장 프로시저를 사용하여 지연된 복제를 구성합니다. AWS Management Console, AWS CLI 또는 Amazon RDS API를 사용하여 지연 복제를 구성할 수 없습니다.
- 다음 버전에 대한 지연된 복제 구성에서 전역 트랜잭션 ID(GTID)를 기반으로 하는 복제를 사용할 수 있습니다.
  - RDS for MySQL 버전 5.7.44 이상 5.7 버전
  - RDS for MySQL 버전 8.0.28 이상 8.0 버전
  - RDS for MySQL 버전 8.4.3 이상 8.4 버전

  GTID 기반 복제를 사용하는 경우 mysql.rds_start_replication_until_gtid 저장 프로시저 대신 mysql.rds_start_replication_until 저장 프로시저를 사용하십시오. GTID 기반 복제에 대한 자세한 내용은 GTID 기반 복제 사용 단원을 참조하십시오.

주제

- 읽기 전용 복제본 생성 중 지연 복제 구성
- 기존 읽기 전용 복제본에 대한 지연 복제 수정
- 읽기 전용 복제본에 대한 복제를 중지할 위치 설정
- 읽기 전용 복제본 승격

읽기 전용 복제본 생성 중 지연 복제 구성

DB 인스턴스에서 향후에 생성되는 읽기 전용 복제본에 대한 지연된 복제를 구성하려면 mysql.rds_set_configuration 파라미터와 함께 `target delay` 저장 프로시저를 실행합니다.

읽기 전용 복제본을 생성하는 동안 지연된 복제를 구성하려면

1. MySQL 클라이언트를 사용하여 마스터 사용자로 읽기 전용 복제본에 대한 원본이 될 MySQL DB 인스턴스에 연결합니다.
2. mysql.rds_set_configuration 파라미터와 함께 `target delay` 저장 프로시저를 실행합니다.

   예를 들어, 현재 DB 인스턴스에서 생성되는 모든 읽기 전용 복제본에 대해 1시간(3,600초) 이상 복제를 지연하도록 지정하려면 다음 저장 프로시저를 실행합니다.

   ```
   call mysql.rds_set_configuration('target delay', 3600);
   ```

> **Note**
>
> 이 저장 프로시저를 실행한 후 AWS CLI 또는 Amazon RDS API를 사용하여 생성하는 모든 읽기 전용 복제본은 지정된 시간(초)만큼 복제를 지연하도록 구성됩니다.

기존 읽기 전용 복제본에 대한 지연 복제 수정

기존 읽기 전용 복제본에 대한 지연된 복제를 수정하려면 mysql.rds_set_source_delay 저장 프로시저를 실행합니다.

기존 읽기 전용 복제본에 대한 지연된 복제를 수정하려면

1. MySQL 클라이언트를 사용하여 마스터 사용자로 읽기 전용 복제본에 연결합니다.
2. mysql.rds_stop_replication 저장 프로시저를 사용하여 복제를 중지합니다.
3. mysql.rds_set_source_delay 저장 프로시저를 실행합니다.

   예를 들어, 읽기 전용 복제본에 대한 복제가 1시간(3,600초) 이상 지연되도록 지정하려면 다음 저장 프로시저를 실행합니다.

   ```
   call mysql.rds_set_source_delay(3600);
   ```
4. mysql.rds_start_replication 저장 프로시저를 사용하여 복제를 시작합니다.

읽기 전용 복제본에 대한 복제를 중지할 위치 설정

읽기 전용 복제본에 대한 복제를 중단한 이후에 mysql.rds_start_replication_until 저장 프로시저를 사용하여 복제를 시작한 다음 지정된 이진 로그 파일 위치에서 복제를 중지할 수 있습니다.

읽기 전용 복제본에 대한 복제를 시작하고 특정 위치에서 복제를 중지하려면

1. MySQL 클라이언트를 사용하여 원본 MySQL DB 인스턴스에 마스터 사용자로 연결합니다.
2. mysql.rds_start_replication_until 저장 프로시저를 실행합니다.

   다음 예제에서는 복제를 시작하고 `120` 바이너리 로그 파일의 `mysql-bin-changelog.000777` 위치에 도달할 때까지 변경 사항을 복제합니다. 재해 복구 시나리오에서 `120`이 재해 직전 위치라고 가정합니다.

```
call mysql.rds_start_replication_until(
  'mysql-bin-changelog.000777',
  120);
```

중지 지점에 도달하면 복제가 자동으로 중지됩니다. `Replication has been stopped since the replica reached the stop point specified by the rds_start_replication_until stored procedure` RDS 이벤트가 생성됩니다.

읽기 전용 복제본 승격

복제가 중지된 후 재해 복구 시나리오에서 읽기 전용 복제본을 새 원본 DB 인스턴스로 승격할 수 있습니다. 읽기 전용 복제본 승격에 대한 자세한 내용은 읽기 전용 복제본을 독립 DB 인스턴스로 승격 단원을 참조하십시오.

MySQL을 사용한 읽기 전용 복제본 업로드

읽기 전용 복제본은 읽기 쿼리를 지원하도록 설계되었지만 경우에 따라 업데이트가 필요할 수 있습니다. 예를 들어 특정 유형의 쿼리가 복제본에 대한 액세스를 최저거화하기 위해 인덱스를 추가해야 할 경우가 있습니다.

읽기 전용 복제본의 DB 파라미터 그룹에서 `read_only` 파라미터를 `0`으로 설정하여 업데이트를 활성화할 수 있지만, 읽기 전용 복제본이 소스 DB 인스턴스와 호환되지 않는 경우 문제가 발생할 수 있으므로 그렇게 하지 않는 것이 좋습니다. 유지 관리 작업의 경우 블루/그린 배포를 사용하는 것이 좋습니다. 자세한 내용은 데이터베이스 업데이트에 블루/그린 배포 사용 단원을 참조하십시오.

읽기 전용 복제본에서 읽기 전용을 비활성화하는 경우 가능한 빨리 `read_only` 파라미터의 값을 `1`로 되돌려 변경합니다.

MySQL을 사용한 다중 AZ 읽기 전용 복제본 배포 작업

단일 AZ 또는 다중 AZ DB 인스턴스 배포를 통해 읽기 전용 복제본을 생성할 수 있습니다. 다중 AZ 배포는 중요 데이터의 내구성과 가용성을 개선하는 데 효과적이지만 읽기 전용 쿼리를 실행하는 데 보조로 사용할 수는 없습니다. 대신 트래픽이 많은 다중 AZ DB 인스턴스에서 읽기 전용 쿼리를 오프로드할 목적으로 읽기 전용 복제본을 생성할 수 있습니다. 다중 AZ 배포의 원본 인스턴스가 보조 인스턴스로 장애 조치되는 경우 연결된 모든 읽기 전용 복제본이 자동으로 전환되어 보조(이제는 기본) 인스턴스를 복제 원본으로 사용합니다. 자세한 내용은 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 섹션을 참조하세요.

다중 AZ DB 인스턴스에서 읽기 전용 복제본을 생성할 수 있습니다. Amazon RDS는 복제본에 대한 장애 조치 지원을 위해 대기 복제본을 다른 가용 영역에 생성합니다. 읽기 전용 복제본을 다중 AZ DB 인스턴스로 생성하는 작업은 원본 데이터베이스가 다중 AZ DB 인스턴스인지 여부와는 무관합니다.

RDS for MySQL에서의 계단식 읽기 전용 복제본 사용

RDS for MySQL은 계단식 읽기 전용 복제본을 지원합니다. 계단식 읽기 전용 복제본을 사용하면 소스 RDS for MySQL DB 인스턴스에 오버헤드를 추가하지 않고도 읽기 전용 복제본 크기를 조정할 수 있습니다.

계단식 읽기 전용 복제본을 사용하면 RDS for MySQL DB 인스턴스가 데이터를 체인의 첫 번째 읽기 전용 복제본으로 전송합니다. 뒤이어 해당 읽기 전용 복제본이 데이터를 체인의 두 번째 복제본으로 전송하는 식으로 이루어집니다. 결과적으로 체인의 모든 읽기 전용 복제본이 소스 DB 인스턴스에만 오버헤드가 발생하는 일 없이 RDS for MySQL DB 인스턴스에서 변경됩니다.

소스 RDS for MyQL DB 인스턴스에서 체인에 최대 3개의 읽기 전용 복제본을 생성할 수 있습니다. 예를 들어 RDS MySQL DB 인스턴스, `mysql-main`이 있다고 가정해봅니다. 다음을 수행할 수 있습니다.

- `mysql-main`부터 시작해서 체인에 첫 번째 읽기 전용 복제본 `read-replica-1`을 생성합니다.
- 다음으로 `read-replica-1`에서 체인에 다음 읽기 전용 복제본 `read-replica-2`를 생성합니다.
- 마지막으로 `read-replica-2`에서 체인에 세 번째 읽기 전용 복제본 `read-replica-3`을 생성합니다.

체인에서 `mysql-main`에 대한 세 번째 계단식 읽기 전용 복제본 다음으로 또 다른 읽기 전용 복제본을 생성할 수 없습니다. RDS for MySQL 소스 DB 인스턴스부터 계단식 읽기 전용 복제본 체인의 마지막에 이르는 전체 인스턴스는 최대 4개의 DB 인스턴스로 구성될 수 있습니다.

읽기 전용 복제본을 계단식으로 실행하려면 각 소스 RDS for MySQL DB 인스턴스에 자동 백업이 켜져 있어야 합니다. 읽기 전용 복제본에서 자동 백업을 켜려면 먼저 읽기 전용 복제본을 생성한 다음 자동 백업이 켜지도록 읽기 전용 복제본을 수정합니다. 자세한 내용은 읽기 전용 복제본 생성 단원을 참조하십시오.

모든 읽기 전용 복제본과 마찬가지로 계단식 구성에 포함된 읽기 전용 복제본을 승격할 수 있습니다. 읽기 전용 복제본 체인의 한 읽기 전용 복제본을 승격하면 체인에서 해당 복제본이 제거됩니다. 예를 들어 `mysql-main` DB 인스턴스의 일부 워크로드를 회계 부서에서만 사용할 수 있도록 새 인스턴스로 옮기려고 합니다. 이 예제에서 3개의 읽기 전용 복제본 체인이 있다고 가정하고 `read-replica-2`를 승격하기로 결정합니다. 체인은 다음과 같이 변화합니다.

- `read-replica-2`를 승격하면 복제 체인에서 제거됩니다.

  - 이제 전체 읽기/쓰기 DB 인스턴스가 됩니다.
  - 승격 전과 마찬가지로 `read-replica-3`으로 계속 복제합니다.
- `mysql-main`은 `read-replica-1`로 계속 복제를 진행합니다.

읽기 전용 복제본 승격에 대한 자세한 내용은 읽기 전용 복제본을 독립 DB 인스턴스로 승격 섹션을 참조하세요.

MySQL 읽기 전용 복제본의 복제 지연 모니터링

MySQL 읽기 전용 복제본의 경우 Amazon RDS `ReplicaLag` 지표를 보면서 Amazon CloudWatch의 복제 지연을 모니터링할 수 있습니다. `ReplicaLag` 메트릭은 `Seconds_Behind_Master` 명령의 `SHOW REPLICA STATUS` 필드의 값을 보고합니다.

이렇게 MySQL에서 복제 지연이 발생하는 공통 원인은 다음과 같습니다.

- 네트워크 중단.
- 읽기 전용 복제본에 대한 서로 다른 인덱스를 가진 테이블에 쓰기 작업 중일 때. 읽기 전용 복제본에 `read_only` 파라미터가 `0`으로 설정된 경우 읽기 전용 복제본이 소스 DB 인스턴스와 호환되지 않으면 복제가 중단될 수 있습니다. 읽기 전용 복제본에 대한 유지 관리 작업을 수행한 후에는 `read_only` 파라미터를 다시 `1`로 설정하는 것이 좋습니다.
- MyISAM과 같은 비트랜잭션 스토리지 엔진 사용. 복제는 MySQL의 InnoDB 스토리지 엔진에서만 지원됩니다.

`ReplicaLag` 지표가 0에 도달하면 복제본이 원본 DB 인스턴스를 따라잡은 것입니다. `ReplicaLag` 지표가 -1을 반환하는 경우 복제가 현재 활성이 아닙니다. `ReplicaLag` = -1은 `Seconds_Behind_Master` = `NULL`과 동등합니다.

MySQL 읽기 전용 복제본을 사용한 복제 시작 및 중지

Amazon RDS DB 인스턴스에서는 시스템에 저장된 프로시저인 mysql.rds_stop_replication과(와) mysql.rds_start_replication을 호출하여 복제 프로세스를 종료하거나 재시작할 수 있습니다. 대용량 인덱스를 생성하는 등 오랜 시간이 걸리는 작업에서 두 Amazon RDS 인스턴스를 서로 복제할 때도 이런 방법이 가능합니다. 또한 데이터베이스를 가져오거나 내보낼 때도 복제를 종료하거나 시작할 필요가 있습니다. 자세한 내용은 가동 중지 시간을 줄이면서 Amazon RDS for MySQL 데이터베이스로 데이터 가져오기 및 복제를 사용하여 MySQL DB 인스턴스에서 데이터 내보내기 단원을 참조하십시오.

수동으로 또는 복제 오류로 인해 연속하여 30일 이상 복제가 중단된 경우에는 Amazon RDS가 원본 DB 인스턴스와 모든 읽기 전용 복제본 사이의 복제를 종료합니다. 이렇게 하는 이유는 소스 DB 인스턴

스에 대한 스토리지 요건 증가와 장애 조치의 장기화를 방지하기 위해서입니다. 읽기 전용 복제본 DB 인스턴스는 계속 사용할 수 있습니다. 그러나 복제가 종료된 후 읽기 전용 복제본에 필요한 이진 로그가 원본 DB 인스턴스에서 삭제되므로 복제를 재개할 수 없습니다. 원본 DB 인스턴스에서 복제를 재설정하려면 새 읽기 전용 복제본을 생성해야 합니다.

MySQL 읽기 전용 복제본의 문제 해결

MySQL DB 인스턴스의 경우 읽기 전용 복제본이 읽기 전용 복제본과 원본 DB 인스턴스 사이에 복제 오류, 데이터 불일치 또는 둘 다를 나타내는 경우가 있습니다. 이 문제는 읽기 전용 복제본 또는 원본 DB 인스턴스에 오류가 발생하는 동안 일부 이진 로그(binlog) 이벤트 또는 InnoDB 다시 실행 로그가 플러시되지 않는 경우에 발생합니다. 이러한 경우 수동으로 읽기 전용 복제본을 삭제한 후 재생성해야 합니다. `sync_binlog=1` 및 `innodb_flush_log_at_trx_commit=1` 파라미터 값을 설정하여 발생하는 이러한 가능성을 줄일 수 있습니다. 단, 이 설정은 성능 감소의 원인이 될 수도 있으므로 변경 사항을 프로덕션 환경에 적용하기 전에 그 효과를 테스트하는 것이 좋습니다.

> **Warning**
>
> 원본 DB 인스턴스와 연결된 파라미터 그룹에서 `sync_binlog=1` 및 `innodb_flush_log_at_trx_commit=1` 파라미터 값을 유지하는 것이 좋습니다. 이러한 파라미터는 동적입니다. 이러한 설정을 사용하지 않으려면 원본 DB 인스턴스에서 이를 다시 시작하게 할 수 있는 작업을 실행하기 전에 해당 값을 임시로 설정하는 것이 좋습니다. 이러한 작업에는 재부팅, 장애 조치를 통한 재부팅, 데이터베이스 버전 업그레이드, DB 인스턴스 클래스 또는 해당 스토리지 변경이 포함되지만 이에 국한되지 않습니다. 원본 DB 인스턴스에 대한 읽기 전용 복제본을 새로 생성할 때도 동일한 권장 사항이 적용됩니다.
>
> 이 지침을 따르지 않으면 읽기 전용 복제본이 읽기 전용 복제본과 원본 DB 인스턴스 간에 복제 오류, 데이터 불일치 또는 둘 다 발생할 위험이 높아집니다.

MySQL의 복제본 기술은 비동기적입니다. 간혹 원본 DB 인스턴스에서 `BinLogDiskUsage`가 증가하면 읽기 전용 복제본의 `ReplicaLag`가 예상되는 이유도 비동기식이기 때문입니다. 예를 들어 원본 DB 인스턴스에 대해 대량의 쓰기 작업이 동시에 발생할 수 있습니다. 반대로 읽기 전용 복제본에 대한 쓰기 작업은 단일 I/O 스레드를 사용하기 때문에 연이어 차례로 발생합니다. 이로 인해 원본 인스턴스와 읽기 전용 복제본 사이에 지연 시간이 있기 마련입니다. 읽기 전용 복제본에 대한 자세한 내용은 MySQL 설명서의 복제 구현 세부 정보를 참조하십시오.

원본 DB 인스턴스와 뒤이어 일어나는 읽기 전용 복제본의 업데이트 간 지연 시간을 줄일 수 있는 방법에는 다음과 같이 몇 가지가 있습니다.

- 읽기 전용 복제본의 크기를 조정하여 원본 DB 인스턴스에 버금가는 스토리지 크기와 DB 인스턴스 클래스를 할당합니다.
- 원본 DB 인스턴스와 읽기 전용 복제본에 사용되는 DB 파라미터 그룹의 파라미터 설정이 서로 호환되는지 확인합니다. 자세한 정보와 예는 이번 섹션 후반의 `max_allowed_packet` 파라미터 관련 설명을 참조하십시오.

Amazon RDS는 읽기 전용 복제본의 복제 상태를 모니터링하고, 어떤 이유로든 복제가 중지되는 경우 읽기 전용 복제본 인스턴스의 `Replication State` 필드를 `Error`로 업데이트합니다. 읽기 전용 복제본에서 실행되는 DML 쿼리가 원본 DB 인스턴스의 업데이트와 충돌하는 경우가 한 예가 될 수 있습니다.

MySQL 엔진에서 발생하는 관련 오류에 대한 세부 정보는 `Replication Error`필드에서 다시 확인할 수 있습니다. RDS-EVENT-0045, RDS-EVENT-0046 및 RDS-EVENT-0047을 포함하여 읽기 전용 복제본의 상태를 나타내는 이벤트도 생성됩니다. 이벤트와 이벤트 구독에 대한 자세한 내용은 Amazon RDS 이벤트 알림 작업 단원을 참조하십시오. MySQL 오류 메시지가 반환되는 경우에는 MySQL 오류 메시지 문서에서 오류 번호를 검토하십시오.

복제 오류의 원인이 되는 공통적인 문제를 하나 꼽으라고 하면 읽기 전용 복제본의 `max_allowed_packet` 파라미터 값이 원본 DB 인스턴스의 `max_allowed_packet` 파라미터 값보다 작을 때입니다. `max_allowed_packet` 파라미터는 DB 파라미터 그룹에서 설정할 수 있는 사용자 지정 파라미터입니다. `max_allowed_packet`을 사용하여 데이터베이스에서 실행할 수 있는 DML 코드의 최대 크기를 지정합니다. 경우에 따라 읽기 전용 복제본과 연결된 DB 파라미터 그룹의 `max_allowed_packet` 값이 원본 DB 인스턴스와 연결된 DB 파라미터 그룹의 `max_allowed_packet` 값보다 작습니다. 이러한 경우 복제 프로세스에서 오류(`Packet bigger than 'max_allowed_packet' bytes`)가 발생하여 복제가 중지될 수도 있습니다. 이 오류를 해결하려면 원본 DB 인스턴스와 읽기 전용 복제본이 동일한 `max_allowed_packet` 파라미터 값을 가진 DB 파라미터 그룹을 사용하도록 합니다.

이밖에 복제 오류의 원인이 되는 공통적인 상황은 다음과 같습니다.

- 읽기 전용 복제본의 테이블에 쓰기 작업 중일 때. 경우에 따라 원본 DB 인스턴스의 인덱스와는 다른 읽기 전용 복제본의 인덱스를 생성할 수 있습니다. 이 경우 `read_only` 파라미터를 `0`으로 설정하여 인덱스를 생성합니다. 읽기 전용 복제본의 테이블에 쓰는 경우 읽기 전용 복제본이 원본 DB 인스턴스와 호환되지 않으면 복제가 중단될 수 있습니다. 읽기 전용 복제본에 대한 유지 관리 작업을 수행한 후에는 `read_only` 파라미터를 다시 `1`로 설정하는 것이 좋습니다.
- MyISAM 같은 비트랜잭션 스토리지 엔진을 사용할 때. 읽기 전용 복제본에는 트랜잭션 스토리지 엔진이 필요합니다. 복제는 MySQL의 InnoDB 스토리지 엔진에서만 지원됩니다.

- SYSDATE()와 같이 안전하지 않은 비결정적 쿼리를 사용하는 경우. 자세한 내용은 바이너리 로깅에서 안전한 문과 안전하지 않은 문 결정을 참조하세요.

오류를 건너뛰어도 안전하다고 판단될 경우에는 RDS for MySQL에 대한 현재 복제 오류 건너뛰기 섹션에 설명한 단계를 따르십시오. 아니면 먼저 읽기 전용 복제본을 삭제할 수 있습니다. 그런 다음 엔드포인트가 이전 읽기 전용 복제본의 엔드포인트와 동일하게 유지되도록 동일한 DB 인스턴스 식별자를 사용하여 인스턴스를 생성합니다. 복제 오류가 해결되면 `Replication State`가 replicating으로 변경됩니다.

GTID 기반 복제 사용

아래 내용에서는 Amazon RDS for MySQL DB 인스턴스 간 이진 로그(binlog) 복제를 통해 전역 트랜잭션 식별자(GTID)를 사용하는 방법이 나와 있습니다.

binlog 복제를 사용하고 있지만 MySQL을 사용한 GTID 기반 복제에 대해 잘 알지 못하는 경우 MySQL 설명서의 전역 트랜잭션 식별자를 사용한 복제를 참조하세요.

GTID 기반 복제는 다음 버전에서 지원됩니다.

- 모든 RDS for MySQL 8.4 버전
- 모든 RDS for MySQL 8.0 버전
- 모든 RDS for MySQL 5.7 버전

복제 구성의 모든 MySQL DB 인스턴스가 이 버전 요구 사항을 충족해야 합니다.

주제

- 전역 트랜잭션 식별자(GTID) 개요
- GTID 기반 복제 파라미터
- RDS for MySQL용 새 읽기 전용 복제본에 대한 GTID 기반 복제 활성화
- RDS for MySQL용 기존 읽기 전용 복제본에 대한 GTID 기반 복제 활성화
- 읽기 전용 복제본이 포함된 MySQL DB 인스턴스에 대해 GTID 기반 복제 비활성화

전역 트랜잭션 식별자(GTID) 개요

전역 트랜잭션 ID(GTIDs) are unique identifiers generated for committed MySQL transactions. GTID를 사용해 binlog 복제 관련 문제를 더 간편하게 해결할 수 있습니다.

MySQL은 binlog 복제에 다음 두 가지 유형의 트랜잭션을 사용합니다.

- GTID 트랜잭션 – GTID로 식별되는 트랜잭션.
- 익명 트랜잭션 – GTID가 할당되지 않은 트랜잭션.

복제 구성의 GTID는 모든 DB 인스턴스에서 고유합니다. GTID를 사용하면 로그 파일 위치를 참조할 필요가 없기 때문에 복제 구성이 간편해집니다. 또한 GTID를 사용하면 복제된 트랜잭션을 추적하고 소스 인스턴스 및 복제본이 일치하는지를 쉽게 확인할 수 있습니다.

GTID 기반 복제를 사용하여 RDS for MySQL 읽기 전용 복제본으로 데이터를 복제할 수 있습니다. 읽기 전용 복제본을 새로 생성할 때 GTID 기반 복제를 구성할 수 있습니다. 또는 기존 읽기 전용 복제본을 GTID 기반 복제를 사용하도록 변환할 수 있습니다.

RDS for MySQL을 사용하여 지연된 복제 구성에서 GTID 기반 복제를 사용할 수도 있습니다. 자세한 정보는 MySQL을 사용한 지연 복제 구성을 참조하세요.

GTID 기반 복제 파라미터

다음 파라미터를 사용하여 GTID 기반 복제를 구성할 수 있습니다.

파라미터	유효한 값	설명
`gtid_mode`	`OFF`, `OFF_PERMISSIVE` , `ON_PERMISSIVE` , `ON`	`OFF`는 새 트랜잭션을 익명 트랜잭션(GTID가 없음)으로 지정하며, 트랜잭션을 복제하려면 익명이어야 합니다. `OFF_PERMISSIVE` 는 새 트랜잭션을 익명 트랜잭션(GTID가 없음)으로 지정하지만, 모든 트랜잭션을 복제할 수 있습니다. `ON_PERMISSIVE` 는 새 트랜잭션을 GTID 트랜잭션으로 지정하지만, 모든 트랜잭션을 복제할 수 있습니다. `ON`은 새 트랜잭션을 GTID 트랜잭션으로 지정하고, 트랜잭션을 복제하려면 GTID 트랜잭션이어야 합니다.

파라미터	유효한 값	설명
enforce_gtid_consistency	OFF, ON, WARN	OFF는 트랜잭션이 GTID 일관성을 위반하는 것을 허용합니다. ON은 트랜잭션이 GTID 일관성을 위반하지 않도록 합니다. WARN은 트랜잭션이 GTID 일관성을 위반하는 것을 허용하지만, 위반이 발생할 경우 경고를 생성합니다.

> **Note**
>
> AWS Management Console 콘솔에서 gtid_mode 파라미터는 gtid-mode로 표시됩니다.

GTID 기반 복제의 경우 DB 인스턴스 또는 읽기 전용 복제본의 파라미터 그룹에 대해 이 설정을 사용하십시오.

- ON 및 ON_PERMISSIVE는 RDS DB 인스턴스에서 밖으로 복제하는 경우에만 적용됩니다. 복제되는 트랜잭션에 대해 이 두 값으로 인해 RDS DB 인스턴스가 GTID를 사용하게 됩니다. ON은 대상 데이터베이스에서도 GTID 기반 복제를 사용할 것을 요구합니다. ON_PERMISSIVE로 인해 외부 데이터베이스에서 GTID 기반 복제는 선택 사항이 됩니다.
- OFF_PERMISSIVE가 설정된 경우 이는 RDS DB 인스턴스가 소스 데이터베이스에서 안으로 복제하는 것을 수락할 수 있음을 뜻합니다. 소스 데이터베이스가 GTID 기반 복제를 사용하든, 사용하지 않든 수락이 가능합니다.
- OFF가 설정된 경우 이는 RDS DB 인스턴스가 GTID 기반 복제를 사용하지 않는 소스 데이터베이스에서 안으로 복제하는 것만을 수락할 수 있음을 뜻합니다.

파라미터 그룹에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.

RDS for MySQL용 새 읽기 전용 복제본에 대한 GTID 기반 복제 활성화

RDS for MySQL DB 인스턴스에 대해 GTID 기반 복제를 활성화하면 DB 인스턴스의 읽기 전용 복제본에 대해 GTID 기반 복제가 자동으로 구성됩니다.

새 읽기 전용 복제본에 대해 GTID 기반 복제를 활성화하려면

1. DB 인스턴스와 연결된 파라미터 그룹에서 다음과 같이 파라미터가 설정되어 있는지 확인합니다.

   - `gtid_mode` – `ON` 또는 `ON_PERMISSIVE`
   - `enforce_gtid_consistency` – `ON`

   파라미터 그룹을 사용한 구성 파라미터 설정에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.
2. DB 인스턴스의 파라미터 그룹을 변경한 경우 DB 인스턴스를 재부팅하십시오. 이 작업을 수행하는 방법에 대한 자세한 내용은 DB 인스턴스 재부팅 단원을 참조하십시오.
3. DB 인스턴스의 읽기 전용 복제본을 한 개 이상 생성합니다. 이 작업을 수행하는 방법에 대한 자세한 내용은 읽기 전용 복제본 생성 단원을 참조하십시오.

Amazon RDS는 `MASTER_AUTO_POSITION`를 사용하여 MySQL DB 인스턴스와 읽기 전용 복제본 간에 GTID 기본 복제를 설정하려고 시도합니다. 이 시도가 실패할 경우 Amazon RDS는 읽기 전용 복제본이 있는 복제의 로그 파일 위치를 사용합니다. `MASTER_AUTO_POSITION`에 대한 자세한 내용은 MySQL 설명서에서 GTID 자동 배치를 참조하십시오.

RDS for MySQL용 기존 읽기 전용 복제본에 대한 GTID 기반 복제 활성화

읽기 전용 복제본이 포함되어 있고 GTID 기반 복제를 사용하지 않는 기존 MySQL DB 인스턴스의 경우에는 DB 인스턴스와 읽기 전용 복제본 간에 GTID 기반 복제를 구성할 수 있습니다.

기존 읽기 전용 복제본에 대해 GTID 기반 복제를 활성화하려면

1. DB 인스턴스 또는 모든 읽기 전용 복제본이 RDS for MySQL 버전 8.0.26보다 낮은 버전 8.0을 사용하는 경우 DB 인스턴스 또는 읽기 전용 복제본을 MySQL 8.0보다 높은 8.0.26으로 업그레이드합니다. 모든 RDS for MySQL 8.4 버전 및 5.7 버전이 GTID 기반 복제를 지원합니다.

   자세한 내용은 RDS for MySQL DB 엔진 업그레이드 단원을 참조하십시오.
2. (선택 사항) GTID 파라미터를 재설정하고 DB 인스턴스와 읽기 전용 복제본의 동작을 테스트합니다.

   a. DB 인스턴스 및 각 읽기 전용 복제본과 연결된 파라미터 그룹에서 `enforce_gtid_consistency` 파라미터가 `WARN`으로 설정되어 있는지 확인합니다.

파라미터 그룹을 사용한 구성 파라미터 설정에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.

b. DB 인스턴스의 파라미터 그룹을 변경한 경우 DB 인스턴스를 재부팅하십시오. 읽기 전용 복제본의 파라미터 그룹을 변경한 경우 읽기 전용 복제본을 재부팅합니다.

자세한 내용은 DB 인스턴스 재부팅 섹션을 참조하세요.

c. 보통의 워크로드로 DB 인스턴스와 읽기 전용 복제본을 실행하고 로그 파일을 모니터링합니다.

GTID 비호환 트랜잭션에 대한 경고가 표시될 경우, GTID 호환 기능만 사용하도록 애플리케이션을 조정하십시오. 다음 단계로 진행하기 전에 DB 인스턴스에서 GTID 비호환 트랜잭션에 대한 경고가 표시되지 않는지 확인합니다.

3. 읽기 전용 복제본이 모든 트랜잭션을 처리할 때까지 익명 트랜잭션을 허용하는 GTID 기반 복제에 대한 GTID 파라미터를 재설정합니다.

a. DB 인스턴스 및 각 읽기 전용 복제본과 연결된 파라미터 그룹에서 다음과 같이 파라미터가 설정되었는지 확인합니다.

- `gtid_mode` – `ON_PERMISSIVE`
- `enforce_gtid_consistency` – `ON`

b. DB 인스턴스의 파라미터 그룹을 변경한 경우 DB 인스턴스를 재부팅하십시오. 읽기 전용 복제본의 파라미터 그룹을 변경한 경우 읽기 전용 복제본을 재부팅합니다.

4. 익명 트랜잭션이 모두 복제될 때까지 기다립니다. 이러한 트랜잭션이 복제되었는지 확인하려면 다음과 같이 합니다.

a. 소스 DB 인스턴스에서 다음 문을 실행합니다.

MySQL 8.4

```
SHOW BINARY LOG STATUS;
```

MySQL 5.7 및 8.0

```
SHOW MASTER STATUS;
```

`File` 및 `Position` 열의 값을 메모합니다.

b. 각 읽기 전용 복제본에서 이전 단계에서 메모한 소스 인스턴스의 파일 및 위치 정보를 사용하여 다음 쿼리를 실행합니다.

```
SELECT MASTER_POS_WAIT('file', position);
```

예를 들어, 파일 이름이 mysql-bin-changelog.000031이고 위치가 107일 경우 다음 명령문을 실행합니다.

```
SELECT MASTER_POS_WAIT('mysql-bin-changelog.000031', 107);
```

읽기 전용 복제본이 지정된 위치에 전달되면 쿼리가 즉시 반환합니다. 그렇지 않으면 함수가 대기합니다. 모든 읽기 전용 복제본에 대해 쿼리가 반환하면 다음 단계로 진행합니다.

5. GTID 기반 복제에 대해서만 GTID 파라미터를 재설정합니다.

a. DB 인스턴스 및 각 읽기 전용 복제본과 연결된 파라미터 그룹에서 다음과 같이 파라미터가 설정되었는지 확인합니다.

- gtid_mode – ON
- enforce_gtid_consistency – ON

b. DB 인스턴스와 각 읽기 전용 복제본을 재부팅합니다.

6. 각각의 읽기 전용 복제본에서 다음 프로시저를 실행합니다.

MySQL 8.4 이상 메이저 버전

```
CALL mysql.rds_set_source_auto_position(1);
```

MySQL 8.0 이하 메이저 버전

```
CALL mysql.rds_set_master_auto_position(1);
```

읽기 전용 복제본이 포함된 MySQL DB 인스턴스에 대해 GTID 기반 복제 비활성화

읽기 전용 복제본이 포함된 MySQL DB 인스턴스입니다.

읽기 전용 복제본이 포함된 MySQL DB 인스턴스에 대해 GTID 기반 복제 사용 중지

1. 각각의 읽기 전용 복제본에서 다음 프로시저를 실행합니다.

MySQL 8.4 이상 메이저 버전

```
CALL mysql.rds_set_source_auto_position(0);
```

MySQL 8.0 이하 메이저 버전

```
CALL mysql.rds_set_master_auto_position(0);
```

2. gtid_mode를 ON_PERMISSIVE로 재설정합니다.

   a. MySQL DB 인스턴스 및 각 읽기 전용 복제본과 연결된 파라미터 그룹에서 gtid_mode 파라미터가 ON_PERMISSIVE로 설정되어 있는지 확인합니다.

      파라미터 그룹을 사용한 구성 파라미터 설정에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.

   b. MySQL DB 인스턴스와 각 읽기 전용 복제본을 재부팅합니다. 재부팅에 대한 자세한 내용은 DB 인스턴스 재부팅 단원을 참조하십시오.

3. gtid_mode를 OFF_PERMISSIVE로 재설정합니다.

   a. MySQL DB 인스턴스 및 각 읽기 전용 복제본과 연결된 파라미터 그룹에서 gtid_mode 파라미터가 OFF_PERMISSIVE로 설정되어 있는지 확인합니다.

   b. MySQL DB 인스턴스와 각 읽기 전용 복제본을 재부팅합니다.

4. 모든 읽기 전용 복제본에서 모든 GTID 트랜잭션이 적용될 때까지 기다립니다. 이러한 사항이 적용되었는지 확인하려면 다음 단계를 수행합니다.

   a. MySQL DB 인스턴스에서 다음 명령을 실행합니다.

      MySQL 8.4

      ```
      SHOW BINARY LOG STATUS
      ```

      MySQL 5.7 및 8.0

      ```
      SHOW MASTER STATUS
      ```

      출력이 다음 출력과 유사해야 합니다.

```
File                          Position
-----------------------------------
mysql-bin-changelog.000031      107
-----------------------------------
```

출력에서 파일 및 위치를 메모합니다.

b. 각 읽기 전용 복제본에서 이전 단계의 소스 인스턴스의 파일 및 위치 정보를 사용하여 다음 쿼리를 실행합니다.

MySQL 8.4 및 MySQL 8.0.26 이상 MySQL 8.0 버전

```
SELECT SOURCE_POS_WAIT('file', position);
```

MySQL 5.7

```
SELECT MASTER_POS_WAIT('file', position);
```

예를 들어 파일 이름이 `mysql-bin-changelog.000031`이고 위치가 `107`일 경우 다음 문을 실행합니다.

MySQL 8.4 및 MySQL 8.0.26 이상 MySQL 8.0 버전

```
SELECT SOURCE_POS_WAIT('mysql-bin-changelog.000031', 107);
```

MySQL 5.7

```
SELECT MASTER_POS_WAIT('mysql-bin-changelog.000031', 107);
```

5. GTID 기반 복제를 비활성화하도록 GTID 파라미터를 재설정합니다.

a. MySQL DB 인스턴스 및 각 읽기 전용 복제본과 연결된 파라미터 그룹에서 다음과 같이 파라미터가 설정되었는지 확인합니다.

- `gtid_mode` – `OFF`
- `enforce_gtid_consistency` – `OFF`

b. MySQL DB 인스턴스와 각 읽기 전용 복제본을 재부팅합니다.

외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성

바이너리 로그 파일 복제를 사용하여 RDS for MySQL 또는 MariaDB DB 인스턴스와 Amazon RDS 외부에 있는 MySQL 또는 MariaDB 인스턴스 간에 복제를 설정할 수 있습니다.

주제

- 시작하기 전에
- 외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성

시작하기 전에

복제된 트랜잭션의 이진 로그 파일 위치를 사용하여 복제를 구성할 수 있습니다.

Amazon RDS DB 인스턴스에서 복제를 시작하는 데 필요한 권한은 제한되고 Amazon RDS 마스터 사용자는 사용할 수 없습니다. 이 때문에 Amazon RDS mysql.rds_set_external_master(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하) 또는 mysql.rds_set_external_source(RDS for MySQL 메이저 버전 8.4 이상) 및 mysql.rds_start_replication 명령을 사용하여 라이브 데이터베이스와 Amazon RDS 데이터베이스 사이의 복제를 설정해야 합니다.

MySQL 또는 MariaDB 데이터베이스의 이진 로깅 형식을 설정하려면 `binlog_format` 파라미터를 업데이트합니다. DB 인스턴스가 기본 DB 인스턴스 파라미터 그룹을 사용하는 경우, `binlog_format` 파라미터를 수정하려면 새로운 DB 파라미터 그룹을 만듭니다. MariaDB 및 MySQL 8.0 이하 버전에서는 `binlog_format`의 기본값이 `MIXED`입니다. 그러나 특정한 이진 로그(binlog) 형식이 필요하다면 `binlog_format`을 `ROW` 또는 `STATEMENT`로 설정할 수도 있습니다. 변경 사항을 적용하려면 DB 인스턴스를 재부팅합니다. MySQL 8.4 이상 버전에서는 `binlog_format`의 기본값이 `ROW`입니다.

`binlog_format` 파라미터 설정에 대한 자세한 내용은 단일 AZ 데이터베이스의 RDS for MySQL 이진 로깅 구성 단원을 참조하십시오. 다양한 MySQL 복제 유형에 대한 자세한 내용은 MySQL 설명서의 문 기반 및 행 기반 복제의 장/단점을 참조하십시오.

외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성

Amazon RDS에서 외부 소스 인스턴스 및 복제본을 설정할 때 다음 지침을 따르십시오.

- 사용자의 복제본인 Amazon RDS DB 인스턴스에 대한 장애 조치 이벤트를 모니터링합니다. 장애 조치가 발생할 경우에는 사용자의 복제본인 DB 인스턴스가 다른 네트워크 주소를 가진 새 호스트에서 다시 생성될 수도 있습니다. 장애 조치 이벤트를 모니터링하는 자세한 방법은 Amazon RDS 이벤트 알림 작업 단원을 참조하십시오.

- binlog가 복제본에 적용된 것으로 확인될 때까지는 소스 인스턴스에서 binlog를 유지 관리합니다. 이렇게 유지 관리해야 오류 발생 시 소스 인스턴스를 복원할 수 있습니다.
- Amazon RDS DB 인스턴스에서 자동 백업을 활성화합니다. 자동 백업을 활성화하면 소스 인스턴스 및 복제본을 다시 동기화할 필요가 있을 때 복제본을 특정 시점으로 복원할 수 있습니다. 백업 및 특정 시점으로 복원에 대한 자세한 내용은 데이터 백업, 복원 및 내보내기 단원을 참조하십시오.

외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제를 구성하려면

1. 원본 MySQL 또는 MariaDB 인스턴스를 읽기 전용으로 설정합니다.

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SET GLOBAL read_only = ON;
```

2. 원본 MySQL 또는 MariaDB 인스턴스에서 `SHOW MASTER STATUS` 명령을 실행하여 binlog 위치를 확인합니다.

   다음 예제와 비슷한 출력 결과를 얻습니다.

```
File                          Position
-------------------------------------
 mysql-bin-changelog.000031      107
-------------------------------------
```

3. `mysqldump`를 사용하여 외부 인스턴스에서 Amazon RDS DB 인스턴스로 데이터베이스를 복사합니다. 매우 큰 데이터베이스의 경우, 가동 중지 시간을 줄이면서 Amazon RDS for MySQL 데이터베이스로 데이터 가져오기에서 이 절차를 사용하고 싶을 것입니다.

   대상 LinuxmacOS, 또는Unix:

```
mysqldump --databases database_name \
    --single-transaction \
    --compress \
    --order-by-primary \
    -u local_user \
    -plocal_password | mysql \
        --host=hostname \
        --port=3306 \
        -u RDS_user_name \
        -pRDS_password
```

Windows의 경우:

```
mysqldump --databases database_name ^
    --single-transaction ^
    --compress ^
    --order-by-primary ^
    -u local_user ^
    -plocal_password | mysql ^
        --host=hostname ^
        --port=3306 ^
        -u RDS_user_name ^
        -pRDS_password
```

> **Note**
>
> -p 옵션과 입력한 암호 사이에 공백이 없어야 합니다.

Amazon RDS DB 인스턴스에 연결하기 위해 호스트 이름, 사용자 이름, 포트 및 암호를 지정하려면 `--host` 명령에 `--user (-u)`, `--port`, `-p`, `mysql` 옵션을 사용합니다. 호스트 이름은 Amazon RDS DB 인스턴스 엔드포인트의 DNS(Domain Name Service) 이름입니다(예: `myinstance.123456789012.us-east-1.rds.amazonaws.com`). AWS Management Console의 인스턴스 세부 정보에서 엔드포인트 값을 확인할 수 있습니다.

4. 원본 MySQL 또는 MariaDB 인스턴스를 다시 쓰기 가능한 상태로 만듭니다.

```
mysql> SET GLOBAL read_only = OFF;
mysql> UNLOCK TABLES;
```

복제에 사용할 백업을 만드는 방법에 대한 자세한 내용은 MySQL 설명서를 참조하십시오.

5. AWS Management Console에서 Amazon RDS DB 인스턴스에 대한 Virtual Private Cloud(VPC) 보안 그룹에 외부 데이터베이스를 호스팅하는 서버의 IP 주소를 추가합니다. VPC 보안 그룹 수정에 대한 자세한 내용은 Amazon Virtual Private Cloud 사용 설명서의 VPC용 보안 그룹을 참조하십시오.

   다음 조건이 충족되면 IP 주소가 바뀔 수 있습니다.

   - 외부 소스 인스턴스와 DB 인스턴스 간 통신에 퍼블릭 IP 주소를 사용하는 경우
   - 외부 소스 인스턴스가 중지되었다가 다시 시작된 경우

위 두 가지 조건이 충족되면 추가하기 전에 IP 주소를 확인하십시오.

Amazon RDS DB 인스턴스의 IP 주소로부터의 연결을 허용하도록 로컬 네트워크를 구성해야 할 수도 있습니다. 이렇게 하면 로컬 네트워크가 외부 MySQL 또는 MariaDB 인스턴스와 통신할 수 있습니다. Amazon RDS DB 인스턴스의 IP 주소를 확인하려면 `host` 명령을 사용합니다.

```
host db_instance_endpoint
```

호스트 이름은 Amazon RDS DB 인스턴스 엔드포인트의 DNS 이름입니다.

6. 선택한 클라이언트를 사용하여 외부 인스턴스에 연결하고 복제에 사용할 사용자를 만듭니다. 이 계정을 복제용으로만 사용하고, 보안을 강화하기 위해 해당 도메인으로만 제한하십시오. 다음은 예입니다.

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

7. 외부 인스턴스의 경우 복제 사용자에게 `REPLICATION CLIENT` 및 `REPLICATION SLAVE` 권한을 부여합니다. 예를 들어 도메인의 `'REPLICATION CLIENT'` 사용자를 위해 모든 데이터베이스에서 `REPLICATION SLAVE` 및 `repl_user` 권한을 부여하려면 다음 명령을 실행합니다.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
```

8. Amazon RDS DB 인스턴스를 복제본으로 만듭니다. 이렇게 하려면 먼저 마스터 사용자로 Amazon RDS DB 인스턴스에 연결합니다. 그런 다음 mysql.rds_set_external_source(RDS for MySQL 메이저 버전 8.4 이상) 또는 mysql.rds_set_external_master(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하) 명령을 사용하여 외부 MySQL 또는 MariaDB 데이터베이스를 소스 인스턴스로 식별합니다. 2단계에서 확인한 마스터 로그 파일 이름과 마스터 로그 위치를 사용합니다. 다음 명령은 예시입니다.

   MySQL 8.4

```
CALL mysql.rds_set_external_source ('mysourceserver.mydomain.com', 3306,
 'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 1);
```

MariaDB 및 MySQL 8.0 및 5.7

```
CALL mysql.rds_set_external_master ('mymasterserver.mydomain.com', 3306,
 'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 1);
```

> **Note**
>
> RDS for MySQL의 경우 mysql.rds_set_external_source_with_delay(RDS for MySQL 메이저 버전 8.4 이상) 또는 mysql.rds_set_external_master_with_delay(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하) 저장 프로시저를 대신 실행하여 지연 복제를 사용하도록 선택할 수 있습니다. RDS for MySQL에서 지연 복제를 사용하는 이유 중 하나는 mysql.rds_start_replication_until 저장 프로시저를 사용하여 재해를 복구할 수 있기 때문입니다. 현재 RDS for MariaDB에서는 지연된 복제를 지원하지만 `mysql.rds_start_replication_until` 절차에서는 지원하지 않습니다.

9. Amazon RDS DB 인스턴스에서 mysql.rds_start_replication 명령을 실행하여 복제를 시작합니다.

```
CALL mysql.rds_start_replication;
```

Amazon RDS for MySQL용 다중 소스 복제 구성

다중 소스 복제를 사용하면 Amazon RDS for MySQL DB 인스턴스를 둘 이상의 RDS for MySQL 소스 DB 인스턴스로부터 바이너리 로그 이벤트를 수신하는 복제본으로 설정할 수 있습니다. 다중 소스 복제는 다음 엔진 버전을 실행하는 RDS for MySQL DB 인스턴스에 지원됩니다.

- 모든 MySQL 8.4 버전
- 8.0.35 이상 마이너 버전
- 5.7.44 이상 마이너 버전

MySQL 다중 소스 복제에 대한 자세한 내용은 MySQL 설명서의 MySQL 다중 소스 복제를 참조하세요. MySQL 설명서에는 이 기능에 대한 자세한 정보가 포함되어 있으며, 이 항목에서는 RDS for MySQL DB 인스턴스에서 다중 소스 복제 채널을 구성하고 관리하는 방법을 설명합니다.

다중 소스 복제 사용 사례

RDS for MySQL에서 다중 소스 복제를 사용하기에 적합한 경우는 다음과 같습니다.

- 개별 DB 인스턴스에 있는 여러 샤드를 단일 샤드로 병합하거나 결합해야 하는 애플리케이션
- 여러 소스에서 통합된 데이터로 보고서를 생성해야 하는 애플리케이션
- 여러 RDS for MySQL용 DB 인스턴스에 분산된 데이터의 통합 장기 백업을 생성하기 위한 요구 사항

다중 소스 복제를 위한 사전 요구 사항

다중 소스 복제를 구성하려면 먼저 다음 사전 조건을 완료하세요.

- 각 소스 RDS for MySQL DB 인스턴스에 자동 백업이 활성화되어 있는지 확인하세요. 자동 백업을 활성화하면 바이너리 로깅이 활성화됩니다. 자동 백업을 활성화하는 방법에 대한 자세한 내용은 the section called “자동 백업 활성화” 섹션을 참조하세요.
- 복제 오류를 방지하려면 소스 DB 인스턴스에 대한 쓰기 작업을 차단하는 것이 좋습니다. RDS for MySQL용 소스 DB 인스턴스에 연결된 사용자 지정 파라미터 그룹에서 `read-only` 파라미터를 `ON`으로 설정하면 됩니다. AWS Management Console 또는 AWS CLI로 사용자 지정 파라미터 그룹을 새로 생성하거나 기존 그룹을 수정할 수 있습니다. 자세한 내용은 the section called “DB 파라미터 그룹 생성” 및 the section called “DB 파라미터 그룹의 파라미터 수정”(을)를 참조하세요.
- 다중 소스 DB 인스턴스에 대한 Virtual Private Cloud(VPC) 보안 그룹에 각 소스 DB 인스턴스에 인스턴스의 IP 주소를 추가합니다. `dig` *`RDS Endpoint`* 명령을 실행하여 소스 DB 인스턴스의 IP 주소를 식별할 수 있습니다. 대상 다중 소스 DB 인스턴스의 VPC와 동일한 VPC의 Amazon EC2에서 명령을 실행합니다.
- 다음 예제와 같이 각 소스 DB 인스턴스에 대해 클라이언트를 사용하여 DB 인스턴스에 연결하고 복제에 필요한 권한을 가진 데이터베이스 사용자를 생성합니다.

```
CREATE USER 'repl_user' IDENTIFIED BY 'password';
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user';
```

> **Note**
>
> MySQL 8.4부터 `REPLICATION SLAVE` 권한은 더 이상 사용되지 않으며 `REPLICATION REPLICA`로 대체되었습니다. MySQL 8.4 이상 버전의 경우 다음 구문을 대신 사용합니다.
>
> ```
> CREATE USER 'repl_user' IDENTIFIED BY 'password';
> GRANT REPLICATION CLIENT, REPLICATION REPLICA ON *.* TO 'repl_user';
> ```

MySQL용 RDS의 다중 소스 복제 채널 구성

다중 소스 복제 채널을 구성하는 것은 단일 소스 복제 구성과 비슷합니다. 다중 소스 복제의 경우 먼저 소스 인스턴스에 대한 이진 로깅을 설정합니다. 그런 다음 소스에서 다중 소스 복제본으로 데이터를 가져옵니다. 그런 다음 이진 로그 좌표를 사용하거나 GTID 자동 위치 지정을 사용하여 각 소스에서 복제를 시작합니다.

RDS for MySQL DB 인스턴스의 다중 소스 복제본인 RDS for MySQL DB 인스턴스의 다중 소스 복제본으로 구성하려면 다음 단계를 수행합니다.

주제

- 1단계: 소스 DB 인스턴스에서 다중 소스 복제본으로 데이터 가져오기
- 2단계: 소스 DB 인스턴스에서 다중 소스 복제본으로의 복제 시작하기

1단계: 소스 DB 인스턴스에서 다중 소스 복제본으로 데이터 가져오기

각 소스 DB 인스턴스에서 다음 단계를 수행합니다.

소스에서 다중 소스 복제본으로 데이터를 가져오기 전에 `SHOW MASTER STATUS` 명령을 실행하여 현재 바이너리 로그 파일과 위치를 확인하세요. 다음 단계에서 사용할 수 있도록 이러한 세부 정보를 적어 둡니다. 이 예제 출력에서 파일은 `mysql-bin-changelog.000031`이고 위치는 `107`입니다.

> **Note**
>
> MySQL 8.4부터 `SHOW MASTER STATUS` 명령이 더 이상 사용되지 않으며 `SHOW BINARY LOG STATUS`로 대체되었습니다. MySQL 8.4 이상 버전의 경우 `SHOW BINARY LOG STATUS`를 대신 사용합니다.

```
File                          Position
-----------------------------------
mysql-bin-changelog.000031      107
-----------------------------------
```

이제 다음 예제와 같이 `mysqldump`를 사용하여 소스 DB 인스턴스에서 다중 소스 복제본으로 데이터베이스를 복사합니다.

```
mysqldump --databases database_name \
 --single-transaction \
```

```
  --compress \
  --order-by-primary \
  -u RDS_user_name \
  -p RDS_password \
  --host=RDS Endpoint | mysql \
  --host=RDS Endpoint \
  --port=3306 \
  -u RDS_user_name \
 -p RDS_password
```

데이터베이스를 복사한 후 소스 DB 인스턴스에서 읽기 전용 파라미터를 OFF로 설정할 수 있습니다.

2단계: 소스 DB 인스턴스에서 다중 소스 복제본으로의 복제 시작하기

각 소스 DB 인스턴스에 대해 관리 사용자 보안 인증을 사용하여 인스턴스에 연결하고 다음 두 개의 저장 프로시저를 실행합니다. 이러한 저장 프로시저는 채널에서 복제를 구성하고 복제를 시작합니다. 이 예에서는 이전 단계의 예제 출력에 있는 binlog 파일 이름과 위치를 사용합니다.

```
CALL mysql.rds_set_external_source_for_channel('mysourcehost.example.com', 3306,
 'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 1, 'channel_1');
CALL mysql.rds_start_replication_for_channel('channel_1');
```

이러한 저장 프로시저 및 기타를 사용하여 복제 채널을 설정하고 관리하는 방법에 대한 자세한 내용은 the section called “다중 소스 복제 관리” 섹션을 참조하세요.

다중 소스 복제와 함께 필터 사용

복제 필터를 사용하여 다중 소스 복제본과 함께 복제할 데이터베이스와 테이블을 지정할 수 있습니다. 복제 필터는 데이터베이스와 테이블을 복제에 포함하거나 복제에서 제외할 수 있습니다. 복제 필터에 대한 자세한 정보는 the section called “복제 필터 구성” 섹션을 참조하세요.

다중 소스 복제를 사용하면 전체적으로 또는 채널 수준에서 복제 필터를 구성할 수 있습니다. 채널 수준 필터링은 버전 8.0 또는 버전 8.4를 실행하는 지원되는 DB 인스턴스에서만 사용할 수 있습니다. 다음 예제에서는 필터를 전역 또는 채널 수준에서 구성하는 방법을 보여줍니다.

다중 소스 복제의 필터링과 관련된 다음 요구 사항 및 동작을 참고하세요.

- 채널 이름 주위에는 역따옴표(``)가 필요합니다.
- 파라미터 그룹에서 복제 필터를 변경하면 업데이트가 적용된 모든 채널에 다중 소스 복제본의 `sql_thread`가 다시 시작되어 변경 내용이 동적으로 적용됩니다. 업데이트에 글로벌 필터가 포함된 경우 실행 상태의 모든 복제 채널이 다시 시작됩니다.

- 모든 글로벌 필터는 채널별 필터보다 먼저 적용됩니다.
- 필터가 전체적으로 채널 수준에서 적용되는 경우 채널 수준 필터만 적용됩니다. 예를 들어, 필터가 `replicate_ignore_db="db1,`channel_22`:db2"`이면 db1에 설정된 `replicate_ignore_db`는 `channel_22`를 제외한 모든 채널에 적용되고, `channel_22`만 db2의 변경 내용을 무시합니다.

예 1: 글로벌 필터 설정

다음 예시에서 `temp_data` 데이터베이스는 모든 채널의 복제에서 제외됩니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
--db-parameter-group-name myparametergroup \
--parameters "ParameterName=replicate-ignore-
db,ParameterValue='temp_data',ApplyMethod=immediate"
```

예 2: 채널 수준 필터 설정

다음 예제에서는 `sample22` 데이터베이스의 변경 내용이 `channel_22` 채널에만 포함됩니다. 마찬가지로 `sample99` 데이터베이스의 변경 사항은 `channel_99` 채널에만 포함됩니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
--db-parameter-group-name myparametergroup \
--parameters "ParameterName=replicate-do-db,ParameterValue='\`channel_22\`:sample22,
\`channel_99\`:sample99',ApplyMethod=immediate"
```

다중 소스 복제 채널 모니터링

다음 방법을 사용하여 다중 소스 복제본의 개별 채널을 모니터링할 수 있습니다.

- 모든 채널 또는 특정 채널의 상태를 모니터링하려면 다중 소스 복제본에 연결하고 `SHOW REPLICA STATUS` 또는 `SHOW REPLICA STATUS FOR CHANNEL 'channel_name'` 명령을 실행합니다. 이러한 필드에 대한 자세한 내용은 MySQL 설명서의 복제 상태 확인을 참조하세요.
- 복제 채널이 시작, 중지 또는 제거될 때 알림을 받으려면 RDS 이벤트 알림을 사용합니다. 자세한 내용은 the section called “Amazon RDS 이벤트 알림 작업” 섹션을 참조하세요.

- 특정 채널의 지연을 모니터링하려면 해당 채널의 `ReplicationChannelLag` 지표를 확인하세요. 이 지표에 대해 기간이 60초(1분)로 설정된 데이터 요소들은 15일 동안 사용할 수 있습니다. 채널의 복제 채널 지연을 찾으려면 인스턴스 식별자와 복제 채널 이름을 사용하세요. 이 지연이 특정 임곗값을 초과할 때 알림을 받으려면 CloudWatch 경보를 설정할 수 있습니다. 자세한 내용은 the section called “CloudWatch를 사용하여 RDS 모니터링” 섹션을 참조하세요.

다중 소스 복제의 고려 사항

RDS for MySQL에서 다중 소스 복제를 사용하기 전에 다음 고려 사항과 모범 사례를 검토하세요.

- 다중 소스 복제본으로 구성된 DB 인스턴스에 처리량, 메모리, CPU, IOPS와 같은 리소스가 충분한지 확인하여 여러 원본 인스턴스의 워크로드를 처리할 수 있는지 확인하세요.
- 다중 소스 복제본의 리소스 사용률을 정기적으로 모니터링하고 리소스에 부담을 주지 않고 워크로드를 처리할 수 있도록 스토리지 또는 인스턴스 구성을 조정하세요.
- 시스템 변수 `replica_parallel_workers`를 `0`보다 큰 값으로 설정하여 다중 소스 복제본에서 다중 스레드 복제를 구성할 수 있습니다. 이 경우 각 채널에 할당된 스레드 수는 이 변수의 값에 해당 스레드를 관리하는 코디네이터 스레드 1개를 더한 값입니다.
- 복제 필터를 적절하게 구성하여 충돌을 방지하세요. 전체 데이터베이스를 복제본의 다른 데이터베이스에 복제하기 위해 `--replicate-rewrite-db` 옵션을 사용할 수 있습니다. 예를 들어 데이터베이스 A의 모든 테이블을 복제 인스턴스의 데이터베이스 B에 복제할 수 있습니다. 이 접근 방식은 모든 원본 인스턴스가 동일한 스키마 명명 규칙을 사용할 때 유용할 수 있습니다. `--replicate-rewrite-db` 옵션에 대한 자세한 내용은 MySQL 설명서의 복제본 서버 옵션 및 변수를 참조하세요.
- 복제 오류를 방지하려면 복제본에 쓰지 마세요. 쓰기 작업을 차단하려면 다중 소스 복제본에서 `read_only` 파라미터를 활성화하는 것이 좋습니다. 이렇게 하면 쓰기 작업 충돌로 인한 복제 문제를 해결하는 데 도움이 됩니다.
- 다중 소스 복제본에서 실행되는 정렬 및 고부하 조인과 같은 읽기 작업의 성능을 높이려면 RDS 최적화된 읽기를 사용하는 것이 좋습니다. 이 기능은 대규모 임시 테이블이나 정렬 파일에 의존하는 쿼리에 유용할 수 있습니다. 자세한 내용은 the section called “RDS Optimized Reads를 통한 쿼리 성능 개선” 섹션을 참조하세요.
- 복제 지연을 최소화하고 다중 소스 복제본의 성능을 개선하려면 쓰기 최적화를 활성화하는 것이 좋습니다. 자세한 내용은 the section called “RDS Optimized Writes for MySQL을 통한 쓰기 성능 개선” 섹션을 참조하세요.
- 한 번에 한 채널에서 관리 작업(예: 구성 변경)을 수행하고 여러 연결에서 여러 채널을 변경하지 마세요. 이러한 경우 복제 작업에서 충돌이 발생할 수 있습니다. 예를 들어 여러 연결에서

`rds_skip_repl_error_for_channel` 및 `rds_start_replication_for_channel` 프로시저를 동시에 실행하면 의도한 것과 다른 채널의 이벤트를 건너뛸 수 있습니다.

- 다중 소스 복제 인스턴스에서 백업을 활성화하고 해당 인스턴스의 데이터를 Amazon S3 버킷으로 내보내 장기간 사용할 수 있도록 저장할 수 있습니다. 하지만 개별 소스 인스턴스에 적절한 보존 기간을 두고 백업을 구성하는 것도 중요합니다. Amazon S3로 DB 스냅샷 데이터 내보내기에 대한 자세한 내용은 the section called “Amazon S3로 DB 스냅샷 데이터 내보내기” 섹션을 참조하세요.
- 다중 소스 복제본에 읽기 워크로드를 분산하려면 다중 소스 복제본에서 읽기 전용 복제본을 생성할 수 있습니다. 애플리케이션의 요구 사항에 따라 다른 AWS 리전에서 이러한 읽기 전용 복제본을 찾을 수 있습니다. 읽기 전용 복제본에 대한 자세한 내용은 the section called “MySQL 읽기 전용 복제본”을 참조하세요.

RDS for MySQL용 다중 소스 복제 제한

RDS for MySQL용 다중 소스에 다음과 같은 제한 사항이 적용됩니다.

- 현재 RDS for MySQL은 다중 소스 복제본에 대해 최대 15개의 채널 구성을 지원합니다.
- 읽기 복제본 인스턴스는 다중 소스 복제본으로 구성할 수 없습니다.
- 엔진 버전 5.7을 실행하는 RDS for MySQL에서 다중 소스 복제를 구성하려면 복제본 인스턴스에서 성능 스키마를 활성화해야 합니다. RDS for MySQL 엔진 버전 8.0 또는 8.4에서 성능 스키마를 활성화 하는 것은 선택 사항입니다.
- 엔진 버전 5.7을 실행하는 RDS for MySQL의 경우 복제 필터가 모든 복제 채널에 적용됩니다. 엔진 버전 8.0 또는 8.4를 실행하는 RDS for MySQL의 경우 모든 복제 채널 또는 개별 채널에 적용되는 필터를 구성할 수 있습니다.
- RDS 스냅샷을 복원하거나 시점 복구(PITR)를 수행해도 다중 소스 복제본 채널 구성은 복원되지 않습니다.
- 다중 소스 복제본의 읽기 전용 복제본을 생성하면 다중 소스 인스턴스의 데이터만 복제됩니다. 채널 구성은 복원되지 않습니다.
- MySQL은 각 채널에 대해 다른 수의 병렬 워커를 설정하는 것을 지원하지 않습니다. 모든 채널은 `replica_parallel_workers` 값에 따라 동일한 수의 병렬 워커를 받습니다.

다중 소스 복제 대상이 다중 AZ DB 클러스터인 경우 다음과 같은 추가 제한 사항이 적용됩니다.

- 소스 RDS for MySQL 인스턴스용 채널을 구성해야 해당 인스턴스에 쓰기 작업을 수행할 수 있습니다.
- RDS for MySQL 인스턴스의 경우 GTID 기반 복제가 활성화되어야 합니다.

- DB 클러스터의 장애 조치 이벤트가 발생하면 다중 소스 복제 구성이 제거됩니다. 해당 구성을 복원하려면 구성 단계를 반복해야 합니다.

RDS for MySQL 액티브-액티브 클러스터 구성

Amazon RDS의 액티브-액티브 클러스터는 여러 노드가 읽기 및 쓰기 작업을 적극적으로 처리하여 인스턴스 간에 워크로드를 분산하고 가용성과 확장성을 개선하는 데이터베이스 구성입니다. 클러스터의 각 노드는 데이터 일관성을 유지하기 위해 동기화되므로 노드 장애 발생 시 고가용성과 더 빠른 장애 조치를 지원합니다.

MySQL 그룹 복제 플러그인을 사용하여 RDS for MySQL 액티브-액티브 클러스터를 설정할 수 있습니다. 그룹 복제 플러그인은 다음 버전을 실행하는 RDS for MySQL DB 인스턴스에 지원됩니다.

- 모든 MySQL 8.4 버전
- MySQL 8.0.35 이상 마이너 버전

MySQL 그룹 복제에 대한 자세한 내용은 MySQL 설명서의 그룹 복제를 참조하세요. MySQL 설명서에는 이 기능에 대한 자세한 정보가 포함되어 있으며, 이 주제에서는 RDS for MySQL의 DB 인스턴스에서 플러그인을 구성하고 관리하는 방법을 설명합니다.

> **Note**
>
> 간결하게 설명하기 위해 이 주제에서 '액티브-액티브' 클러스터에 대한 모든 언급은 MySQL 그룹 복제 플러그인을 사용하는 액티브-액티브 클러스터를 가리킵니다.

액티브-액티브 클러스터의 사용 사례

액티브-액티브 클러스터를 사용하기에 좋은 경우는 다음과 같습니다.

- 쓰기 작업을 지원하기 위해 클러스터의 모든 DB 인스턴스가 필요한 애플리케이션입니다. 그룹 복제 플러그인은 액티브-액티브 클러스터의 각 DB 인스턴스에서 데이터를 일관되게 유지합니다. 작동 방식에 대한 자세한 내용은 MySQL 설명서의 그룹 복제를 참조하세요.
- 데이터베이스의 지속적인 가용성이 필요한 애플리케이션입니다. 액티브-액티브 클러스터의 경우 데이터는 클러스터의 모든 DB 인스턴스에 유지됩니다. 한 DB 인스턴스에 장애가 발생하는 경우 애플리케이션은 클러스터의 다른 DB 인스턴스로 트래픽을 다시 라우팅할 수 있습니다.
- 로드 밸런싱을 위해 클러스터의 여러 DB 인스턴스 간에 읽기 및 쓰기 작업을 분할해야 할 수 있는 애플리케이션입니다. 액티브-액티브 클러스터를 사용하면 애플리케이션이 특정 DB 인스턴스로 읽기 트래픽을 보내고 다른 인스턴스에는 쓰기 트래픽을 보낼 수 있습니다. 또한 언제든지 읽기 또는 쓰기를 전송할 DB 인스턴스를 전환할 수 있습니다.

주제

- 액티브-액티브 클러스터의 제한 및 고려 사항
- 교차 VPC 액티브-액티브 클러스터 준비
- 액티브-액티브 클러스터의 필수 파라미터 설정
- 기존 DB 인스턴스를 액티브-액티브 클러스터로 변환
- 새 DB 인스턴스로 액티브-액티브 클러스터 설정
- 액티브-액티브 클러스터에 DB 인스턴스 추가
- 액티브-액티브 클러스터 모니터링
- 액티브-액티브 클러스터의 DB 인스턴스에서 그룹 복제 중지
- 액티브-액티브 클러스터의 DB 인스턴스 이름 변경
- 액티브-액티브 클러스터에서 DB 인스턴스 제거

액티브-액티브 클러스터의 제한 및 고려 사항

Amazon RDS의 액티브-액티브 클러스터는 여러 인스턴스에 워크로드를 분산하여 향상된 가용성과 확장성을 제공합니다. 그러나 이 아키텍처를 사용할 때 유의해야 할 중요한 제한 사항과 고려 사항이 있습니다.

다음 섹션에서는 복제 지연, 충돌 해결, 리소스 할당 및 장애 조치 동작과 같은 주요 요인을 간략하게 설명합니다. 이러한 고려 사항을 이해하면 액티브-액티브 클러스터 배포에서 최적의 성능과 안정성을 보장하는 데 도움이 될 수 있습니다.

주제

- RDS for MySQL 액티브-액티브 클러스터의 제한 사항
- RDS for MySQL 액티브-액티브 클러스터에 대한 고려 사항 및 모범 사례

RDS for MySQL 액티브-액티브 클러스터의 제한 사항

RDS for MySQL의 액티브-액티브 클러스터에 다음과 같은 제한 사항이 적용됩니다.

- 액티브-액티브 클러스터의 DB 인스턴스에 대한 마스터 사용자 이름으로 `rdsgrprepladmin`을 지정할 수 없습니다. 이 사용자 이름은 그룹 복제 연결용으로 예약되어 있습니다.
- 액티브-액티브 클러스터에 읽기 전용 복제본이 있는 DB 인스턴스의 경우, `Replicating` 이외의 연장된 복제 상태가 로그 파일이 저장 제한을 초과하게끔 만들 수 있습니다. 읽기 전용 복제본 상태에 대한 자세한 내용은 읽기 전용 복제본 모니터링 섹션을 참조하세요.

- 액티브-액티브 클러스터의 DB 인스턴스에는 블루/그린 배포가 지원되지 않습니다. 자세한 내용은 데이터베이스 업데이트에 Amazon RDS 블루/그린 배포 사용 단원을 참조하십시오.
- 액티브-액티브 클러스터의 DB 인스턴스에는 Kerberos 인증이 지원되지 않습니다. 자세한 내용은 Amazon RDS for MySQL에 Kerberos 인증 사용 단원을 참조하십시오.
- 다중 AZ DB 클러스터의 DB 인스턴스는 액티브-액티브 클러스터에 추가할 수 없습니다. 그러나 다중 AZ DB 인스턴스 배포의 DB 인스턴스는 액티브-액티브 클러스터에 추가할 수 있습니다. 자세한 내용은 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 단원을 참조하십시오.
- 그룹 복제 플러그인이 쓰기를 거부하기 때문에 프라이머리 키가 없는 표는 액티브-액티브 클러스터에 복제되지 않습니다.
- InnoDB가 아닌 표는 액티브-액티브 클러스터에 복제되지 않습니다.
- 액티브-액티브 클러스터는 클러스터의 여러 DB 인스턴스에 대한 동시 DML 및 DDL 문을 지원하지 않습니다.
- 그룹의 복제 모드에 단일 기본 모드를 사용하도록 액티브-액티브 클러스터를 구성할 수 없습니다. 이 구성의 경우 다중 AZ DB 클러스터를 대신 사용하는 것이 좋습니다. 자세한 내용은 Amazon RDS용 다중 AZ DB 클러스터 배포 단원을 참조하십시오.
- 액티브-액티브 클러스터의 DB 인스턴스에는 다중 소스 복제가 지원되지 않습니다.
- 리전 간 액티브-액티브 클러스터는 그룹 복제 연결에 인증 기관(CA) 검증을 적용할 수 없습니다.

RDS for MySQL 액티브-액티브 클러스터에 대한 고려 사항 및 모범 사례

RDS for MySQL 액티브-액티브 클러스터를 사용하려면 먼저 다음 고려 사항 및 모범 사례를 검토하세요.

- 액티브-액티브 클러스터의 DB 인스턴스는 9개를 초과할 수 없습니다.
- 그룹 복제 플러그인을 사용하면 액티브-액티브 클러스터의 트랜잭션 일관성 보장을 제어할 수 있습니다. 자세한 내용은 MySQL 설명서의 트랜잭션 일관성 보장을 참조하세요.
- 액티브-액티브 클러스터에서 서로 다른 DB 인스턴스가 동일한 행을 업데이트하는 경우 충돌이 발생할 수 있습니다. 충돌 및 충돌 해결에 대한 자세한 내용은 MySQL 설명서의 그룹 복제를 참조하세요.
- 내결함성을 위해 액티브-액티브 클러스터에 DB 인스턴스를 3개 이상 포함합니다. 1~2개의 DB 인스턴스만으로도 액티브-액티브 클러스터를 구성할 수는 있지만, 클러스터에 내결함성이 갖춰지지 않습니다. 내결함성에 대한 자세한 내용은 MySQL 설명서의 내결함성을 참조하세요.
- DB 인스턴스가 기존 액티브-액티브 클러스터에 조인하고 클러스터에서 가장 낮은 엔진 버전과 동일한 엔진 버전을 실행하는 경우 DB 인스턴스는 읽기-쓰기 모드로 조인됩니다.

- DB 인스턴스가 기존 액티브-액티브 클러스터에 조인하고 클러스터의 최하위 엔진 버전보다 높은 엔진 버전을 실행하는 경우 DB 인스턴스는 읽기 전용 모드를 유지해야 합니다.
- DB 파라미터 그룹에서 `rds.group_replication_enabled` 파라미터를 1로 설정하여 DB 인스턴스의 그룹 복제를 활성화했지만, 복제가 시작되지 않았거나 시작에 실패한 경우 DB 인스턴스는 슈퍼 읽기 전용 모드로 전환되어 데이터 불일치를 방지합니다. 슈퍼 읽기 전용 모드에 대한 자세한 내용은 MySQL 설명서를 참조하세요.
- 액티브-액티브 클러스터에서 DB 인스턴스를 업그레이드할 수 있지만, 액티브-액티브 클러스터의 다른 모든 DB 인스턴스가 동일한 엔진 버전 또는 상위 엔진 버전으로 업그레이드될 때까지 DB 인스턴스는 읽기 전용입니다. DB 인스턴스를 업그레이드한 경우 업그레이드가 완료되면 DB 인스턴스가 동일한 액티브-액티브 클러스터에 자동으로 조인합니다. 의도치 않게 DB 인스턴스가 읽기 전용 모드로 전환되는 것을 방지하려면 해당 인스턴스의 자동 마이너 버전 업그레이드를 비활성화하세요. MySQL DB 인스턴스 업그레이드에 대한 자세한 내용은 RDS for MySQL DB 엔진 업그레이드를 참조하십시오.
- 다중 AZ DB 인스턴스 배포의 DB 인스턴스를 기존 액티브-액티브 클러스터에 추가할 수 있습니다. 액티브-액티브 클러스터의 단일 AZ DB 인스턴스를 다중 AZ DB 인스턴스 배포로 변환할 수도 있습니다. 다중 AZ 배포의 기본 DB 인스턴스에 장애가 발생하면 기본 인스턴스가 대기 인스턴스로 장애 조치됩니다. 새 기본 DB 인스턴스는 장애 조치가 완료된 후 동일한 클러스터에 자동으로 조인합니다. 다중 AZ DB 인스턴스 배포에 대한 자세한 내용은 Amazon RDS에 대한 다중 AZ 인스턴스 배포 섹션을 참조하세요.
- 액티브-액티브 클러스터의 DB 인스턴스는 유지 관리 기간의 시간 범위를 다르게 설정하는 것이 좋습니다. 이렇게 하면 클러스터의 여러 DB 인스턴스가 유지 관리를 위해 동시에 오프라인 상태가 되는 것을 방지할 수 있습니다. 자세한 내용은 Amazon RDS 유지 관리 기간 단원을 참조하십시오.
- 액티브-액티브 클러스터는 DB 인스턴스 간 연결에 SSL을 사용할 수 있습니다. SSL 연결을 구성하려면 group_replication_recovery_use_ssl 및 group_replication_ssl_mode 파라미터를 설정하세요. 이러한 파라미터의 값은 액티브-액티브 클러스터의 모든 DB 인스턴스에서 일치해야 합니다.

  현재 액티브-액티브 클러스터는 AWS 리전 간 연결에 대한 인증 기관(CA) 검증을 지원하지 않습니다. 따라서 group_replication_ssl_mode 파라미터를 `DISABLED`(기본값)로 설정하거나, 리전 간 클러스터의 경우 `REQUIRED`로 설정해야 합니다.
- RDS for MySQL 액티브-액티브 클러스터는 다중 기본 모드에서 실행됩니다. group_replication_enforce_update_everywhere_checks의 기본값은 `ON`이며, 파라미터는 정적입니다. 이 파라미터를 `ON`으로 설정하면 애플리케이션은 계단식 외래 키 제약 조건이 있는 표에 삽입할 수 없습니다.
- RDS for MySQL 액티브-액티브 클러스터는 XCOM 대신 MySQL 통신 스택을 연결 보안에 사용합니다. 자세한 내용은 MySQL 설명서의 연결 보안 관리를 위한 통신 스택을 참조하세요.

- DB 파라미터 그룹을 액티브-액티브 클러스터의 DB 인스턴스와 연결할 때는 이 DB 파라미터 그룹을 클러스터에 있는 다른 DB 인스턴스에만 연결하는 것이 좋습니다.
- 액티브-액티브 클러스터는 RDS for MySQL DB 인스턴스만 지원합니다. 이러한 DB 인스턴스는 지원되는 버전의 DB 엔진을 실행해야 합니다.
- 액티브-액티브 클러스터의 DB 인스턴스에 예상치 못한 장애가 발생하면 RDS는 자동으로 DB 인스턴스 복구를 시작합니다. DB 인스턴스가 복구되지 않는 경우 클러스터의 정상 DB 인스턴스를 통해 특정 시점으로 복구를 수행하여 새 DB 인스턴스로 교체하는 것이 좋습니다. 지침은 특정 시점으로 복구를 사용하여 액티브-액티브 클러스터에 DB 인스턴스 추가 단원을 참조하십시오.
- 클러스터의 다른 DB 인스턴스에 영향을 주지 않고 액티브-액티브 클러스터의 DB 인스턴스를 삭제할 수 있습니다. DB 인스턴스 삭제에 대한 자세한 내용은 DB 인스턴스 삭제 단원을 참조하세요.
- DB 인스턴스가 의도치 않게 활성-활성 클러스터를 벗어나면 `group_replication_exit_state_action` 파라미터가 기본적으로 `OFFLINE_MODE`로 변경됩니다. 이 상태에서는 DB 인스턴스에 액세스할 수 없으며 DB 인스턴스를 재부팅하여 다시 온라인 상태로 전환하고 클러스터에 다시 조인해야 합니다. 사용자 지정 파라미터 그룹에서 `group_replication_exit_state_action` 파라미터를 수정하여 이 동작을 변경할 수 있습니다. 파라미터를 `READ_ONLY`로 설정하면 DB 인스턴스가 의도하지 않게 클러스터를 떠날 때 오프라인 상태가 아닌 슈퍼 읽기 전용 상태가 됩니다.

교차 VPC 액티브-액티브 클러스터 준비

2개 이상의 VPC에서 Amazon RDS for MySQL DB 인스턴스를 사용하여 액티브-액티브 클러스터를 구성할 수 있습니다. VPC는 동일한 AWS 리전에 있을 수도 있고, 다른 AWS 리전에 있을 수도 있습니다.

> **Note**
>
> 여러 AWS 리전 간에 트래픽을 전송하면 추가 비용이 발생할 수 있습니다. 자세한 내용은 일반적인 아키텍처의 데이터 전송 비용 개요를 참조하세요.

단일 VPC에서 액티브-액티브 클러스터를 구성하는 경우 이 단계를 건너뛰고 새 DB 인스턴스로 액티브-액티브 클러스터 설정으로 이동합니다.

2개 이상의 VPC에 DB 인스턴스가 있는 액티브-액티브 클러스터를 준비하려면

1. CIDR 블록의 IPv4 주소 범위가 다음 요구 사항을 충족하는지 확인합니다.
   - VPC의 CIDR 블록에 있는 IPv4 주소 범위는 겹칠 수 없습니다.

- CIDR 블록의 모든 IPv4 주소 범위는 128.0.0.0/*subnet_mask*보다 작거나 128.0.0.0/*subnet_mask*보다 커야 합니다.

  다음 범위는 이러한 요구 사항을 보여줍니다.

  - 한 VPC에서는 10.1.0.0/16이 지원되고, 다른 VPC에서는 10.2.0.0/16이 지원됩니다.
  - 한 VPC에서는 172.1.0.0/16이 지원되고, 다른 VPC에서는 172.2.0.0/16이 지원됩니다.
  - 한 VPC의 10.1.0.0/16 및 다른 VPC의 10.1.0.0/16은 범위가 겹치기 때문에 지원되지 않습니다.
  - 한 VPC의 10.1.0.0/16 및 다른 VPC의 172.1.0.0/16은 하나는 128.0.0.0/*subnet_mask*보다 낮고 다른 하나는 128.0.0.0/*subnet_mask*보다 높기 때문에 지원되지 않습니다.

  CIDR 블록에 대한 자세한 내용은 Amazon VPC 사용 설명서의 VPC CIDR 블록을 참조하세요.

2. 각 VPC에서 DNS 확인 및 DNS 호스트 이름이 모두 활성화되어 있는지 확인합니다.

   지침은 Amazon VPC 사용 설명서의 VPC에 대한 DNS 속성 보기 및 업데이트를 참조하세요.

3. VPC 간 트래픽을 다음 방법 중 하나로 라우팅할 수 있도록 VPC를 구성합니다.

   - VPC 간에 VPC 피어링 연결을 만듭니다.

     지침은 Amazon VPC 피어링 가이드의 VPC 피어링 연결 생성을 참조하세요. 각 VPC에 피어링된 VPC의 보안 그룹을 참조하는 보안 그룹의 인바운드 규칙이 있는지 확인합니다. 그렇게 하면 피어링된 VPC의 참조 보안 그룹과 연결된 인스턴스 간에 트래픽을 주고받을 수 있습니다. 지침은 Amazon VPC 피어링 가이드의 피어 보안 그룹을 참조하도록 보안 그룹 업데이트를 참조하세요.

   - VPC 간에 전송 게이트웨이를 만듭니다.

     지침은 Amazon VPC 전송 게이트웨이의 전송 게이트웨이 시작하기를 참조하세요. 각 VPC에 다른 VPC의 트래픽을 허용하는 보안 그룹의 인바운드 규칙(예: 다른 VPC의 CIDR을 지정하는 인바운드 규칙)이 있는지 확인합니다. 이렇게 하면 액티브-액티브 클러스터의 참조된 보안 그룹과 연결된 인스턴스 간에 트래픽을 주고받을 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 보안 그룹을 사용하여 AWS 리소스에 대한 트래픽 제어를 참조하세요.

액티브-액티브 클러스터의 필수 파라미터 설정

Amazon RDS for MySQL에서 액티브-액티브 클러스터의 파라미터를 구성하는 것은 일관된 성능과 운영 안정성을 유지하는 데 필수적입니다. 이 표에서는 복제, 충돌 해결 및 워크로드 배포를 제어하는 주요 파라미터를 자세히 설명합니다. 올바른 구성은 노드 간의 효율적인 동기화를 보장하고, 복제 지연을 최소화하며, 분산 또는 트래픽이 많은 환경에서 리소스 사용률을 최적화합니다.

파라미터	설명	필수 설정
`binlog_format`	이진 로깅 형식을 설정합니다. RDS for MySQL 8.0 버전 이하에서 기본값은 `MIXED`입니다. RDS for MySQL 8.4 버전에서 기본값은 `ROW`입니다. 자세한 내용은 MySQL 설명서를 참조하세요.	`ROW`
`enforce_gtid_consistency`	문 실행에 GTID 일관성을 적용합니다. RDS for MySQL의 기본값은 `OFF`입니다. 자세한 내용은 MySQL 설명서를 참조하세요.	`ON`
`group_replication_group_name`	그룹 복제 이름을 UUID로 설정합니다. UUID 형식은 `11111111-2222-3333-4444-555555555555` 입니다. MySQL DB 인스턴스에 연결하고 `SELECT UUID()`를 실행하여 MySQL UUID를 생성할 수 있습니다. 액티브-액티브 클러스터의 모든 DB 인스턴스에서 값이 동일해야 합니다. 자세한 내용은 MySQL 설명서를 참조하세요.	MySQL UUID
`gtid-mode`	GTID 기반 로깅을 제어합니다. RDS for MySQL의 기본값은	`ON`

파라미터	설명	필수 설정
	OFF_PERMISSIVE 입니다. 자세한 내용은 MySQL 설명서를 참조하세요.	
rds.custom_dns_resolution	VPC의 Amazon DNS 서버로부터의 DNS 확인을 허용할지 여부를 지정합니다. rds.group_replication_enabled 파라미터로 그룹 복제를 활성화한 경우 DNS 확인을 활성화해야 합니다. rds.group_replication_enabled 파라미터로 그룹 복제를 비활성화한 경우 DNS 확인을 활성화할 수 없습니다. 자세한 내용은 Amazon VPC 사용 설명서의 Amazon DNS 서버를 참조하세요.	1
rds.group_replication_enabled	그룹 복제를 DB 인스턴스에 활성화할지 여부를 지정합니다. 액티브-액티브 클러스터의 DB 인스턴스에서 그룹 복제를 활성화해야 합니다.	1
replica_preserve_commit_order (RDS for MySQL 8.4 이상 버전) 또는 slave_preserve_commit_order (RDS for MySQL 8.0 버전)	복제본에서 트랜잭션이 커밋되는 순서를 제어합니다. RDS for MySQL의 기본값은 ON입니다. 자세한 내용은 MySQL 설명서를 참조하세요.	ON

기존 DB 인스턴스를 액티브-액티브 클러스터로 변환

활성-활성 클러스터로 마이그레이션하려는 DB 인스턴스의 DB 엔진 버전은 다음 중 하나여야 합니다.

- 모든 MySQL 8.4 버전
- MySQL 8.0.35 이상 마이너 버전

엔진 버전을 업그레이드해야 하는 경우 RDS for MySQL DB 엔진 업그레이드 섹션을 참조하세요.

2개 이상의 VPC에 DB 인스턴스가 있는 액티브-액티브 클러스터를 설정하는 경우 교차 VPC 액티브-액티브 클러스터 준비에 나와 있는 사전 조건을 완료해야 합니다.

다음 단계를 완료하여 기존 DB 인스턴스를 RDS for MySQL용 액티브-액티브 클러스터로 마이그레이션하세요.

주제

- 1단계: 하나 이상의 사용자 지정 파라미터 그룹에서 액티브-액티브 클러스터 파라미터 설정
- 2단계: 필수 그룹 복제 파라미터가 설정된 DB 파라미터 그룹과 DB 인스턴스 연결
- 3단계: 액티브-액티브 클러스터 생성
- 4단계: 액티브-액티브 클러스터를 위한 추가 RDS for MySQL DB 인스턴스 생성
- 5단계: 변환하려는 DB 인스턴스에서 그룹 초기화
- 6단계: 액티브-액티브 클러스터의 다른 DB 인스턴스에서 복제 시작
- 7단계: (권장) 액티브-액티브 클러스터의 상태 확인

1단계: 하나 이상의 사용자 지정 파라미터 그룹에서 액티브-액티브 클러스터 파라미터 설정

액티브-액티브 클러스터의 RDS for MySQL DB 인스턴스는 필수 파라미터가 올바르게 설정된 사용자 지정 파라미터 그룹과 연결되어야 합니다. 파라미터 및 각 파라미터에 대한 필수 설정에 대한 정보는 액티브-액티브 클러스터의 필수 파라미터 설정 섹션을 참조하세요.

새 파라미터 그룹이나 기존 파라미터 그룹에서 이러한 파라미터를 설정할 수 있습니다. 하지만 액티브-액티브 클러스터에 속하지 않은 DB 인스턴스에 실수로 영향을 주지 않으려면 새 사용자 지정 파라미터 그룹을 생성하는 것이 좋습니다. 액티브-액티브 클러스터의 DB 인스턴스를 동일한 DB 파라미터 그룹이나 다른 DB 파라미터 그룹과 연결할 수 있습니다.

AWS Management Console 또는 AWS CLI로 사용자 지정 파라미터 그룹을 새로 생성할 수 있습니다. 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹 생성 단원을 참조하십시오. 다음은 create-db-parameter-group AWS CLI 명령을 실행하여 RDS for MySQL 8.0에 대해 이름이 *myactivepg*인 사용자 지정 DB 파라미터 그룹을 만드는 예시입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-parameter-group \
  --db-parameter-group-name myactivepg \
  --db-parameter-group-family mysql8.0 \
  --description "Parameter group for active-active clusters"
```

Windows의 경우:

```
aws rds create-db-parameter-group ^
  --db-parameter-group-name myactivepg ^
  --db-parameter-group-family mysql8.0 ^
  --description "Parameter group for active-active clusters"
```

AWS Management Console 또는 AWS CLI로 사용자 지정 파라미터 그룹에서 파라미터를 설정할 수도 있습니다. 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 단원을 참조하십시오.

다음 예시에서는 modify-db-parameter-group AWS CLI 명령을 실행하여 RDS for MySQL 8.0에 대해 파라미터를 설정합니다. 이 예시를 RDS for MySQL 8.4와 함께 사용하려면 `slave_preserve_commit_order`를 `replica_preserve_commit_order`로 변경합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myactivepg \
  --parameters
 "ParameterName='rds.group_replication_enabled',ParameterValue='1',ApplyMethod=pending-
reboot" \

 "ParameterName='rds.custom_dns_resolution',ParameterValue='1',ApplyMethod=pending-
reboot" \

 "ParameterName='enforce_gtid_consistency',ParameterValue='ON',ApplyMethod=pending-
reboot" \
               "ParameterName='gtid-mode',ParameterValue='ON',ApplyMethod=pending-
reboot" \

 "ParameterName='binlog_format',ParameterValue='ROW',ApplyMethod=immediate" \

 "ParameterName='slave_preserve_commit_order',ParameterValue='ON',ApplyMethod=immediate"
 \
```

```
 "ParameterName='group_replication_group_name',ParameterValue='11111111-2222-3333-4444-55555555
reboot"
```

Windows의 경우:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myactivepg ^
  --parameters
 "ParameterName='rds.group_replication_enabled',ParameterValue='1',ApplyMethod=pending-
reboot" ^

 "ParameterName='rds.custom_dns_resolution',ParameterValue='1',ApplyMethod=pending-
reboot" ^

 "ParameterName='enforce_gtid_consistency',ParameterValue='ON',ApplyMethod=pending-
reboot" ^
               "ParameterName='gtid-mode',ParameterValue='ON',ApplyMethod=pending-
reboot" ^

 "ParameterName='binlog_format',ParameterValue='ROW',ApplyMethod=immediate" ^

 "ParameterName='slave_preserve_commit_order',ParameterValue='ON',ApplyMethod=immediate"
 ^

 "ParameterName='group_replication_group_name',ParameterValue='11111111-2222-3333-4444-55555555
reboot"
```

2단계: 필수 그룹 복제 파라미터가 설정된 DB 파라미터 그룹과 DB 인스턴스 연결

DB 인스턴스를 이전 단계에서 만들거나 수정한 파라미터 그룹과 연결합니다. 지침은 Amazon RDS의 DB 인스턴스에 DB 파라미터 그룹 연결 단원을 참조하십시오.

새 파라미터 설정을 적용하려면 DB 인스턴스를 재부팅하세요. 지침은 DB 인스턴스 재부팅 단원을 참조하십시오.

3단계: 액티브-액티브 클러스터 생성

DB 인스턴스와 연결된 DB 파라미터 그룹에서 `group_replication_group_seeds` 파라미터를 변환할 DB 인스턴스의 엔드포인트로 설정합니다.

AWS Management Console 또는 AWS CLI를 사용하여 파라미터를 설정할 수 있습니다. 이 파라미터를 설정한 후 DB 인스턴스를 재부팅할 필요가 없습니다. 파라미터 설정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요.

다음 예제에서는 modify-db-parameter-group AWS CLI 명령을 실행하여 파라미터를 설정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myactivepg \
  --parameters
 "ParameterName='group_replication_group_seeds',ParameterValue='myactivedb1.123456789012.us-east-1.rds.amazonaws.com:3306',ApplyMethod=immediate"
```

Windows의 경우:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myactivepg ^
  --parameters
 "ParameterName='group_replication_group_seeds',ParameterValue='myactivedb1.123456789012.us-east-1.rds.amazonaws.com:3306',ApplyMethod=immediate"
```

4단계: 액티브-액티브 클러스터를 위한 추가 RDS for MySQL DB 인스턴스 생성

액티브-액티브 클러스터용 추가 DB 인스턴스를 만들려면 변환하려는 DB 인스턴스에서 특정 시점으로 복구를 수행합니다. 지침은 특정 시점으로 복구를 사용하여 액티브-액티브 클러스터에 DB 인스턴스 추가 단원을 참조하십시오.

액티브-액티브 클러스터에는 최대 9개의 DB 인스턴스가 있을 수 있습니다. 클러스터에 사용할 DB 인스턴스 수를 원하는 수준으로 갖출 때까지 DB 인스턴스에서 특정 시점으로 복구를 수행합니다. 특정 시점으로 복구를 수행할 때 추가할 DB 인스턴스를 `rds.group_replication_enabled`가 `1`로 설정된 DB 파라미터 그룹과 연결해야 합니다. 그렇지 않으면 새로 추가된 DB 인스턴스에서 그룹 복제가 시작되지 않습니다.

5단계: 변환하려는 DB 인스턴스에서 그룹 초기화

그룹을 초기화하고 복제를 시작합니다.

1. SQL 클라이언트에서 변환하려는 DB 인스턴스에 연결합니다. RDS for MySQL DB 인스턴스 연결에 대한 자세한 내용은 MySQL DB 인스턴스에 연결 섹션을 참조하세요.

2. SQL 클라이언트에서 다음 저장 프로시저를 실행하고 *group_replication_user_password*를 rdsgrprepladmin 사용자의 암호로 대체합니다. rdsgrprepladmin 사용자는 액티브-액티브 클러스터의 그룹 복제 연결에만 사용할 수 있습니다. 이 사용자의 암호는 액티브-액티브 클러스터의 모든 DB 인스턴스에서 동일해야 합니다.

```
call mysql.rds_set_configuration('binlog retention hours', 168); -- 7 days binlog
call mysql.rds_group_replication_create_user('group_replication_user_password');
call
 mysql.rds_group_replication_set_recovery_channel('group_replication_user_password');
call mysql.rds_group_replication_start(1);
```

   이 예제에서는 binlog retention hours 값을 168로 설정합니다. 즉, 이진 로그 파일은 DB 인스턴스에 7일 동안 유지됩니다. 값은 사용자의 요구 사항에 맞게 조정할 수 있습니다.

   이 예제는 mysql.rds_group_replication_start 저장 프로시저에서 현재 DB 인스턴스로 새 그룹을 초기화하도록 1을 지정합니다.

   예제에서 호출한 저장 프로시저에 대한 자세한 내용은 활성-활성 클러스터 섹션을 참조하세요.

6단계: 액티브-액티브 클러스터의 다른 DB 인스턴스에서 복제 시작

액티브-액티브 클러스터의 각 DB 인스턴스에 대해 SQL 클라이언트를 사용하여 인스턴스에 연결하고 다음 저장 프로시저를 실행합니다. *group_replication_user_password*를 rdsgrprepladmin 사용자의 암호로 대체합니다.

```
call mysql.rds_set_configuration('binlog retention hours', 168); -- 7 days binlog
call mysql.rds_group_replication_create_user('group_replication_user_password');
call
 mysql.rds_group_replication_set_recovery_channel('group_replication_user_password');
call mysql.rds_group_replication_start(0);
```

이 예제에서는 binlog retention hours 값을 168로 설정합니다. 즉, 이진 로그 파일은 각 DB 인스턴스에 7일 동안 유지됩니다. 값은 사용자의 요구 사항에 맞게 조정할 수 있습니다.

이 예제는 mysql.rds_group_replication_start 저장 프로시저에서 현재 DB 인스턴스를 기존 그룹에 조인하도록 0을 지정합니다.

> **Tip**
>
> 액티브-액티브 클러스터의 다른 모든 DB 인스턴스에서 이러한 저장 프로시저를 실행해야 합니다.

7단계: (권장) 액티브-액티브 클러스터의 상태 확인

클러스터의 각 멤버가 올바르게 구성되었는지 확인하려면 액티브-액티브 클러스터의 DB 인스턴스에 연결하고 다음 SQL 명령을 실행하여 클러스터의 상태를 확인합니다.

```
SELECT * FROM performance_schema.replication_group_members;
```

출력은 다음 샘플 출력과 같이 각 DB 인스턴스의 MEMBER_STATE에 대해 ONLINE으로 표시되어야 합니다.

```
+---------------------------+--------------------------------------
+----------------+-------------+--------------+-------------+----------------
+----------------------------+
| CHANNEL_NAME              | MEMBER_ID                            | MEMBER_HOST    |
 MEMBER_PORT | MEMBER_STATE | MEMBER_ROLE | MEMBER_VERSION | MEMBER_COMMUNICATION_STACK
 |
+---------------------------+--------------------------------------
+----------------+-------------+--------------+-------------+----------------
+----------------------------+
| group_replication_applier | 9854d4a2-5d7f-11ee-b8ec-0ec88c43c251 | ip-10-15-3-137 |
      3306 | ONLINE       | PRIMARY     | 8.0.35         | MySQL                       |
| group_replication_applier | 9e2e9c28-5d7f-11ee-8039-0e5d58f05fef | ip-10-15-3-225 |
      3306 | ONLINE       | PRIMARY     | 8.0.35         | MySQL                       |
| group_replication_applier | a6ba332d-5d7f-11ee-a025-0a5c6971197d | ip-10-15-1-83  |
      3306 | ONLINE       | PRIMARY     | 8.0.35         | MySQL                       |
+---------------------------+--------------------------------------
+----------------+-------------+--------------+-------------+----------------
+----------------------------+
3 rows in set (0.00 sec)
```

가능한 MEMBER_STATE 값에 대한 자세한 내용은 MySQL 설명서의 그룹 복제 서버 상태를 참조하세요.

새 DB 인스턴스로 액티브-액티브 클러스터 설정

새 Amazon RDS for MySQL DB 인스턴스를 사용하여 액티브-액티브 클러스터를 설정하려면 다음 단계를 완료하세요.

2개 이상의 VPC에 DB 인스턴스가 있는 액티브-액티브 클러스터를 설정하는 경우 교차 VPC 액티브-액티브 클러스터 준비에 나와 있는 사전 조건을 완료해야 합니다.

주제

- 1단계: 하나 이상의 사용자 지정 파라미터 그룹에서 액티브-액티브 클러스터 파라미터 설정
- 2단계: 액티브-액티브 클러스터를 위한 새 RDS for MySQL DB 인스턴스 생성
- 3단계: 액티브-액티브 클러스터의 DB 인스턴스 지정
- 4단계: DB 인스턴스에서 그룹 초기화 및 복제 시작
- 5단계: 액티브-액티브 클러스터의 다른 DB 인스턴스에서 복제 시작
- 6단계: (권장) 액티브-액티브 클러스터의 상태 확인
- 7단계: (선택 사항) 액티브-액티브 클러스터의 DB 인스턴스로 데이터 가져오기

1단계: 하나 이상의 사용자 지정 파라미터 그룹에서 액티브-액티브 클러스터 파라미터 설정

액티브-액티브 클러스터의 RDS for MySQL DB 인스턴스는 필수 파라미터가 올바르게 설정된 사용자 지정 파라미터 그룹과 연결되어야 합니다. 파라미터 및 각 파라미터에 대한 필수 설정에 대한 정보는 액티브-액티브 클러스터의 필수 파라미터 설정 섹션을 참조하세요.

새 파라미터 그룹이나 기존 파라미터 그룹에서 이러한 파라미터를 설정할 수 있습니다. 하지만 액티브-액티브 클러스터에 속하지 않은 DB 인스턴스에 실수로 영향을 주지 않으려면 새 사용자 지정 파라미터 그룹을 생성하는 것이 좋습니다. 액티브-액티브 클러스터의 DB 인스턴스를 동일한 DB 파라미터 그룹이나 다른 DB 파라미터 그룹과 연결할 수 있습니다.

AWS Management Console 또는 AWS CLI로 사용자 지정 파라미터 그룹을 새로 생성할 수 있습니다. 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹 생성 단원을 참조하십시오. 다음은 create-db-parameter-group AWS CLI 명령을 실행하여 RDS for MySQL 8.0에 대해 이름이 *myactivepg*인 사용자 지정 DB 파라미터 그룹을 만드는 예시입니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-parameter-group \
```

```
  --db-parameter-group-name myactivepg \
  --db-parameter-group-family mysql8.0 \
  --description "Parameter group for active-active clusters"
```

Windows의 경우:

```
aws rds create-db-parameter-group ^
  --db-parameter-group-name myactivepg ^
  --db-parameter-group-family mysql8.0 ^
  --description "Parameter group for active-active clusters"
```

AWS Management Console 또는 AWS CLI로 사용자 지정 파라미터 그룹에서 파라미터를 설정할 수도 있습니다. 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 단원을 참조하십시오.

다음 예시에서는 modify-db-parameter-group AWS CLI 명령을 실행하여 RDS for MySQL 8.0에 대해 파라미터를 설정합니다. 이 예시를 RDS for MySQL 8.4와 함께 사용하려면 `slave_preserve_commit_order`를 `replica_preserve_commit_order`로 변경합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myactivepg \
  --parameters
 "ParameterName='rds.group_replication_enabled',ParameterValue='1',ApplyMethod=pending-
reboot" \

 "ParameterName='rds.custom_dns_resolution',ParameterValue='1',ApplyMethod=pending-
reboot" \

 "ParameterName='enforce_gtid_consistency',ParameterValue='ON',ApplyMethod=pending-
reboot" \
               "ParameterName='gtid-mode',ParameterValue='ON',ApplyMethod=pending-
reboot" \

 "ParameterName='binlog_format',ParameterValue='ROW',ApplyMethod=immediate" \

 "ParameterName='slave_preserve_commit_order',ParameterValue='ON',ApplyMethod=immediate"
 \

 "ParameterName='group_replication_group_name',ParameterValue='11111111-2222-3333-4444-55555555
reboot"
```

Windows의 경우:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myactivepg ^
  --parameters
 "ParameterName='rds.group_replication_enabled',ParameterValue='1',ApplyMethod=pending-
reboot" ^

 "ParameterName='rds.custom_dns_resolution',ParameterValue='1',ApplyMethod=pending-
reboot" ^

 "ParameterName='enforce_gtid_consistency',ParameterValue='ON',ApplyMethod=pending-
reboot" ^
               "ParameterName='gtid-mode',ParameterValue='ON',ApplyMethod=pending-
reboot" ^

 "ParameterName='binlog_format',ParameterValue='ROW',ApplyMethod=immediate" ^

 "ParameterName='slave_preserve_commit_order',ParameterValue='ON',ApplyMethod=immediate"
 ^

 "ParameterName='group_replication_group_name',ParameterValue='11111111-2222-3333-4444-55555555
reboot"
```

2단계: 액티브-액티브 클러스터를 위한 새 RDS for MySQL DB 인스턴스 생성

활성-활성 클러스터는 다음 버전의 RDS for MySQL DB 인스턴스에서 지원됩니다.

- 모든 MySQL 8.4 버전
- MySQL 버전 8.0.35 이상 마이너 버전

클러스터에 대해 새 DB 인스턴스를 9개까지 생성할 수 있습니다.

AWS Management Console 또는 AWS CLI를 사용하여 새 DB 인스턴스를 생성할 수 있습니다. DB 인스턴스를 생성하는 방법에 대한 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요. DB 인스턴스를 만들 때 이전 단계에서 만들거나 수정한 DB 파라미터 그룹과 연결합니다.

3단계: 액티브-액티브 클러스터의 DB 인스턴스 지정

각 DB 인스턴스와 연결된 DB 파라미터 그룹에서 클러스터에 포함하려는 DB 인스턴스의 엔드포인트로 `group_replication_group_seeds` 파라미터를 설정합니다.

AWS Management Console 또는 AWS CLI를 사용하여 파라미터를 설정할 수 있습니다. 이 파라미터를 설정한 후 DB 인스턴스를 재부팅할 필요가 없습니다. 파라미터 설정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요.

다음 예제에서는 modify-db-parameter-group AWS CLI 명령을 실행하여 파라미터를 설정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myactivepg \
  --parameters
 "ParameterName='group_replication_group_seeds',ParameterValue='myactivedb1.123456789012.us-
east-1.rds.amazonaws.com:3306,myactivedb2.123456789012.us-
east-1.rds.amazonaws.com:3306,myactivedb3.123456789012.us-
east-1.rds.amazonaws.com:3306',ApplyMethod=immediate"
```

Windows의 경우:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myactivepg ^
  --parameters
 "ParameterName='group_replication_group_seeds',ParameterValue='myactivedb1.123456789012.us-
east-1.rds.amazonaws.com:3306,myactivedb2.123456789012.us-
east-1.rds.amazonaws.com:3306,myactivedb3.123456789012.us-
east-1.rds.amazonaws.com:3306',ApplyMethod=immediate"
```

> **Tip**
>
> 액티브-액티브 클러스터의 DB 인스턴스와 연결된 각 DB 파라미터 그룹에서 `group_replication_group_seeds` 파라미터를 설정해야 합니다.

4단계: DB 인스턴스에서 그룹 초기화 및 복제 시작

새 DB를 선택하여 그룹을 초기화하고 복제를 시작할 수 있습니다. 이렇게 하려면 다음 단계를 완료합니다.

1. 액티브-액티브 클러스터에서 DB 인스턴스를 선택하고 SQL 클라이언트에서 해당 DB 인스턴스에 연결합니다. RDS for MySQL DB 인스턴스 연결에 대한 자세한 내용은 MySQL DB 인스턴스에 연결 섹션을 참조하세요.

2. SQL 클라이언트에서 다음 저장 프로시저를 실행하고 *group_replication_user_password*를 rdsgrprepladmin 사용자의 암호로 대체합니다. rdsgrprepladmin 사용자는 액티브-액티브 클러스터의 그룹 복제 연결에만 사용할 수 있습니다. 이 사용자의 암호는 액티브-액티브 클러스터의 모든 DB 인스턴스에서 동일해야 합니다.

```
call mysql.rds_set_configuration('binlog retention hours', 168); -- 7 days binlog
call mysql.rds_group_replication_create_user('group_replication_user_password');
call
 mysql.rds_group_replication_set_recovery_channel('group_replication_user_password');
call mysql.rds_group_replication_start(1);
```

   이 예제에서는 binlog retention hours 값을 168로 설정합니다. 즉, 이진 로그 파일은 DB 인스턴스에 7일 동안 유지됩니다. 값은 사용자의 요구 사항에 맞게 조정할 수 있습니다.

   이 예제는 mysql.rds_group_replication_start 저장 프로시저에서 현재 DB 인스턴스로 새 그룹을 초기화하도록 1을 지정합니다.

   예제에서 호출한 저장 프로시저에 대한 자세한 내용은 활성-활성 클러스터 섹션을 참조하세요.

5단계: 액티브-액티브 클러스터의 다른 DB 인스턴스에서 복제 시작

액티브-액티브 클러스터의 각 DB 인스턴스에 대해 SQL 클라이언트를 사용하여 인스턴스에 연결하고 다음 저장 프로시저를 실행합니다. *group_replication_user_password*를 rdsgrprepladmin 사용자의 암호로 대체합니다.

```
call mysql.rds_set_configuration('binlog retention hours', 168); -- 7 days binlog
call mysql.rds_group_replication_create_user('group_replication_user_password');
call
 mysql.rds_group_replication_set_recovery_channel('group_replication_user_password');
call mysql.rds_group_replication_start(0);
```

이 예제에서는 binlog retention hours 값을 168로 설정합니다. 즉, 이진 로그 파일은 각 DB 인스턴스에 7일 동안 유지됩니다. 값은 사용자의 요구 사항에 맞게 조정할 수 있습니다.

이 예제는 mysql.rds_group_replication_start 저장 프로시저에서 현재 DB 인스턴스를 기존 그룹에 조인하도록 0을 지정합니다.

> **Tip**
>
> 액티브-액티브 클러스터의 다른 모든 DB 인스턴스에서 이러한 저장 프로시저를 실행해야 합니다.

6단계: (권장) 액티브-액티브 클러스터의 상태 확인

클러스터의 각 멤버가 올바르게 구성되었는지 확인하려면 액티브-액티브 클러스터의 DB 인스턴스에 연결하고 다음 SQL 명령을 실행하여 클러스터의 상태를 확인합니다.

```
SELECT * FROM performance_schema.replication_group_members;
```

출력은 다음 샘플 출력과 같이 각 DB 인스턴스의 MEMBER_STATE에 대해 ONLINE으로 표시되어야 합니다.

```
+---------------------------+--------------------------------------
+---------------+-------------+--------------+-------------+----------------
+----------------------------+
| CHANNEL_NAME              | MEMBER_ID                            | MEMBER_HOST    |
 MEMBER_PORT | MEMBER_STATE | MEMBER_ROLE | MEMBER_VERSION | MEMBER_COMMUNICATION_STACK
 |
+---------------------------+--------------------------------------
+---------------+-------------+--------------+-------------+----------------
+----------------------------+
| group_replication_applier | 9854d4a2-5d7f-11ee-b8ec-0ec88c43c251 | ip-10-15-3-137 |
      3306 | ONLINE       | PRIMARY     | 8.0.35         | MySQL                       |
| group_replication_applier | 9e2e9c28-5d7f-11ee-8039-0e5d58f05fef | ip-10-15-3-225 |
      3306 | ONLINE       | PRIMARY     | 8.0.35         | MySQL                       |
| group_replication_applier | a6ba332d-5d7f-11ee-a025-0a5c6971197d | ip-10-15-1-83  |
      3306 | ONLINE       | PRIMARY     | 8.0.35         | MySQL                       |
+---------------------------+--------------------------------------
+---------------+-------------+--------------+-------------+----------------
+----------------------------+
3 rows in set (0.00 sec)
```

가능한 MEMBER_STATE 값에 대한 자세한 내용은 MySQL 설명서의 그룹 복제 서버 상태를 참조하세요.

7단계: (선택 사항) 액티브-액티브 클러스터의 DB 인스턴스로 데이터 가져오기

MySQL 데이터베이스의 데이터를 액티브-액티브 클러스터의 DB 인스턴스로 가져올 수 있습니다. 데이터를 가져온 후 그룹 복제는 데이터를 클러스터의 다른 DB 인스턴스로 복제합니다.

데이터 가져오기에 대한 자세한 내용은 가동 중지 시간을 줄이면서 Amazon RDS for MySQL 데이터베이스로 데이터 가져오기 섹션을 참조하세요.

액티브-액티브 클러스터에 DB 인스턴스 추가

DB 스냅샷을 복원하거나 DB 인스턴스를 특정 시점으로 복원하여 DB 인스턴스를 Amazon RDS for MySQL 액티브-액티브 클러스터에 추가할 수 있습니다. 액티브-액티브 클러스터에는 최대 9개의 DB 인스턴스가 포함될 수 있습니다.

DB 인스턴스를 특정 시점으로 복구할 때는 일반적으로 DB 스냅샷에서 복원된 DB 인스턴스보다 최근 트랜잭션이 더 많이 포함됩니다. DB 인스턴스에 최근 트랜잭션이 더 많으면 복제를 시작할 때 적용해야 하는 트랜잭션 수가 줄어듭니다. 따라서 특정 시점으로 복구를 사용하여 클러스터에 DB 인스턴스를 추가하는 것이 일반적으로 DB 스냅샷에서 복원하는 것보다 빠릅니다.

주제

- 특정 시점으로 복구를 사용하여 액티브-액티브 클러스터에 DB 인스턴스 추가
- DB 스냅샷을 사용하여 액티브-액티브 클러스터에 DB 인스턴스 추가

특정 시점으로 복구를 사용하여 액티브-액티브 클러스터에 DB 인스턴스 추가

클러스터의 DB 인스턴스에서 특정 시점으로 복구를 수행하여 액티브-액티브 클러스터에 DB 인스턴스를 추가할 수 있습니다.

다른 AWS 리전에서 DB 인스턴스를 특정 시점으로 복구하는 방법에 대한 자세한 내용은 다른 AWS 리전에 자동 백업 복제 섹션을 참조하세요.

특정 시점으로 복구를 사용하여 액티브-액티브 클러스터에 DB 인스턴스를 추가하려면

1. 액티브-액티브 클러스터의 DB 인스턴스에서 특정 시점으로 복구를 수행하여 새 DB 인스턴스를 생성합니다.

   클러스터의 모든 DB 인스턴스에서 특정 시점으로 복구를 수행하여 새 DB 인스턴스를 생성할 수 있습니다. 지침은 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 단원을 참조하십시오.

> **Important**
>
> 특정 시점으로 복구 중에는 액티브-액티브 클러스터 파라미터가 설정된 DB 파라미터 그룹에 새 DB 인스턴스를 연결합니다. 그렇지 않으면 새 DB 인스턴스에서 그룹 복제가 시작되지 않습니다. 파라미터 및 각 파라미터에 대한 필수 설정에 대한 정보는 액티브-액티브 클러스터의 필수 파라미터 설정 섹션을 참조하세요.

> **Tip**
>
> 특정 시점으로 복구를 시작하기 전에 DB 인스턴스의 스냅샷을 생성하면 새 DB 인스턴스에서 트랜잭션을 적용하는 데 필요한 시간을 줄일 수 있습니다.

2. 새 DB 인스턴스와 연결한 DB 파라미터 그룹을 포함하여 액티브-액티브 클러스터의 DB 인스턴스와 연결된 각 DB 파라미터 그룹의 `group_replication_group_seeds` 파라미터에 DB 인스턴스를 추가합니다.

   파라미터 설정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요.

3. SQL 클라이언트에서 새 DB 인스턴스에 연결하고 mysql.rds_group_replication_set_recovery_channel 저장 프로시저를 호출합니다. *`group_replication_user_password`*를 `rdsgrprepladmin` 사용자의 암호로 대체합니다.

```
call
 mysql.rds_group_replication_set_recovery_channel('group_replication_user_password');
```

4. SQL 클라이언트를 통해 mysql.rds_group_replication_start 저장 프로시저를 호출하여 복제를 시작합니다.

```
call mysql.rds_group_replication_start(0);
```

DB 스냅샷을 사용하여 액티브-액티브 클러스터에 DB 인스턴스 추가

클러스터에 있는 DB 인스턴스의 DB 스냅샷을 만든 다음 DB 스냅샷을 복원하여 액티브-액티브 클러스터에 DB 인스턴스를 추가할 수 있습니다.

스냅샷을 다른 AWS 리전으로 복사하는 방법에 대한 자세한 내용은 the section called “리전 간 복사” 섹션을 참조하세요.

DB 스냅샷을 사용하여 액티브-액티브 클러스터에 DB 인스턴스를 추가하려면

1. 액티브-액티브 클러스터에서 DB 인스턴스의 DB 스냅샷을 생성합니다.

   클러스터에 있는 모든 DB 인스턴스의 DB 스냅샷을 생성할 수 있습니다. 지침은 Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성 단원을 참조하십시오.

2. DB 스냅샷에서 DB 인스턴스를 복원합니다.

   스냅샷 복원 작업 중에 액티브-액티브 클러스터 파라미터가 설정된 DB 파라미터 그룹에 새 DB 인스턴스를 연결합니다. 파라미터 및 각 파라미터에 대한 필수 설정에 대한 정보는 액티브-액티브 클러스터의 필수 파라미터 설정 섹션을 참조하세요.

   DB 스냅샷에서 DB 인스턴스를 복원하는 방법에 대한 자세한 내용은 DB 인스턴스 복원 섹션을 참조하세요.

3. 새 DB 인스턴스와 연결한 DB 파라미터 그룹을 포함하여 액티브-액티브 클러스터의 DB 인스턴스와 연결된 각 DB 파라미터 그룹의 `group_replication_group_seeds` 파라미터에 DB 인스턴스를 추가합니다.

   파라미터 설정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요.

4. SQL 클라이언트에서 새 DB 인스턴스에 연결하고 mysql.rds_group_replication_set_recovery_channel 저장 프로시저를 호출합니다. *`group_replication_user_password`*를 `rdsgrprepladmin` 사용자의 암호로 대체합니다.

   ```
   call
    mysql.rds_group_replication_set_recovery_channel('group_replication_user_password');
   ```

5. SQL 클라이언트를 통해 mysql.rds_group_replication_start 저장 프로시저를 호출하여 복제를 시작합니다.

   ```
   call mysql.rds_group_replication_start(0);
   ```

액티브-액티브 클러스터 모니터링

Amazon RDS for MySQL에서 액티브-액티브 클러스터를 모니터링하는 것은 성능, 복제 무결성 및 노드 동기화를 추적하는 데 매우 중요합니다. 클러스터의 DB 인스턴스에 연결하고 다음 SQL 명령을 실행하여 액티브-액티브 클러스터를 모니터링할 수 있습니다.

```
SELECT * FROM performance_schema.replication_group_members;
```

출력은 다음 샘플 출력과 같이 각 DB 인스턴스의 MEMBER_STATE에 대해 ONLINE으로 표시되어야 합니다.

```
+---------------------------+--------------------------------------
+---------------+-------------+--------------+-------------+---------------
+-----------------------------+
| CHANNEL_NAME              | MEMBER_ID                            | MEMBER_HOST    |
 MEMBER_PORT | MEMBER_STATE | MEMBER_ROLE | MEMBER_VERSION | MEMBER_COMMUNICATION_STACK
 |
+---------------------------+--------------------------------------
+---------------+-------------+--------------+-------------+---------------
+-----------------------------+
| group_replication_applier | 9854d4a2-5d7f-11ee-b8ec-0ec88c43c251 | ip-10-15-3-137 |
      3306 | ONLINE       | PRIMARY     | 8.0.35         | MySQL                       |
| group_replication_applier | 9e2e9c28-5d7f-11ee-8039-0e5d58f05fef | ip-10-15-3-225 |
      3306 | ONLINE       | PRIMARY     | 8.0.35         | MySQL                       |
| group_replication_applier | a6ba332d-5d7f-11ee-a025-0a5c6971197d | ip-10-15-1-83  |
      3306 | ONLINE       | PRIMARY     | 8.0.35         | MySQL                       |
+---------------------------+--------------------------------------
+---------------+-------------+--------------+-------------+---------------
+-----------------------------+
3 rows in set (0.00 sec)
```

가능한 MEMBER_STATE 값에 대한 자세한 내용은 MySQL 설명서의 그룹 복제 서버 상태를 참조하세요.

액티브-액티브 클러스터의 DB 인스턴스에서 그룹 복제 중지

액티브-액티브 클러스터의 DB 인스턴스에서 그룹 복제를 중지할 수 있습니다. 그룹 복제를 중지하면 복제가 다시 시작되거나 해당 DB 인스턴스가 액티브-액티브 클러스터에서 제거될 때까지 DB 인스턴스는 슈퍼 읽기 전용 모드로 전환됩니다. 슈퍼 읽기 전용 모드에 대한 자세한 내용은 MySQL 설명서를 참조하세요.

액티브-액티브 클러스터의 그룹 복제를 일시적으로 중지하려면

1. SQL 클라이언트를 사용하여 액티브-액티브 클러스터의 DB 인스턴스에 연결합니다.

   RDS for MySQL DB 인스턴스 연결에 대한 자세한 내용은 MySQL DB 인스턴스에 연결 섹션을 참조하세요.

2. SQL 클라이언트에서 mysql.rds_group_replication_stop 저장 프로시저를 호출합니다.

```
call mysql.rds_group_replication_stop();
```

액티브-액티브 클러스터의 DB 인스턴스 이름 변경

액티브-액티브 클러스터에서 DB 인스턴스의 이름을 변경할 수 있습니다. 액티브-액티브 클러스터에서 둘 이상의 DB 인스턴스 이름을 바꾸려면 DB 인스턴스 이름을 한 번에 하나씩 바꾸세요. 즉, 다음 DB 인스턴스의 이름을 바꾸기 전에 한 DB 인스턴스의 이름을 변경하고 클러스터에 다시 조인해야 합니다.

액티브-액티브 클러스터의 DB 인스턴스 이름을 변경하려면

1. SQL 클라이언트에서 DB 인스턴스에 연결하고 mysql.rds_group_replication_stop 저장 프로시저를 호출합니다.

```
call mysql.rds_group_replication_stop();
```

2. DB 인스턴스 이름 변경의 지침에 따라 DB 인스턴스의 이름을 바꿉니다.
3. 액티브-액티브 클러스터의 DB 인스턴스와 연결된 각 DB 파라미터 그룹의 `group_replication_group_seeds` 파라미터를 수정합니다.

   파라미터 설정에서 이전 DB 인스턴스 엔드포인트를 새 DB 인스턴스 엔드포인트로 교체합니다. 파라미터 설정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요.

4. SQL 클라이언트에서 DB 인스턴스에 연결하고 mysql.rds_group_replication_start 저장 프로시저를 호출합니다.

```
call mysql.rds_group_replication_start(0);
```

액티브-액티브 클러스터에서 DB 인스턴스 제거

액티브-액티브 클러스터에서 DB 인스턴스를 제거하면 독립형 DB 인스턴스로 되돌아갑니다.

액티브-액티브 클러스터에서 DB 인스턴스를 제거하려면

1. SQL 클라이언트에서 DB 인스턴스에 연결하고 mysql.rds_group_replication_stop 저장 프로시저를 호출합니다.

   ```
   call mysql.rds_group_replication_stop();
   ```

2. 액티브-액티브 클러스터에 남아 있는 DB 인스턴스의 group_replication_group_seeds 파라미터를 수정합니다.

   group_replication_group_seeds 파라미터에서 액티브-액티브 클러스터에서 제거하려는 DB 인스턴스를 삭제합니다. 파라미터 설정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요.

3. 액티브-액티브 클러스터에서 제거하려는 DB 인스턴스가 더 이상 클러스터에 속하지 않도록 파라미터를 수정합니다.

   DB 인스턴스를 다른 파라미터 그룹과 연결하거나 DB 인스턴스와 연결된 DB 파라미터 그룹의 파라미터를 수정할 수 있습니다. 수정할 파라미터로는 group_replication_group_name, rds.group_replication_enabled, group_replication_group_seeds가 있습니다. 액티브-액티브 클러스터 파라미터에 대한 자세한 내용은 액티브-액티브 클러스터의 필수 파라미터 설정 섹션을 참조하세요.

   DB 파라미터 그룹에서 파라미터를 수정하는 경우 DB 파라미터 그룹이 액티브-액티브 클러스터의 다른 DB 인스턴스와 연결되어 있지 않은지 확인합니다.

4. 새 파라미터 설정을 적용하려면 액티브-액티브 클러스터에서 제거한 DB 인스턴스를 재부팅하세요.

   지침은 DB 인스턴스 재부팅 단원을 참조하세요.

복제를 사용하여 MySQL DB 인스턴스에서 데이터 내보내기

RDS for MySQL DB 인스턴스에서 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스로 데이터를 내보내려면 복제 기능을 사용하면 됩니다. 이 시나리오에서 MySQL DB 인스턴스는 소스 MySQL DB 인스턴스이고, Amazon RDS 외부에서 실행 중인 MySQL 인스턴스는 외부 MySQL 데이터베이스입니다.

외부 MySQL 데이터베이스는 데이터 센터의 온프레미스 또는 Amazon EC2 인스턴스에서 실행할 수 있습니다. 외부 MySQL 데이터베이스는 원본 MySQL DB 인스턴스와 동일한 버전 또는 해당 버전 이상을 실행해야 합니다.

외부 MySQL 데이터베이스에 대한 복제는 원본 MySQL DB 인스턴스에서 데이터베이스를 내보내는 데 걸리는 시간 중에만 지원됩니다. 데이터 내보내기가 완료되어 애플리케이션이 외부 MySQL 인스턴스에 대한 액세스를 시작할 수 있을 때 복제가 종료되어야 합니다.

아래 목록에 수행할 단계가 나와 있습니다. 각 단계에 대해서는 이후 섹션에서 자세히 설명합니다.

1. 외부 MySQL DB 인스턴스를 준비합니다.
2. 복제할 원본 MySQL DB 인스턴스를 준비합니다.
3. mysqldump 유틸리티를 사용하여 원본 MySQL DB 인스턴스에서 외부 MySQL 데이터베이스로 데이터베이스를 전송합니다.
4. 외부 MySQL 데이터베이스에 대한 복제를 시작합니다.
5. 내보내기가 완료된 후 복제를 중지합니다.

외부 MySQL 데이터베이스 준비

외부 MySQL 데이터베이스를 준비하려면 다음 단계를 수행하세요.

외부 MySQL 데이터베이스를 준비하려면

1. 외부 MySQL 데이터베이스를 설치합니다.
2. 마스터 사용자로 외부 MySQL 데이터베이스에 연결합니다. 그런 다음, 데이터베이스에 액세스하는 서비스, 애플리케이션 및 관리자를 지원하는 데 필요한 사용자를 생성합니다.
3. MySQL 설명서의 지침에 따라 외부 MySQL 데이터베이스를 복제본으로 준비합니다. 자세한 내용은 MySQL 설명서의 Setting the Replica Configuration을 참조하세요.
4. 내보내기 중에 외부 MySQL 데이터베이스가 읽기 전용 복제본으로 작동하도록 송신 규칙을 구성합니다. 송신 규칙을 사용하면 복제 중에 외부 MySQL 데이터베이스를 원본 MySQL DB 인스턴스

에 연결할 수 있습니다. 원본 MySQL DB 인스턴스의 포트와 IP 주소에 대한 Transmission Control Protocol(TCP) 연결을 허용하는 송신 규칙을 지정합니다.

환경에 적합한 송신 규칙을 지정합니다.

- 외부 MySQL 데이터베이스가 Amazon VPC 서비스를 기반으로 하는 Virtual Private Cloud(VPC)의 Amazon EC2 인스턴스에서 실행 중인 경우 VPC 보안 그룹에서 송신 규칙을 지정합니다. 자세한 내용은 보안 그룹을 통한 액세스 제어 단원을 참조하십시오.
- 외부 MySQL 데이터베이스가 온프레미스에 설치되어 있는 경우 방화벽에서 송신 규칙을 지정합니다.

5. 외부 MySQL 데이터베이스가 VPC에서 실행 중인 경우, 보안 그룹 송신 규칙 외에 VPC ACL(액세스 제어 목록) 규칙에 대한 규칙을 구성합니다.

   - 원본 MySQL DB 인스턴스의 IP 주소에서 포트 1024–65535로의 TCP 트래픽을 허용하는 ACL 송신 규칙을 구성합니다.
   - 원본 MySQL DB 인스턴스의 포트와 IP 주소로의 아웃바운드 TCP 트래픽을 허용하는 ACL 송신 규칙을 구성합니다.

   Amazon VPC 네트워크 ACL에 대한 자세한 내용은 Amazon VPC 사용 설명서의 네트워크 ACL을 참조하세요.

6. (선택 사항) 복제 오류를 방지하려면 `max_allowed_packet` 파라미터를 최대 크기로 설정하세요. 이 설정을 사용하는 것이 좋습니다.

원본 MySQL DB 인스턴스 준비

원본 MySQL DB 인스턴스를 복제 소스로 준비하려면 다음 단계를 수행하세요.

원본 MySQL DB 인스턴스를 준비하려면

1. 클라이언트 컴퓨터에서 복제를 설정하는 동안 이진 로그를 저장하기에 충분한 디스크 공간을 사용할 수 있는지 확인합니다.
2. 소스 MySQL DB 인스턴스에 연결하고 MySQL 설명서의 복제를 위한 사용자 생성에 나와 있는 지침에 따라 복제 계정을 생성합니다.
3. 원본 MySQL DB 인스턴스를 실행 중인 시스템에서 복제 중에 외부 MySQL 데이터베이스의 연결을 허용하도록 송신 규칙을 구성합니다. 외부 MySQL 데이터베이스의 IP 주소에서 MySQL DB 인스턴스가 사용하는 포트에 대한 TCP 연결을 허용하는 수신 규칙을 지정합니다.

4. 송신 규칙을 지정합니다.

   - VPC에서 원본 MySQL DB 인스턴스가 실행 중인 경우 VPC 보안 그룹에서 수신 규칙을 지정합니다. 자세한 내용은 보안 그룹을 통한 액세스 제어 단원을 참조하십시오.

5. MySQL DB 인스턴스가 VPC에서 실행 중인 경우 보안 그룹 수신 규칙 이외에 VPC ACL 규칙을 구성합니다.

   - 외부 MySQL 데이터베이스의 IP 주소에서 Amazon RDS 인스턴스가 사용하는 포트에 대한 TCP 연결을 허용하도록 ACL 수신 규칙을 구성합니다.
   - 포트 1024–65535에서 외부 MySQL 데이터베이스의 IP 주소로 TCP 연결을 허용하도록 ACL 송신 규칙을 구성합니다.

   Amazon VPC 네트워크 ACL에 대한 자세한 내용은 Amazon VPC 사용 설명서의 네트워크 ACL을 참조하세요.

6. 내보내기 중에 아무런 이진 로그도 제거되지 않을 만큼 백업 보존 기간이 충분히 길게 설정되어 있는지 확인합니다. 내보내기가 완료되기 전에 제거되는 로그가 있으면, 복제를 처음부터 다시 시작해야 합니다. 백업 보존 기간 설정에 대한 자세한 정보는 백업 소개 단원을 참조하십시오.

7. `mysql.rds_set_configuration` 저장 프로시저를 사용하여 내보내기 중에 이진 로그가 제거되지 않을 만큼 충분히 길게 이진 로그 보존 기간을 설정합니다. 자세한 내용은 MySQL 이진 로그 액세스 단원을 참조하십시오.

8. 원본 MySQL DB 인스턴스의 이진 로그가 제거되지 않도록 더욱 확실히 하려면 원본 MySQL DB 인스턴스에서 Amazon RDS 읽기 전용 복제본을 생성합니다. 자세한 내용은 읽기 전용 복제본 생성 단원을 참조하십시오.

9. Amazon RDS 읽기 전용 복제본이 생성된 후 `mysql.rds_stop_replication` 저장 프로시저를 호출하여 복제 프로세스를 중지합니다. 원본 MySQL DB 인스턴스가 더 이상 이진 로그 파일을 제거하지 않을 것이므로, 복제 프로세스에 해당 파일을 사용할 수 있습니다.

10. (선택 사항) 복제 오류를 방지하려면 `max_allowed_packet` 파라미터와 `slave_max_allowed_packet` 파라미터 모두 최대 크기로 설정하세요. 두 파라미터의 최대 크기는 모두 1GB입니다. 두 파라미터 모두에 대해 이 설정을 사용하는 것이 좋습니다. 파라미터 설정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정을 참조하세요.

데이터베이스 복사

데이터베이스를 복사하려면 다음 단계를 수행하세요.

데이터베이스를 복사하려면

1. 소스 MySQL DB 인스턴스의 RDS 읽기 전용 복제본에 연결하고 MySQL `SHOW REPLICA STATUS\G` 문을 실행합니다. 다음에 대한 값을 확인합니다.

   - `Master_Host`
   - `Master_Port`
   - `Master_Log_File`
   - `Exec_Master_Log_Pos`

   > **Note**
   >
   > 이전 버전의 MySQL에는 `SHOW SLAVE STATUS` 대신 `SHOW REPLICA STATUS`가 사용되었습니다. 8.0.23 이전 MySQL 버전을 사용하는 경우 `SHOW SLAVE STATUS`를 사용합니다.

2. mysqldump 유틸리티를 사용하여 Amazon RDS에서 로컬 클라이언트 컴퓨터로 데이터를 복사하는 스냅샷을 생성합니다. 클라이언트 컴퓨터에 복제할 데이터베이스의 `mysqldump` 파일을 보관하기에 충분한 공간이 있는지 확인합니다. 매우 큰 데이터베이스의 경우 이 프로세스는 여러 시간이 걸릴 수 있습니다. MySQL 설명서의 mysqldump를 사용하여 데이터 스냅샷 생성에 나와 있는 지침을 따르세요.

   다음 예에서는 클라이언트에서 `mysqldump`를 실행하고 덤프를 파일에 씁니다.

   대상 LinuxmacOS, 또는Unix:

   ```
   mysqldump -h source_MySQL_DB_instance_endpoint \
       -u user \
       -ppassword \
       --port=3306 \
       --single-transaction \
       --routines \
       --triggers \
       --databases  database database2 > path/rds-dump.sql
   ```

   Windows의 경우:

   ```
   mysqldump -h source_MySQL_DB_instance_endpoint ^
   ```

```
    -u user ^
    -ppassword ^
    --port=3306 ^
    --single-transaction ^
    --routines ^
    --triggers ^
    --databases  database database2 > path\rds-dump.sql
```

외부 MySQL 데이터베이스에 백업 파일을 로드할 수 있습니다. 자세한 내용은 MySQL 설명서의 SQL 형식 백업 다시 로드를 참조하세요. 다른 유틸리티를 실행하여 외부 MySQL 데이터베이스로 데이터를 로드할 수 있습니다.

내보내기 완료

내보내기를 완료하려면 다음 단계를 수행하세요.

내보내기를 완료하려면

1. MySQL CHANGE MASTER 문을 사용하여 외부 MySQL 데이터베이스를 구성합니다. 사용자가 부여한 REPLICATION SLAVE 권한의 ID와 암호를 지정합니다. RDS 읽기 전용 복제본에서 실행한 MySQL SHOW REPLICA STATUS\G 문에서 얻은 Master_Host, Master_Port, Relay_Master_Log_File 및 Exec_Master_Log_Pos 값을 지정합니다. 자세한 내용은 MySQL 설명서의 CHANGE MASTER TO 문을 참조하세요.

   > **Note**
   >
   > 이전 버전의 MySQL에는 SHOW SLAVE STATUS 대신 SHOW REPLICA STATUS가 사용되었습니다. 8.0.23 이전 MySQL 버전을 사용하는 경우 SHOW SLAVE STATUS를 사용합니다.

2. MySQL START REPLICA 명령을 사용하여 소스 MySQL DB 인스턴스에서 외부 MySQL 데이터베이스로 복제를 시작합니다.

   이렇게 하면 원본 MySQL DB 인스턴스에서 복제가 시작되고 Amazon RDS 읽기 전용 복제본에서 복제를 중지한 후 발생한 모든 원본 변경 사항이 내보내집니다.

> **Note**
>
> 이전 버전의 MySQL에는 `START SLAVE` 대신 `START REPLICA`가 사용되었습니다. 8.0.23 이전 MySQL 버전을 사용하는 경우 `START SLAVE`를 사용합니다.

3. 외부 MySQL 데이터베이스에서 MySQL `SHOW REPLICA STATUS\G` 명령을 실행하여 이 데이터베이스가 읽기 전용 복제본으로 작동 중인지 확인합니다. 결과 해석에 대한 자세한 내용은 MySQL 설명서의 SHOW SLAVE | REPLICA STATUS 문을 참조하세요.
4. 외부 MySQL 데이터베이스에서의 복제가 원본 MySQL DB 인스턴스를 따라잡은 후, MySQL `STOP REPLICA` 명령을 사용하여 원본 MySQL DB 인스턴스에서의 복제를 중지하세요.

> **Note**
>
> 이전 버전의 MySQL에는 `STOP SLAVE` 대신 `STOP REPLICA`가 사용되었습니다. 8.0.23 이전 MySQL 버전을 사용하는 경우 `STOP SLAVE`를 사용합니다.

5. Amazon RDS 읽기 전용 복제본에서 `mysql.rds_start_replication` 저장 프로시저를 호출합니다. 이렇게 하면 Amazon RDS가 원본 MySQL DB 인스턴스에서 이진 로그 파일 제거를 시작할 수 있습니다.

MySQL DB 인스턴스 옵션

아래에는 MySQL DB 엔진을 실행하는 Amazon RDS 인스턴스에 사용 가능한 옵션 또는 추가 기능에 대한 설명이 나와 있습니다. 이러한 옵션들을 활성화하려면 먼저 사용자 정의 옵션 그룹에 추가한 다음 옵션 그룹과 DB 인스턴스를 연동시켜야 합니다. 옵션 그룹 작업에 대한 자세한 내용은 옵션 그룹 작업 단원을 참조하십시오.

Amazon RDS는 다음의 MySQL 옵션을 지원합니다.

옵션	옵션 ID	엔진 버전
MySQL에 대한 MariaDB 감사 플러그인 지원	`MARIADB_AUDIT_PLUGIN`	모든 MySQL 8.4 버전 MySQL 8.0.28 이상의 8.0 버전 모든 MySQL 5.7 버전
MySQL memcached 지원	`MEMCACHED`	모든 MySQL 5.7 및 8.0 버전

MySQL에 대한 MariaDB 감사 플러그인 지원

Amazon RDS는 오픈 소스 MariaDB 감사 플러그인을 기반으로 MySQL 데이터베이스 인스턴스용 감사 플러그인을 제공합니다. 자세한 내용은 MySQL Server GitHub 리포지토리의 감사 플러그인을 참조하세요.

> **Note**
>
> MySQL용 감사 플러그인은 MariaDB 감사 플러그인을 기반으로 합니다. 이 문서에서는 이를 MariaDB 감사 플러그인이라고 합니다.

MariaDB 감사 플러그인은 사용자의 데이터베이스 로그온, 데이터베이스에 대해 실행되는 쿼리 등의 데이터베이스 활동을 기록합니다. 데이터베이스 활동 기록은 로그 파일에 저장됩니다.

감사 플러그인 옵션 설정

Amazon RDS는 MariaDB 감사 플러그인 옵션의 다음 설정을 지원합니다.

옵션 설정	유효한 값	기본값	설명
SERVER_AUDIT_FILE_PATH	/rdsdbdata/log/audit/	/rdsdbdata/log/audit/	로그 파일의 위치. 로그 파일에는 SERVER_AUDIT_EVENTS 에서 지정된 활동 기록이 포함되어 있습니다. 자세한 내용은 데이터베이스 로그 파일 보기 및 나열 및 MySQL 데이터베이스 로그 파일 단원을 참조하십시오.
SERVER_AUDIT_FILE_ROTATE_SIZE	1–1000000000	1000000	도달 시 파일 로테이션을 초래하는 바이트 크기. 자세한 내용은 RDS for MySQL 데이터베이스 로그 개요 섹션을 참조하세요.
SERVER_AUDIT_FILE_ROTATIONS	0–100	9	server_audit_output_type=file 일 경우 저장할 로그 교체 수입니다. 0으로 설정하면 로그 파일이 교체되지 않습니다. 자세한 내용은 RDS for MySQL 데이터베이스 로그 개요 및 데이터베이스 로그 파일 다운로드 단원을 참조하세요.

옵션 설정	유효한 값	기본값	설명
SERVER_AUDIT_EVENTS	CONNECT, QUERY, QUERY_DDL, QUERY_DML, QUERY_DML_NO_SELECT, QUERY_DCL	CONNECT, QUERY	로그에 기록할 활동 유형. MariaDB 감사 플러그인 설치 자체가 로깅됩니다. • CONNECT: 성공/실패한 데이터베이스 연결과 데이터베이스 연결 해제를 로깅합니다. • QUERY: 데이터베이스에 대해 실행된 모든 쿼리의 텍스트를 로깅합니다. • QUERY_DDL : QUERY 이벤트와 유사하지만 데이터 정의 언어(DDL) 쿼리(CREATE, ALTER 등)만 반환합니다. • QUERY_DML : QUERY 이벤트와 유사하지만 데이터 조작 언어(DML) 쿼리(INSERT, UPDATE 등과 SELECT)만 반환합니다. • QUERY_DML_NO_SELECT : QUERY_DML 이벤트와 유사하지만 SELECT 쿼리를 로깅하지 않습니다. • QUERY_DCL : QUERY 이벤트와 유사하지만 데이터 제어 언어(DCL) 쿼리(GRANT, REVOKE 등)만 반환합니다. MySQL에서 TABLE이 지원되지 않습니다.
SERVER_AUDIT_INCL_USERS	복수의 쉼표로 분리된 값	없음	지정된 사용자들의 활동만을 포함하십시오. 기본적으로 활동은 모든 사용자에 대해 기록됩니다. SERVER_AUDIT_INCL_USERS 및 SERVER_AUDIT_EXCL_USERS 는 상호 배타적입니다. SERVER_AUDIT_INCL_USERS 에 값을 추가하는 경우 SERVER_AUDIT_EXCL_USERS 에 값이 추가되지 않았는지 확인합니다.

옵션 설정	유효한 값	기본값	설명
SERVER_AUDIT_EXCL_USERS	복수의 쉼표로 분리된 값	없음	지정된 사용자들의 활동을 제외하십시오. 기본적으로 활동은 모든 사용자에 대해 기록됩니다. SERVER_AUDIT_INCL_USERS 및 SERVER_AUDIT_EXCL_USERS 는 상호 배타적입니다. SERVER_AUDIT_EXCL_USERS 에 값을 추가하는 경우 SERVER_AUDIT_INCL_USERS 에 값이 추가되지 않았는지 확인합니다. rdsadmin 사용자는 데이터베이스 상태를 확인하기 위해 1초마다 데이터베이스에 쿼리를 요청합니다. 다른 설정에 따라 이 활동은 로그 파일의 크기를 아주 빨리 대폭 증가시킬 수 있습니다. 이 활동을 기록할 필요가 없는 경우, rdsadmin 사용자를 SERVER_AUDIT_EXCL_USERS 목록에 추가하십시오. **Note** CONNECT 활동은 해당 사용자가 이 옵션 설정에 지정되었다 해도 모든 사용자에 대해 기록됩니다.
SERVER_AUDIT_LOGGING	ON	ON	로깅이 활성화되었습니다. 유일한 유효 값은 ON입니다. Amazon RDS는 로깅 비활성화를 지원하지 않습니다. 로깅을 비활성화하려면 MariaDB 감사 플러그인을 제거하십시오. 자세한 내용은 MariaDB 감사 플러그인 제거하기 섹션을 참조하세요.
SERVER_AUDIT_QUERY_LOG_LIMIT	0–2147483647	1024	레코드에서 쿼리 문자열의 길이 제한.

MariaDB 감사 플러그인 추가하기

MariaDB 감사 플러그인을 DB 인스턴스에 추가하는 일반적인 프로세스는 다음과 같습니다.

- 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 복사 또는 수정합니다.
- 옵션을 옵션 그룹에 추가합니다.
- 옵션 그룹을 DB 인스턴스에 연동시킵니다.

MariaDB 감사 플러그인을 추가한 후 DB 인스턴스를 재시작할 필요가 없습니다. 옵션 그룹이 활성화되자마자 감사가 즉시 시작됩니다.

> **Important**
>
> MariaDB 감사 플러그인을 DB 인스턴스에 추가하면 작동이 중단될 수 있습니다. 유지 관리 기간 또는 데이터베이스 워크로드가 적은 시간에 MariaDB 감사 플러그인을 추가하는 것이 좋습니다.

MariaDB 감사 플러그인을 추가하려면,

1. 사용할 옵션 그룹을 결정합니다. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 사용합니다. 기존 옵션 그룹을 사용하려면 다음 단계로 건너뛰십시오. 그렇지 않다면 사용자 지정 DB 옵션 그룹을 생성합니다. 엔진으로 mysql을 선택하고, 메이저 엔진 버전으로 5.7, 8.0 또는 8.4를 선택합니다. 자세한 내용은 옵션 그룹 생성 단원을 참조하십시오.
2. MARIADB_AUDIT_PLUGIN 옵션을 옵션 그룹에 추가하고 옵션 설정을 구성하십시오. 옵션 추가에 대한 자세한 내용은 옵션 그룹에 옵션 추가 섹션을 참조하세요. 각 설정에 대한 자세한 내용은 감사 플러그인 옵션 설정 단원을 참조하십시오.
3. 옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스에 적용합니다.
   - 새 DB 인스턴스의 경우, 인스턴스를 시작할 때 옵션 그룹을 적용합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
   - 기존 DB 인스턴스의 경우, 해당 인스턴스를 수정하고 새 옵션 그룹을 연결하여 옵션 그룹을 적용합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

감사 로그 형식

로그 파일은 UTF-8 형식의 쉼표로 구분된 변수(CSV) 파일로 표시됩니다.

> **Tip**
>
> 로그 파일 항목은 순서대로 나열되지 않습니다. 항목의 순서를 정하려면 타임스탬프 값을 사용합니다. 최신 이벤트를 보기 위해 모든 로그 파일을 확인해야 하는 경우가 있을 수 있습니다.
> 로그 데이터를 보다 유연하게 정렬하고 검색하려면 CloudWatch에 감사 로그를 업로드하고 CloudWatch 인터페이스를 사용하여 보는 설정을 켜세요.
> 더 많은 유형의 필드와 JSON 형식의 출력이 있는 감사 데이터를 보기 위해 데이터베이스 활동 스트림 기능을 사용할 수도 있습니다. 자세한 내용은 데이터베이스 활동 스트림을 사용하여 Amazon RDS 모니터링 단원을 참조하십시오.

감사 로그 파일은 지정된 순서대로 다음의 쉼표로 구분된 정보를 행에 포함합니다.

필드	설명
타임스탬프	로깅된 이벤트의 YYYYMMDD 다음 HH:MI:SS(24시간 형식).
serverhost	이벤트가 기록되는 인스턴스의 이름입니다.
사용자 이름	사용자의 연결된 사용자 이름입니다.
host	사용자가 연결한 호스트입니다.
connectionid	기록된 작업의 연결 ID 번호입니다.
queryid	관계형 테이블 이벤트 및 관련 쿼리를 검색하는 데 사용할 수 있는 쿼리 ID 번호입니다. TABLE 이벤트의 경우 여러 줄이 추가됩니다.
작업을 통해 처리 속도를 높일 수 있습니다	기록된 작업 유형입니다. 가능한 값은 CONNECT, QUERY, READ, WRITE, CREATE, ALTER, RENAME 및 DROP입니다.
데이터베이스	USE 명령에 의해 설정된 활성 데이터베이스입니다.
객체	QUERY 이벤트의 경우 이 값은 데이터베이스에서 수행한 쿼리를 나타냅니다. TABLE 이벤트의 경우 이 값은 테이블 이름을 나타냅니다.
retcode	기록된 작업의 반환 코드입니다.

필드	설명
connection_type	서버 연결의 보안 상태입니다. 가능한 값은 다음과 같습니다. • 0 - 정의되지 않음 • 1 – TCP/IP • 2 – 소켓 • 3 – 명명된 파이프 • 4 – SSL/TLS • 5 – 공유 메모리

MariaDB 감사 플러그인 로그 보기 및 다운로드

MariaDB 감사 플러그인을 활성화한 후 다른 텍스트 기반 로그 파일에 액세스하는 것과 동일한 방식으로 로그 파일의 결과에 액세스할 수 있습니다. 감사 로그 파일은 `/rdsdbdata/log/audit/`에 있습니다. 콘솔에서 로그 파일 보기에 대한 자세한 내용은 데이터베이스 로그 파일 보기 및 나열을(를) 참조하십시오. 로그 파일 다운로드에 대한 자세한 내용은 데이터베이스 로그 파일 다운로드을(를) 참조하십시오.

MariaDB 감사 플러그인 설정 수정

MariaDB 감사 플러그인을 활성화한 후 설정을 수정할 수 있습니다. 옵션 설정을 변경하는 방법에 대한 자세한 내용은 옵션 설정 수정을(를) 참조하십시오. 각 설정에 대한 자세한 내용은 감사 플러그인 옵션 설정 단원을 참조하십시오.

MariaDB 감사 플러그인 제거하기

Amazon RDS는 MariaDB 감사 플러그인에서의 로깅 끄기를 지원하지 않습니다. 다만 DB 인스턴스에서 플러그인을 제거할 수는 있습니다. MariaDB 감사 플러그인을 제거할 때 DB 인스턴스가 자동으로 재시작하여 감사가 중지됩니다.

MariaDB 감사 플러그인을 DB 인스턴스에서 제거하려면 다음 중 하나를 실행하십시오.

- MariaDB 감사 플러그인이 속한 옵션 그룹에서 MariaDB 감사 플러그인 옵션을 제거합니다. 이 변경은 해당 옵션 그룹을 사용하는 모든 DB 인스턴스에 영향을 미칩니다. 자세한 내용은 옵션 그룹에서 옵션 제거 섹션을 참조하세요.

- DB 인스턴스를 수정하고, 플러그인이 포함되지 않은 다른 옵션 그룹을 지정합니다. 이 변경은 단일 DB 인스턴스에 영향을 미칩니다. 기본(빈) 옵션 그룹을 지정하거나 다른 사용자 지정 옵션 그룹을 지정할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

MySQL memcached 지원

Amazon RDS는 MySQL 5.6에서 도입된 InnoDB 테이블에 대한 memcached 인터페이스 사용을 지원합니다. memcached API가 있으면 애플리케이션에서 NoSQL 키-값 데이터 스토어와 비슷한 방식으로 InnoDB 테이블을 사용할 수 있습니다.

> **Note**
>
> MySQL 8.4에서는 더 이상 memcached 인터페이스를 사용할 수 없습니다. DB 인스턴스를 MySQL 8.4로 업그레이드할 때는 기존 옵션 그룹에서 memcached를 비활성화해야 합니다.

memcached 인터페이스는 간단한 키 기반 캐시입니다. 애플리케이션은 이 memcached를 사용하여 키-값 데이터 쌍을 삽입하거나, 조작하거나, 캐시에서 가져옵니다. MySQL 5.6은 데몬 서비스를 이용한 플러그인이 추가되면서 memcached 프로토콜을 통해 InnoDB 테이블의 데이터를 노출시킵니다. MySQL memcached 플러그인에 대한 자세한 내용은 InnoDB와 memcached의 통합을 참조하세요.

RDS for MySQL DB 인스턴스에 memcached 지원을 활성화하는 방법

1. 먼저 memcached 인터페이스에 대한 액세스를 제어하는 데 사용할 보안 그룹을 결정합니다. 기존에 SQL 인터페이스를 사용하는 애플리케이션과 앞으로 memcached 인터페이스에 액세스할 애플리케이션이 동일한 경우에는 기존에 SQL 인터페이스에서 사용하는 VPC 또는 DB 보안 그룹을 그대로 사용할 수 있습니다. 하지만 memcached 인터페이스에 액세스하는 애플리케이션이 다를 경우에는 새로운 VPC 또는 DB 보안 그룹을 정의해야 합니다. 보안 그룹 관리에 대한 자세한 내용은 보안 그룹을 통한 액세스 제어 단원을 참조하세요.
2. 사용자 정의 DB 옵션 그룹을 생성한 후 엔진 유형 및 버전으로 MySQL을 선택합니다. 옵션 그룹의 생성에 대한 자세한 내용은 옵션 그룹 생성을(를) 참조하세요.
3. [MEMCACHED] 옵션을 옵션 그룹에 추가합니다. memcached 인터페이스가 사용할 포트를 비롯해 인터페이스에 대한 액세스 제어에 사용할 보안 그룹을 지정합니다. 옵션 추가에 대한 자세한 내용은 옵션 그룹에 옵션 추가 섹션을 참조하세요.
4. 필요하다면 옵션 설정을 변경하여 memcached 파라미터를 구성합니다. 옵션 설정을 변경하는 방법에 대한 자세한 내용은 옵션 설정 수정을(를) 참조하십시오.
5. 옵션 그룹을 인스턴스에 적용합니다. Amazon RDS는 옵션 그룹이 적용될 때 해당 인스턴스에 대한 memcached 지원을 활성화합니다.

- 인스턴스 실행 시 사용자 정의 옵션 그룹을 지정하여 새로운 인스턴스에 대한 memcached 지원을 활성화합니다. MySQL 인스턴스 실행에 대한 자세한 내용은 Amazon RDS DB 인스턴스 생성 단원을 참조하십시오.
- 인스턴스 설정 변경 시 사용자 정의 옵션 그룹을 지정하여 기존 인스턴스에 대한 memcached 지원을 활성화합니다. DB 인스턴스 변경에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

6. MySQL 테이블에서 memcached 인터페이스를 통해 액세스할 수 있는 열을 지정합니다. memcached 플러그인은 containers라는 이름의 전용 데이터베이스에 innodb_memcache라는 이름의 카탈로그 테이블을 생성합니다. 행을 containers 테이블에 삽입하여 memcached를 통해 액세스할 수 있도록 InnoDB 테이블을 매핑합니다. InnoDB 테이블에서 memcached 키 값을 저장하는 데 사용할 열 하나와, 이 키와 연동된 데이터 값을 저장하는 데 사용할 하나 이상의 열을 지정합니다. 또한 memcached 애플리케이션이 해당 열 세트를 참조하는 데 사용할 이름도 지정합니다. 행을 containers 테이블에 삽입하는 방법에 대한 자세한 내용은 InnoDB memcached 플러그인의 내부 요소를 참조하십시오. InnoDB 테이블을 매핑하고 memcached를 통해 액세스하는 예제는 InnoDB memcached 플러그인에 대한 애플리케이션 작성을 참조하세요.
7. memcached 인터페이스에 액세스하는 애플리케이션이 SQL 인터페이스를 사용하는 애플리케이션과 다른 컴퓨터 또는 EC2 인스턴스에 연결되어 있는 경우에는 이 컴퓨터에 대한 연결 정보를 MySQL 인스턴스와 연동되어 있는 VPC 보안 그룹에 추가합니다. 보안 그룹 관리에 대한 자세한 내용은 보안 그룹을 통한 액세스 제어 단원을 참조하세요.

인스턴스에 대한 memcached 지원을 비활성화하려면 인스턴스를 변경하여 MySQL 버전의 기본 옵션 그룹을 지정합니다. DB 인스턴스 변경에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

MySQL memcached 보안 고려 사항

memcached 프로토콜은 사용자 인증을 지원하지 않습니다. MySQL memcached 보안 고려 사항에 대한 자세한 내용은 MySQL 설명서의 InnoDB memcached 플러그인의 보안 고려 사항을 참조하세요.

다음은 memcached 인터페이스의 보안을 강화하는 데 효과적인 작업입니다.

- MEMCACHED 옵션을 옵션 그룹에 추가할 때 기본 11211이 아닌 다른 포트를 지정합니다.
- 기존에 신뢰할 수 있는 클라이언트 주소 및 EC2 인스턴스에 대한 액세스를 제한하는 VPC 보안 그룹과 memcached 인터페이스가 연동되어 있는지 확인합니다. 보안 그룹 관리에 대한 자세한 내용은 보안 그룹을 통한 액세스 제어 섹션을 참조하세요.

MySQL memcached 연결 정보

memcached 인터페이스에 연결하려면 애플리케이션이 Amazon RDS 인스턴스의 DNS 이름과 memcached 포트 번호를 모두 지정해야 합니다. 예를 들어 인스턴스의 DNS 이름이 my-cache-instance.cg034hpkmmjt.region.rds.amazonaws.com이고, memcached 인스턴스가 포트 11212를 사용한다면 PHP에 지정되는 연결 정보는 다음과 같습니다.

```
<?php

$cache = new Memcache;
$cache->connect('my-cache-instance.cg034hpkmmjt.region.rds.amazonaws.com',11212);
?>
```

MySQL DB 인스턴스의 DNS 이름과 memcached 포트를 찾으려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. AWS Management Console 우측 상단 모서리에서 DB 인스턴스가 속한 리전을 선택합니다.
3. 탐색 창에서 데이터베이스를 선택합니다.
4. 세부 정보를 표시하고자 하는 MySQL DB 인스턴스 이름을 선택합니다.
5. [Connect] 섹션에 있는 [Endpoint] 필드의 값을 기록해둡니다. DNS 이름은 엔드포인트와 같습니다. 또한 [Connect] 섹션에 있는 포트는 memcached 인터페이스에 액세스하는 데 사용되지 않습니다.
6. 세부 정보 섹션의 옵션 그룹 필드에 나열된 이름을 기록해둡니다.
7. 탐색 창에서 옵션 그룹을 선택합니다.
8. MySQL DB 인스턴스가 사용하는 옵션 그룹의 이름을 선택하여 옵션 그룹 세부 정보를 표시합니다. [Options] 섹션에 있는 [MEMCACHED] 옵션의 [Port] 설정 값을 기록해둡니다.

MySQL memcached 옵션 설정

Amazon RDS는 MySQL memcached 파라미터를 Amazon RDS MEMCACHED 옵션의 옵션 설정으로 노출시킵니다.

MySQL memcached 파라미터

- DAEMON_MEMCACHED_R_BATCH_SIZE – 커밋으로 새로운 트랜잭션을 시작하기 전에 실행해야 할 memcached 읽기 연산(get) 수를 지정하는 정수입니다. 허용 값은 1~4294967295이고, 기본값은 1입니다. 옵션을 적용하려면 인스턴스를 다시 시작해야 합니다.
- DAEMON_MEMCACHED_W_BATCH_SIZE – 커밋으로 새로운 트랜잭션을 시작하기 전에 실행해야 할 memcached 쓰기 연산(add, set, incr 등) 수를 지정하는 정수입니다. 허용 값은 1~4294967295이고, 기본값은 1입니다. 옵션을 적용하려면 인스턴스를 다시 시작해야 합니다.
- INNODB_API_BK_COMMIT_INTERVAL – InnoDB memcached 인터페이스를 사용하는 자동 커밋 모드의 유휴 연결(idle connection) 횟수를 지정하는 정수입니다. 허용 값은 1~1073741824이고, 기본값은 5입니다. 인스턴스를 다시 시작할 필요 없이 옵션이 바로 적용됩니다.
- INNODB_API_DISABLE_ROWLOCK – InnoDB memcached 인터페이스 사용 시 행 잠금 기능을 활성화(1. true) 또는 비활성화(0. false)하는 부울입니다. 기본 값은 0(false)입니다. 옵션을 적용하려면 인스턴스를 다시 시작해야 합니다.
- INNODB_API_ENABLE_MDL – 0(false)으로 설정 시 InnoDB memcached 플러그인이 사용하는 테이블을 잠그는 부울입니다. 따라서 SQL 인터페이스에서 DDL을 통해 삭제 또는 변경이 불가능합니다. 기본 값은 0(false)입니다. 옵션을 적용하려면 인스턴스를 다시 시작해야 합니다.
- INNODB_API_TRX_LEVEL – memcached 인터페이스에서 처리하는 쿼리의 트랜잭션 격리 수준을 지정하는 정수입니다. 허용 값은 0~3입니다. 기본값은 0입니다. 옵션을 적용하려면 인스턴스를 다시 시작해야 합니다.

Amazon RDS가 위 MySQL memcached 파라미터를 구성하지만 다음 파라미터는 변경할 수 없습니다. DAEMON_MEMCACHED_LIB_NAME, DAEMON_MEMCACHED_LIB_PATH 및 INNODB_API_ENABLE_BINLOG. MySQL 관리자가 daemon_memcached_options를 사용하여 설정하는 파라미터는 Amazon RDS의 개별 MEMCACHED 옵션 설정으로 이용할 수 있습니다.

MySQL daemon_memcached_options 파라미터

- BINDING_PROTOCOL – 사용할 바인딩 프로토콜을 지정하는 문자열입니다. 허용 값은 auto, ascii 또는 binary입니다. 기본 값은 auto이고, 이 경우 서버가 자동으로 클라이언트와 프로토콜을 협상합니다. 옵션을 적용하려면 인스턴스를 다시 시작해야 합니다.
- BACKLOG_QUEUE_LIMIT – 의 처리를 기다리는 네트워크 연결 수를 지정하는 정수입니다.memcached 이 파라미터의 최대 값을 높이면 memcached 인스턴스에 연결할 수 없다는 클라이언트의 오류 메시지가 줄어들 수 있지만 그렇다고 서버 성능이 향상되는 것은 아닙니다. 허용 값은 1~2048이고, 기본값은 1024입니다. 옵션을 적용하려면 인스턴스를 다시 시작해야 합니다.

- CAS_DISABLED – Compare and Swap(CAS) 기능을 활성화(1. true)하거나 비활성화(0. false)하는 부울로서, 이 기능을 활성화하면 항목 1개의 크기가 8바이트까지 줄어듭니다. 기본 값은 0(false)입니다. 옵션을 적용하려면 인스턴스를 다시 시작해야 합니다.
- CHUNK_SIZE – 가장 작은 항목의 키, 값 및 플래그에 할당할 최소 청크 크기(바이트)를 지정하는 정수입니다. 허용 값은 1~48입니다. 기본 값은 48이며, 값이 작을수록 메모리 효율이 크게 개선됩니다. 옵션을 적용하려면 인스턴스를 다시 시작해야 합니다.
- CHUNK_SIZE_GROWTH_FACTOR – 새로운 청크 크기를 조절하는 부동 소수점입니다. 이전 청크 크기와 CHUNK_SIZE_GROWTH_FACTOR를 곱한 값이 새로운 청크의 크기가 됩니다. 허용되는 값은 1~2이고, 기본값은 1.25입니다. 옵션을 적용하려면 인스턴스를 다시 시작해야 합니다.
- ERROR_ON_MEMORY_EXHAUSTED – 1(true)로 설정할 경우, 항목을 저장할 메모리가 없으면 memcached가 항목을 제거하지 않고 오류를 반환하는 부울입니다. 0(false)으로 설정할 경우 메모리가 없으면 memcached가 항목을 제거합니다. 기본 값은 0(false)입니다. 옵션을 적용하려면 인스턴스를 다시 시작해야 합니다.
- MAX_SIMULTANEOUS_CONNECTIONS – 동시에 접속할 수 있는 최대 수를 지정하는 정수입니다. 이 값을 10보다 작게 설정하면 MySQL을 시작하지 못합니다. 허용 값은 10~1024이고, 기본값은 1024입니다. 옵션을 적용하려면 인스턴스를 다시 시작해야 합니다.
- VERBOSITY – MySQL 오류 로그에서 memcached 서비스가 기록할 정보 수준을 지정하는 문자열입니다. 기본값은 v입니다. 옵션을 적용하려면 인스턴스를 다시 시작해야 합니다. 허용 값은 다음과 같습니다.
  - v – 주요 이벤트 루프 실행 중 발생하는 오류와 경고를 기록합니다.
  - vv – v에서 기록하는 정보에 더하여 각 클라이언트 명령과 응답을 기록합니다.
  - vvv – vv에서 기록하는 정보에 더하여 내부의 상태 전환을 기록합니다.

Amazon RDS가 위 MySQL DAEMON_MEMCACHED_OPTIONS 파라미터를 구성하지만 다음 파라미터는 변경할 수 없습니다. DAEMON_PROCESS, LARGE_MEMORY_PAGES, MAXIMUM_CORE_FILE_LIMIT, MAX_ITEM_SIZE, LOCK_DOWN_PAGE_MEMORY, MASK, IDFILE, REQUESTS_PER_EVENT, SOCKET 및 USER.

MySQL 파라미터

기본적으로, MySQL DB 인스턴스는 MySQL 데이터베이스에만 해당되는 DB 파라미터 그룹을 사용합니다. 이 파라미터 그룹에는 MySQL 데이터베이스 엔진에 대한 파라미터가 포함되어 있습니다. 파라미터 그룹 작업 및 파라미터 설정에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.

RDS for MySQL 파라미터는 사용자가 선택한 스토리지 엔진의 기본값으로 설정됩니다. MySQL 파라미터에 대한 자세한 내용은 MySQL 설명서를 참조하세요. MySQL 스토리지 엔진에 대한 자세한 내용은 RDS for MySQL에 대해 지원되는 스토리지 엔진 섹션을 참조하세요.

RDS 콘솔이나 AWS CLI를 사용하여 특정 RDS for MySQL 버전에 대해 사용할 수 있는 파라미터를 볼 수 있습니다. RDS 콘솔의 파라미터 그룹에서 MySQL 파라미터 보기에 대한 자세한 정보는 Amazon RDS에서 DB 파라미터 그룹의 파라미터 값 보기 섹션을 참조하세요.

AWS CLI로 `describe-engine-default-parameters` 명령을 실행하여 RDS for MySQL 버전의 파라미터를 볼 수 있습니다. `--db-parameter-group-family` 옵션에 대해 다음 값 중 하나를 지정할 수 있습니다.

- `mysql8.4`
- `mysql8.0`
- `mysql5.7`

예를 들어 RDS for MySQL 버전 8.0에 대한 파라미터를 보려면 다음 명령을 실행합니다.

```
aws rds describe-engine-default-parameters --db-parameter-group-family mysql8.0
```

출력 결과는 다음과 비슷합니다.

```
{
    "EngineDefaults": {
        "Parameters": [
            {
                "ParameterName": "activate_all_roles_on_login",
                "ParameterValue": "0",
                "Description": "Automatically set all granted roles as active after the
 user has authenticated successfully.",
                "Source": "engine-default",
                "ApplyType": "dynamic",
```

```
                "DataType": "boolean",
                "AllowedValues": "0,1",
                "IsModifiable": true
            },
            {
                "ParameterName": "allow-suspicious-udfs",
                "Description": "Controls whether user-defined functions that have only
 an xxx symbol for the main function can be loaded",
                "Source": "engine-default",
                "ApplyType": "static",
                "DataType": "boolean",
                "AllowedValues": "0,1",
                "IsModifiable": false
            },
            {
                "ParameterName": "auto_generate_certs",
                "Description": "Controls whether the server autogenerates SSL key and
 certificate files in the data directory, if they do not already exist.",
                "Source": "engine-default",
                "ApplyType": "static",
                "DataType": "boolean",
                "AllowedValues": "0,1",
                "IsModifiable": false
            },
        ...
```

RDS for MySQL 버전 8.0에 대한 수정 가능 파라미터를 나열하려면 다음 명령을 실행합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-engine-default-parameters --db-parameter-group-family mysql8.0 \
   --query 'EngineDefaults.Parameters[?IsModifiable==`true`]'
```

Windows의 경우:

```
aws rds describe-engine-default-parameters --db-parameter-group-family mysql8.0 ^
   --query "EngineDefaults.Parameters[?IsModifiable==`true`]"
```

MySQL DB 인스턴스에 대한 공통 DBA 작업

다음 콘텐츠에서는 MySQL 데이터베이스 엔진을 실행하는 DB 인스턴스의 일부 공통 DBA 작업에 대한 Amazon RDS별 구현을 설명합니다. 관리형 서비스 환경을 제공하기 위해 Amazon RDS는 DB 인스턴스에 대해 shell 액세스를 제공하지 않습니다. 또한 고급 권한이 필요한 특정 시스템 절차와 테이블에 대한 액세스를 제한합니다.

Amazon RDS의 MySQL 로그 파일 작업 방법에 대한 자세한 내용은 MySQL 데이터베이스 로그 파일 단원을 참조하십시오.

사전 정의된 사용자 이해

Amazon RDS는 새 RDS for MySQL DB 인스턴스를 사용하여 미리 정의된 여러 사용자를 자동으로 생성합니다. 미리 정의된 사용자 및 권한은 변경할 수 없습니다. 미리 정의된 사용자에 대한 권한은 삭제하거나 이름을 바꾸거나 수정할 수 없습니다. 이를 시도할 시에는 오류가 발생합니다.

- rdsadmin - `superuser` 권한이 있는 관리자가 독립 실행형 MySQL 데이터베이스에서 수행할 수 있는 많은 관리 작업을 처리하기 위해 생성된 사용자입니다. 이 사용자는 RDS for MySQL에서 다양한 관리 작업에 내부적으로 사용됩니다.
- rdsrepladmin – Amazon RDS에서 RDS for MySQL DB 인스턴스 및 클러스터의 복제 작업을 지원하기 위해 내부적으로 사용하는 사용자입니다.

다른 일반적인 DBA 작업에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- RDS for MySQL에 대한 역할 기반 권한 모델
- RDS for MySQL에 대한 동적 권한
- RDS for MySQL에 대한 세션 또는 쿼리 종료
- RDS for MySQL에 대한 현재 복제 오류 건너뛰기
- RDS for MySQL의 충돌 복구 시간 개선을 위한 InnoDB 테이블스페이스 작업
- RDS for MySQL의 글로벌 상태 기록 관리
- MySQL 8.4에서 버퍼 풀 크기 및 다시 실행 로그 용량 구성

RDS for MySQL에 대한 역할 기반 권한 모델

RDS for MySQL 버전 8.0.36부터는 mysql 데이터베이스의 테이블을 직접 수정할 수 없습니다. 특히 grant 테이블에서 데이터 조작 언어(DML) 작업을 수행하여 데이터베이스 사용자를 만들 수 없습니다. 대신 CREATE USER, GRANT, REVOKE 등의 MySQL 계정 관리 문을 사용하여 사용자에게 역할 기반 권한을 부여합니다. 또한 mysql 데이터베이스에 저장 프로시저를 비롯한 다른 종류의 객체를 생성할 수 없습니다. 여전히 mysql 테이블을 쿼리할 수 있습니다. 이진 로그 복제를 사용하는 경우 소스 DB 인스턴스의 mysql 테이블에 대한 직접 변경은 대상 클러스터로 복제되지 않습니다.

경우에 따라 애플리케이션에서 바로 가기를 사용하여 mysql 테이블에 삽입함으로써 사용자 또는 다른 객체를 만들 수 있습니다. 그렇다면 애플리케이션 코드를 변경하여 CREATE USER와 같은 해당 명령문을 사용합니다.

외부 MySQL 데이터베이스에서 마이그레이션하는 동안 데이터베이스 사용자의 메타데이터를 내보내려면 다음 메서드 중 하나를 사용합니다.

- MySQL Shell의 인스턴스 덤프 유틸리티를 필터와 함께 사용하여 사용자, 역할 및 권한 부여를 제외합니다. 다음 예제는 사용하는 명령 구문을 보여 줍니다. outputUrl이 비어 있는지 확인합니다.

  ```
  mysqlsh user@host -- util.dumpInstance(outputUrl,{excludeSchemas:['mysql'],users:
   true})
  ```

  자세한 내용은 MySQL 참조 설명서의 Instance Dump Utility, Schema Dump Utility, Table Dump Utility를 참조하세요.
- mysqlpump 클라이언트 유틸리티를 사용하세요. 이 예제에는 mysql 시스템 데이터베이스의 테이블을 제외한 모든 테이블이 포함됩니다. 또한 마이그레이션된 데이터베이스의 모든 MySQL 사용자를 재현하는 CREATE USER 및 GRANT 문이 포함됩니다.

  ```
  mysqlpump --exclude-databases=mysql --users
  ```

  MySQL 8.4에서는 mysqlpump 클라이언트 유틸리티를 더 이상 사용할 수 없습니다. 대신 mysqldump를 사용합니다.

많은 사용자 또는 애플리케이션에 대한 권한 관리를 단순화하려면 CREATE ROLE 문을 사용하여 권한 집합이 있는 역할을 만들 수 있습니다. 그런 다음, GRANT 및 SET ROLE 문과 current_role 함수를 사용하여 사용자 또는 애플리케이션에 역할을 할당하고 현재 역할을 전환하며 어떤 역할이 유효한지

확인할 수 있습니다. MySQL 8.0의 역할 기반 권한 시스템에 대한 자세한 내용은 MySQL 참조 설명서의 역할 사용을 참조하세요.

> **Important**
>
> 애플리케이션에서 직접 마스터 사용자를 사용하지 않는 것이 좋습니다. 대신에 애플리케이션에 필요한 최소 권한으로 생성한 데이터베이스 사용자를 사용하는 모범 사례를 준수하십시오.

버전 8.0.36부터 RDS for MySQL에는 다음과 같은 모든 권한이 있는 특수 역할이 포함됩니다. 이 역할의 이름은 `rds_superuser_role`입니다. 각 DB 인스턴스의 기본 관리 사용자에게는 이미 이 역할이 부여되었습니다. `rds_superuser_role` 역할에는 모든 데이터베이스 객체에 대한 다음과 같은 권한이 포함됩니다.

- `ALTER`
- `APPLICATION_PASSWORD_ADMIN`
- `ALTER ROUTINE`
- `CREATE`
- `CREATE ROLE`
- `CREATE ROUTINE`
- `CREATE TEMPORARY TABLES`
- `CREATE USER`
- `CREATE VIEW`
- `DELETE`
- `DROP`
- `DROP ROLE`
- `EVENT`
- `EXECUTE`
- `INDEX`
- `INSERT`
- `LOCK TABLES`
- `PROCESS`
- `REFERENCES`

- RELOAD
- REPLICATION CLIENT
- REPLICATION SLAVE
- ROLE_ADMIN
- SET_USER_ID
- SELECT
- SHOW DATABASES
- SHOW VIEW
- TRIGGER
- UPDATE
- XA_RECOVER_ADMIN

역할 정의에는 WITH GRANT OPTION이 포함되므로 관리 사용자가 다른 사용자에게 해당 역할을 부여할 수 있습니다. 특히 관리자는 MySQL 클러스터를 대상으로 사용하여 이진 로그 복제를 수행하는 데 필요한 모든 권한을 부여해야 합니다.

> **Tip**
>
> 권한의 세부 정보를 모두 확인하려면 다음 문을 사용합니다.
>
> ```
> SHOW GRANTS FOR rds_superuser_role@'%';
> ```

RDS for MySQL 버전 8.0.36 이상에서 역할을 사용하여 액세스 권한을 부여하는 경우 SET ROLE *role_name* 또는 SET ROLE ALL 문을 사용하여 해당 역할을 활성화합니다. 다음 예에서는 이 작업을 수행하는 방법을 보여줍니다. 적합한 역할 이름을 CUSTOM_ROLE로 대체합니다.

```
# Grant role to user
mysql> GRANT CUSTOM_ROLE TO 'user'@'domain-or-ip-address'

# Check the current roles for your user. In this case, the CUSTOM_ROLE role has not
 been activated.
# Only the rds_superuser_role is currently in effect.
mysql> SELECT CURRENT_ROLE();
+--------------------------+
```

```
| CURRENT_ROLE()           |
+--------------------------+
| `rds_superuser_role`@`%` |
+--------------------------+
1 row in set (0.00 sec)

# Activate all roles associated with this user using SET ROLE.
# You can activate specific roles or all roles.
# In this case, the user only has 2 roles, so we specify ALL.
mysql> SET ROLE ALL;
Query OK, 0 rows affected (0.00 sec)

# Verify role is now active
mysql> SELECT CURRENT_ROLE();
+-------------------------------------------------+
| CURRENT_ROLE()                                  |
+-------------------------------------------------+
| `CUSTOM_ROLE`@`%`,`rds_superuser_role`@`%` |
+-------------------------------------------------+
```

RDS for MySQL에 대한 동적 권한

동적 권한은 GRANT 문을 사용하여 명시적으로 부여할 수 있는 MySQL 권한입니다. 사용 중인 RDS for MySQL 버전에 따라 RDS에서 특정 동적 권한만 부여할 수 있습니다. RDS는 이러한 권한 중 일부를 허용하지 않습니다. 이러한 권한은 복제 및 백업과 같은 특정 데이터베이스 작업에 방해가 될 수 있기 때문입니다.

다음 표에는 다양한 MySQL 버전에 부여할 수 있는 이러한 권한이 나와 있습니다. 8.0.36 미만의 MySQL 버전에서 8.0.36 이상 버전으로 업그레이드하는 경우 특정 권한 부여가 더 이상 허용되지 않으면 애플리케이션 코드를 업데이트해야 할 수 있습니다.

권한	MySQL 8.0.35 이하	MySQL 8.0.36 이상 마이너 버전	MySQL 8.4.3 이상
ALLOW_NONEXISTENT_DEFINER	사용할 수 없음	사용할 수 없음	허용되지 않음
APPLICATION_PASSWORD_ADMIN	Allowed	허용됨	허용됨

권한	MySQL 8.0.35 이하	MySQL 8.0.36 이상 마이너 버전	MySQL 8.4.3 이상
AUDIT_ABORT_EXEMPT	허용	허용되지 않음	허용되지 않음
AUDIT_ADMIN	허용되지 않음	허용되지 않음	허용되지 않음
AUTHENTICATION_POLICY_ADMIN	허용	허용되지 않음	허용되지 않음
BACKUP_ADMIN	허용	허용되지 않음	허용되지 않음
BINLOG_ADMIN	허용	허용되지 않음	허용되지 않음
BINLOG_ENCRYPTION_ADMIN	허용되지 않음	허용되지 않음	허용되지 않음
CLONE_ADMIN	허용되지 않음	허용되지 않음	허용되지 않음
CONNECTION_ADMIN	허용	허용되지 않음	허용되지 않음
ENCRYPTION_KEY_ADMIN	허용되지 않음	허용되지 않음	허용되지 않음
FIREWALL_ADMIN	허용되지 않음	허용되지 않음	허용되지 않음
FIREWALL_EXEMPT	허용	허용되지 않음	허용되지 않음
FIREWALL_USER	허용되지 않음	허용되지 않음	허용되지 않음
FLUSH_OPTIMIZER_COSTS	Allowed	허용됨	허용됨
FLUSH_PRIVILEGES	사용할 수 없음	사용할 수 없음	허용
FLUSH_STATUS	Allowed	허용됨	허용됨
FLUSH_TABLES	Allowed	허용됨	허용됨

권한	MySQL 8.0.35 이하	MySQL 8.0.36 이상 마이너 버전	MySQL 8.4.3 이상
FLUSH_USER_RESOURCES	Allowed	허용됨	허용됨
GROUP_REPLICATION_ADMIN	허용되지 않음	허용되지 않음	허용되지 않음
GROUP_REPLICATION_STREAM	허용되지 않음	허용되지 않음	허용되지 않음
INNODB_REDO_LOG_ARCHIVE	허용되지 않음	허용되지 않음	허용되지 않음
INNODB_REDO_LOG_ENABLE	허용되지 않음	허용되지 않음	허용되지 않음
MASKING_DICTIONARIES_ADMIN	허용되지 않음	허용되지 않음	허용되지 않음
NDB_STORED_USER	허용되지 않음	허용되지 않음	허용되지 않음
OPTIMIZE_LOCAL_TABLE	사용할 수 없음	사용할 수 없음	허용되지 않음
PASSWORDLESS_USER_ADMIN	허용되지 않음	허용되지 않음	허용되지 않음
PERSIST_RO_VARIABLES_ADMIN	허용되지 않음	허용되지 않음	허용되지 않음
REPLICATION_APPLIER	허용	허용되지 않음	허용되지 않음

권한	MySQL 8.0.35 이하	MySQL 8.0.36 이상 마이너 버전	MySQL 8.4.3 이상
REPLICATION_SLAVE_ADMIN	허용되지 않음	허용되지 않음	허용되지 않음
RESOURCE_GROUP_ADMIN	허용	허용되지 않음	허용되지 않음
RESOURCE_GROUP_USER	허용	허용되지 않음	허용되지 않음
ROLE_ADMIN	Allowed	허용됨	허용됨
SENSITIVE_VARIABLES_OBSERVER	Allowed	허용됨	허용됨
SERVICE_CONNECTION_ADMIN	허용	허용되지 않음	허용되지 않음
SESSION_VARIABLES_ADMIN	Allowed	허용됨	허용됨
SET_ANY_DEFINER	사용할 수 없음	사용할 수 없음	허용
SET_USER_ID	Allowed	허용됨	사용할 수 없음
SHOW_ROUTINE	Allowed	허용됨	허용됨
SKIP_QUERY_REWRITE	허용되지 않음	허용되지 않음	허용되지 않음
SYSTEM_USER	허용되지 않음	허용되지 않음	허용되지 않음
SYSTEM_VARIABLES_ADMIN	허용되지 않음	허용되지 않음	허용되지 않음
TABLE_ENCRYPTION_ADMIN	허용되지 않음	허용되지 않음	허용되지 않음

권한	MySQL 8.0.35 이하	MySQL 8.0.36 이상 마이너 버전	MySQL 8.4.3 이상
TELEMETRY_LOG_ADMIN	허용	허용되지 않음	허용되지 않음
TP_CONNECTION_ADMIN	허용되지 않음	허용되지 않음	허용되지 않음
TRANSACTION_GTID_TAG	사용할 수 없음	사용할 수 없음	허용되지 않음
VERSION_TOKEN_ADMIN	허용되지 않음	허용되지 않음	허용되지 않음
XA_RECOVER_ADMIN	Allowed	허용됨	허용됨

RDS for MySQL에 대한 세션 또는 쿼리 종료

`rds_kill` 및 `rds_kill_query` 명령을 사용하여 DB 인스턴스에서 사용자 세션이나 쿼리를 종료할 수 있습니다. 먼저 MySQL DB 인스턴스에 연결한 후 다음과 같이 해당 명령을 실행합니다. 자세한 내용은 MySQL DB 인스턴스에 연결 섹션을 참조하세요.

```
CALL mysql.rds_kill(thread-ID)
CALL mysql.rds_kill_query(thread-ID)
```

예를 들어, 스레드 99에서 실행 중인 세션을 종료하려면 다음과 같이 입력합니다.

```
CALL mysql.rds_kill(99);
```

스레드 99에서 실행 중인 쿼리를 종료하려면 다음과 같이 입력합니다.

```
CALL mysql.rds_kill_query(99);
```

RDS for MySQL에 대한 현재 복제 오류 건너뛰기

에러가 읽기 전용 복제본의 응답을 중지시키고 데이터 무결성에 영향을 미치지 않는다면 읽기 전용 복사본의 에러를 건너뛸 수 있습니다.

> **Note**
>
> 먼저 안전하게 건너뛸 수 있는 오류인지 확인해야 합니다. MySQL 유틸리티에서 읽기 전용 복제본에 연결한 후 다음 MySQL 명령을 실행합니다.
>
> ```
> SHOW REPLICA STATUS\G
> ```
>
> 반환된 값에 대한 자세한 내용은 MySQL 설명서를 참조하세요.
> 이전 버전의 MySQL에는 `SHOW SLAVE STATUS` 대신 `SHOW REPLICA STATUS`가 사용되었습니다. 8.0.23 이전 MySQL 버전을 사용하는 경우 `SHOW SLAVE STATUS`를 사용합니다.

다음과 같은 방법으로 읽기 복제본에 대한 오류를 건너뛸 수 있습니다.

주제

- mysql.rds_skip_repl_error 프로시저 호출
- slave_skip_errors 파라미터 설정

mysql.rds_skip_repl_error 프로시저 호출

Amazon RDS는 읽기 전용 복제본에서 오류를 건너뛰기 위해 호출할 수 있는 저장 프로시저를 제공합니다. 먼저 읽기 전용 복제본에 연결한 후 다음과 같이 적합한 명령을 실행합니다. 자세한 내용은 MySQL DB 인스턴스에 연결 단원을 참조하십시오.

오류를 건너뛰려면 다음 명령을 실행합니다.

```
CALL mysql.rds_skip_repl_error;
```

이 명령은 원본 DB 인스턴스에서 실행하거나, 혹은 복제 오류가 발생하지 않은 읽기 전용 복제본에서 실행하는 경우 아무런 효과도 없습니다.

`mysql.rds_skip_repl_error`가 지원되는 MySQL 버전 등에 대한 자세한 내용은 mysql.rds_skip_repl_error 단원을 참조하십시오.

> **Important**
>
> `mysql.rds_skip_repl_error`을 호출하려고 할 때 `ERROR 1305 (42000): PROCEDURE mysql.rds_skip_repl_error does not exist`와 같은 오류가 발생한 경우에는 MySQL DB 인스턴스를 최신 마이너 버전이나 mysql.rds_skip_repl_error에 등록된 최소 마이너 버전 중 하나로 업그레이드해야 합니다.

slave_skip_errors 파라미터 설정

하나 이상의 오류를 건너뛰려면 읽기 전용 복제본에 `slave_skip_errors` 정적 파라미터를 설정해야 합니다. 하나 이상의 특정 복제 오류 코드를 건너뛰도록 이 파라미터를 설정할 수 있습니다. 현재는 MySQL 5.7 DB 인스턴스용 RDS에 대해서만 이 파라미터를 설정할 수 있습니다. 파라미터에 대한 설정을 변경하면 DB 인스턴스를 재부팅해야만 새 설정이 적용됩니다. 이러한 파라미터의 설정에 대한 자세한 내용은 MySQL 설명서를 참조하세요.

이 파라미터를 별도의 DB 파라미터 그룹에 설정하는 것이 좋습니다. 이 DB 파라미터 그룹은 오류를 건너뛰어야 하는 읽기 전용 복제본과만 연결할 수 있습니다. 이 모범 사례를 따르면 다른 DB 인스턴스 및 읽기 전용 복제본에 미치는 잠재적 영향을 줄일 수 있습니다.

> **Important**
>
> 이 파라미터에 기본값이 아닌 값을 설정하면 복제 불일치가 발생할 수 있습니다. 문제를 해결하기 위해 다른 옵션을 다 써 버렸고 읽기 전용 복제본의 데이터에 미칠 수 있는 잠재적인 영향을 확신하는 경우에만 이 파라미터를 기본값이 아닌 값으로 설정하세요.

RDS for MySQL의 충돌 복구 시간 개선을 위한 InnoDB 테이블스페이스 작업

MySQL의 모든 테이블은 테이블 정의, 데이터 및 인덱스로 구성되어 있습니다. InnoDB는 MySQL 스토리지 엔진으로서 테이블 데이터와 인덱스를 테이블스페이스에 저장하는 역할을 합니다. 이 스토리지 엔진은 전역적 공유 테이블스페이스를 생성하여 데이터 사전을 비롯한 기타 관련 메타데이터, 그리고 테이블 데이터와 인덱스도 저장합니다. 또한 테이블 및 파티션마다 별도의 테이블스페이스를 생성할 수도 있습니다. 이렇게 별도로 생성된 테이블스페이스는 확장자가 .ibd인 파일에 저장되며, 각 테이블스페이스 헤더에는 식별할 수 있도록 고유 번호가 포함됩니다.

Amazon RDS는 MySQL 파라미터 그룹을 통해 `innodb_file_per_table`이라고 하는 파라미터를 하나 제공합니다. 이 파라미터는 InnoDB가 파라미터 값을 0으로 설정하여 새 테이블 데이터와 인덱스

를 공유 테이블스페이스에 추가할지 또는 파라미터 값을 1로 설정하여 개별 테이블스페이스에 추가할지 제어합니다. Amazon RDS는 innodb_file_per_table 파라미터의 기본값을 1로 설정하여 개별 InnoDB 테이블을 삭제하고 해당 테이블에서 DB 인스턴스에 사용하는 스토리지를 회수할 수 있습니다. 대부분 사용 사례에서 innodb_file_per_table 파라미터는 1로 설정하는 것이 바람직합니다.

하지만 표준 스토리지(마그네틱)나 일반 SSD 스토리지를 사용하여 테이블 수가 1,000개를 넘거나, 혹은 프로비저닝된 IOPS 스토리지를 사용하여 테이블 수가 10,000개를 넘는 등 테이블 수가 많을 때는 innodb_file_per_table 파라미터를 0으로 설정해야 합니다. 이 파라미터를 0으로 설정하면 테이블스페이스가 개별적으로 생성되지 않기 때문에 데이터베이스 충돌 복구에 걸리는 시간을 개선할 수 있습니다.

MySQL은 충돌 복구 주기에서 테이블스페이스가 저장된 메타데이터 파일을 각각 처리합니다. MySQL이 공유 테이블스페이스에 저장된 메타데이터 정보를 처리하는 데 걸리는 시간은 다수의 테이블스페이스로 인해 수천 개의 테이블스페이스 파일을 처리하는 데 걸리는 시간에 비하면 무시해도 될 정도입니다. 테이블스페이스 번호는 각 파일의 헤더에 저장되기 때문에 모든 테이블스페이스 파일을 읽으려면 최대 몇 시간까지 걸릴 수 있습니다. 예를 들어 표준 스토리지에 InnoDB 테이블스페이스가 수백만 개 저장되어 있다면 충돌 복구 주기에서 처리하는 데만 5~8시간이 소요됩니다. 경우에 따라 충돌 복구 주기가 끝나더라도 InnoDB가 추가 정리가 필요하다고 판단할 경우에는 또 다른 충돌 복구 주기가 시작되면서 복구 시간이 연장됩니다. 또 한 가지, 충돌 복구 주기는 테이블스페이스 정보 처리 외에도 롤링 백 트랜잭션, 손상된 페이지 복구, 그리고 그 밖의 작업까지 수반한다는 점도 잊어서는 안 됩니다.

innodb_file_per_table 파라미터는 파라미터 그룹에 속하기 때문에 DB 인스턴스에 사용되는 파라미터 그룹만 편집하면 파라미터 값이 변경됩니다. 따라서 DB 인스턴스를 재부팅할 필요가 없습니다. 예를 들어 설정을 1(개별 테이블 생성)에서 0(공유 테이블스페이스 사용)으로 변경하면 새로운 InnoDB 테이블이 공유 테이블스페이스에 추가되는 반면 기존 테이블은 개별 테이블스페이스를 그대로 유지합니다. InnoDB 테이블을 공유 테이블스페이스로 이동하려면 ALTER TABLE 명령을 사용해야 합니다.

여러 테이블스페이스를 공유 테이블스페이스로 마이그레이션

InnoDB 테이블의 메타데이터를 자체 테이블스페이스에서 공유 테이블스페이스로 이동할 수 있습니다. 이렇게 하면 innodb_file_per_table 파라미터 설정에 따라 테이블 메타데이터가 다시 작성됩니다. 먼저 MySQL DB 인스턴스에 연결한 후 다음과 같이 해당 명령을 실행합니다. 자세한 내용은 MySQL DB 인스턴스에 연결 섹션을 참조하세요.

```
ALTER TABLE table_name ENGINE = InnoDB, ALGORITHM=COPY;
```

예를 들어 다음 쿼리는 공유 테이블스페이스에 없는 모든 InnoDB 테이블에 대해 ALTER TABLE 문을 반환합니다.

MySQL 5.7 DB 인스턴스의 경우:

```
SELECT CONCAT('ALTER TABLE `',
REPLACE(LEFT(NAME , INSTR((NAME), '/') - 1), '`', '``'), '`.`',
REPLACE(SUBSTR(NAME FROM INSTR(NAME, '/') + 1), '`', '``'), '` ENGINE=InnoDB,
 ALGORITHM=COPY;') AS Query
FROM INFORMATION_SCHEMA.INNODB_SYS_TABLES
WHERE SPACE <> 0 AND LEFT(NAME, INSTR((NAME), '/') - 1) NOT IN ('mysql','');
```

MySQL 8.4 및 8.0 DB 인스턴스의 경우:

```
SELECT CONCAT('ALTER TABLE `',
REPLACE(LEFT(NAME , INSTR((NAME), '/') - 1), '`', '``'), '`.`',
REPLACE(SUBSTR(NAME FROM INSTR(NAME, '/') + 1), '`', '``'), '` ENGINE=InnoDB,
 ALGORITHM=COPY;') AS Query
FROM INFORMATION_SCHEMA.INNODB_TABLES
WHERE SPACE <> 0 AND LEFT(NAME, INSTR((NAME), '/') - 1) NOT IN ('mysql','');
```

MySQL 테이블을 리빌드하여 테이블의 메타데이터를 공유 테이블스페이스로 이동하려면 테이블을 리빌드할 수 있는 스토리지 공간이 임시로 필요합니다. DB 인스턴스에 여유 스토리지 공간이 있어야 하는 이유도 바로 이 때문입니다. 리빌드 단계에서는 테이블이 잠겨서 쿼리에 액세스하지 못합니다. 작은 용량의 테이블이나 자주 액세스하지 않는 테이블의 경우 이것이 문제가 되지는 않습니다. 하지만 대용량의 테이블이나 동시 접속자 수가 많은 환경에서 자주 액세스하는 테이블이라면 테이블을 읽기 전용 복제본에 다시 빌드할 수 있습니다.

읽기 전용 복제본을 생성한 후 테이블 메타데이터를 읽기 전용 복제본의 공유 테이블스페이스로 마이그레이션할 수 있습니다. ALTER TABLE 문이 읽기 전용 복제본에 대한 액세스를 차단하더라도 원본 DB 인스턴스는 영향을 받지 않습니다. 따라서 테이블 리빌딩 프로세스 중 읽기 전용 복제본이 지연되더라도 원본 DB 인스턴스는 계속해서 이진 로그를 생성합니다. 리빌딩 프로세스에는 스토리지 공간이 추가로 필요할 뿐만 아니라 재생 로그 파일이 커질 수도 있기 때문에 원본 DB 인스턴스보다 큰 용량의 스토리지를 할당하여 읽기 전용 복제본을 생성해야 합니다.

읽기 전용 복제본을 생성하여 InnoDB 테이블을 다시 빌드한 후 공유 테이블스페이스를 사용하려면 다음 단계를 따라야 합니다.

1. 이진 로깅을 계속 할 수 있도록 원본 DB 인스턴스에 백업 보존 기간이 활성화되어 있는지 확인합니다.
2. AWS Management Console 또는 AWS CLI를 사용하여 원본 DB 인스턴스의 읽기 전용 복제본을 생성합니다. 읽기 전용 복제본을 생성하려면 충돌 복구와 같이 다수의 동일한 프로세스를 거쳐야 하기 때문에 InnoDB 테이블스페이스가 많을 경우에는 생성 프로세스에 다소 시간이 걸릴 수 있습니다.

이때 읽기 전용 복제본에 할당하는 스토리지 공간은 현재 원본 DB 인스턴스에 사용 중인 스토리지 공간보다 많아야 합니다.

3. 읽기 전용 복제본이 생성되면 파라미터 설정 `read_only = 0` 및 `innodb_file_per_table = 0`을 사용하여 파라미터 그룹을 생성합니다. 그런 다음 파라미터 그룹을 읽기 전용 복제본과 연결합니다.
4. 복제본에서 마이그레이션할 모든 테이블에 대해 다음 SQL 문을 실행합니다.

```
ALTER TABLE name ENGINE = InnoDB
```

5. 읽기 전용 복제본에서 `ALTER TABLE` 문을 모두 완료한 후에는 읽기 전용 복제본이 소스 DB 인스턴스에 연결되어 있고 두 인스턴스가 동기화되어 있는지 확인합니다.
6. 콘솔 또는 CLI를 사용하여 읽기 전용 복제본을 인스턴스로 승격합니다. 새로운 독립형 DB 인스턴스에 사용한 파라미터 그룹에서 `innodb_file_per_table` 파라미터가 0으로 설정되어 있는지 확인합니다. 새로운 독립형 DB 인스턴스의 이름을 변경하고 애플리케이션을 새로운 독립형 DB 인스턴스로 지정합니다.

RDS for MySQL의 글로벌 상태 기록 관리

> **Tip**
>
> 데이터베이스 성능을 분석하고자 Amazon RDS의 성능 개선 도우미를 사용할 수도 있습니다. 자세한 내용은 성능 개선 도우미를 통한 Amazon RDS 모니터링 단원을 참조하십시오.

MySQL은 작업 관련 정보를 알 수 있는 다수의 상태 변수를 유지하고 있습니다. 이 변수 값은 DB 인스턴스에서 잠금 또는 메모리 문제를 파악하는 데 효과적입니다. DB 인스턴스를 마지막으로 시작한 때부터 계속해서 누적되기 때문입니다. 대부분 상태 변수는 `FLUSH STATUS` 명령을 사용해 0으로 재설정할 수 있습니다.

Amazon RDS는 시간이 지나면서 이 변수 값의 스냅샷을 캡처하거나, 마지막 스냅샷 이후 모든 변경 사항과 함께 변수 값을 테이블에 기록하는 등 시간 경과에 따른 상태 변수 값을 모니터링할 수 있는 프로시저를 지원합니다. 이러한 인프라를 전역적 상태 이력(GoSH)이라고 부릅니다. GoSH는 버전 5.5.23부터 모든 MySQL DB 인스턴스에 설치되기 시작했지만 기본적으로 비활성화되어 있습니다.

GoSH를 활성화하려면 먼저 파라미터 `event_scheduler`를 `ON`으로 설정하여 DB 파라미터 그룹의 이벤트 스케줄러를 활성화해야 합니다. 또한 MySQL 5.7을 실행하는 MySQL DB 인스턴스의 경우 `show_compatibility_56` 파라미터를 1에 설정해야 합니다. DB 파라미터 그룹의 생성 및 변경에 대

한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오. 이 파라미터를 활성화할 때 생기는 부작용에 관한 내용은 MySQL 5.7 참조 설명서에 나온 show_compatibility_56을 참조하세요.

그런 다음 아래 표의 프로시저에 따라 GoSH를 활성화 및 구성할 수 있습니다. 먼저 MySQL DB 인스턴스에 연결한 후 다음과 같이 해당 명령을 실행합니다. 자세한 내용은 MySQL DB 인스턴스에 연결 단원을 참조하십시오. 각 절차에 대해 다음 명령을 실행하고 *procedure-name*을 바꿉니다.

```
CALL procedure-name;
```

다음 표에는 이전 명령에서 *procedure-name*에 사용할 수 있는 모든 절차가 나열되어 있습니다.

프로시저	설명
mysql.rds_enable_gsh_collector	rds_set_gsh_collector 에서 설정한 주기에 따라 기본 스냅샷을 캡처하도록 GoSH를 활성화합니다.
mysql.rds_set_gsh_collector	스냅샷 캡처 주기(분)를 지정합니다. 기본 값은 5입니다.
mysql.rds_disable_gsh_collector	스냅샷을 비활성화합니다.
mysql.rds_collect_global_status_history	필요할 경우에만 스냅샷을 캡처합니다.
mysql.rds_enable_gsh_rotation	mysql.rds_global_status_history 테이블의 내용이 mysql.rds_global_status_history_old 에서 설정한 주기에 따라 rds_set_gsh_rotation 로 로테이션됩니다.
mysql.rds_set_gsh_rotation	테이블 로테이션 주기(일)를 지정합니다. 기본 값은 7입니다.
mysql.rds_disable_gsh_rotation	테이블 로테이션을 비활성화합니다.
mysql.rds_rotate_global_status_history	필요에 따라 mysql.rds_global_status_history 테이블의 내용을 mysql.rds_global_status_history_old 로 로테이션합니다.

GoSH가 활성화되어 있을 때는 쓰기가 가능한 테이블에 쿼리를 요청할 수 있습니다. 예를 들어 Innodb 버퍼 풀의 적중률에 대한 쿼리를 요청하려면 다음과 같이 쿼리를 실행합니다.

```
select a.collection_end, a.collection_start, (( a.variable_Delta-b.variable_delta)/
a.variable_delta)*100 as "HitRatio"
    from mysql.rds_global_status_history as a join mysql.rds_global_status_history as b
 on a.collection_end = b.collection_end
    where a. variable_name = 'Innodb_buffer_pool_read_requests' and b.variable_name =
 'Innodb_buffer_pool_reads'
```

MySQL 8.4에서 버퍼 풀 크기 및 다시 실행 로그 용량 구성

MySQL 8.4에서 Amazon RDS는 기본적으로 `innodb_dedicated_server` 파라미터를 활성화합니다. `innodb_dedicated_server` 파라미터를 사용하면 데이터베이스 엔진이 `innodb_buffer_pool_size` 및 `innodb_redo_log_capacity` 파라미터를 계산합니다. 이러한 파라미터를 계산하는 방법에 대한 자세한 내용은 MySQL 설명서의 Configuring InnoDB Buffer Pool Size 및 Redo Log를 참조하세요.

`innodb_dedicated_server`가 활성화되면 DB 인스턴스 클래스 메모리를 기반으로 `innodb_buffer_pool_size` 파라미터가 계산됩니다. 다음 표에는 감지된 서버 메모리와 해당 버퍼 풀 크기가 나와 있습니다.

감지된 서버 메모리	버퍼 풀 크기
< 1GB	기본값 128MB
1GB~4GB	*### ## ###* * 0.5
> 4GB	*### ## ###* * 0.75

`innodb_redo_log_capacity` 파라미터는 인스턴스 클래스에 따라 (vCPUs 개수/2)GB로 최대 16GB까지 자동으로 확장됩니다. 인스턴스 클래스가 클수록 다시 실행 로그 용량이 커지므로 쓰기 집약적 워크로드의 성능과 복원력이 향상될 수 있습니다.

MySQL 8.0에서 MySQL 8.4로 업그레이드하기 전에 업그레이드 완료 후 발생할 수 있는 재실행 로그의 크기 증가를 수용할 수 있도록 스토리지 공간을 늘려야 합니다. 자세한 내용은 DB 인스턴스 스토리지 용량 증가 단원을 참조하십시오.

innodb_dedicated_server 파라미터가 innodb_buffer_pool_size 및 innodb_redo_log_capacity 파라미터의 값을 계산하지 않도록 하려면 사용자 지정 파라미터 그룹에서 이러한 파라미터에 특정 값을 설정하여 값을 재정의할 수 있습니다. 또는 innodb_dedicated_server 파라미터를 비활성화하고 사용자 지정 파라미터 그룹에서 innodb_buffer_pool_size 및 innodb_redo_log_capacity 파라미터의 값을 설정할 수 있습니다. 자세한 내용은 기본 및 사용자 지정 파라미터 그룹 단원을 참조하십시오.

innodb_dedicated_server 파라미터를 0으로 설정하여 비활성화하고 innodb_buffer_pool_size 및 innodb_redo_log_capacity 파라미터에 대한 값을 설정하지 않으면 Amazon RDS는 후자의 두 파라미터를 각각 128MB 및 100MB로 설정합니다. 이러한 기본값은 더 큰 인스턴스 클래스에서 성능 저하를 일으킵니다.

MySQL DB 인스턴스의 현지 시간대

기본적으로 MySQL DB 인스턴스의 시간대는 협정 세계시(UTC)입니다. 대신 DB 인스턴스의 시간대를 애플리케이션의 현지 시간대로 설정할 수 있습니다.

DB 인스턴스의 현지 시간대를 설정하려면 DB 인스턴스의 파라미터 그룹에서 `time_zone` 파라미터를 이 섹션의 뒤에 나오는 지원되는 값 중 하나로 설정합니다. 파라미터 그룹에 대한 `time_zone` 파라미터를 설정하면 해당 파라미터 그룹을 사용 중인 모든 DB 인스턴스와 읽기 전용 복제본이 새로운 현지 시간대를 사용하도록 변경됩니다. 파라미터 그룹에서 파라미터를 설정하는 방법에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하세요.

현지 시간대를 설정하면 데이터베이스에 대한 모든 새 연결에 변경 사항이 반영됩니다. 현지 시간대를 변경할 때 데이터베이스에 대해 열린 연결이 있는 경우 연결을 닫고 새 연결을 열어야 현지 시간대 업데이트가 표시됩니다.

DB 인스턴스와 하나 이상의 읽기 전용 복제본에 대해 서로 다른 현지 시간대를 설정할 수 있습니다. 이렇게 하려면 DB 인스턴스와 복제본에 대해 서로 다른 파라미터 그룹을 사용하고 각 파라미터 그룹에서 `time_zone` 파라미터를 다른 현지 시간대로 설정합니다.

AWS 리전 간 복제 중인 경우 소스 DB 인스턴스와 읽기 전용 복제본은 서로 다른 파라미터 그룹을 사용합니다. 파라미터 그룹은 AWS 리전마다 고유합니다. 각 인스턴스에 대해 동일한 현지 시간대를 사용하려면 인스턴스의 파라미터 그룹과 읽기 전용 복제본의 파라미터 그룹에서 `time_zone` 파라미터를 설정해야 합니다.

DB 스냅샷에서 DB 인스턴스를 복원할 경우 현지 시간대가 UTC로 설정됩니다. 복원이 완료된 후 시간대를 현지 시간대로 업데이트할 수 있습니다. DB 인스턴스를 특정 시점으로 복원할 경우 복원된 DB 인스턴스의 현지 시간대는 복원된 DB 인스턴스의 파라미터 그룹에서 설정한 시간대입니다.

IANA(Internet Assigned Numbers Authority)에서는 https://www.iana.org/time-zones에서 일 년에 여러 번 새로운 표준 시간대를 게시합니다. RDS에서 MySQL의 새로운 마이너 유지 관리 릴리스를 릴리스할 때마다 릴리스 시점의 최신 표준 시간대 데이터가 함께 제공됩니다. 최신 RDS for MySQL 버전을 사용하면 RDS의 최신 표준 시간대 데이터를 갖게 됩니다. DB 인스턴스에 최신 표준 시간대 데이터가 있는지 확인하려면 상위 DB 엔진 버전으로 업그레이드하는 것이 좋습니다. 또는 MariaDB DB 인스턴스의 표준 시간대 테이블을 수동으로 수정할 수 있습니다. 이렇게 하려면 SQL 명령을 사용하거나 SQL 클라이언트에서 mysql_tzinfo_to_sql 도구를 실행할 수 있습니다. 표준 시간대 데이터를 수동으로 업데이트한 후 변경 사항을 적용하려면 DB 인스턴스를 재부팅합니다. RDS는 실행 중인 DB 인스턴스의 표준 시간대 데이터를 수정하거나 재설정하지 않습니다. 새 표준 시간대 데이터는 데이터베이스 엔진 버전 업그레이드를 수행할 때만 설치됩니다.

현지 시간대를 다음 값 중 하나로 설정할 수 있습니다.

영역	시간대
아프리카	Africa/Cairo, Africa/Casablanca, Africa/Harare, Africa/Monrovia, Africa/Nairobi, Africa/Tripoli, Africa/Windhoek
아메리카	America/Araguaina, America/Asuncion, America/Bogota, America/Buenos_Aires, America/Caracas, America/Chihuahua, America/Cuiaba, America/Denver, America/Fortaleza, America/Guatemala, America/Halifax, America/Manaus, America/Matamoros, America/Monterrey, America/Montevideo, America/Phoenix, America/Santiago, America/Tijuana
아시아	Asia/Amman, Asia/Ashgabat, Asia/Baghdad, Asia/Baku, Asia/Bangkok, Asia/Beirut, Asia/Calcutta, Asia/Damascus, Asia/Dhaka, Asia/Irkutsk, Asia/Jerusalem, Asia/Kabul, Asia/Karachi, Asia/Kathmandu, Asia/Krasnoyarsk, Asia/Magadan, Asia/Muscat, Asia/Novosibirsk, Asia/Riyadh, Asia/Seoul, Asia/Shanghai, Asia/Singapore, Asia/Taipei, Asia/Tehran, Asia/Tokyo, Asia/Ulaanbaatar, Asia/Vladivostok, Asia/Yakutsk, Asia/Yerevan
대서양	Atlantic/Azores
호주	Australia/Adelaide, Australia/Brisbane, Australia/Darwin, Australia/Hobart, Australia/Perth, Australia/Sydney
브라질	Brazil/DeNoronha, Brazil/East
캐나다	Canada/Newfoundland, Canada/Saskatchewan, Canda/Yukon
유럽	Europe/Amsterdam, Europe/Athens, Europe/Dublin, Europe/Helsinki, Europe/Istanbul, Europe/Kaliningrad Europe/Moscow, Europe/Paris, Europe/Prague, Europe/Sarajevo
태평양	Pacific/Auckland, Pacific/Fiji, Pacific/Guam, Pacific/Honolulu, Pacific/Samoa
US	US/Alaska, US/Central, US/East-Indiana, US/Eastern, US/Pacific
UTC	UTC

Amazon RDS for MySQL에 대해 알려진 문제 및 제한

Amazon RDS for MySQL 작업에 대해 알려진 문제 및 제한은 다음과 같습니다.

주제

- InnoDB 예약어
- Amazon RDS for MySQL에 대해 스토리지가 가득 찬 동작
- 일관되지 않은 InnoDB 버퍼 풀 크기
- 인덱스 병합 최적화가 잘못된 결과를 반환
- Amazon RDS DB 인스턴스에 대한 MySQL 파라미터 예외
- Amazon RDS의 MySQL 파일 크기 제한
- MySQL Keyring Plugin 지원 안 됨
- 사용자 지정 포트
- MySQL 저장 프로시저 제한 사항
- 외부 소스 인스턴스를 사용하여 GTID 기반 복제
- MySQL 기본 인증 플러그인
- innodb_buffer_pool_size 재정의
- MySQL 5.7에서 MySQL 8.4로 업그레이드
- InnoDB 페이지 압축

InnoDB 예약어

`InnoDB`는 RDS for MySQL의 예약어입니다. MySQL 데이터베이스에는 이 이름을 사용할 수 없습니다.

Amazon RDS for MySQL에 대해 스토리지가 가득 찬 동작

MySQL DB 인스턴스의 스토리지가 가득 차면 메타데이터 불일치, 사전 불일치 및 고아 테이블이 발생할 수 있습니다. 이러한 문제를 방지하기 위해 Amazon RDS에서는 `storage-full` 상태에 도달한 DB 인스턴스를 자동으로 중지합니다.

MySQL DB 인스턴스는 다음과 같은 경우에 `storage-full` 상태에 도달합니다.

- DB 인스턴스의 스토리지는 20,000MiB 미만이며 사용 가능한 스토리지는 200MiB 이하에 이릅니다.

- DB 인스턴스에는 102,400MiB 이상의 스토리지가 있으며 사용 가능한 스토리지는 1024MiB 이하에 이릅니다.
- DB 인스턴스의 스토리지는 20,000MiB에서 102,400MiB 사이이며 사용 가능한 스토리지의 1% 미만입니다.

DB 인스턴스가 `storage-full` 상태에 도달했기 때문에 Amazon RDS에서 DB 인스턴스를 자동으로 중지하더라도 DB 인스턴스를 변경할 수 있습니다. DB 인스턴스를 다시 시작하려면 다음 중 하나 이상을 완료하세요.

- 스토리지 자동 크기 조정을 활성화하도록 DB 인스턴스를 수정합니다.

  스토리지 자동 크기 조정에 대한 자세한 내용은 Amazon RDS 스토리지 Autoscaling을 사용한 용량 자동 관리 섹션을 참조하세요.
- DB 인스턴스를 수정하여 스토리지 용량을 늘립니다.

  스토리지 용량 증가에 대한 자세한 내용은 DB 인스턴스 스토리지 용량 증가 섹션을 참조하세요.

이러한 변경 사항 중 하나를 수행한 후에는 DB 인스턴스가 자동으로 다시 시작됩니다. DB 인스턴스 수정에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하세요.

일관되지 않은 InnoDB 버퍼 풀 크기

MySQL 5.7에는 현재 InnoDB 버퍼 풀 크기가 관리되지 않는 버그가 있습니다. MySQL 5.7에서 `innodb_buffer_pool_size` 파라미터 값을 InnoDB 버퍼 풀 크기를 너무 많이 늘려 너무 많은 메모리를 소모하도록 하는 큰 값으로 조정할 수 있습니다. 이에 따라 MySQL 데이터베이스 엔진의 실행이 중지되거나 시작되지 않을 수 있습니다. 이 문제는 사용 가능한 메모리가 적은 DB 인스턴스 클래스 문제보다 더 일반적으로 발생합니다.

이 문제를 해결하려면 `innodb_buffer_pool_size` 파라미터 값을 `innodb_buffer_pool_instances` 파라미터 값과 `innodb_buffer_pool_chunk_size` 파라미터 값의 곱의 배수로 설정해야 합니다. 예를 들어, 다음 예에서와 같이 `innodb_buffer_pool_size` 파라미터 값을 `innodb_buffer_pool_instances`와 `innodb_buffer_pool_chunk_size` 파라미터 값의 곱의 8배로 설정할 수 있습니다.

```
innodb_buffer_pool_chunk_size = 536870912
innodb_buffer_pool_instances = 4
innodb_buffer_pool_size = (536870912 * 4) * 8 = 17179869184
```

MySQL 5.7 버그에 대한 자세한 내용은 MySQL 설명서의 https://bugs.mysql.com/bug.php?id=79379를 참조하세요.

인덱스 병합 최적화가 잘못된 결과를 반환

인덱스 병합 최적화를 사용하는 쿼리는 MySQL 5.5.37에서 도입된 MySQL 쿼리 옵티마이저의 버그로 인해 잘못된 결과를 반환할 수 있습니다. 여러 개의 인덱스가 있는 테이블에 대해 쿼리를 실행하면 최적화 프로그램이 여러 개의 인덱스를 기반으로 여러 범위의 행을 스캔하지만, 결과를 올바르게 함께 병합하지 않습니다. 쿼리 옵티마이저 버그에 대한 자세한 내용은 MySQL 버그 데이터베이스의 http://bugs.mysql.com/bug.php?id=72745 및 http://bugs.mysql.com/bug.php?id=68194를 참조하세요.

예를 들면, 검색 인수가 인덱싱된 열을 참조하는 2개의 인덱스가 있는 테이블에 대한 쿼리를 고려합니다.

```
SELECT * FROM table1
WHERE indexed_col1 = 'value1' AND indexed_col2 = 'value2';
```

이 경우 검색 엔진이 두 인덱스를 모두 검색합니다. 그러나 버그로 인해 병합 결과가 정확하지 않습니다.

이 문제를 해결하려면 다음 중 한 가지 방법을 시도하면 됩니다.

- MySQL DB 인스턴스용 DB 파라미터 그룹에서 `optimizer_switch` 파라미터를 `index_merge=off`로 설정합니다. DB 파라미터 그룹 파라미터 설정에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하세요.
- MySQL DB 인스턴스를 MySQL 버전 5.7 또는 8.0으로 업그레이드합니다. 자세한 내용은 RDS for MySQL DB 엔진 업그레이드 단원을 참조하십시오.
- 인스턴스를 업그레이드하거나 `optimizer_switch` 파라미터를 변경할 수 없는 경우, 쿼리에 대한 인덱스를 명시적으로 확인하여 버그를 해결할 수 있습니다. 예:

  ```
  SELECT * FROM table1
  USE INDEX covering_index
  WHERE indexed_col1 = 'value1' AND indexed_col2 = 'value2';
  ```

자세한 내용은 MySQL 설명서의 인덱스 병합 최적화를 참조하세요.

Amazon RDS DB 인스턴스에 대한 MySQL 파라미터 예외

일부 MySQL 파라미터의 경우 Amazon RDS DB 인스턴스와 함께 사용할 때 특별히 고려해야 할 사항이 있습니다.

lower_case_table_names

Amazon RDS는 대소문자 구분 파일 시스템을 사용하므로 `lower_case_table_names` 서버 파라미터의 값을 2(이름은 지정된 대로 저장되지만 소문자로 비교됨)로 설정하는 것은 지원하지 않습니다. 다음은 Amazon RDS for MySQL DB 인스턴스에 대해 지원되는 값입니다.

- 모든 RDS for MySQL 버전에는 0(이름은 지정된 대로 저장되며 비교 시 대소문자 구분)이 지원됩니다.
- RDS for MySQL 버전 5.7, 버전 8.0.28 이상의 8.0 버전, 버전 8.4에는 1(이름은 소문자로 저장되며 비교 시 대소문자 구분 안 함)이 지원됩니다.

`lower_case_table_names` 파라미터는 DB 인스턴스를 생성하기 전에 사용자 지정 DB 파라미터 그룹을 설정합니다. DB 인스턴스를 생성할 때 사용자 지정 DB 파라미터 그룹을 지정합니다.

파라미터 그룹이 8.0보다 낮은 버전의 MySQL DB 인스턴스와 연결된 경우 파라미터 그룹에서 `lower_case_table_names` 파라미터를 변경하지 않는 것이 좋습니다. 변경할 경우 특정 시점으로 복구 백업과 읽기 전용 복제본 DB 인스턴스에서 불일치가 발생할 수 있습니다.

파라미터 그룹이 버전 8.0 또는 8.4 MySQL DB 인스턴스와 연결된 경우 파라미터 그룹에서 `lower_case_table_names` 파라미터를 수정할 수 없습니다.

읽기 전용 복제본은 항상 소스 DB 인스턴스와 동일한 `lower_case_table_names` 파라미터 값을 사용해야 합니다.

long_query_time

마이크로초 해상도로 느린 쿼리를 MySQL의 느린 쿼리 로그에 기록할 수 있도록 `long_query_time` 파라미터를 부동 소수점 값으로 설정합니다. 100밀리초에 해당하는 0.1초와 같은 값을 설정하여 1초 이내의 시간이 걸리는 느린 트랜잭션을 디버깅할 때 도움을 받을 수 있습니다.

Amazon RDS의 MySQL 파일 크기 제한

MySQL 버전 8.0 이상 DB 인스턴스의 경우 최대 파일 크기는 16TiB입니다. file-per-table 테이블스페이스를 사용하는 경우 최대 파일 크기는 InnoDB 테이블의 크기를 16TiB로 제한합니다. MySQL DB 인스

턴스에서는 테이블이 각각 자체 테이블스페이스에 들어 있는 InnoDB 테이블당 파일 테이블스페이스가 기본적으로 설정됩니다. 자세한 내용은 MySQL 설명서의 InnoDB 한도를 참조하세요.

> **Note**
>
> 일부 기존 DB 인스턴스에는 하한이 있습니다. 예를 들어, 2014년 4월 이전에 생성된 MySQL DB 인스턴스의 파일 및 테이블 크기 제한이 2TB입니다. 이 2TB 파일 크기 제한은 DB 인스턴스 생성 시기와 상관없이 2014년 4월 이전에 캡처된 DB 스냅샷으로 생성된 DB 인스턴스 또는 읽기 전용 복제본에도 적용됩니다.

애플리케이션에 따라 InnoDB 테이블당 파일 테이블스페이스 사용에 대한 장점과 단점은 서로 다릅니다. 애플리케이션에 가장 적합한 접근 방식을 확인하려면 MySQL 설명서의 테이블당 파일 테이블스페이스를 참조하세요.

테이블을 최대 파일 크기로 늘리도록 허용하는 것은 권장하지 않습니다. 일반적으로 모범 사례는 성능 및 복구 시간을 향상할 수 있도록 데이터를 더 작은 테이블로 분할하는 것입니다.

대형 테이블을 여러 개의 작은 테이블로 분할하는 데 사용할 수 있는 한 가지 옵션으로는 파티셔닝이 있습니다. 파티셔닝을 수행하면 사용자가 지정하는 규칙에 따라 라지 테이블의 일부가 개별 파일로 배포됩니다. 예를 들어, 트랜잭션을 날짜별로 저장하는 경우 파티셔닝을 사용하여 이전 트랜잭션을 개별 파일로 배포하는 파티셔닝 규칙을 생성할 수 있습니다. 이렇게 하면 애플리케이션에서 즉시 사용할 필요가 없는 이전 트랜잭션 데이터를 주기적으로 보관할 수 있습니다. 자세한 내용은 MySQL 설명서의 파티셔닝을 참조하세요.

모든 테이블과 InnoDB 시스템 테이블스페이스의 크기를 지원하는 단일 시스템 테이블 또는 보기가 없기 때문에 여러 테이블을 쿼리하여 테이블스페이스의 크기를 확인해야 합니다.

InnoDB 시스템 테이블스페이스와 데이터 딕셔너리 테이블스페이스의 크기를 확인하려면

- 다음 SQL 명령을 사용하여 크기가 너무 커서 파티셔닝을 수행해야 하는 테이블스페이스가 있는지 확인합니다.

  > **Note**
  >
  > 데이터 딕셔너리 테이블스페이스는 MySQL 8.0 이상 버전에만 해당됩니다.

```
select FILE_NAME,TABLESPACE_NAME, ROUND(((TOTAL_EXTENTS*EXTENT_SIZE)
```

```
/1024/1024/1024), 2)  as "File Size (GB)" from information_schema.FILES
where tablespace_name in ('mysql','innodb_system');
```

InnoDB 시스템 테이블스페이스 외부의 InnoDB 사용자 테이블 크기를 확인하려면(MySQL 5.7 버전의 경우)

- 다음 SQL 명령을 사용하여 크기가 너무 커서 파티셔닝을 수행해야 하는 테이블이 있는지 확인합니다.

```
SELECT SPACE,NAME,ROUND((ALLOCATED_SIZE/1024/1024/1024), 2)
as "Tablespace Size (GB)"
FROM information_schema.INNODB_SYS_TABLESPACES ORDER BY 3 DESC;
```

InnoDB 시스템 테이블스페이스 외부의 InnoDB 사용자 테이블 크기를 확인하려면(MySQL 8.0 이상 버전의 경우)

- 다음 SQL 명령을 사용하여 크기가 너무 커서 파티셔닝을 수행해야 하는 테이블이 있는지 확인합니다.

```
SELECT SPACE,NAME,ROUND((ALLOCATED_SIZE/1024/1024/1024), 2)
as "Tablespace Size (GB)"
FROM information_schema.INNODB_TABLESPACES ORDER BY 3 DESC;
```

비 InnoDB 사용자 테이블의 크기를 확인하려면

- 다음 SQL 명령을 사용하여 비 InnoDB 사용자 테이블이 너무 큰지 확인합니다.

```
SELECT TABLE_SCHEMA, TABLE_NAME, round(((DATA_LENGTH + INDEX_LENGTH+DATA_FREE)
/ 1024 / 1024/ 1024), 2) As "Approximate size (GB)" FROM information_schema.TABLES
WHERE TABLE_SCHEMA NOT IN ('mysql', 'information_schema', 'performance_schema')
and ENGINE<>'InnoDB';
```

InnoDB 테이블당 파일 테이블스페이스를 활성화하는 방법

- DB 인스턴스의 파라미터 그룹에서 innodb_file_per_table 파라미터를 1로 설정합니다.

InnoDB 테이블당 파일 테이블스페이스를 비활성화하는 방법

- DB 인스턴스의 파라미터 그룹에서 innodb_file_per_table 파라미터를 0으로 설정합니다.

파라미터 그룹 업데이트에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹을(를) 참조하십시오.

InnoDB 테이블당 파일 테이블스페이스를 활성화하거나 비활성화한 경우 ALTER TABLE 명령을 실행하여 아래의 예와 같이 테이블을 전역 테이블스페이스에서 자체 테이블스페이스로 이동하거나 자체 테이블스페이스에서 전역 테이블스페이스로 이동할 수 있습니다.

```
ALTER TABLE table_name TABLESPACE=innodb_file_per_table;
```

MySQL Keyring Plugin 지원 안 됨

현재 Amazon RDS for MySQL에서는 MySQL keyring_aws Amazon Web Services Keyring Plugin이 지원되지 않습니다.

사용자 지정 포트

Amazon RDS는 MySQL 엔진의 사용자 지정 포트 33060으로의 연결을 차단합니다. MySQL 엔진에 사용할 다른 포트를 선택하세요.

MySQL 저장 프로시저 제한 사항

다음 RDS for MySQL 버전에서는 사용자 이름이 16자보다 긴 MySQL 사용자가 소유한 세션 또는 쿼리를 mysql.rds_kill 및 mysql.rds_kill_query 저장 프로시저로 종료할 수 없습니다.

- 8.0.32 이하 8 버전
- 5.7.41 이하 5.7 버전

외부 소스 인스턴스를 사용하여 GTID 기반 복제

Amazon RDS는 외부 MySQL 인스턴스에서 구성 중에 GTID_PURGED를 설정해야 하는 Amazon RDS for MySQL DB 인스턴스로의 글로벌 트랜잭션 식별자(GTID) 기반 복제를 지원합니다. 그러나 RDS for MySQL 8.0.37 이상 버전에서만 이 기능을 지원합니다.

MySQL 기본 인증 플러그인

RDS for MySQL 버전 8.0.34 이상 8.0 버전에서 `mysql_native_password` 플러그인을 사용합니다. `default_authentication_plugin` 설정은 변경할 수 없습니다.

RDS for MySQL 버전 8.4 이상 버전은 `caching_sha2_password` 플러그인을 기본 인증 플러그인으로 사용합니다. MySQL 8.4의 기본 인증 플러그인을 변경할 수 있습니다. `mysql_native_password` 플러그인은 여전히 MySQL 8.4에서 작동하지만 이 플러그인의 지원은 MySQL 8.4와 함께 종료됩니다. 기본 인증 플러그인을 변경하려면 사용자 지정 파라미터 그룹을 만들고 `authentication_policy` 파라미터 값을 수정합니다. 자세한 내용은 the section called “기본 및 사용자 지정 파라미터 그룹” 단원을 참조하십시오.

innodb_buffer_pool_size 재정의

마이크로 또는 소형 DB 인스턴스 클래스의 경우 `innodb_buffer_pool_size` 파라미터의 기본값이 다음 명령을 실행하여 반환되는 값과 다를 수 있습니다.

```
mysql> SELECT @@innodb_buffer_pool_size;
```

이러한 차이는 Amazon RDS가 DB 인스턴스 클래스 관리의 일환으로 기본값을 재정의해야 할 때 발생할 수 있습니다. 필요한 경우 기본값을 재정의하고 DB 인스턴스 클래스가 지원하는 값으로 설정할 수 있습니다. 유효한 값을 결정하려면 메모리 사용량과 DB 인스턴스의 총 가용 메모리를 더하세요. 자세한 내용은 Amazon RDS 인스턴스 유형을 참조하세요.

DB 인스턴스의 메모리가 4GB뿐인 경우 `innodb_buffer_pool_size`를 8GB로 설정할 수 없지만 다른 파라미터에 할당한 메모리 양에 따라 3GB로 설정할 수 있습니다.

입력한 값이 너무 크면 Amazon RDS는 값을 다음 한도까지 낮춥니다.

- 마이크로 DB 인스턴스 클래스: 256MB
- db.t4g 마이크로 DB 인스턴스 클래스: 128MB

MySQL 5.7에서 MySQL 8.4로 업그레이드

MySQL 5.7에서 MySQL 8.4로 곧바로 업그레이드할 수 없습니다. 먼저 MySQL 5.7에서 MySQL 8.0으로 업그레이드한 다음 MySQL 8.0에서 MySQL 8.4로 업그레이드해야 합니다. 자세한 내용은 RDS for MySQL 메이저 버전 업그레이드 단원을 참조하십시오.

InnoDB 페이지 압축

InnoDB 페이지 압축은 파일 시스템 블록 크기가 1만 6,000인 Amazon RDS DB 인스턴스에서는 작동하지 않습니다. 파일 시스템 블록 크기가 InnoDB 페이지 크기보다 작아야 하기 때문입니다. 2024년 2월부터 새로 만들어지는 모든 DB 인스턴스의 파일 시스템 블록 크기는 1만 6,000이므로 페이지 플러싱 중에 처리량이 증가하고 IOPS 소비가 줄어듭니다.

RDS for MySQL 저장 프로시저 참조

이 주제에서는 MySQL DB 엔진을 실행 중인 Amazon RDS 인스턴스에 사용할 수 있는 시스템 저장 프로시저를 설명합니다. 마스터 사용자는 이 프로시저를 실행해야 합니다.

주제

- 글로벌 상태 기록 수집 및 유지
- 이진수 로그(binlog) 복제 구성, 시작 및 중지
- 세션 또는 쿼리 종료
- 활성-활성 클러스터
- 다중 소스 복제 관리
- GTID를 사용한 트랜잭션 복제
- 쿼리 로그 교체
- 이진수 로그 구성 설정 및 표시
- InnoDB 캐시 워밍

글로벌 상태 기록 수집 및 유지

Amazon RDS는 시간에 따른 상태 변수 값의 스냅샷을 생성하고 이를 마지막 스냅샷 이후의 변경 사항과 함께 테이블에 쓰는 일련의 프로시저를 제공합니다. 이 인프라를 전역적 상태 이력이라고 합니다. 자세한 내용은 RDS for MySQL의 글로벌 상태 기록 관리를 참조하세요.

다음 저장 프로시저는 전역적 상태 이력을 수집하고 유지하는 방법을 관리합니다.

주제

- mysql.rds_collect_global_status_history
- mysql.rds_disable_gsh_collector
- mysql.rds_disable_gsh_rotation
- mysql.rds_enable_gsh_collector
- mysql.rds_enable_gsh_rotation
- mysql.rds_rotate_global_status_history
- mysql.rds_set_gsh_collector
- mysql.rds_set_gsh_rotation

mysql.rds_collect_global_status_history

전역적 상태 이력에 대해 요청 시 스냅샷을 생성합니다.

구문

```
CALL mysql.rds_collect_global_status_history;
```

mysql.rds_disable_gsh_collector

전역적 상태 이력에서 생성한 스냅샷을 비활성화합니다.

구문

```
CALL mysql.rds_disable_gsh_collector;
```

mysql.rds_disable_gsh_rotation

`mysql.global_status_history` 테이블 회전을 비활성화합니다.

구문

```
CALL mysql.rds_disable_gsh_rotation;
```

mysql.rds_enable_gsh_collector

전역적 상태 이력을 활성화하여 `rds_set_gsh_collector`로 지정한 간격에 따라 기본 스냅샷을 생성합니다.

구문

```
CALL mysql.rds_enable_gsh_collector;
```

mysql.rds_enable_gsh_rotation

`mysql.global_status_history` 테이블의 내용이 `rds_set_gsh_rotation`에서 설정한 주기에 따라 `mysql.global_status_history_old`로 회전되도록 회전을 활성화합니다.

구문

```
CALL mysql.rds_enable_gsh_rotation;
```

mysql.rds_rotate_global_status_history

필요에 따라 `mysql.global_status_history` 테이블의 내용을 `mysql.global_status_history_old`로 로테이션합니다.

구문

```
CALL mysql.rds_rotate_global_status_history;
```

mysql.rds_set_gsh_collector

전역적 상태 이력에서 생성하는 스냅샷 간격(분)을 지정합니다.

구문

```
CALL mysql.rds_set_gsh_collector(intervalPeriod);
```

파라미터

intervalPeriod

스냅샷 주기(분)입니다. 기본값은 5입니다.

mysql.rds_set_gsh_rotation

`mysql.global_status_history` 테이블의 로테이션 주기(일)을 지정합니다.

구문

```
CALL mysql.rds_set_gsh_rotation(intervalPeriod);
```

파라미터

intervalPeriod

테이블 로테이션 주기(일)입니다. 기본값은 7입니다.

이진수 로그(binlog) 복제 구성, 시작 및 중지

다음 저장 프로시저는 트랜잭션이 외부 데이터베이스에서 RDS for MySQL로 또는 RDS for MySQL에서 외부 데이터베이스로 복제되는 방식을 제어합니다.

`caching_sha2_password`로 구성된 복제 사용자로 복제를 관리하기 위해 이러한 저장 프로시저를 사용하는 경우, `SOURCE_SSL=1`을 지정하여 TLS를 구성해야 합니다. `caching_sha2_password`는 RDS for MySQL 8.4의 기본 인증 플러그인입니다. 자세한 내용은 SSL/TLS를 사용하여 암호화 섹션을 참조하세요.

읽기 전용 복제본 구성, 사용 및 관리에 대한 자세한 내용은 the section called “MySQL 읽기 전용 복제본” 섹션을 참조하세요.

주제

- mysql.rds_next_master_log(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하)
- mysql.rds_next_source_log(RDS for MySQL 메이저 버전 8.4 이상)
- mysql.rds_reset_external_master(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하)
- mysql.rds_reset_external_source(RDS for MySQL 메이저 버전 8.4 이상)
- mysql.rds_set_external_master(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하)
- mysql.rds_set_external_source(RDS for MySQL 메이저 버전 8.4 이상)
- mysql.rds_set_external_master_with_auto_position(RDS for MySQL 메이저 버전 8.0 이하)
- mysql.rds_set_external_source_with_auto_position(RDS for MySQL 메이저 버전 8.4 이상)
- mysql.rds_set_external_master_with_delay(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하)
- mysql.rds_set_external_source_with_delay(RDS for MySQL 메이저 버전 8.4 이상)
- mysql.rds_set_external_source_gtid_purged
- mysql.rds_set_master_auto_position(RDS for MySQL 메이저 버전 8.0 이하)
- mysql.rds_set_source_auto_position(RDS for MySQL 메이저 버전 8.4 이상)
- mysql.rds_set_source_delay
- mysql.rds_skip_repl_error
- mysql.rds_start_replication
- mysql.rds_start_replication_until

- mysql.rds_stop_replication

mysql.rds_next_master_log(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하)

소스 데이터베이스 인스턴스 로그 위치를 소스 데이터베이스 인스턴스에서 다음 이진 로그의 시작으로 변경합니다. 읽기 전용 복제본에 대한 복제 I/O 오류 1236을 수신 중일 경우에만 이 프로시저를 사용하십시오.

구문

```
CALL mysql.rds_next_master_log(
curr_master_log
);
```

파라미터

curr_master_log

현재 마스터 로그 파일의 인덱스입니다. 예를 들어, 현재 파일의 이름이 `mysql-bin-changelog.012345`이면 인덱스는 12345입니다. 현재 마스터 로그 파일 이름을 확인하려면 `SHOW REPLICA STATUS` 명령을 실행하고 `Master_Log_File` 필드를 봅니다.

사용 노트

마스터 사용자는 `mysql.rds_next_master_log` 프로시저를 실행해야 합니다.

> **Warning**
>
> 복제 원본인 다중 AZ DB 인스턴스의 장애 조치 후 복제가 실패하고 `mysql.rds_next_master_log`의 `Last_IO_Errno`필드에서 I/O 오류 1236을 보고하는 경우에만 `SHOW REPLICA STATUS`를 호출합니다.
> 장애 조치 이벤트가 발생하기 전에 원본 인스턴스의 트랜잭션이 디스크의 바이너리 로그에 기록되지 않은 경우 `mysql.rds_next_master_log`를 호출하면 읽기 전용 복제본에서 데이터가 손실될 수 있습니다. 소스 인스턴스 파라미터 `sync_binlog` 및 `innodb_support_xa`를 1로 설정하여 이런 상황이 발생할 가능성을 줄일 수 있으나, 성능이 저하될 수 있습니다. 자세한 내용은 MySQL 읽기 전용 복제본의 문제 해결 섹션을 참조하세요.

예시

RDS for MySQL 읽기 전용 복제본에서 복제가 실패한다고 가정해 보겠습니다. 읽기 전용 복제본에서 `SHOW REPLICA STATUS\G`를 실행하면 다음 결과가 반환됩니다.

```
*************************** 1. row ***************************
             Replica_IO_State:
                  Source_Host: myhost.XXXXXXXXXXXXXX.rr-rrrr-1.rds.amazonaws.com
                  Source_User: MasterUser
                  Source_Port: 3306
                Connect_Retry: 10
              Source_Log_File: mysql-bin-changelog.012345
          Read_Source_Log_Pos: 1219393
               Relay_Log_File: relaylog.012340
                Relay_Log_Pos: 30223388
        Relay_Source_Log_File: mysql-bin-changelog.012345
           Replica_IO_Running: No
          Replica_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Source_Log_Pos: 30223232
              Relay_Log_Space: 5248928866
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Source_SSL_Allowed: No
           Source_SSL_CA_File:
           Source_SSL_CA_Path:
              Source_SSL_Cert:
            Source_SSL_Cipher:
               Source_SSL_Key:
        Seconds_Behind_Master: NULL
Source_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 1236
                Last_IO_Error: Got fatal error 1236 from master when reading data from
 binary log: 'Client requested master to start replication from impossible position;
 the first event 'mysql-bin-changelog.013406' at 1219393, the last event read from
```

```
 '/rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4, the last byte read from '/
rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4.'
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
            Source_Server_Id: 67285976
```

Last_IO_Errno 필드가 인스턴스에서 I/O 오류 1236을 수신하고 있음을 보여 줍니다. Master_Log_File 필드는 파일 이름이 mysql-bin-changelog.012345임을 보여 줍니다. 이는 로그 파일 인덱스가 12345임을 의미합니다. 오류를 해결하려면 다음 파라미터와 함께 mysql.rds_next_master_log를 호출합니다.

```
CALL mysql.rds_next_master_log(12345);
```

mysql.rds_next_source_log(RDS for MySQL 메이저 버전 8.4 이상)

소스 데이터베이스 인스턴스 로그 위치를 소스 데이터베이스 인스턴스에서 다음 이진 로그의 시작으로 변경합니다. 읽기 전용 복제본에 대한 복제 I/O 오류 1236을 수신 중일 경우에만 이 프로시저를 사용하십시오.

구문

```
CALL mysql.rds_next_source_log(
curr_source_log
);
```

파라미터

curr_source_log

현재 소스 로그 파일의 인덱스입니다. 예를 들어, 현재 파일의 이름이 mysql-bin-changelog.012345이면 인덱스는 12345입니다. 현재 소스 로그 파일 이름을 확인하려면 SHOW REPLICA STATUS 명령을 실행하고 Source_Log_File 필드를 봅니다.

사용 노트

관리 사용자는 mysql.rds_next_source_log 프로시저를 실행해야 합니다.

> **Warning**
>
> 복제 원본인 다중 AZ DB 인스턴스의 장애 조치 후 복제가 실패하고 `mysql.rds_next_source_log`의 `Last_IO_Errno`필드에서 I/O 오류 1236을 보고하는 경우에만 `SHOW REPLICA STATUS`를 호출합니다.
> 장애 조치 이벤트가 발생하기 전에 원본 인스턴스의 트랜잭션이 디스크의 바이너리 로그에 기록되지 않은 경우 `mysql.rds_next_source_log`를 호출하면 읽기 전용 복제본에서 데이터가 손실될 수 있습니다. 소스 인스턴스 파라미터 `innodb_support_xa` 및 `sync_binlog`를 1로 설정하여 이런 상황이 발생할 가능성을 줄일 수 있으나, 성능이 저하될 수 있습니다. 자세한 내용은 MySQL 읽기 전용 복제본의 문제 해결 섹션을 참조하세요.

예시

RDS for MySQL 읽기 전용 복제본에서 복제가 실패한다고 가정해 보겠습니다. 읽기 전용 복제본에서 `SHOW REPLICA STATUS\G`를 실행하면 다음 결과가 반환됩니다.

```
*************************** 1. row ***************************
             Replica_IO_State:
                  Source_Host: myhost.XXXXXXXXXXXXXX.rr-rrrr-1.rds.amazonaws.com
                  Source_User: MasterUser
                  Source_Port: 3306
                Connect_Retry: 10
              Source_Log_File: mysql-bin-changelog.012345
          Read_Source_Log_Pos: 1219393
               Relay_Log_File: relaylog.012340
                Relay_Log_Pos: 30223388
        Relay_Source_Log_File: mysql-bin-changelog.012345
           Replica_IO_Running: No
          Replica_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Source_Log_Pos: 30223232
              Relay_Log_Space: 5248928866
```

```
               Until_Condition: None
                Until_Log_File:
                 Until_Log_Pos: 0
            Source_SSL_Allowed: No
            Source_SSL_CA_File:
            Source_SSL_CA_Path:
               Source_SSL_Cert:
             Source_SSL_Cipher:
                Source_SSL_Key:
        Seconds_Behind_Source: NULL
Source_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 1236
                Last_IO_Error: Got fatal error 1236 from source when reading data from
 binary log: 'Client requested source to start replication from impossible position;
 the first event 'mysql-bin-changelog.013406' at 1219393, the last event read from
 '/rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4, the last byte read from '/
rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4.'
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Source_Server_Id: 67285976
```

`Last_IO_Errno` 필드가 인스턴스에서 I/O 오류 1236을 수신하고 있음을 보여 줍니다. `Source_Log_File` 필드는 파일 이름이 `mysql-bin-changelog.012345`임을 보여 줍니다. 이는 로그 파일 인덱스가 12345임을 의미합니다. 오류를 해결하려면 다음 파라미터와 함께 `mysql.rds_next_source_log`를 호출합니다.

```
CALL mysql.rds_next_source_log(12345);
```

mysql.rds_reset_external_master(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하)

이제 RDS for MySQL DB 인스턴스가 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 읽기 전용 복제본이 되지 않도록 다시 구성합니다.

> **Important**
>
> 이 프로시저를 실행하려면 `autocommit`를 활성화해야 합니다. 활성화하려면 `autocommit` 파라미터를 1로 설정합니다. 파라미터 수정에 대한 자세한 정보는 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 단원을 참조하십시오.

구문

```
CALL mysql.rds_reset_external_master;
```

사용 노트

마스터 사용자는 `mysql.rds_reset_external_master` 프로시저를 실행해야 합니다. 이 프로시저는 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 읽기 전용 복제본으로 제거되도록 MySQL DB 인스턴스에서 실행해야 합니다.

> **Note**
>
> 가능하면 읽기 전용 복제본을 사용하여 두 Amazon RDS DB 인스턴스 간 복제를 관리하는 것이 좋습니다. 그렇게 하면 이 복제 및 다른 복제 관련 저장 프로시저만 사용하는 것이 좋습니다. 이러한 방법을 사용하면 Amazon RDS DB 인스턴스 간 더욱 복잡한 복제 토폴로지를 사용할 수 있습니다. 이러한 저장 프로시저는 주로 Amazon RDS 외부에서 실행되는 MySQL 인스턴스를 사용하여 복제할 수 있도록 하기 위한 것입니다. Amazon RDS DB 인스턴스 간 복제 관리에 대한 자세한 내용은 DB 인스턴스 읽기 전용 복제본 작업 주제를 참조하십시오.

복제를 사용하여 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스에서 데이터 가져오기에 대한 자세한 내용은 외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성 주제를 참조하십시오.

mysql.rds_reset_external_source(RDS for MySQL 메이저 버전 8.4 이상)

이제 RDS for MySQL DB 인스턴스가 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 읽기 전용 복제본이 되지 않도록 다시 구성합니다.

> **Important**
>
> 이 프로시저를 실행하려면 `autocommit`를 활성화해야 합니다. 활성화하려면 `autocommit` 파라미터를 1로 설정합니다. 파라미터 수정에 대한 자세한 정보는 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 단원을 참조하십시오.

구문

```
CALL mysql.rds_reset_external_source;
```

사용 노트

관리 사용자는 `mysql.rds_reset_external_source` 프로시저를 실행해야 합니다. 이 프로시저는 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 읽기 전용 복제본으로 제거되도록 MySQL DB 인스턴스에서 실행해야 합니다.

> **Note**
>
> 가능하면 읽기 전용 복제본을 사용하여 두 Amazon RDS DB 인스턴스 간 복제를 관리하는 것이 좋습니다. 그렇게 하면 이 복제 및 다른 복제 관련 저장 프로시저만 사용하는 것이 좋습니다. 이러한 방법을 사용하면 Amazon RDS DB 인스턴스 간 더욱 복잡한 복제 토폴로지를 사용할 수 있습니다. 이러한 저장 프로시저는 주로 Amazon RDS 외부에서 실행되는 MySQL 인스턴스를 사용하여 복제할 수 있도록 하기 위한 것입니다.
>
> Amazon RDS DB 인스턴스 간 복제 관리에 대한 자세한 내용은 DB 인스턴스 읽기 전용 복제본 작업 주제를 참조하십시오. 복제를 사용하여 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스에서 데이터 가져오기에 대한 자세한 내용은 외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성 주제를 참조하십시오.

mysql.rds_set_external_master(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하)

RDS for MySQL DB 인스턴스가 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 읽기 전용 복제본이 되도록 구성합니다.

> **Important**
>
> 이 프로시저를 실행하려면 `autocommit`를 활성화해야 합니다. 활성화하려면 `autocommit` 파라미터를 1로 설정합니다. 파라미터 수정에 대한 자세한 정보는 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 단원을 참조하십시오.

> **Note**
>
> mysql.rds_set_external_master_with_delay(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하) 저장 프로시저를 사용하여 외부 소스 데이터베이스 인스턴스 및 지연된 복제를 구성할 수 있습니다.

구문

```
CALL mysql.rds_set_external_master (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
);
```

파라미터

host_name

소스 데이터베이스 인스턴스가 될 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 호스트 이름 또는 IP 주소입니다.

host_port

소스 데이터베이스 인스턴스로 구성될 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스에서 사용하는 포트입니다. 네트워크 구성에 포트 번호를 변환하는 SSH 포트 복제가 포함되는 경우 SSH(Secure Shell)에 의해 공개되는 포트 이름을 지정하십시오.

replication_user_name

Amazon RDS 외부에서 실행 중인 MySQL 인스턴스에서 `REPLICATION CLIENT` 및 `REPLICATION SLAVE` 권한을 보유한 사용자의 ID입니다. 외부 인스턴스를 사용한 복제에만 사용되는 계정을 제공하는 것이 좋습니다.

replication_user_password

`replication_user_name`에 지정된 사용자 ID의 암호입니다.

mysql_binary_log_file_name

복제 정보를 포함하는 소스 데이터베이스 인스턴스의 이진 로그 이름입니다.

mysql_binary_log_file_location

복제 시 복제 정보를 읽기 시작하는 `mysql_binary_log_file_name` 이진수 로그 내 위치입니다.

소스 데이터베이스 인스턴스의 SHOW MASTER STATUS를 실행하여 binlog 파일 이름 및 위치를 확인할 수 있습니다.

ssl_encryption

복제 연결에 보안 소켓 계층(SSL) 암호화를 사용할지 여부를 지정하는 값입니다. 1은 SSL 암호화 사용, 0은 암호화 사용 안 함입니다. 기본값은 0입니다.

> **Note**
>
> MASTER_SSL_VERIFY_SERVER_CERT 옵션은 지원되지 않습니다. 이 옵션은 0으로 설정되어 있는데, 이는 연결이 암호화되었지만 인증서는 확인되지 않았음을 의미합니다.

사용 노트

마스터 사용자는 mysql.rds_set_external_master 프로시저를 실행해야 합니다. 이 프로시저는 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 읽기 전용 복제본으로 구성되도록 MySQL DB 인스턴스에서 실행해야 합니다.

mysql.rds_set_external_master을 실행하기 전에 소스 데이터베이스 인스턴스가 되도록 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스를 구성해야 합니다. Amazon RDS 외부에서 실행하는 MySQL 인스턴스에 연결하려면 MySQL의 외부 인스턴스에서 replication_user_name 및 replication_user_password 권한이 있는 복제 사용자를 나타내는 REPLICATION CLIENT 및 REPLICATION SLAVE 값을 지정해야 합니다.

MySQL의 외부 인스턴스를 소스 데이터베이스 인스턴스로 구성하려면

1. 선택한 MySQL 클라이언트를 사용하여 MySQL의 외부 인스턴스에 연결하고 복제에 사용될 사용자 계정을 생성합니다. 다음은 예입니다.

   MySQL 5.7

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
   ```

   MySQL 8.0

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED WITH mysql_native_password BY
    'password';
   ```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

2. MySQL의 외부 인스턴스에서 복제 사용자에게 `REPLICATION CLIENT` 및 `REPLICATION SLAVE` 권한을 부여합니다. 다음 예제에서는 도메인의 'repl_user' 사용자에게 모든 데이터베이스에 대한 `REPLICATION CLIENT` 및 `REPLICATION SLAVE` 권한을 부여합니다.

   MySQL 5.7

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
    IDENTIFIED BY 'password';
   ```

   MySQL 8.0

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
   ```

암호화된 복제를 사용하려면 SSL 연결을 사용하도록 소스 데이터베이스 인스턴스를 구성합니다.

> **Note**
>
> 가능하면 읽기 전용 복제본을 사용하여 두 Amazon RDS DB 인스턴스 간 복제를 관리하는 것이 좋습니다. 그렇게 하면 이 복제 및 다른 복제 관련 저장 프로시저만 사용하는 것이 좋습니다. 이러한 방법을 사용하면 Amazon RDS DB 인스턴스 간 더욱 복잡한 복제 토폴로지를 사용할 수 있습니다. 이러한 저장 프로시저는 주로 Amazon RDS 외부에서 실행되는 MySQL 인스턴스를 사용하여 복제할 수 있도록 하기 위한 것입니다. Amazon RDS DB 인스턴스 간 복제 관리에 대한 자세한 내용은 DB 인스턴스 읽기 전용 복제본 작업 주제를 참조하십시오.

`mysql.rds_set_external_master`을 호출하여 Amazon RDS DB 인스턴스를 읽기 전용 복제본으로 구성한 후 읽기 전용 복제본에서 mysql.rds_start_replication을 호출하여 복제 프로세스를 시작할 수 있습니다. mysql.rds_reset_external_master(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하)를 호출하여 읽기 전용 복제본 구성을 제거할 수 있습니다.

`mysql.rds_set_external_master`를 호출하면 Amazon RDS는 `set master`의 시간, 사용자 및 작업을 `mysql.rds_history` 및 `mysql.rds_replication_status` 테이블에 기록합니다.

예시

MySQL DB 인스턴스에서 다음 예제를 실행하면 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 읽기 전용 복제본이 되도록 DB 인스턴스가 구성됩니다.

```
call mysql.rds_set_external_master(
  'Externaldb.some.com',
  3306,
  'repl_user',
  'password',
  'mysql-bin-changelog.0777',
  120,
  1);
```

mysql.rds_set_external_source(RDS for MySQL 메이저 버전 8.4 이상)

RDS for MySQL DB 인스턴스가 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 읽기 전용 복제본이 되도록 구성합니다.

> **Important**
>
> 이 프로시저를 실행하려면 `autocommit`를 활성화해야 합니다. 활성화하려면 `autocommit` 파라미터를 1로 설정합니다. 파라미터 수정에 대한 자세한 정보는 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 단원을 참조하십시오.

구문

```
CALL mysql.rds_set_external_source (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
);
```

파라미터

host_name

소스 데이터베이스 인스턴스가 될 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 호스트 이름 또는 IP 주소입니다.

host_port

소스 데이터베이스 인스턴스로 구성될 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스에서 사용하는 포트입니다. 네트워크 구성에 포트 번호를 변환하는 SSH 포트 복제가 포함되는 경우 SSH(Secure Shell)에 의해 공개되는 포트 이름을 지정하십시오.

replication_user_name

Amazon RDS 외부에서 실행 중인 MySQL 인스턴스에서 `REPLICATION CLIENT` 및 `REPLICATION SLAVE` 권한을 보유한 사용자의 ID입니다. 외부 인스턴스를 사용한 복제에만 사용되는 계정을 제공하는 것이 좋습니다.

replication_user_password

`replication_user_name`에 지정된 사용자 ID의 암호입니다.

mysql_binary_log_file_name

복제 정보를 포함하는 소스 데이터베이스 인스턴스의 이진 로그 이름입니다.

mysql_binary_log_file_location

복제 시 복제 정보를 읽기 시작하는 `mysql_binary_log_file_name` 이진수 로그 내 위치입니다.

소스 데이터베이스 인스턴스의 `SHOW MASTER STATUS`를 실행하여 binlog 파일 이름 및 위치를 확인할 수 있습니다.

ssl_encryption

복제 연결에 보안 소켓 계층(SSL) 암호화를 사용할지 여부를 지정하는 값입니다. 1은 SSL 암호화 사용, 0은 암호화 사용 안 함입니다. 기본값은 0입니다.

> **Note**
>
> `SOURCE_SSL_VERIFY_SERVER_CERT` 옵션은 지원되지 않습니다. 이 옵션은 0으로 설정되어 있는데, 이는 연결이 암호화되었지만 인증서는 확인되지 않았음을 의미합니다.

사용 노트

관리 사용자는 `mysql.rds_set_external_source` 프로시저를 실행해야 합니다. 이 프로시저는 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 읽기 전용 복제본으로 구성되도록 RDS for MySQL DB 인스턴스에서 실행해야 합니다.

`mysql.rds_set_external_source`을 실행하기 전에 소스 데이터베이스 인스턴스가 되도록 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스를 구성해야 합니다. Amazon RDS 외부에서 실행하는 MySQL 인스턴스에 연결하려면 MySQL의 외부 인스턴스에서 `replication_user_name` 및 `replication_user_password` 권한이 있는 복제 사용자를 나타내는 `REPLICATION CLIENT` 및 `REPLICATION SLAVE` 값을 지정해야 합니다.

MySQL의 외부 인스턴스를 소스 데이터베이스 인스턴스로 구성하려면

1. 선택한 MySQL 클라이언트를 사용하여 MySQL의 외부 인스턴스에 연결하고 복제에 사용될 사용자 계정을 생성합니다. 다음은 예입니다.

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
   ```

   > **Note**
   >
   > 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

2. MySQL의 외부 인스턴스에서 복제 사용자에게 `REPLICATION CLIENT` 및 `REPLICATION SLAVE` 권한을 부여합니다. 다음 예제에서는 도메인의 'repl_user' 사용자에게 모든 데이터베이스에 대한 `REPLICATION CLIENT` 및 `REPLICATION SLAVE` 권한을 부여합니다.

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
   ```

암호화된 복제를 사용하려면 SSL 연결을 사용하도록 소스 데이터베이스 인스턴스를 구성합니다. 또한 mysql.rds_import_binlog_ssl_material 프로시저를 사용하여 인증 기관(CA) 인증서, 클라이언트 인증서, 클라이언트 키를 DB 인스턴스나 DB 클러스터로 가져옵니다.

> **Note**
>
> 가능하면 읽기 전용 복제본을 사용하여 두 Amazon RDS DB 인스턴스 간 복제를 관리하는 것이 좋습니다. 그렇게 하면 이 복제 및 다른 복제 관련 저장 프로시저만 사용하는 것이 좋습니다. 이러한 방법을 사용하면 Amazon RDS DB 인스턴스 간 더욱 복잡한 복제 토폴로지를 사용할

수 있습니다. 이러한 저장 프로시저는 주로 Amazon RDS 외부에서 실행되는 MySQL 인스턴스를 사용하여 복제할 수 있도록 하기 위한 것입니다. Amazon RDS DB 인스턴스 간 복제 관리에 대한 자세한 내용은 DB 인스턴스 읽기 전용 복제본 작업 주제를 참조하십시오.

`mysql.rds_set_external_source`를 직접 호출하여 RDS for MySQL DB 인스턴스를 읽기 전용 복제본으로 구성한 후 읽기 전용 복제본에서 mysql.rds_start_replication을 직접 호출하여 복제 프로세스를 시작할 수 있습니다. mysql.rds_reset_external_source(RDS for MySQL 메이저 버전 8.4 이상)를 호출하여 읽기 전용 복제본 구성을 제거할 수 있습니다.

`mysql.rds_set_external_source`를 호출하면 Amazon RDS는 `set master`의 시간, 사용자 및 작업을 `mysql.rds_history` 및 `mysql.rds_replication_status` 테이블에 기록합니다.

예시

RDS for MySQL DB 인스턴스에서 실행할 경우 다음 예시는 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 읽기 전용 복제본이 되도록 DB 인스턴스를 구성합니다.

```
call mysql.rds_set_external_source(
  'Externaldb.some.com',
  3306,
  'repl_user',
  'password',
  'mysql-bin-changelog.0777',
  120,
  1);
```

mysql.rds_set_external_master_with_auto_position(RDS for MySQL 메이저 버전 8.0 이하)

RDS for MySQL DB 인스턴스를 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 읽기 전용 복제본으로 구성합니다. 또한 이 프로시저는 전역 트랜잭션 식별자(GTID)를 기반으로 한 지연된 복제 및 복제를 구성합니다.

Important

이 프로시저를 실행하려면 `autocommit`를 활성화해야 합니다. 활성화하려면 `autocommit` 파라미터를 1로 설정합니다. 파라미터 수정에 대한 자세한 정보는 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 단원을 참조하십시오.

구문

```
CALL mysql.rds_set_external_master_with_auto_position (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , ssl_encryption
  , delay
);
```

파라미터

host_name

소스 데이터베이스 인스턴스가 될 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 호스트 이름 또는 IP 주소입니다.

host_port

소스 데이터베이스 인스턴스로 구성될 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스에서 사용하는 포트입니다. 네트워크 구성에 포트 번호를 변환하는 SSH 포트 복제가 포함되는 경우 SSH(Secure Shell)에 의해 공개되는 포트 이름을 지정하십시오.

replication_user_name

Amazon RDS 외부에서 실행 중인 MySQL 인스턴스에서 `REPLICATION CLIENT` 및 `REPLICATION SLAVE` 권한을 보유한 사용자의 ID입니다. 외부 인스턴스를 사용한 복제에만 사용되는 계정을 제공하는 것이 좋습니다.

replication_user_password

`replication_user_name`에 지정된 사용자 ID의 암호입니다.

ssl_encryption

복제 연결에 보안 소켓 계층(SSL) 암호화를 사용할지 여부를 지정하는 값입니다. 1은 SSL 암호화 사용, 0은 암호화 사용 안 함입니다. 기본값은 0입니다.

> **Note**
>
> `MASTER_SSL_VERIFY_SERVER_CERT` 옵션은 지원되지 않습니다. 이 옵션은 0으로 설정되어 있는데, 이는 연결이 암호화되었지만 인증서는 확인되지 않았음을 의미합니다.

delay

소스 데이터베이스 인스턴스에서 복제를 지연할 최소 시간(초)입니다.

이 파라미터에 대한 제한은 1일(86,400초)입니다.

사용 노트

마스터 사용자는 mysql.rds_set_external_master_with_auto_position 프로시저를 실행해야 합니다. 이 프로시저는 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 읽기 전용 복제본으로 구성되도록 MySQL DB 인스턴스에서 실행해야 합니다.

이 프로시저는 모든 RDS for MySQL 5.7 버전 및 RDS for MySQL 8.0.26 이상의 8.0 버전에서 지원됩니다.

mysql.rds_set_external_master_with_auto_position을 실행하기 전에 소스 데이터베이스 인스턴스가 되도록 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스를 구성해야 합니다. Amazon RDS 외부에서 실행하는 MySQL 인스턴스에 연결하려면 replication_user_name 및 replication_user_password의 값을 지정해야 합니다. 이러한 값은 MySQL의 외부 인스턴스에 대해 REPLICATION CLIENT 및 REPLICATION SLAVE 권한을 가진 복제 사용자를 나타내야 합니다.

MySQL의 외부 인스턴스를 소스 데이터베이스 인스턴스로 구성하려면

1. 선택한 MySQL 클라이언트를 사용하여 MySQL의 외부 인스턴스에 연결하고 복제에 사용될 사용자 계정을 생성합니다. 다음은 예제입니다.

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'SomePassW0rd'
   ```

2. MySQL의 외부 인스턴스에서 복제 사용자에게 REPLICATION CLIENT 및 REPLICATION SLAVE 권한을 부여합니다. 다음 예제에서는 도메인의 REPLICATION CLIENT 사용자에게 모든 데이터베이스에 대한 REPLICATION SLAVE 및 'repl_user' 권한을 부여합니다.

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
   IDENTIFIED BY 'SomePassW0rd'
   ```

자세한 내용은 외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성 섹션을 참조하세요.

> **Note**
>
> 가능하면 읽기 전용 복제본을 사용하여 두 Amazon RDS DB 인스턴스 간 복제를 관리하는 것이 좋습니다. 그렇게 하면 이 복제 및 다른 복제 관련 저장 프로시저만 사용하는 것이 좋습니다. 이러한 방법을 사용하면 Amazon RDS DB 인스턴스 간 더욱 복잡한 복제 토폴로지를 사용할 수 있습니다. 이러한 저장 프로시저는 주로 Amazon RDS 외부에서 실행되는 MySQL 인스턴스를 사용하여 복제할 수 있도록 하기 위한 것입니다. Amazon RDS DB 인스턴스 간 복제 관리에 대한 자세한 내용은 DB 인스턴스 읽기 전용 복제본 작업 주제를 참조하십시오.

`mysql.rds_set_external_master_with_auto_position` 직접 호출 전에 the section called "mysql.rds_set_external_source_gtid_purged"를 직접 호출하여 외부 소스의 지정된 GTID 범위로 시스템 변수를 설정합니다.

`mysql.rds_set_external_master_with_auto_position`을 호출하여 Amazon RDS DB 인스턴스를 읽기 전용 복제본으로 구성한 후 읽기 전용 복제본에서 mysql.rds_start_replication을 호출하여 복제 프로세스를 시작할 수 있습니다. mysql.rds_reset_external_master(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하)를 호출하여 읽기 전용 복제본 구성을 제거할 수 있습니다.

`mysql.rds_set_external_master_with_auto_position`를 호출하면 Amazon RDS는 `set master`의 시간, 사용자 및 작업을 `mysql.rds_history` 및 `mysql.rds_replication_status` 테이블에 기록합니다.

재해 복구의 경우 이 프로시저를 mysql.rds_start_replication_until 또는 mysql.rds_start_replication_until_gtid 저장 프로시저와 함께 사용할 수 있습니다. 지연된 읽기 전용 복제본에 대한 변경 사항을 재해 직전 시간으로 롤포워드하려면 `mysql.rds_set_external_master_with_auto_position` 프로시저를 실행할 수 있습니다. `mysql.rds_start_replication_until_gtid` 프로시저에서 복제를 중지한 이후에 읽기 전용 복제본을 독립 DB 인스턴스로 승격의 지침에 따라 읽기 전용 복제본을 새 기본 DB 인스턴스가 되도록 승격할 수 있습니다.

`mysql.rds_rds_start_replication_until_gtid` 프로시저를 사용하려면 GTID를 기반으로 한 복제를 활성화해야 합니다. 재해 원인으로 알려진 특정 GTID 기반 트랜잭션을 건너 뛰려면 mysql.rds_skip_transaction_with_gtid 저장 프로시저를 사용할 수 있습니다. GTID 기반 복제 작업에 대한 자세한 내용은 GTID 기반 복제 사용 단원을 참조하십시오.

예시

MySQL DB 인스턴스에서 다음 예제를 실행하면 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 읽기 전용 복제본이 되도록 DB 인스턴스가 구성됩니다. MySQL DB 인스턴스에서 최소 복제 지연을 1시간(3,600초)으로 설정합니다. Amazon RDS 외부에서 실행 중인 MySQL 소스 데이터베이스 인스턴스의 변경 사항은 최소 1시간 동안 MySQL DB 인스턴스 읽기 전용 복제본에 적용되지 않습니다.

```
call mysql.rds_set_external_master_with_auto_position(
  'Externaldb.some.com',
  3306,
  'repl_user',
  'SomePassW0rd',
  1,
  3600);
```

mysql.rds_set_external_source_with_auto_position(RDS for MySQL 메이저 버전 8.4 이상)

RDS for MySQL DB 인스턴스를 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 읽기 전용 복제본으로 구성합니다. 또한 이 프로시저는 전역 트랜잭션 식별자(GTID)를 기반으로 한 지연된 복제 및 복제를 구성합니다.

> **Important**
>
> 이 프로시저를 실행하려면 `autocommit`를 활성화해야 합니다. 활성화하려면 `autocommit` 파라미터를 1로 설정합니다. 파라미터 수정에 대한 자세한 정보는 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 단원을 참조하십시오.

구문

```
CALL mysql.rds_set_external_source_with_auto_position (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , ssl_encryption
  , delay
);
```

파라미터

host_name

소스 데이터베이스 인스턴스가 될 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 호스트 이름 또는 IP 주소입니다.

host_port

소스 데이터베이스 인스턴스로 구성될 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스에서 사용하는 포트입니다. 네트워크 구성에 포트 번호를 변환하는 SSH 포트 복제가 포함되는 경우 SSH(Secure Shell)에 의해 공개되는 포트 이름을 지정하십시오.

replication_user_name

Amazon RDS 외부에서 실행 중인 MySQL 인스턴스에서 `REPLICATION CLIENT` 및 `REPLICATION SLAVE` 권한을 보유한 사용자의 ID입니다. 외부 인스턴스를 사용한 복제에만 사용되는 계정을 제공하는 것이 좋습니다.

replication_user_password

`replication_user_name`에 지정된 사용자 ID의 암호입니다.

ssl_encryption

복제 연결에 보안 소켓 계층(SSL) 암호화를 사용할지 여부를 지정하는 값입니다. 1은 SSL 암호화 사용, 0은 암호화 사용 안 함입니다. 기본값은 0입니다.

> **Note**
>
> `SOURCE_SSL_VERIFY_SERVER_CERT` 옵션은 지원되지 않습니다. 이 옵션은 0으로 설정되어 있는데, 이는 연결이 암호화되었지만 인증서는 확인되지 않았음을 의미합니다.

delay

소스 데이터베이스 인스턴스에서 복제를 지연할 최소 시간(초)입니다.

이 파라미터에 대한 제한은 1일(86,400초)입니다.

사용 노트

관리 사용자는 `mysql.rds_set_external_source_with_auto_position` 프로시저를 실행해야 합니다. 이 프로시저는 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 읽기 전용 복제본으로 구성되도록 MySQL DB 인스턴스에서 실행해야 합니다.

`mysql.rds_set_external_source_with_auto_position`을 실행하기 전에 소스 데이터베이스 인스턴스가 되도록 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스를 구성해야 합니다. Amazon RDS 외부에서 실행하는 MySQL 인스턴스에 연결하려면 `replication_user_name` 및 `replication_user_password`의 값을 지정해야 합니다. 이러한 값은 MySQL의 외부 인스턴스에 대해 `REPLICATION CLIENT` 및 `REPLICATION SLAVE` 권한을 가진 복제 사용자를 나타내야 합니다.

MySQL의 외부 인스턴스를 소스 데이터베이스 인스턴스로 구성하려면

1. 선택한 MySQL 클라이언트를 사용하여 MySQL의 외부 인스턴스에 연결하고 복제에 사용될 사용자 계정을 생성합니다. 다음은 예제입니다.

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'SomePassW0rd'
   ```

2. MySQL의 외부 인스턴스에서 복제 사용자에게 `REPLICATION CLIENT` 및 `REPLICATION SLAVE` 권한을 부여합니다. 다음 예제에서는 도메인의 `REPLICATION CLIENT` 사용자에게 모든 데이터베이스에 대한 `REPLICATION SLAVE` 및 `'repl_user'` 권한을 부여합니다.

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
   IDENTIFIED BY 'SomePassW0rd'
   ```

자세한 내용은 외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성 섹션을 참조하세요.

> **Note**
>
> 가능하면 읽기 전용 복제본을 사용하여 두 Amazon RDS DB 인스턴스 간 복제를 관리하는 것이 좋습니다. 그렇게 하면 이 복제 및 다른 복제 관련 저장 프로시저만 사용하는 것이 좋습니다. 이러한 방법을 사용하면 Amazon RDS DB 인스턴스 간 더욱 복잡한 복제 토폴로지를 사용할 수 있습니다. 이러한 저장 프로시저는 주로 Amazon RDS 외부에서 실행되는 MySQL 인스턴스를 사용하여 복제할 수 있도록 하기 위한 것입니다. Amazon RDS DB 인스턴스 간 복제 관리에 대한 자세한 내용은 DB 인스턴스 읽기 전용 복제본 작업 주제를 참조하십시오.

`mysql.rds_set_external_source_with_auto_position` 직접 호출 전에 the section called "mysql.rds_set_external_source_gtid_purged"를 직접 호출하여 외부 소스의 지정된 GTID 범위로 시스템 변수를 설정합니다.

`mysql.rds_set_external_source_with_auto_position`을 호출하여 Amazon RDS DB 인스턴스를 읽기 전용 복제본으로 구성한 후 읽기 전용 복제본에서 mysql.rds_start_replication을 호출하여 복제 프로세스를 시작할 수 있습니다. mysql.rds_reset_external_source(RDS for MySQL 메이저 버전 8.4 이상)를 호출하여 읽기 전용 복제본 구성을 제거할 수 있습니다.

`mysql.rds_set_external_source_with_auto_position`를 호출하면 Amazon RDS는 `set master`의 시간, 사용자 및 작업을 `mysql.rds_history` 및 `mysql.rds_replication_status` 테이블에 기록합니다.

재해 복구의 경우 이 프로시저를 mysql.rds_start_replication_until 또는 mysql.rds_start_replication_until_gtid 저장 프로시저와 함께 사용할 수 있습니다. 지연된 읽기 전용 복제본에 대한 변경 사항을 재해 직전 시간으로 롤포워드하려면 `mysql.rds_set_external_source_with_auto_position` 프로시저를 실행할 수 있습니다. `mysql.rds_start_replication_until_gtid` 프로시저에서 복제를 중지한 이후에 읽기 전용 복제본을 독립 DB 인스턴스로 승격의 지침에 따라 읽기 전용 복제본을 새 기본 DB 인스턴스가 되도록 승격할 수 있습니다.

`mysql.rds_rds_start_replication_until_gtid` 프로시저를 사용하려면 GTID를 기반으로 한 복제를 활성화해야 합니다. 재해 원인으로 알려진 특정 GTID 기반 트랜잭션을 건너 뛰려면 mysql.rds_skip_transaction_with_gtid 저장 프로시저를 사용할 수 있습니다. GTID 기반 복제 작업에 대한 자세한 내용은 GTID 기반 복제 사용 단원을 참조하십시오.

예시

MySQL DB 인스턴스에서 다음 예제를 실행하면 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 읽기 전용 복제본이 되도록 DB 인스턴스가 구성됩니다. MySQL DB 인스턴스에서 최소 복제 지연을 1시간(3,600초)으로 설정합니다. Amazon RDS 외부에서 실행 중인 MySQL 소스 데이터베이스 인스턴스의 변경 사항은 최소 1시간 동안 MySQL DB 인스턴스 읽기 전용 복제본에 적용되지 않습니다.

```
call mysql.rds_set_external_source_with_auto_position(
  'Externaldb.some.com',
  3306,
  'repl_user',
  'SomePassW0rd',
  1,
```

```
  3600);
```

mysql.rds_set_external_master_with_delay(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하)

Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 읽기 전용 복제본이 되도록 RDS for MySQL DB 인스턴스를 구성하고 지연 복제를 구성합니다.

> **Important**
>
> 이 프로시저를 실행하려면 `autocommit`를 활성화해야 합니다. 활성화하려면 `autocommit` 파라미터를 1로 설정합니다. 파라미터 수정에 대한 자세한 정보는 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 단원을 참조하십시오.

구문

```
CALL mysql.rds_set_external_master_with_delay(
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
  , delay
);
```

파라미터

host_name

소스 데이터베이스 인스턴스가 될 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 호스트 이름 또는 IP 주소입니다.

host_port

소스 데이터베이스 인스턴스로 구성될 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스에서 사용하는 포트입니다. 네트워크 구성에 포트 번호를 변환하는 SSH 포트 복제가 포함되는 경우 SSH에 의해 공개되는 포트 이름을 지정하십시오.

replication_user_name

Amazon RDS 외부에서 실행 중인 MySQL 인스턴스에서 REPLICATION CLIENT 및 REPLICATION SLAVE 권한을 보유한 사용자의 ID입니다. 외부 인스턴스를 사용한 복제에만 사용되는 계정을 제공하는 것이 좋습니다.

replication_user_password

replication_user_name에 지정된 사용자 ID의 암호입니다.

mysql_binary_log_file_name

복제 정보를 포함하는 소스 데이터베이스 인스턴스의 이진 로그 이름입니다.

mysql_binary_log_file_location

복제 시 복제 정보를 읽기 시작할 mysql_binary_log_file_name 이진 로그 내 위치입니다.

소스 데이터베이스 인스턴스의 SHOW MASTER STATUS를 실행하여 binlog 파일 이름 및 위치를 확인할 수 있습니다.

ssl_encryption

복제 연결에 보안 소켓 계층(SSL) 암호화를 사용할지 여부를 지정하는 값입니다. 1은 SSL 암호화 사용, 0은 암호화 사용 안 함입니다. 기본값은 0입니다.

> **Note**
>
> MASTER_SSL_VERIFY_SERVER_CERT 옵션은 지원되지 않습니다. 이 옵션은 0으로 설정되어 있는데, 이는 연결이 암호화되었지만 인증서는 확인되지 않았음을 의미합니다.

delay

소스 데이터베이스 인스턴스에서 복제를 지연할 최소 시간(초)입니다.

이 파라미터에 대한 제한은 1일(86,400초)입니다.

사용 노트

마스터 사용자는 mysql.rds_set_external_master_with_delay 프로시저를 실행해야 합니다. 이 프로시저는 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 읽기 전용 복제본으로 구성되도록 MySQL DB 인스턴스에서 실행해야 합니다.

`mysql.rds_set_external_master_with_delay`을 실행하기 전에 소스 데이터베이스 인스턴스가 되도록 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스를 구성해야 합니다. Amazon RDS 외부에서 실행하는 MySQL 인스턴스에 연결하려면 `replication_user_name` 및 `replication_user_password`의 값을 지정해야 합니다. 이러한 값은 MySQL의 외부 인스턴스에 대해 `REPLICATION CLIENT` 및 `REPLICATION SLAVE` 권한을 가진 복제 사용자를 나타내야 합니다.

MySQL의 외부 인스턴스를 소스 데이터베이스 인스턴스로 구성하려면

1. 선택한 MySQL 클라이언트를 사용하여 MySQL의 외부 인스턴스에 연결하고 복제에 사용될 사용자 계정을 생성합니다. 다음은 예제입니다.

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'SomePassW0rd'
   ```

2. MySQL의 외부 인스턴스에서 복제 사용자에게 `REPLICATION CLIENT` 및 `REPLICATION SLAVE` 권한을 부여합니다. 다음 예제에서는 도메인의 `REPLICATION CLIENT` 사용자에게 모든 데이터베이스에 대한 `REPLICATION SLAVE` 및 `'repl_user'` 권한을 부여합니다.

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
   IDENTIFIED BY 'SomePassW0rd'
   ```

자세한 내용은 외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성 섹션을 참조하세요.

> **Note**
>
> 가능하면 읽기 전용 복제본을 사용하여 두 Amazon RDS DB 인스턴스 간 복제를 관리하는 것이 좋습니다. 그렇게 하면 이 복제 및 다른 복제 관련 저장 프로시저만 사용하는 것이 좋습니다. 이러한 방법을 사용하면 Amazon RDS DB 인스턴스 간 더욱 복잡한 복제 토폴로지를 사용할 수 있습니다. 이러한 저장 프로시저는 주로 Amazon RDS 외부에서 실행되는 MySQL 인스턴스를 사용하여 복제할 수 있도록 하기 위한 것입니다. Amazon RDS DB 인스턴스 간 복제 관리에 대한 자세한 내용은 DB 인스턴스 읽기 전용 복제본 작업 주제를 참조하십시오.

`mysql.rds_set_external_master_with_delay`을 호출하여 Amazon RDS DB 인스턴스를 읽기 전용 복제본으로 구성한 후 읽기 전용 복제본에서 mysql.rds_start_replication을 호출하여 복제 프로세스를 시작할 수 있습니다. mysql.rds_reset_external_master(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하)를 호출하여 읽기 전용 복제본 구성을 제거할 수 있습니다.

mysql.rds_set_external_master_with_delay를 호출하면 Amazon RDS는 set master의 시간, 사용자 및 작업을 mysql.rds_history 및 mysql.rds_replication_status 테이블에 기록합니다.

재해 복구의 경우 이 프로시저를 mysql.rds_start_replication_until 또는 mysql.rds_start_replication_until_gtid 저장 프로시저와 함께 사용할 수 있습니다. 지연된 읽기 전용 복제본에 대한 변경 사항을 재해 직전 시간으로 롤포워드하려면 mysql.rds_set_external_master_with_delay 프로시저를 실행할 수 있습니다. mysql.rds_start_replication_until 프로시저에서 복제를 중지한 이후에 읽기 전용 복제본을 독립 DB 인스턴스로 승격의 지침에 따라 읽기 전용 복제본을 새 기본 DB 인스턴스가 되도록 승격할 수 있습니다.

mysql.rds_rds_start_replication_until_gtid 프로시저를 사용하려면 GTID를 기반으로 한 복제를 활성화해야 합니다. 재해 원인으로 알려진 특정 GTID 기반 트랜잭션을 건너 뛰려면 mysql.rds_skip_transaction_with_gtid 저장 프로시저를 사용할 수 있습니다. GTID 기반 복제 작업에 대한 자세한 내용은 GTID 기반 복제 사용 단원을 참조하십시오.

mysql.rds_set_external_master_with_delay 프로시저는 다음 버전의 RDS for MySQL에서 사용할 수 있습니다.

- MySQL 8.0.26 및 8.0 버전 이상
- 모든 5.6 버전

예시

MySQL DB 인스턴스에서 다음 예제를 실행하면 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 읽기 전용 복제본이 되도록 DB 인스턴스가 구성됩니다. MySQL DB 인스턴스에서 최소 복제 지연을 1시간(3,600초)으로 설정합니다. Amazon RDS 외부에서 실행 중인 MySQL 소스 데이터베이스 인스턴스의 변경 사항은 최소 1시간 동안 MySQL DB 인스턴스 읽기 전용 복제본에 적용되지 않습니다.

```
call mysql.rds_set_external_master_with_delay(
  'Externaldb.some.com',
  3306,
  'repl_user',
  'SomePassW0rd',
  'mysql-bin-changelog.000777',
  120,
  1,
  3600);
```

mysql.rds_set_external_source_with_delay(RDS for MySQL 메이저 버전 8.4 이상)

Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 읽기 전용 복제본이 되도록 RDS for MySQL DB 인스턴스를 구성하고 지연 복제를 구성합니다.

> **Important**
>
> 이 프로시저를 실행하려면 `autocommit`를 활성화해야 합니다. 활성화하려면 `autocommit` 파라미터를 1로 설정합니다. 파라미터 수정에 대한 자세한 정보는 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 단원을 참조하십시오.

구문

```
CALL mysql.rds_set_external_source_with_delay (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
  , delay
);
```

파라미터

host_name

소스 데이터베이스 인스턴스가 될 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 호스트 이름 또는 IP 주소입니다.

host_port

소스 데이터베이스 인스턴스로 구성될 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스에서 사용하는 포트입니다. 네트워크 구성에 포트 번호를 변환하는 SSH 포트 복제가 포함되는 경우 SSH에 의해 공개되는 포트 이름을 지정하십시오.

replication_user_name

Amazon RDS 외부에서 실행 중인 MySQL 인스턴스에서 REPLICATION CLIENT 및 REPLICATION SLAVE 권한을 보유한 사용자의 ID입니다. 외부 인스턴스를 사용한 복제에만 사용되는 계정을 제공하는 것이 좋습니다.

replication_user_password

replication_user_name에 지정된 사용자 ID의 암호입니다.

mysql_binary_log_file_name

복제 정보를 포함하는 소스 데이터베이스 인스턴스의 이진 로그 이름입니다.

mysql_binary_log_file_location

복제 시 복제 정보를 읽기 시작할 mysql_binary_log_file_name 이진 로그 내 위치입니다.

소스 데이터베이스 인스턴스의 SHOW MASTER STATUS를 실행하여 binlog 파일 이름 및 위치를 확인할 수 있습니다.

ssl_encryption

복제 연결에 보안 소켓 계층(SSL) 암호화를 사용할지 여부를 지정하는 값입니다. 1은 SSL 암호화 사용, 0은 암호화 사용 안 함입니다. 기본값은 0입니다.

> **Note**
>
> SOURCE_SSL_VERIFY_SERVER_CERT 옵션은 지원되지 않습니다. 이 옵션은 0으로 설정되어 있는데, 이는 연결이 암호화되었지만 인증서는 확인되지 않았음을 의미합니다.

delay

소스 데이터베이스 인스턴스에서 복제를 지연할 최소 시간(초)입니다.

이 파라미터에 대한 제한은 1일(86,400초)입니다.

사용 노트

관리 사용자는 mysql.rds_set_external_source_with_delay 프로시저를 실행해야 합니다. 이 프로시저는 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 읽기 전용 복제본으로 구성되도록 MySQL DB 인스턴스에서 실행해야 합니다.

mysql.rds_set_external_source_with_delay을 실행하기 전에 소스 데이터베이스 인스턴스가 되도록 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스를 구성해야 합니다. Amazon RDS 외부에서 실행하는 MySQL 인스턴스에 연결하려면 replication_user_name 및 replication_user_password의 값을 지정해야 합니다. 이러한 값은 MySQL의 외부 인스턴스에 대해 REPLICATION CLIENT 및 REPLICATION SLAVE 권한을 가진 복제 사용자를 나타내야 합니다.

MySQL의 외부 인스턴스를 소스 데이터베이스 인스턴스로 구성하려면

1. 선택한 MySQL 클라이언트를 사용하여 MySQL의 외부 인스턴스에 연결하고 복제에 사용될 사용자 계정을 생성합니다. 다음은 예제입니다.

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'SomePassW0rd'
```

2. MySQL의 외부 인스턴스에서 복제 사용자에게 REPLICATION CLIENT 및 REPLICATION SLAVE 권한을 부여합니다. 다음 예제에서는 도메인의 REPLICATION CLIENT 사용자에게 모든 데이터베이스에 대한 REPLICATION SLAVE 및 'repl_user' 권한을 부여합니다.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
IDENTIFIED BY 'SomePassW0rd'
```

자세한 내용은 외부 소스 인스턴스를 사용하여 이진 로그 파일 위치 복제 구성 섹션을 참조하세요.

> **Note**
>
> 가능하면 읽기 전용 복제본을 사용하여 두 Amazon RDS DB 인스턴스 간 복제를 관리하는 것이 좋습니다. 그렇게 하면 이 복제 및 다른 복제 관련 저장 프로시저만 사용하는 것이 좋습니다. 이러한 방법을 사용하면 Amazon RDS DB 인스턴스 간 더욱 복잡한 복제 토폴로지를 사용할 수 있습니다. 이러한 저장 프로시저는 주로 Amazon RDS 외부에서 실행되는 MySQL 인스턴스를 사용하여 복제할 수 있도록 하기 위한 것입니다. Amazon RDS DB 인스턴스 간 복제 관리에 대한 자세한 내용은 DB 인스턴스 읽기 전용 복제본 작업 주제를 참조하십시오.

mysql.rds_set_external_source_with_delay을 호출하여 Amazon RDS DB 인스턴스를 읽기 전용 복제본으로 구성한 후 읽기 전용 복제본에서 mysql.rds_start_replication을 호출하여 복제 프로세스를 시작할 수 있습니다. mysql.rds_reset_external_source(RDS for MySQL 메이저 버전 8.4 이상)를 호출하여 읽기 전용 복제본 구성을 제거할 수 있습니다.

mysql.rds_set_external_source_with_delay를 호출하면 Amazon RDS는 set master의 시간, 사용자 및 작업을 mysql.rds_history 및 mysql.rds_replication_status 테이블에 기록합니다.

재해 복구의 경우 이 프로시저를 mysql.rds_start_replication_until 또는 mysql.rds_start_replication_until_gtid 저장 프로시저와 함께 사용할 수 있습니다. 지연된 읽기 전용 복제본에 대한 변경 사항을 재해 직전 시간으로 롤포워드하려면 mysql.rds_set_external_source_with_delay 프로시저를 실행할 수 있습니다. mysql.rds_start_replication_until 프로시저에서 복제를 중지한 이후에 읽기 전용 복제본을 독립 DB 인스턴스로 승격의 지침에 따라 읽기 전용 복제본을 새 기본 DB 인스턴스가 되도록 승격할 수 있습니다.

mysql.rds_rds_start_replication_until_gtid 프로시저를 사용하려면 GTID를 기반으로 한 복제를 활성화해야 합니다. 재해 원인으로 알려진 특정 GTID 기반 트랜잭션을 건너 뛰려면 mysql.rds_skip_transaction_with_gtid 저장 프로시저를 사용할 수 있습니다. GTID 기반 복제 작업에 대한 자세한 내용은 GTID 기반 복제 사용 단원을 참조하십시오.

예시

MySQL DB 인스턴스에서 다음 예제를 실행하면 Amazon RDS 외부에서 실행 중인 MySQL 인스턴스의 읽기 전용 복제본이 되도록 DB 인스턴스가 구성됩니다. MySQL DB 인스턴스에서 최소 복제 지연을 1시간(3,600초)으로 설정합니다. Amazon RDS 외부에서 실행 중인 MySQL 소스 데이터베이스 인스턴스의 변경 사항은 최소 1시간 동안 MySQL DB 인스턴스 읽기 전용 복제본에 적용되지 않습니다.

```
call mysql.rds_set_external_source_with_delay(
  'Externaldb.some.com',
  3306,
  'repl_user',
  'SomePassW0rd',
  'mysql-bin-changelog.000777',
  120,
  1,
  3600);
```

mysql.rds_set_external_source_gtid_purged

외부 소스의 지정된 GTID 범위를 사용하여 gtid_purged 시스템 변수를 설정합니다. gtid_purged 값은 자동 위치 지정을 사용하여 복제를 재개하도록 GTID 기반 복제를 구성하는 데 필요합니다.

> **Important**
>
> 이 프로시저를 실행하려면 `autocommit`를 활성화해야 합니다. 활성화하려면 `autocommit` 파라미터를 1로 설정합니다. 파라미터 수정에 대한 자세한 정보는 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 단원을 참조하십시오.

구문

```
CALL mysql.rds_set_external_source_gtid_purged(
  server_uuid
  , start_pos
  , end_pos
);
```

파라미터

server_uuid

GTID 범위를 가져오는 외부 서버의 범용 고유 식별자(UUID)입니다.

start_pos

설정할 GTID 범위의 시작 위치입니다.

end_pos

설정할 GTID 범위의 종료 위치입니다.

사용 노트

`mysql.rds_set_external_source_gtid_purged` 프로시저는 MySQL 8.0.37 이상의 8.0 버전에서만 사용할 수 있습니다.

mysql.rds_set_external_master_with_auto_position(RDS for MySQL 메이저 버전 8.0 이하), mysql.rds_set_external_source_with_auto_position(RDS for MySQL 메이저 버전 8.4 이상) 또는 mysql.rds_set_external_source_with_auto_position_for_channel을 직접 호출하기 전에 `mysql.rds_set_external_source_gtid_purged`를 직접 호출합니다.

`mysql.rds_set_external_source_gtid_purged` 직접 호출 전에 데이터베이스의 모든 활성 복제 채널을 중지해야 합니다. 채널 상태를 확인하려면 `SHOW REPLICA STATUS` MySQL 문을 사용합니

다. 채널에서 복제를 중지하려면 the section called “mysql.rds_stop_replication_for_channel”을 직접 호출합니다.

지정하는 GTID 범위는 기존 GTID_PURGED 값의 상위 집합이어야 합니다. 이 저장 프로시저는 GTID_PURGED 값을 설정하기 전에 다음 값을 확인합니다.

- server_uuid는 유효합니다.
- start_pos 값이 0보다 크거나 end_pos보다 작습니다.
- end_pos 값이 start_pos보다 크거나 같습니다.

외부 서버의 GTID 집합에 여러 범위의 값이 포함된 경우 다른 GTID 집합 값을 사용하여 프로시저를 여러 번 직접 호출해 보세요.

mysql.rds_set_external_source_gtid_purged를 직접 호출하면 Amazon RDS는 set gtid_purged의 시간, 사용자 및 작업을 mysql.rds_history 테이블에 기록합니다.

복제에 사용하는 백업의 gtid_purged 값을 적절하게 설정하지 않으면 복제 프로세스 중에 트랜잭션이 누락되거나 중복될 수 있습니다. 다음 단계를 수행하여 올바른 gtid_purged 값을 설정합니다.

복제본에 gtid_purged 값을 설정하려면

1. 복제 시작 지점으로 사용할 시점 또는 특정 백업 파일을 결정합니다. 이는 논리적 백업(mysqldump 파일) 또는 물리적 백업(Amazon RDS 스냅샷)일 수 있습니다.
2. gtid_executed 값을 결정합니다. 이 값은 서버에서 커밋된 모든 GTID 집합을 나타냅니다. 이 값을 검색하려면 소스 인스턴스에서 다음 중 하나를 수행합니다.
   - 백업을 수행할 때 SQL 문(SELECT @@GLOBAL.GTID_EXECUTED;)을 실행합니다.
   - 관련 옵션이 해당 백업 유틸리티에 포함되어 있는 경우 백업 파일에서 값을 추출합니다. 자세한 내용은 MySQL 설명서의 set-gtid-purged 옵션을 참조하세요.
3. mysql.rds_set_external_source_gtid_purged에 대한 직접 호출에 사용할 gtid_purged 값을 결정합니다. gtid_purged 값에는 소스 인스턴스에서 실행되어 복제에 더 이상 필요하지 않은 GTID가 모두 포함되어야 합니다. 따라서 gtid_purged 값은 이전 단계에서 검색한 gtid_executed 값의 하위 집합이어야 합니다.

   gtid_purged 값을 확인하려면 백업에 포함되지 않아 복제에 더 이상 필요하지 않은 GTID를 식별합니다. 바이너리 로그를 분석하거나 mysqlbinlog와 같은 도구를 사용하여 바이너리 로그에서 제거된 GTID를 찾으면 됩니다.

또는 백업 지점까지의 모든 바이너리 로그를 포함하는 일관된 백업이 있는 경우 백업 지점의 `gtid_executed` 값과 동일하게 `gtid_purged` 값을 설정할 수 있습니다.

4. 백업과 일치하는 적절한 `gtid_purged` 값을 결정한 후 RDS for MySQL DB 인스턴스에서 `mysql.rds_set_external_source_gtid_purged` 저장 프로시저를 직접 호출하여 값을 설정합니다.

예시

MySQL DB 인스턴스에서 실행하는 경우 다음 예제에서는 UUID가 `12345678-abcd-1234-efgh-123456789abc`, 시작 위치가 `1`, 종료 위치가 `100`인 외부 MySQL 서버의 GTID 범위를 설정합니다. 결과 GTID 값은 `+12345678-abcd-1234-efgh-123456789abc:1-100`으로 설정됩니다.

```
CALL mysql.rds_set_external_source_gtid_purged('12345678-abcd-1234-efgh-123456789abc',
 1, 100);
```

mysql.rds_set_master_auto_position(RDS for MySQL 메이저 버전 8.0 이하)

복제 모드를 바이너리 로그 파일 위치 또는 전역 트랜잭션 식별자(GTID)를 기반으로 설정합니다.

구문

```
CALL mysql.rds_set_master_auto_position (
auto_position_mode
);
```

파라미터

auto_position_mode

로그 파일 위치 복제 또는 GTID를 기반으로 하는 복제를 사용할지 여부를 나타내는 값:

- `0` – 바이너리 로그 파일 위치를 기반으로 한 복제 방법을 사용합니다. 기본값은 `0`입니다.
- `1` – GTID 기반 복제 방법을 사용합니다.

사용 노트

마스터 사용자는 `mysql.rds_set_master_auto_position` 프로시저를 실행해야 합니다.

이 프로시저는 모든 RDS for MySQL 5.7 버전 및 RDS for MySQL 8.0.26 이상의 8.0 버전에서 지원됩니다.

mysql.rds_set_source_auto_position(RDS for MySQL 메이저 버전 8.4 이상)

복제 모드를 바이너리 로그 파일 위치 또는 전역 트랜잭션 식별자(GTID)를 기반으로 설정합니다.

구문

```
CALL mysql.rds_set_source_auto_position (auto_position_mode);
```

파라미터

auto_position_mode

로그 파일 위치 복제 또는 GTID를 기반으로 하는 복제를 사용할지 여부를 나타내는 값:

- `0` – 바이너리 로그 파일 위치를 기반으로 한 복제 방법을 사용합니다. 기본값은 `0`입니다.
- `1` – GTID 기반 복제 방법을 사용합니다.

사용 노트

관리 사용자는 `mysql.rds_set_source_auto_position` 프로시저를 실행해야 합니다.

mysql.rds_set_source_delay

소스 데이터베이스 인스턴스에서 현재 읽기 전용 복제본으로의 복제를 지연할 최소 시간(초)을 설정합니다. 읽기 전용 복제본에 연결된 경우에 이 프로시저를 사용하여 소스 데이터베이스 인스턴스에서 복제를 지연합니다.

구문

```
CALL mysql.rds_set_source_delay(
delay
);
```

파라미터

delay

소스 데이터베이스 인스턴스에서 복제를 지연할 최소 시간(초)입니다.

이 파라미터에 대한 제한은 1일(86,400초)입니다.

사용 노트

마스터 사용자는 mysql.rds_set_source_delay 프로시저를 실행해야 합니다.

재해 복구의 경우 이 프로시저를 mysql.rds_start_replication_until 저장 프로시저 또는 mysql.rds_start_replication_until_gtid 저장 프로시저와 함께 사용할 수 있습니다. 지연된 읽기 전용 복제본에 대한 변경 사항을 재해 직전 시간으로 롤포워드하려면 mysql.rds_set_source_delay 프로시저를 실행할 수 있습니다. mysql.rds_start_replication_until 또는 mysql.rds_start_replication_until_gtid 프로시저에서 복제를 중지한 이후에 읽기 전용 복제본을 독립 DB 인스턴스로 승격의 지침에 따라 읽기 전용 복제본을 새 기본 DB 인스턴스가 되도록 승격할 수 있습니다.

mysql.rds_rds_start_replication_until_gtid 프로시저를 사용하려면 GTID를 기반으로 한 복제를 활성화해야 합니다. 재해 원인으로 알려진 특정 GTID 기반 트랜잭션을 건너 뛰려면 mysql.rds_skip_transaction_with_gtid 저장 프로시저를 사용할 수 있습니다. GTID 기반 복제에 대한 자세한 내용은 GTID 기반 복제 사용 단원을 참조하십시오.

mysql.rds_set_source_delay 프로시저는 다음 버전의 RDS for MySQL에서 사용할 수 있습니다.

- 모든 RDS for MySQL 8.4 버전
- MySQL 8.0.26 및 8.0 버전 이상
- 모든 5.6 버전

예시

소스 데이터베이스 인스턴스에서 현재 읽기 전용 복제본으로의 복제를 최소 1시간(3,600초) 동안 지연하려면 다음 파라미터를 사용하여 mysql.rds_set_source_delay를 호출하면 됩니다.

```
CALL mysql.rds_set_source_delay(3600);
```

mysql.rds_skip_repl_error

MySQL DB 읽기 전용 복제본의 복제 오류를 건너뛰고 삭제합니다.

구문

```
CALL mysql.rds_skip_repl_error;
```

사용 노트

마스터 사용자는 읽기 전용 복제본에서 `mysql.rds_skip_repl_error` 프로시저를 실행해야 합니다. 이 프로시저에 대한 자세한 내용은 mysql.rds_skip_repl_error 프로시저 호출를 참조하세요.

오류가 있는지 여부를 판별하려면 MySQL `SHOW REPLICA STATUS\G` 명령을 실행합니다. 중대한 복제 오류가 아닌 경우 `mysql.rds_skip_repl_error`를 사용하여 오류를 건너뛸 수 있습니다. 오류가 여러 개인 경우 `mysql.rds_skip_repl_error`는 첫 번째 오류를 삭제한 후 다른 오류가 있음을 경고합니다. `SHOW REPLICA STATUS\G`를 사용하여 다음 오류에 대한 적합한 조치를 결정할 수 있습니다. 반환된 값에 대한 자세한 내용은 MySQL 설명서의 SHOW SLAVE STATUS 문을 참조하세요.

Amazon RDS의 복제 오류 해결에 대한 자세한 내용은 MySQL 읽기 전용 복제본의 문제 해결 주제를 참조하십시오.

복제 중지 오류

`mysql.rds_skip_repl_error` 프로시저를 호출할 때 복제본이 중단되었거나 사용 중지되었다는 오류 메시지가 표시될 수 있습니다.

읽기 전용 복제본 대신 기본 인스턴스에서 프로시저를 실행하면 이 오류 메시지가 나타납니다. 이 프로시저가 작동하려면 읽기 전용 복제본에서 이 프로시저를 실행해야 합니다.

이 오류 메시지는 읽기 전용 복제본에서 프로시저를 실행하지만 복제를 성공적으로 다시 시작할 수 없는 경우에도 나타날 수 있습니다.

많은 수의 오류를 건너뛰어야 하는 경우, 복제 지연이 바이너리 로그(binlog) 파일의 기본값 보관 기간 이상으로 늘어날 수 있습니다. 이 경우 binlog 파일이 읽기 전용 복제본에서 재생되기 전에 지워지기 때문에 치명적 오류가 발생할 수 있습니다. 이 제거는 복제를 중지시키며, 복제 오류를 건너뛰기 위해 더 이상 `mysql.rds_skip_repl_error` 명령을 호출할 수 없습니다.

이 문제는 소스 데이터베이스 인스턴스에서 binlog 파일이 보관되는 시간을 늘려서 완화할 수 있습니다. binlog 보관 시간을 늘린 후에 복제를 재시작하고 필요에 따라 `mysql.rds_skip_repl_error` 명령을 호출할 수 있습니다.

binlog 보관 기간을 설정하려면 mysql.rds_set_configuration 프로시저를 사용하여 `'binlog retention hours'` 구성 파라미터와 DB 클러스터에 binlog 파일을 보관할 시간을 함께 지정하십시오. 다음 예제에서는 binlog 파일의 보관 기간을 48시간으로 설정합니다.

```
CALL mysql.rds_set_configuration('binlog retention hours', 48);
```

mysql.rds_start_replication

RDS for MySQL DB 인스턴스에서 복제를 시작합니다.

> **Note**
>
> mysql.rds_start_replication_until 또는 mysql.rds_start_replication_until_gtid 저장 프로시저를 사용하여 RDS for MySQL DB 인스턴스에서 복제를 시작하고 지정된 바이너리 로그 파일 위치에서 복제를 중지할 수 있습니다.

구문

```
CALL mysql.rds_start_replication;
```

사용 노트

마스터 사용자는 `mysql.rds_start_replication` 프로시저를 실행해야 합니다.

Amazon RDS 외부에서 실행 중인 MySQL 인스턴스에서 데이터를 가져오려면 mysql.rds_set_external_master(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하) 또는 mysql.rds_set_external_source(RDS for MySQL 메이저 버전 8.4 이상)를 직접 호출하여 복제 구성을 빌드한 다음, 읽기 전용 복제본에서 `mysql.rds_start_replication`을 직접 호출하여 복제 프로세스를 시작합니다. 자세한 내용은 MySQL DB 인스턴스로 백업 복원 단원을 참조하십시오.

Amazon RDS 외부의 MySQL 인스턴스로 데이터를 내보내려면 읽기 전용 복제본에서 `mysql.rds_start_replication` 및 `mysql.rds_stop_replication`을 호출하여 바이너리 로그 삭제 등의 몇 가지 복제 작업을 제어합니다. 자세한 내용은 복제를 사용하여 MySQL DB 인스턴스에서 데이터 내보내기 단원을 참조하십시오.

읽기 전용 복제본에서 `mysql.rds_start_replication`을 호출하여 이전에 `mysql.rds_stop_replication`을 호출하여 중지한 복제 프로세스를 재시작할 수도 있습니다. 자세한 내용은 DB 인스턴스 읽기 전용 복제본 작업 단원을 참조하십시오.

mysql.rds_start_replication_until

RDS for MySQL DB 인스턴스에서 복제를 시작하고 지정된 바이너리 로그 파일 위치에서 복제를 중지합니다.

구문

```
CALL mysql.rds_start_replication_until (
replication_log_file
  , replication_stop_point
);
```

파라미터

replication_log_file

복제 정보를 포함하는 소스 데이터베이스 인스턴스의 이진 로그 이름입니다.

replication_stop_point

복제를 중지할 `replication_log_file` 바이너리 로그 내 위치입니다.

사용 노트

마스터 사용자는 `mysql.rds_start_replication_until` 프로시저를 실행해야 합니다.

`mysql.rds_start_replication_until` 프로시저는 다음 버전의 RDS for MySQL에서 사용할 수 있습니다.

- 모든 RDS for MySQL 8.4 버전
- MySQL 8.0.26 및 8.0 버전 이상
- 모든 5.6 버전

이 프로시저는 지연 복제에서 재해 복구를 위해 사용할 있습니다. 지연된 복제를 구성한 경우 이 프로시저를 사용하여 지연된 읽기 전용 복제본에 대한 변경 사항을 재해 직전 시간으로 롤포워드할 수 있습니다. 이 프로시저에서 복제를 중지한 이후에 읽기 전용 복제본을 독립 DB 인스턴스로 승격의 지침에 따라 읽기 전용 복제본을 새 기본 DB 인스턴스가 되도록 승격할 수 있습니다.

다음 저장 프로시저를 사용하여 지연 복제를 구성할 수 있습니다.

- mysql.rds_set_configuration
- mysql.rds_set_external_master_with_delay(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하)
- mysql.rds_set_external_source_with_delay(RDS for MySQL 메이저 버전 8.4 이상)

- mysql.rds_set_source_delay

replication_log_file 파라미터에 지정된 파일 이름은 소스 데이터베이스 인스턴스 binlog 파일 이름과 일치해야 합니다.

replication_stop_point 파라미터가 과거에 해당하는 중지 위치를 지정하는 경우 복제가 즉시 중지됩니다.

예시

다음 예에서는 복제를 시작하고 120 바이너리 로그 파일의 mysql-bin-changelog.000777 위치에 도달할 때까지 변경 사항을 복제합니다.

```
call mysql.rds_start_replication_until(
  'mysql-bin-changelog.000777',
  120);
```

mysql.rds_stop_replication

MySQL DB 인스턴스에서 복제를 중지합니다.

구문

```
CALL mysql.rds_stop_replication;
```

사용 노트

마스터 사용자는 mysql.rds_stop_replication 프로시저를 실행해야 합니다.

Amazon RDS 외부에서 실행 중인 MySQL 인스턴스에서 데이터를 가져오도록 복제를 구성하는 경우 가져오기가 완료된 후 읽기 전용 복제본에서 mysql.rds_stop_replication을 호출하여 복제 프로세스를 중지합니다. 자세한 내용은 MySQL DB 인스턴스로 백업 복원 단원을 참조하십시오.

Amazon RDS 외부의 MySQL 인스턴스로 데이터를 내보내도록 복제를 구성할 경우 읽기 전용 복제본에서 mysql.rds_start_replication 및 mysql.rds_stop_replication을 호출하여 이진 로그 삭제 등의 몇 가지 복제 작업을 제어합니다. 자세한 내용은 복제를 사용하여 MySQL DB 인스턴스에서 데이터 내보내기 단원을 참조하십시오.

mysql.rds_stop_replication을 사용하여 두 Amazon RDS DB 인스턴스 간 복제를 중지할 수도 있습니다. 일반적으로 읽기 전용 복제본에서 대규모 인덱스 생성과 같이 읽기 전용 복제본에서 장기 실

행 작업을 수행하려는 경우 복제를 중지합니다. 읽기 전용 복제본에서 mysql.rds_start_replication을 호출하여 중지한 복제 프로세스를 재시작할 수 있습니다. 자세한 내용은 DB 인스턴스 읽기 전용 복제본 작업 단원을 참조하십시오.

세션 또는 쿼리 종료

다음 저장 프로시저는 세션 또는 쿼리를 종료합니다.

주제

- mysql.rds_kill
- mysql.rds_kill_query

mysql.rds_kill

MySQL Server와의 연결을 종료합니다.

구문

```
CALL mysql.rds_kill(processID);
```

파라미터

processID

종료할 연결 스레드의 ID입니다.

사용 노트

MySQL Server에 대한 각 연결은 별개의 스레드로 실행됩니다. 연결을 종료하려면 `mysql.rds_kill` 프로시저를 사용하여 해당 연결의 스레드 ID를 전달합니다. 스레드 ID를 확인하려면 MySQL SHOW PROCESSLIST 명령을 사용합니다.

제한 사항에 대한 자세한 내용은 MySQL 저장 프로시저 제한 사항 섹션을 참조하세요.

예시

다음 예제는 스레드 ID가 4243인 연결을 종료합니다.

```
CALL mysql.rds_kill(4243);
```

mysql.rds_kill_query

MySQL Server에서 실행 중인 쿼리를 종료합니다.

구문

```
CALL mysql.rds_kill_query(processID);
```

파라미터

processID

종료할 쿼리를 실행 중인 프로세스 또는 스레드의 ID입니다.

사용 노트

MySQL Server에서 실행 중인 쿼리를 중지하려면 `mysql_rds_kill_query` 프로시저를 사용하여 해당 쿼리를 실행 중인 스레드의 연결 ID를 전달합니다. 그러면 프로시저가 연결을 종료합니다.

ID를 확인하려면 MySQL INFORMATION_SCHEMA PROCESSLIST 테이블을 쿼리하거나 MySQL SHOW PROCESSLIST 명령을 사용합니다. `SHOW PROCESSLIST` 또는 `SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST`에서 가져온 ID 열의 값은 *processID*입니다.

제한 사항에 대한 자세한 내용은 MySQL 저장 프로시저 제한 사항 섹션을 참조하세요.

예시

다음 예시는 쿼리 스레드 ID가 230040인 쿼리를 중지합니다.

```
CALL mysql.rds_kill_query(230040);
```

활성-활성 클러스터

다음 저장 프로시저는 RDS for MySQL의 활성-활성 클러스터를 설정하고 관리합니다. 자세한 내용은 the section called “액티브-액티브 클러스터 구성” 단원을 참조하십시오.

이러한 저장 프로시저는 다음 버전을 실행하는 RDS for MySQL DB 인스턴스에서만 사용할 수 있습니다.

- 모든 MySQL 8.4 버전
- MySQL 8.0.35 이상 마이너 버전

주제

- mysql.rds_group_replication_advance_gtid
- mysql.rds_group_replication_create_user
- mysql.rds_group_replication_set_recovery_channel
- mysql.rds_group_replication_start
- mysql.rds_group_replication_stop

mysql.rds_group_replication_advance_gtid

현재 DB 인스턴스에 자리 표시자 GTID를 생성합니다.

구문

```
CALL mysql.rds_group_replication_advance_gtid(
begin_id
, end_id
, server_uuid
);
```

파라미터

begin_id

생성할 시작 트랜잭션 ID입니다.

end_id

생성할 종료 트랜잭션 ID입니다.

begin_id

생성할 트랜잭션을 위한 group_replication_group_name입니다. group_replication_group_name은 DB 인스턴스와 연결된 DB 파라미터 그룹에 UUID로 지정됩니다.

사용 노트

활성-활성 클러스터에서 DB 인스턴스가 그룹에 가입하려면 새 DB 인스턴스에서 실행되는 모든 GTID 트랜잭션이 클러스터의 다른 구성원에 존재해야 합니다. 드문 경우지만, 인스턴스를 그룹에 조인하기 전에 트랜잭션을 실행하면 새 DB 인스턴스에서 더 많은 트랜잭션이 발생할 수 있습니다. 이 경우 기존 트랜잭션을 제거할 수는 없지만 이 프로시저를 사용하여 그룹의 다른 DB 인스턴스에 해당하는 자리 표시자 GTID를 만들 수 있습니다. 그러기 전에 트랜잭션이 복제된 데이터에 영향을 주지 않는지 확인하세요.

이 프로시저를 직접 호출하면 server_uuid:begin_id-end_id의 GTID 트랜잭션이 빈 콘텐츠와 함께 생성됩니다. 복제 문제를 방지하려면 다른 조건에서는 이 프로시저를 사용하지 마세요.

> **Important**
>
> 활성-활성 클러스터가 정상적으로 작동할 때는 이 프로시저를 직접 호출하지 마세요. 생성 중인 트랜잭션으로 인해 발생할 수 있는 결과를 이해하지 못하면 이 프로시저를 직접 호출하지 마세요. 이 프로시저를 직접 호출하면 데이터가 일치하지 않을 수 있습니다.

예제

다음 예제는 현재 DB 인스턴스에 자리 표시자 GTID를 만듭니다.

```
CALL mysql.rds_group_replication_advance_gtid(5, 6,
 '11111111-2222-3333-4444-555555555555');
```

mysql.rds_group_replication_create_user

DB 인스턴스의 그룹 복제를 위한 복제 사용자 rdsgrprepladmin를 생성합니다.

구문

```
CALL mysql.rds_group_replication_create_user(
```

```
replication_user_password
);
```

파라미터

replication_user_password

복제 사용자 rdsgrprepladmin의 암호입니다.

사용 노트

- 활성-활성 클러스터의 모든 DB 인스턴스에서 복제 사용자 rdsgrprepladmin의 암호가 동일해야 합니다.
- rdsgrprepladmin 사용자 이름은 그룹 복제 연결용으로 예약되어 있습니다. 마스터 사용자를 포함한 다른 사용자는 이 사용자 이름을 가질 수 없습니다.

예제

다음 예제는 DB 인스턴스에서 그룹 복제를 위한 복제 사용자 rdsgrprepladmin를 생성합니다.

```
CALL mysql.rds_group_replication_create_user('password');
```

mysql.rds_group_replication_set_recovery_channel

활성-활성 클러스터의 group_replication_recovery 채널을 설정합니다. 이 프로시저에서는 예약된 rdsgrprepladmin 사용자를 사용하여 채널을 구성합니다.

구문

```
CALL mysql.rds_group_replication_set_recovery_channel(
replication_user_password);
```

파라미터

replication_user_password

복제 사용자 rdsgrprepladmin의 암호입니다.

사용 노트

활성-활성 클러스터의 모든 DB 인스턴스에서 복제 사용자 rdsgrprepladmin의 암호가 동일해야 합니다. mysql.rds_group_replication_create_user를 직접 호출하면 암호가 지정됩니다.

예제

다음 예에서는 활성-활성 클러스터의 group_replication_recovery 채널을 설정합니다.

```
CALL mysql.rds_group_replication_set_recovery_channel('password');
```

mysql.rds_group_replication_start

현재 DB 인스턴스에서 그룹 복제를 시작합니다.

구문

```
CALL mysql.rds_group_replication_start(
bootstrap
);
```

파라미터

#####

새 그룹을 초기화할지 기존 그룹에 가입할지를 지정하는 값입니다. 1은 현재 DB 인스턴스로 새 그룹을 초기화합니다. 0은 DB 인스턴스와 연결된 DB 파라미터 그룹의 group_replication_group_seeds 파라미터에 정의된 엔드포인트에 연결하여 현재 DB 인스턴스를 기존 그룹에 조인합니다.

예제

다음 예제는 현재 DB 인스턴스로 새 그룹을 초기화합니다.

```
CALL mysql.rds_group_replication_start(1);
```

mysql.rds_group_replication_stop

현재 DB 인스턴스에서 그룹 복제를 중지합니다.

구문

```
CALL mysql.rds_group_replication_stop();
```

사용 노트

DB 인스턴스에서 복제를 중지해도 활성-활성 클러스터의 다른 DB 인스턴스에는 영향을 주지 않습니다.

다중 소스 복제 관리

다음 저장 프로시저는 RDS for MySQL의 다중 소스 복제본에서 복제 채널을 설정하고 관리합니다. 자세한 내용은 the section called “다중 소스 복제 구성” 단원을 참조하십시오.

이러한 저장 프로시저는 다음 엔진 버전을 실행하는 RDS for MySQL DB 인스턴스에서만 사용할 수 있습니다.

- 모든 8.4 버전
- 8.0.35 이상 마이너 버전
- 5.7.44 이상 마이너 버전

`caching_sha2_passwword`로 구성된 복제 사용자로 복제를 관리하기 위해 저장 프로시저를 사용하는 경우, `SOURCE_SSL=1`을 지정하여 TLS를 구성해야 합니다. `caching_sha2_password`는 RDS for MySQL 8.4의 기본 인증 플러그인입니다.

> **Note**
>
> 이 설명서에서는 소스 DB 인스턴스를 RDS for MySQL DB 인스턴스라고 하지만, 이러한 프로시저는 Amazon RDS 외부에서 실행되는 MySQL 인스턴스에도 적용됩니다.

주제

- mysql.rds_next_source_log_for_channel
- mysql.rds_reset_external_source_for_channel
- mysql.rds_set_external_source_for_channel
- mysql.rds_set_external_source_with_auto_position_for_channel
- mysql.rds_set_external_source_with_delay_for_channel
- mysql.rds_set_source_auto_position_for_channel
- mysql.rds_set_source_delay_for_channel
- mysql.rds_skip_repl_error_for_channel
- mysql.rds_start_replication_for_channel
- mysql.rds_start_replication_until_for_channel
- mysql.rds_start_replication_until_gtid_for_channel
- mysql.rds_stop_replication_for_channel

mysql.rds_next_source_log_for_channel

소스 DB 인스턴스 로그 위치를 채널에 대한 소스 DB 인스턴스에서 다음 이진 로그의 시작으로 변경합니다. 다중 소스 복제본에 대한 복제 I/O 오류 1236을 수신 중일 경우에만 이 프로시저를 사용하세요.

구문

```
CALL mysql.rds_next_source_log_for_channel(
curr_master_log,
channel_name
);
```

파라미터

curr_master_log

현재 소스 로그 파일의 인덱스입니다. 예를 들어, 현재 파일의 이름이 `mysql-bin-changelog.012345`이면 인덱스는 12345입니다. 현재 소스 로그 파일 이름을 확인하려면 `SHOW REPLICA STATUS FOR CHANNEL 'channel_name'` 명령을 실행하고 `Source_Log_File` 필드를 봅니다.

channel_name

다중 소스 복제본의 복제 채널 이름입니다. 각 복제 채널은 특정 호스트 및 포트에서 실행되는 단일 소스 RDS for MySQL DB 인스턴스로부터 바이너리 로그 이벤트를 수신합니다.

사용 노트

마스터 사용자는 `mysql.rds_next_source_log_for_channel` 프로시저를 실행해야 합니다. 예를 들어 IO_Thread 오류가 있는 경우 이 프로시저를 사용하여 현재 바이너리 로그 파일의 모든 이벤트를 건너뛰고 `channel_name`에서 지정된 채널의 다음 바이너리 로그 파일에서 복제를 재개할 수 있습니다.

예제

다중 소스 복제본의 채널에서 복제가 실패했다고 가정해 보겠습니다. 다중 소스 복제본에서 `SHOW REPLICA STATUS FOR CHANNEL 'channel_1'\G`를 실행하면 다음 결과가 반환됩니다.

```
mysql> SHOW REPLICA STATUS FOR CHANNEL 'channel_1'\G
```

```
*************************** 1. row ***************************
             Replica_IO_State: Waiting for source to send event
                  Source_Host: myhost.XXXXXXXXXXXXXX.rr-rrrr-1.rds.amazonaws.com
                  Source_User: ReplicationUser
                  Source_Port: 3306
                Connect_Retry: 60
              Source_Log_File: mysql-bin-changelog.012345
          Read_Source_Log_Pos: 1219393
               Relay_Log_File: replica-relay-bin.000003
                Relay_Log_Pos: 30223388
        Relay_Source_Log_File: mysql-bin-changelog.012345
           Replica_IO_Running: No
          Replica_SQL_Running: Yes
              Replicate_Do_DB:.
              .
              .
                Last_IO_Errno: 1236
                Last_IO_Error: Got fatal error 1236 from master when reading data from
 binary log: 'Client requested master to start replication from impossible position;
 the first event 'mysql-bin-changelog.013406' at 1219393, the last event read from
 '/rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4, the last byte read from '/
rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4.'
               Last_SQL_Errno: 0
               Last_SQL_Error:
               .
               .
                 Channel_name: channel_1
              .
              .
 -- Some fields are omitted in this example output
```

Last_IO_Errno 필드가 인스턴스에서 I/O 오류 1236을 수신하고 있음을 보여 줍니다. Source_Log_File 필드는 파일 이름이 mysql-bin-changelog.012345임을 보여 줍니다. 이는 로그 파일 인덱스가 12345임을 의미합니다. 오류를 해결하려면 다음 파라미터와 함께 mysql.rds_next_source_log_for_channel을 직접 호출합니다.

```
CALL mysql.rds_next_source_log_for_channel(12345,'channel_1');
```

mysql.rds_reset_external_source_for_channel

지정된 채널에서 복제 프로세스를 중지하고 다중 소스 복제본에서 채널 및 관련 구성을 제거합니다.

> **Important**
>
> 이 프로시저를 실행하려면 autocommit를 활성화해야 합니다. 활성화하려면 autocommit 파라미터를 1로 설정합니다. 파라미터 수정에 대한 자세한 정보는 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 단원을 참조하십시오.

구문

```
CALL mysql.rds_reset_external_source_for_channel (channel_name);
```

파라미터

channel_name

다중 소스 복제본의 복제 채널 이름입니다. 각 복제 채널은 특정 호스트 및 포트에서 실행되는 단일 소스 RDS for MySQL DB 인스턴스로부터 바이너리 로그 이벤트를 수신합니다.

사용 노트

마스터 사용자는 mysql.rds_reset_external_source_for_channel 프로시저를 실행해야 합니다. 이 프로시저를 수행하면 제거되는 채널에 속하는 모든 릴레이 로그가 삭제됩니다.

mysql.rds_set_external_source_for_channel

RDS for MySQL DB 인스턴스의 복제 채널을 구성하여 다른 RDS for MySQL DB 인스턴스의 데이터를 복제합니다.

> **Important**
>
> 이 프로시저를 실행하려면 autocommit를 활성화해야 합니다. 활성화하려면 autocommit 파라미터를 1로 설정합니다. 파라미터 수정에 대한 자세한 정보는 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 단원을 참조하십시오.

> **Note**
>
> 대신 the section called “mysql.rds_set_external_source_with_delay_for_channel” 저장 프로시저를 사용하여 이 채널에 지연 복제를 구성할 수 있습니다.

구문

```
CALL mysql.rds_set_external_source_for_channel (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
  , channel_name
);
```

파라미터

host_name

RDS for MySQL 소스 DB 인스턴스의 호스트 이름 또는 IP 주소입니다.

host_port

RDS for MySQL 소스 DB 인스턴스에서 사용한 포트입니다. 네트워크 구성에 포트 번호를 변환하는 SSH 포트 복제가 포함되는 경우 SSH(Secure Shell)에 의해 공개되는 포트 이름을 지정하세요.

replication_user_name

RDS for MySQL 소스 DB 인스턴스에서 `REPLICATION CLIENT` 및 `REPLICATION SLAVE` 권한을 보유한 사용자의 ID입니다. 소스 인스턴스를 사용한 복제에만 사용되는 계정을 제공하는 것이 좋습니다.

replication_user_password

`replication_user_name`에 지정된 사용자 ID의 암호입니다.

mysql_binary_log_file_name

복제 정보를 포함하는 소스 DB 인스턴스의 이진 로그 이름입니다.

mysql_binary_log_file_location

복제 시 복제 정보를 읽기 시작하는 mysql_binary_log_file_name 이진수 로그 내 위치입니다.

소스 DB 인스턴스의 SHOW BINARY LOG STATUS를 실행하여 binlog 파일 이름 및 위치를 확인할 수 있습니다.

> **Note**
>
> 이전 버전의 MySQL에는 SHOW MASTER STATUS 대신 SHOW BINARY LOG STATUS가 사용되었습니다. 8.4 이전 MySQL 버전을 사용하는 경우 SHOW MASTER STATUS를 사용합니다.

ssl_encryption

복제 연결에 보안 소켓 계층(SSL) 암호화를 사용할지 여부를 지정하는 값입니다. 1은 SSL 암호화 사용, 0은 암호화 사용 안 함입니다. 기본값은 0입니다.

> **Note**
>
> SOURCE_SSL_VERIFY_SERVER_CERT 옵션은 지원되지 않습니다. 이 옵션은 0으로 설정되어 있는데, 이는 연결이 암호화되었지만 인증서는 확인되지 않았음을 의미합니다.

channel_name

복제 채널의 이름입니다. 각 복제 채널은 특정 호스트 및 포트에서 실행되는 단일 소스 RDS for MySQL DB 인스턴스로부터 바이너리 로그 이벤트를 수신합니다.

사용 노트

마스터 사용자는 mysql.rds_set_external_source_for_channel 프로시저를 실행해야 합니다. 이 프로시저는 복제 채널을 생성할 대상 RDS for MySQL DB 인스턴스에서 실행해야 합니다.

mysql.rds_set_external_source_for_channel을 실행하기 전에 다중 소스 복제본에 필요한 권한을 가진 소스 DB 인스턴스의 복제 사용자를 구성하세요. 다중 소스 복제본을 소스 DB 인스턴스에 연결하려면 소스 DB 인스턴스에 대해 REPLICATION CLIENT 및 REPLICATION SLAVE 권한을 가진

복제 사용자의 replication_user_name 값과 replication_user_password 값을 지정해야 합니다.

소스 DB 인스턴스에서 복제 사용자를 구성하려면

1. 선택한 MySQL 클라이언트를 사용하여 소스 DB 인스턴스에 연결하고 복제에 사용될 사용자 계정을 생성합니다. 다음은 예입니다.

   > **Important**
   >
   > 보안 모범 사례로 다음 예제에 표시된 자리 표시자 값 이외의 암호를 지정하는 것이 좋습니다.

   ```
   CREATE USER 'repl_user'@'example.com' IDENTIFIED BY 'password';
   ```

2. 소스 DB 인스턴스의 경우 복제 사용자에게 REPLICATION CLIENT 및 REPLICATION SLAVE 권한을 부여합니다. 다음 예제에서는 도메인의 'repl_user' 사용자에게 모든 데이터베이스에 대한 REPLICATION CLIENT 및 REPLICATION SLAVE 권한을 부여합니다.

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'example.com';
   ```

암호화된 복제를 사용하려면 SSL 연결을 사용하도록 소스 DB 인스턴스를 구성합니다.

mysql.rds_set_external_source_for_channel을 직접 호출하여 이 복제 채널을 구성한 후 mysql.rds_start_replication_for_channel을 직접 호출하여 복제 프로세스를 시작할 수 있습니다. the section called “mysql.rds_reset_external_source_for_channel”을 직접 호출하여 채널에서 복제를 중지하고 복제본에서 채널 구성을 제거할 수 있습니다.

mysql.rds_set_external_source_for_channel을 직접 호출하면 Amazon RDS는 채널별 세부 정보 없이 mysql.rds_history 테이블에 시간, 사용자 및 set channel source의 작업을 기록하고 mysql.rds_replication_status 테이블에는 채널 이름과 함께 이들 정보를 기록합니다. 이 정보는 내부 사용 및 모니터링 목적으로만 기록됩니다. 감사 목적으로 전체 프로시저 직접 호출을 기록하려면 애플리케이션의 특정 요구 사항에 따라 감사 로그 또는 일반 로그를 활성화하는 것이 좋습니다.

예시

RDS for MySQL DB 인스턴스에서 실행할 때 다음 예제는 이 DB 인스턴스에 channel_1 이름이 지정된 복제 채널을 구성하여 sourcedb.example.com 호스트와 3306 포트로 지정된 소스의 데이터를 복제합니다.

```
call mysql.rds_set_external_source_for_channel(
  'sourcedb.example.com',
  3306,
  'repl_user',
  'password',
  'mysql-bin-changelog.0777',
  120,
  0,
  'channel_1');
```

mysql.rds_set_external_source_with_auto_position_for_channel

RDS for MySQL DB 인스턴스에서 복제 채널을 구성하고 복제 지연 옵션을 설정합니다. 전역 트랜잭션 식별자(GTID)를 기반으로 한 복제입니다.

> **Important**
>
> 이 프로시저를 실행하려면 autocommit를 활성화해야 합니다. 활성화하려면 autocommit 파라미터를 1로 설정합니다. 파라미터 수정에 대한 자세한 정보는 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 단원을 참조하십시오.

구문

```
CALL mysql.rds_set_external_source_with_auto_position_for_channel (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , ssl_encryption
  , delay
  , channel_name
);
```

파라미터

host_name

RDS for MySQL 소스 DB 인스턴스의 호스트 이름 또는 IP 주소입니다.

host_port

RDS for MySQL 소스 DB 인스턴스에서 사용한 포트입니다. 네트워크 구성에 포트 번호를 변환하는 SSH 포트 복제가 포함되는 경우 SSH(Secure Shell)에 의해 공개되는 포트 이름을 지정하세요.

replication_user_name

RDS for MySQL 소스 DB 인스턴스에서 `REPLICATION CLIENT` 및 `REPLICATION SLAVE` 권한을 보유한 사용자의 ID입니다. 소스 인스턴스를 사용한 복제에만 사용되는 계정을 제공하는 것이 좋습니다.

replication_user_password

`replication_user_name`에 지정된 사용자 ID의 암호입니다.

ssl_encryption

복제 연결에 보안 소켓 계층(SSL) 암호화를 사용할지 여부를 지정하는 값입니다. 1은 SSL 암호화 사용, 0은 암호화 사용 안 함입니다. 기본값은 0입니다.

> **Note**
>
> `SOURCE_SSL_VERIFY_SERVER_CERT` 옵션은 지원되지 않습니다. 이 옵션은 0으로 설정되어 있는데, 이는 연결이 암호화되었지만 인증서는 확인되지 않았음을 의미합니다.

delay

소스 DB 인스턴스에서 복제를 지연할 최소 시간(초)입니다.

이 파라미터에 대한 제한은 1일(86,400초)입니다.

channel_name

복제 채널의 이름입니다. 각 복제 채널은 특정 호스트 및 포트에서 실행되는 단일 소스 RDS for MySQL DB 인스턴스로부터 바이너리 로그 이벤트를 수신합니다.

사용 노트

마스터 사용자는 mysql.rds_set_external_source_with_auto_position_for_channel 프로시저를 실행해야 합니다. 이 프로시저는 복제 채널을 생성할 대상 RDS for MySQL DB 인스턴스에서 실행해야 합니다.

rds_set_external_source_with_auto_position_for_channel을 실행하기 전에 다중 소스 복제본에 필요한 권한을 가진 소스 DB 인스턴스의 복제 사용자를 구성하세요. 다중 소스 복제본을 소스 DB 인스턴스에 연결하려면 소스 DB 인스턴스에 대해 REPLICATION CLIENT 및 REPLICATION SLAVE 권한을 가진 복제 사용자의 replication_user_name 값과 replication_user_password 값을 지정해야 합니다.

소스 DB 인스턴스에서 복제 사용자를 구성하려면

1. 선택한 MySQL 클라이언트를 사용하여 소스 DB 인스턴스에 연결하고 복제에 사용될 사용자 계정을 생성합니다. 다음은 예입니다.

   > **Important**
   >
   > 보안 모범 사례로 다음 예제에 표시된 자리 표시자 값 이외의 암호를 지정하는 것이 좋습니다.

   ```
   CREATE USER 'repl_user'@'example.com' IDENTIFIED BY 'password';
   ```

2. 소스 DB 인스턴스의 경우 복제 사용자에게 REPLICATION CLIENT 및 REPLICATION SLAVE 권한을 부여합니다. 다음 예제에서는 도메인의 'repl_user' 사용자에게 모든 데이터베이스에 대한 REPLICATION CLIENT 및 REPLICATION SLAVE 권한을 부여합니다.

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'example.com';
   ```

암호화된 복제를 사용하려면 SSL 연결을 사용하도록 소스 DB 인스턴스를 구성합니다.

mysql.rds_set_external_source_with_auto_position_for_channel 직접 호출 전에 the section called “mysql.rds_set_external_source_gtid_purged”를 직접 호출하여 외부 소스의 지정된 GTID 범위로 시스템 변수를 설정합니다.

mysql.rds_set_external_source_with_auto_position_for_channel을 직접 호출하여 Amazon RDS DB 인스턴스를 지정된 채널의 읽기 복제본으로 구성한 후, 읽기 복제본에서 the section

called “mysql.rds_start_replication_for_channel”을 직접 호출하여 해당 채널에서 복제 프로세스를 시작할 수 있습니다.

mysql.rds_set_external_source_with_auto_position_for_channel을 직접 호출하여 이 복제 채널을 구성한 후 mysql.rds_start_replication_for_channel을 직접 호출하여 복제 프로세스를 시작할 수 있습니다. the section called “mysql.rds_reset_external_source_for_channel”을 직접 호출하여 채널에서 복제를 중지하고 복제본에서 채널 구성을 제거할 수 있습니다.

예시

MySQL용 RDS DB 인스턴스에서 실행할 때 다음 예제는 이 DB 인스턴스에 channel_1 이름이 지정된 복제 채널을 구성하여 sourcedb.example.com 호스트 및 3306 포트로 지정된 소스에서 데이터를 복제하도록 구성하고 최소 복제 지연 시간을 1시간(3,600초)으로 설정합니다. 즉 소스 RDS for MySQL DB 인스턴스의 변경 사항은 최소 1시간 동안 다중 소스 복제본에 적용되지 않습니다.

```
call mysql.rds_set_external_source_with_auto_position_for_channel(
  'sourcedb.example.com',
  3306,
  'repl_user',
  'password',
  1,
  3600,
  'channel_1');
```

mysql.rds_set_external_source_with_delay_for_channel

RDS for MySQL DB 인스턴스의 복제 채널을 지정된 복제 지연으로 구성합니다.

> **Important**
>
> 이 프로시저를 실행하려면 autocommit를 활성화해야 합니다. 활성화하려면 autocommit 파라미터를 1로 설정합니다. 파라미터 수정에 대한 자세한 정보는 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 단원을 참조하십시오.

구문

```
CALL mysql.rds_set_external_source_with_delay_for_channel (
```

```
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
  , delay
  , channel_name
);
```

파라미터

host_name

RDS for MySQL 소스 DB 인스턴스의 호스트 이름 또는 IP 주소입니다.

host_port

RDS for MySQL 소스 DB 인스턴스에서 사용한 포트입니다. 네트워크 구성에 포트 번호를 변환하는 SSH 포트 복제가 포함되는 경우 SSH(Secure Shell)에 의해 공개되는 포트 이름을 지정하세요.

replication_user_name

RDS for MySQL 소스 DB 인스턴스에서 `REPLICATION CLIENT` 및 `REPLICATION SLAVE` 권한을 보유한 사용자의 ID입니다. 소스 인스턴스를 사용한 복제에만 사용되는 계정을 제공하는 것이 좋습니다.

replication_user_password

`replication_user_name`에 지정된 사용자 ID의 암호입니다.

mysql_binary_log_file_name

복제 정보를 포함하는 소스 DB 인스턴스의 이진 로그 이름입니다.

mysql_binary_log_file_location

복제 시 복제 정보를 읽기 시작할 `mysql_binary_log_file_name` 이진 로그 내 위치입니다.

소스 데이터베이스 인스턴스의 `SHOW BINARY LOG STATUS`를 실행하여 binlog 파일 이름 및 위치를 확인할 수 있습니다.

> **Note**
>
> 이전 버전의 MySQL에는 `SHOW MASTER STATUS` 대신 `SHOW BINARY LOG STATUS`가 사용되었습니다. 8.4 이전 MySQL 버전을 사용하는 경우 `SHOW MASTER STATUS`를 사용합니다.

ssl_encryption

복제 연결에 보안 소켓 계층(SSL) 암호화를 사용할지 여부를 지정하는 값입니다. 1은 SSL 암호화 사용, 0은 암호화 사용 안 함입니다. 기본값은 0입니다.

> **Note**
>
> `SOURCE_SSL_VERIFY_SERVER_CERT` 옵션은 지원되지 않습니다. 이 옵션은 0으로 설정되어 있는데, 이는 연결이 암호화되었지만 인증서는 확인되지 않았음을 의미합니다.

delay

소스 DB 인스턴스에서 복제를 지연할 최소 시간(초)입니다.

이 파라미터에 대한 제한은 1일(86,400초)입니다.

channel_name

복제 채널의 이름입니다. 각 복제 채널은 특정 호스트 및 포트에서 실행되는 단일 소스 RDS for MySQL DB 인스턴스로부터 바이너리 로그 이벤트를 수신합니다.

사용 노트

마스터 사용자는 `mysql.rds_set_external_source_with_delay_for_channel` 프로시저를 실행해야 합니다. 이 프로시저는 복제 채널을 생성할 대상 RDS for MySQL DB 인스턴스에서 실행해야 합니다.

`mysql.rds_set_external_source_with_delay_for_channel`을 실행하기 전에 다중 소스 복제본에 필요한 권한을 가진 소스 DB 인스턴스의 복제 사용자를 구성하세요. 다중 소스 복제본을 소스 DB 인스턴스에 연결하려면 소스 DB 인스턴스에 대해 `REPLICATION CLIENT` 및 `REPLICATION SLAVE` 권한을 가진 복제 사용자의 `replication_user_name` 값과 `replication_user_password` 값을 지정해야 합니다.

소스 DB 인스턴스에서 복제 사용자를 구성하려면

1. 선택한 MySQL 클라이언트를 사용하여 소스 DB 인스턴스에 연결하고 복제에 사용될 사용자 계정을 생성합니다. 다음은 예입니다.

   > **Important**
   >
   > 보안 모범 사례로 다음 예제에 표시된 자리 표시자 값 이외의 암호를 지정하는 것이 좋습니다.

   ```
   CREATE USER 'repl_user'@'example.com' IDENTIFIED BY 'password';
   ```

2. 소스 DB 인스턴스의 경우 복제 사용자에게 REPLICATION CLIENT 및 REPLICATION SLAVE 권한을 부여합니다. 다음 예제에서는 도메인의 'repl_user' 사용자에게 모든 데이터베이스에 대한 REPLICATION CLIENT 및 REPLICATION SLAVE 권한을 부여합니다.

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'example.com';
   ```

암호화된 복제를 사용하려면 SSL 연결을 사용하도록 소스 DB 인스턴스를 구성합니다.

mysql.rds_set_external_source_with_delay_for_channel을 직접 호출하여 이 복제 채널을 구성한 후 mysql.rds_start_replication_for_channel을 직접 호출하여 복제 프로세스를 시작할 수 있습니다. the section called “mysql.rds_reset_external_source_for_channel”을 직접 호출하여 채널에서 복제를 중지하고 복제본에서 채널 구성을 제거할 수 있습니다.

mysql.rds_set_external_source_with_delay_for_channel을 직접 호출하면 Amazon RDS는 채널별 세부 정보 없이 mysql.rds_history 테이블에 시간, 사용자 및 set channel source의 작업을 기록하고 mysql.rds_replication_status 테이블에는 채널 이름과 함께 이들 정보를 기록합니다. 이 정보는 내부 사용 및 모니터링 목적으로만 기록됩니다. 감사 목적으로 전체 프로시저 직접 호출을 기록하려면 애플리케이션의 특정 요구 사항에 따라 감사 로그 또는 일반 로그를 활성화하는 것이 좋습니다.

예시

MySQL용 RDS DB 인스턴스에서 실행할 때 다음 예제는 이 DB 인스턴스에 channel_1 이름이 지정된 복제 채널을 구성하여 sourcedb.example.com 호스트 및 3306 포트로 지정된 소스에서 데이

터를 복제하도록 구성하고 최소 복제 지연 시간을 1시간(3,600초)으로 설정합니다. 즉 소스 RDS for MySQL DB 인스턴스의 변경 사항은 최소 1시간 동안 다중 소스 복제본에 적용되지 않습니다.

```
call mysql.rds_set_external_source_with_delay_for_channel(
  'sourcedb.example.com',
  3306,
  'repl_user',
  'password',
  'mysql-bin-changelog.000777',
  120,
  1,
  3600,
  'channel_1');
```

mysql.rds_set_source_auto_position_for_channel

지정된 채널의 복제 모드를 바이너리 로그 파일 위치 또는 글로벌 트랜잭션 식별자(GTID)를 기반으로 설정합니다.

구문

```
CALL mysql.rds_set_source_auto_position_for_channel (
auto_position_mode
 , channel_name
);
```

파라미터

auto_position_mode

로그 파일 위치 복제 또는 GTID를 기반으로 하는 복제를 사용할지 여부를 나타내는 값:

- `0` – 바이너리 로그 파일 위치를 기반으로 한 복제 방법을 사용합니다. 기본값은 `0`입니다.
- `1` – GTID 기반 복제 방법을 사용합니다.

channel_name

다중 소스 복제본의 복제 채널 이름입니다. 각 복제 채널은 특정 호스트 및 포트에서 실행되는 단일 소스 RDS for MySQL DB 인스턴스로부터 바이너리 로그 이벤트를 수신합니다.

사용 노트

마스터 사용자는 mysql.rds_set_source_auto_position_for_channel 프로시저를 실행해야 합니다. 이 프로시저는 지정된 채널에서 복제를 다시 시작하여 지정된 자동 위치 모드를 적용합니다.

예시

다음 예제에서는 GTID 기반 복제 방법을 사용하도록 channel_1의 자동 위치 모드를 설정합니다.

```
call mysql.rds_set_source_auto_position_for_channel(1,'channel_1');
```

mysql.rds_set_source_delay_for_channel

소스 데이터베이스 인스턴스에서 지정된 채널의 다중 소스 복제본으로 복제를 지연할 최소 시간(초)을 설정합니다.

구문

```
CALL mysql.rds_set_source_delay_for_channel(delay, channel_name);
```

파라미터

delay

소스 DB 인스턴스에서 복제를 지연할 최소 시간(초)입니다.

이 파라미터에 대한 제한은 1일(86,400초)입니다.

channel_name

다중 소스 복제본의 복제 채널 이름입니다. 각 복제 채널은 특정 호스트 및 포트에서 실행되는 단일 소스 RDS for MySQL DB 인스턴스로부터 바이너리 로그 이벤트를 수신합니다.

사용 노트

마스터 사용자는 mysql.rds_set_source_delay_for_channel 프로시저를 실행해야 합니다. 프로시저를 사용하려면 우선 mysql.rds_stop_replication_for_channel을 직접 호출하여 복제를 중지합니다. 그런 다음 이 프로시저를 직접 호출하여 복제 지연 값을 설정합니다. 지연이 설정되면 mysql.rds_start_replication_for_channel을 직접 호출하여 복제를 다시 시작합니다.

예시

다음 예제에서는 다중 소스 복제본의 channel_1에 있는 소스 데이터베이스 인스턴스에서 최소 1시간(3,600초) 동안 지연을 설정합니다.

```
CALL mysql.rds_set_source_delay_for_channel(3600,'channel_1');
```

mysql.rds_skip_repl_error_for_channel

지정된 채널의 MySQL DB 다중 소스 복제본의 바이너리 로그 이벤트를 건너뛰고 지정된 채널의 복제 오류를 삭제합니다.

구문

```
CALL mysql.rds_skip_repl_error_for_channel(channel_name);
```

파라미터

channel_name

다중 소스 복제본의 복제 채널 이름입니다. 각 복제 채널은 특정 호스트 및 포트에서 실행되는 단일 소스 RDS for MySQL DB 인스턴스로부터 바이너리 로그 이벤트를 수신합니다.

사용 노트

마스터 사용자는 읽기 전용 복제본에서 mysql.rds_skip_repl_error_for_channel 프로시저를 실행해야 합니다. mysql.rds_skip_repl_error가 읽기 복제본의 오류를 건너뛰는 것과 비슷한 방식으로 이 프로시저를 사용할 수 있습니다. 자세한 내용은 mysql.rds_skip_repl_error 프로시저 호출 단원을 참조하십시오.

> **Note**
>
> GTID 기반 복제의 오류를 건너뛰려면 이 the section called "mysql.rds_skip_transaction_with_gtid" 프로시저를 대신 사용할 것을 권장합니다.

오류가 있는지 여부를 판별하려면 MySQL SHOW REPLICA STATUS FOR CHANNEL '*channel_name*'\G 명령을 실행합니다. 중대한 복제 오류가 아닌 경우

mysql.rds_skip_repl_error_for_channel를 사용하여 오류를 건너뛸 수 있습니다. 오류가 여러 개인 경우 mysql.rds_skip_repl_error_for_channel은 지정된 복제 채널의 첫 번째 오류를 삭제한 후 다른 오류가 있음을 경고합니다. SHOW REPLICA STATUS FOR CHANNEL '*channel_name*'\G를 사용하여 다음 오류에 대한 적합한 조치를 결정할 수 있습니다. 반환된 값에 대한 자세한 내용은 MySQL 설명서의 SHOW SLAVE STATUS 문을 참조하세요.

mysql.rds_start_replication_for_channel

RDS for MySQL DB 인스턴스에서 지정된 채널의 다중 소스 복제본으로의 복제를 시작합니다.

> **Note**
>
> mysql.rds_start_replication_until_for_channel 또는 mysql.rds_start_replication_until_gtid_for_channel 저장 프로시저를 사용하여 RDS for MySQL DB 인스턴스에서 복제를 시작하고 지정된 바이너리 로그 파일 위치에서 복제를 중지할 수 있습니다.

구문

```
CALL mysql.rds_start_replication_for_channel(channel_name);
```

파라미터

channel_name

다중 소스 복제본의 복제 채널 이름입니다. 각 복제 채널은 특정 호스트 및 포트에서 실행되는 단일 소스 RDS for MySQL DB 인스턴스로부터 바이너리 로그 이벤트를 수신합니다.

사용 노트

마스터 사용자는 mysql.rds_start_replication_for_channel 프로시저를 실행해야 합니다. 소스 RDS for MySQL DB 인스턴스에서 데이터를 가져온 후 다중 소스 복제본에서 이 명령을 실행하여 지정된 채널에서 복제를 시작합니다.

예시

다음 예제에서는 다중 소스 복제본의 channel_1에서 복제를 시작합니다.

```
CALL mysql.rds_start_replication_for_channel('channel_1');
```

mysql.rds_start_replication_until_for_channel

지정된 채널의 RDS for MySQL DB 인스턴스에서 복제를 시작하고 지정된 바이너리 로그 파일 위치에서 복제를 중지합니다.

구문

```
CALL mysql.rds_start_replication_until_for_channel (
replication_log_file
  , replication_stop_point
  , channel_name
);
```

파라미터

replication_log_file

복제 정보를 포함하는 소스 DB 인스턴스의 이진 로그 이름입니다.

replication_stop_point

복제를 중지할 replication_log_file 바이너리 로그 내 위치입니다.

channel_name

다중 소스 복제본의 복제 채널 이름입니다. 각 복제 채널은 특정 호스트 및 포트에서 실행되는 단일 소스 RDS for MySQL DB 인스턴스로부터 바이너리 로그 이벤트를 수신합니다.

사용 노트

마스터 사용자는 mysql.rds_start_replication_until_for_channel 프로시저를 실행해야 합니다. 이 프로시저를 사용하면 복제가 시작되었다가 지정된 binlog 파일 위치에 도달하면 복제가 중지됩니다. 이 프로시저는 SQL_THREAD 및 IO_THREAD를 모두 중지합니다.

replication_log_file 파라미터에 지정된 파일 이름은 소스 DB 인스턴스 binlog 파일 이름과 일치해야 합니다.

replication_stop_point 파라미터가 과거에 해당하는 중지 위치를 지정하는 경우 복제가 즉시 중지됩니다.

예시

다음 예에서는 channel_1에서 복제를 시작하고 mysql-bin-changelog.000777 바이너리 로그 파일의 120 위치에 도달할 때까지 변경 사항을 복제합니다.

```
call mysql.rds_start_replication_until_for_channel(
  'mysql-bin-changelog.000777',
  120,
  'channel_1'
  );
```

mysql.rds_start_replication_until_gtid_for_channel

RDS for MySQL DB 인스턴스에서 지정된 채널에 복제를 시작하고 지정된 전역 트랜잭션 식별자(GTID)에서 복제를 중지합니다.

구문

```
CALL mysql.rds_start_replication_until_gtid_for_channel(gtid,channel_name);
```

파라미터

gtid

GTID 이후 복제를 중지합니다.

channel_name

다중 소스 복제본의 복제 채널 이름입니다. 각 복제 채널은 특정 호스트 및 포트에서 실행되는 단일 소스 RDS for MySQL DB 인스턴스로부터 바이너리 로그 이벤트를 수신합니다.

사용 노트

마스터 사용자는 mysql.rds_start_replication_until_gtid_for_channel 프로시저를 실행해야 합니다. 이 프로시저는 지정된 채널에서 복제를 시작하고 지정된 GTID 값까지 모든 변경 내용을 적용합니다. 그런 다음 채널에서의 복제가 중지됩니다.

gtid 파라미터가 복제본으로 이미 실행한 트랜잭션을 지정하는 경우 복제가 즉시 중지됩니다.

이 프로시저를 실행하기 전에 replica_parallel_workers 또는 slave_parallel_workers 값을 0으로 설정하여 멀티스레드 복제를 비활성화해야 합니다.

예시

다음 예제에서는 channel_1에서 복제를 시작하고 GTID 3E11FA47-71CA-11E1-9E33-C80AA9429562:23에 도달할 때까지 변경 사항을 복제합니다.

```
call mysql.rds_start_replication_until_gtid_for_channel('3E11FA47-71CA-11E1-9E33-
C80AA9429562:23','channel_1');
```

mysql.rds_stop_replication_for_channel

지정된 채널의 MySQL DB 인스턴스에서 복제를 중지합니다.

구문

```
CALL mysql.rds_stop_replication_for_channel(channel_name);
```

파라미터

channel_name

다중 소스 복제본의 복제 채널 이름입니다. 각 복제 채널은 특정 호스트 및 포트에서 실행되는 단일 소스 RDS for MySQL DB 인스턴스로부터 바이너리 로그 이벤트를 수신합니다.

사용 노트

마스터 사용자는 mysql.rds_stop_replication_for_channel 프로시저를 실행해야 합니다.

예시

다음 예시에서는 다중 소스 복제본의 channel_1에서 복제를 중지합니다.

```
CALL mysql.rds_stop_replication_for_channel('channel_1');
```

GTID를 사용한 트랜잭션 복제

다음 저장 프로시저는 RDS for MySQL에서 전역 트랜잭션 식별자(GTID)를 사용하여 트랜잭션을 복제하는 방법을 제어합니다. RDS for MySQL에서 GTID 기반 복제에 대한 자세한 내용은 GTID 기반 복제 사용 섹션을 참조하세요.

`caching_sha2_password`로 구성된 복제 사용자로 복제를 관리하기 위해 저장 프로시저를 사용하는 경우, `SOURCE_SSL=1`을 지정하여 TLS를 구성해야 합니다. `caching_sha2_password`는 RDS for MySQL 8.4의 기본 인증 플러그인입니다.

주제

- mysql.rds_skip_transaction_with_gtid
- mysql.rds_start_replication_until_gtid

mysql.rds_skip_transaction_with_gtid

MySQL DB 인스턴스에서 지정된 전역 트랜잭션 식별자(GTID)를 사용하여 트랜잭션 복제를 건너 뜁니다.

특정 GTID 트랜잭션이 문제의 원인으로 알려진 경우 재해 복구를 위해 이 프로시저를 사용할 수 있습니다. 이 저장 프로시저를 사용하여 문제의 트랜잭션을 건너 뛰십시오. 문제의 트랜잭션의 예로는 복제를 비활성화하거나 중요한 데이터를 삭제하거나 DB 인스턴스를 사용할 수 없도록 하는 트랜잭션이 포함됩니다.

구문

```
CALL mysql.rds_skip_transaction_with_gtid (
gtid_to_skip
);
```

파라미터

gtid_to_skip

건너 뛸 복제 트랜잭션의 GTID입니다.

사용 노트

마스터 사용자는 `mysql.rds_skip_transaction_with_gtid` 프로시저를 실행해야 합니다.

이 프로시저는 모든 RDS for MySQL 5.7 버전, 모든 RDS for MySQL 8.0 버전, 모든 RDS for MySQL 8.4 버전에서 지원됩니다.

예시

다음 예에서는 GTID `3E11FA47-71CA-11E1-9E33-C80AA9429562:23`을 사용하여 트랜잭션의 복제를 건너뜁니다.

```
CALL mysql.rds_skip_transaction_with_gtid('3E11FA47-71CA-11E1-9E33-C80AA9429562:23');
```

mysql.rds_start_replication_until_gtid

RDS for MySQL DB 인스턴스에서 복제를 시작하고 지정된 글로벌 트랜잭션 식별자(GTID) 바로 다음에서 복제를 중지합니다.

구문

```
CALL mysql.rds_start_replication_until_gtid(gtid);
```

파라미터

gtid

GTID 이후 복제를 중지해야 합니다.

사용 노트

마스터 사용자는 `mysql.rds_start_replication_until_gtid` 프로시저를 실행해야 합니다.

이 프로시저는 모든 RDS for MySQL 5.7 버전, 모든 RDS for MySQL 8.0 버전, 모든 RDS for MySQL 8.4 버전에서 지원됩니다.

이 프로시저는 지연 복제에서 재해 복구를 위해 사용할 있습니다. 지연된 복제를 구성한 경우 이 프로시저를 사용하여 지연된 읽기 전용 복제본에 대한 변경 사항을 재해 직전 시간으로 롤포워드할 수 있습니다. 이 프로시저에서 복제를 중지한 이후에 읽기 전용 복제본을 독립 DB 인스턴스로 승격의 지침에 따라 읽기 전용 복제본을 새 기본 DB 인스턴스가 되도록 승격할 수 있습니다.

다음 저장 프로시저를 사용하여 지연 복제를 구성할 수 있습니다.

- mysql.rds_set_configuration

- mysql.rds_set_external_master_with_delay(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하)
- mysql.rds_set_external_source_with_delay(RDS for MySQL 메이저 버전 8.4 이상)
- mysql.rds_set_source_delay

gtid 파라미터가 복제본으로 이미 실행한 트랜잭션을 지정하는 경우 복제가 즉시 중지됩니다.

예시

다음 예제에서는 복제를 시작하고 GTID 3E11FA47-71CA-11E1-9E33-C80AA9429562:23에 도달할 때까지 변경 사항을 복제합니다.

```
call mysql.rds_start_replication_until_gtid('3E11FA47-71CA-11E1-9E33-C80AA9429562:23');
```

쿼리 로그 교체

다음 저장 프로시저는 MySQL 로그를 백업 테이블로 로테이션합니다. 자세한 내용은 MySQL 데이터베이스 로그 파일 단원을 참조하십시오.

주제

- mysql.rds_rotate_general_log
- mysql.rds_rotate_slow_log

mysql.rds_rotate_general_log

mysql.general_log 테이블을 백업 테이블로 로테이션합니다.

구문

```
CALL mysql.rds_rotate_general_log;
```

사용 노트

mysql.rds_rotate_general_log 프로시저를 호출하여 mysql.general_log 테이블을 백업 테이블로 로테이션할 수 있습니다. 로그 테이블이 순환되면 현재 로그 테이블은 백업 로그 테이블에 복사되며 현재 로그 테이블의 해당 항목들은 제거됩니다. 백업 로그 테이블이 이미 존재할 경우, 현재 로그 테이블이 백업으로 복사되기 전에 백업 로그 테이블이 삭제됩니다. 필요하다면 백업 로그 테이블을 쿼리할 수 있습니다. mysql.general_log 테이블에 대한 백업 로그 테이블 이름은 mysql.general_log_backup으로 지정됩니다.

이 절차는 log_output 파라미터를 TABLE로 설정한 경우에만 실행할 수 있습니다.

mysql.rds_rotate_slow_log

mysql.slow_log 테이블을 백업 테이블로 로테이션합니다.

구문

```
CALL mysql.rds_rotate_slow_log;
```

사용 노트

mysql.rds_rotate_slow_log 프로시저를 호출하여 mysql.slow_log 테이블을 백업 테이블로 로테이션할 수 있습니다. 로그 테이블이 순환되면 현재 로그 테이블은 백업 로그 테이블에 복사되며 현재 로그 테이블의 해당 항목들은 제거됩니다. 백업 로그 테이블이 이미 존재할 경우, 현재 로그 테이블이 백업으로 복사되기 전에 백업 로그 테이블이 삭제됩니다.

필요하다면 백업 로그 테이블을 쿼리할 수 있습니다. mysql.slow_log 테이블에 대한 백업 로그 테이블 이름은 mysql.slow_log_backup으로 지정됩니다.

이진수 로그 구성 설정 및 표시

다음 저장 프로시저는 바이너리 로그 파일 유지에 대한 파라미터와 같은 구성 파라미터를 설정하고 표시합니다.

주제

- mysql.rds_set_configuration
- mysql.rds_show_configuration

mysql.rds_set_configuration

바이너리 로그를 보관할 기간(시간) 또는 복제를 지연할 시간(초)을 지정합니다.

구문

```
CALL mysql.rds_set_configuration(name,value);
```

파라미터

##

설정할 구성 파라미터의 이름입니다.

USD ##

구성 파라미터의 값입니다.

사용 노트

`mysql.rds_set_configuration` 프로시저는 다음 구성 파라미터를 지원합니다.

- binlog retention hours
- 소스 지연
- target delay

구성 파라미터는 영구적으로 저장되며 DB 인스턴스 재부팅 또는 장애 조치 이후에도 유지됩니다.

binlog retention hours

`binlog retention hours` 파라미터는 이진 로그 파일을 보관할 기간(시간)을 지정하는 데 사용됩니다. Amazon RDS는 일반적으로 가능한 한 빨리 바이너리 로그를 삭제하지만 RDS 외부의 MySQL 데이터베이스 복제에는 바이너리 로그가 필요할 수 있습니다.

`binlog retention hours`의 기본값은 `NULL`입니다. RDS for MySQL의 경우 `NULL`은 바이너리 로그가 유지되지 않음을 의미합니다(0시간).

DB 인스턴스에 대한 바이너리 로그를 유지할 기간(시간)을 지정하려면 다음 예에 나와 있는 것처럼 `mysql.rds_set_configuration` 저장 프로시저를 사용하여 복제 수행에 충분한 기간을 지정합니다.

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

> **Note**
>
> `binlog retention hours`에는 `0` 값을 사용할 수 없습니다.

MySQL DB 인스턴스의 경우 최대 `binlog retention hours` 값은 168(7일)입니다.

보존 기간을 설정한 후, DB 인스턴스 스토리지의 사용량을 모니터링하여 보존된 바이너리 로그가 너무 많은 스토리지를 차지하지 않도록 합니다.

다중 AZ DB 클러스터 배포의 경우 라이터 DB 인스턴스에서만 이진 로그 유지를 구성할 수 있으며, 설정은 모든 리더 DB 인스턴스에 비동기적으로 전파됩니다. DB 클러스터의 이진 로그가 총 로컬 스토리지 공간의 절반을 초과하는 경우 Amazon RDS는 자동으로 오래된 로그를 EBS 볼륨으로 이동합니다. 하지만 최신 로그는 로컬 스토리지에 남아 있으므로 호스트 교체가 필요한 장애가 발생하거나 데이터베이스를 확장 또는 축소하는 경우 손실될 수 있습니다.

소스 지연

`source delay` 파라미터를 사용하여 읽기 전용 복제본에서 소스 DB 인스턴스로의 복제를 지연할 시간(초)을 지정합니다. Amazon RDS에서는 일반적으로 변경 사항을 최대한 빨리 복제하지만, 일부 환경에서는 복제를 지연하기를 원할 수 있습니다. 예를 들어, 복제가 지연될 경우 지연된 읽기 전용 복제본을 재해 직전 시간으로 롤포워드할 수 있습니다. 실수로 테이블이 삭제된 경우 지연된 복제를 사용하여 테이블을 빠르게 복구할 수 있습니다. `target delay`의 기본값은 `0`입니다(복제를 지연하지 않음).

이 파라미터를 사용하면 mysql.rds_set_source_delay 실행되고 CHANGE primary TO MASTER_DELAY = input value가 적용됩니다. 성공하면 프로시저가 `source delay` 파라미터를 `mysql.rds_configuration` 테이블에 저장합니다.

Amazon RDS에서 소스 DB 인스턴스에 대한 복제를 지연할 시간(초)을 지정하려면 `mysql.rds_set_configuration` 저장 프로시저를 사용하고 복제를 지연할 시간(초)을 지정합니다. 다음 예제에서는 복제를 1시간(3,600초) 이상 지연하도록 지정합니다.

```
call mysql.rds_set_configuration('source delay', 3600);
```

그러면 프로시저가 `mysql.rds_set_source_delay(3600)`를 실행합니다.

`source delay` 파라미터에 대한 제한은 1일(86,400초)입니다.

target delay

`target delay` 파라미터를 사용하여 DB 인스턴스와 이 인스턴스에서 생성되는 향후 RDS 관리 읽기 전용 복제본 간의 복제를 지연할 시간(초)을 지정합니다. RDS에서 관리되지 않는 읽기 전용 복제본에서는 이 파라미터가 무시됩니다. Amazon RDS에서는 일반적으로 변경 사항을 최대한 빨리 복제하지만, 일부 환경에서는 복제를 지연하기를 원할 수 있습니다. 예를 들어, 복제가 지연될 경우 지연된 읽기 전용 복제본을 재해 직전 시간으로 롤포워드할 수 있습니다. 실수로 테이블이 삭제된 경우 지연된 복제를 사용하여 테이블을 빠르게 복구할 수 있습니다. `target delay`의 기본값은 `0`입니다(복제를 지연하지 않음).

재해 복구의 경우 이 구성 파라미터를 mysql.rds_start_replication_until 저장 프로시저 또는 mysql.rds_start_replication_until_gtid 저장 프로시저와 함께 사용할 수 있습니다. 지연된 읽기 전용 복제본에 대한 변경 사항을 재해 직전 시간으로 롤포워드하려면 이 파라미터를 설정한 상태에서 `mysql.rds_set_configuration` 프로시저를 실행할 수 있습니다. `mysql.rds_start_replication_until` 또는 `mysql.rds_start_replication_until_gtid` 프로시저에서 복제를 중지한 이후에 읽기 전용 복제본을 독립 DB 인스턴스로 승격의 지침에 따라 읽기 전용 복제본을 새 기본 DB 인스턴스가 되도록 승격할 수 있습니다.

`mysql.rds_rds_start_replication_until_gtid` 프로시저를 사용하려면 GTID를 기반으로 한 복제를 활성화해야 합니다. 재해 원인으로 알려진 특정 GTID 기반 트랜잭션을 건너 뛰려면 mysql.rds_skip_transaction_with_gtid 저장 프로시저를 사용할 수 있습니다. GTID 기반 복제 작업에 대한 자세한 내용은 GTID 기반 복제 사용 단원을 참조하십시오.

Amazon RDS에서 읽기 전용 복제본에 대한 복제를 지연할 시간(초)을 지정하려면 `mysql.rds_set_configuration` 저장 프로시저를 사용하고 복제를 지연할 시간(초)을 지정합니다. 다음 예제에서는 복제를 1시간(3,600초) 이상 지연하도록 지정합니다.

```
call mysql.rds_set_configuration('target delay', 3600);
```

`target delay` 파라미터에 대한 제한은 1일(86,400초)입니다.

mysql.rds_show_configuration

바이너리 로그가 유지되는 시간입니다.

구문

```
CALL mysql.rds_show_configuration;
```

사용 노트

Amazon RDS의 이진수 로그 보관 시간을 확인하려면 `mysql.rds_show_configuration` 저장 프로시저를 사용합니다.

예시

다음 예제는 보존 기간을 표시합니다.

```
call mysql.rds_show_configuration;
                name                          value     description
                binlog retention hours        24        binlog retention hours specifies
 the duration in hours before binary logs are automatically deleted.
```

InnoDB 캐시 워밍

다음 저장 프로시저는 RDS for MySQL DB 인스턴스에서 InnoDB 버퍼 풀을 저장 또는 로드하거나 로드를 취소합니다. 자세한 내용은 Amazon RDS의 MySQL용 InnoDB 캐시 워밍 단원을 참조하십시오.

주제

- mysql.rds_innodb_buffer_pool_dump_now
- mysql.rds_innodb_buffer_pool_load_abort
- mysql.rds_innodb_buffer_pool_load_now

mysql.rds_innodb_buffer_pool_dump_now

버퍼 풀의 현재 상태를 디스크에 덤프합니다.

구문

```
CALL mysql.rds_innodb_buffer_pool_dump_now();
```

사용 노트

마스터 사용자는 `mysql.rds_innodb_buffer_pool_dump_now` 프로시저를 실행해야 합니다.

mysql.rds_innodb_buffer_pool_load_abort

진행 중인 저장된 버퍼 풀 상태 로드를 취소합니다.

구문

```
CALL mysql.rds_innodb_buffer_pool_load_abort();
```

사용 노트

마스터 사용자는 `mysql.rds_innodb_buffer_pool_load_abort` 프로시저를 실행해야 합니다.

mysql.rds_innodb_buffer_pool_load_now

디스크에서 버퍼 풀의 저장된 상태를 디스크에 로드합니다.

구문

```
CALL mysql.rds_innodb_buffer_pool_load_now();
```

사용 노트

마스터 사용자는 `mysql.rds_innodb_buffer_pool_load_now` 프로시저를 실행해야 합니다.

Amazon RDS for Oracle

Amazon RDS는 Oracle Database의 다음 버전 및 에디션을 실행하는 DB 인스턴스를 지원합니다.

- Oracle Database 21c(21.0.0.0)
- Oracle Database 19c(19.0.0.0)

> **Note**
>
> Oracle Database 11g, Oracle Database 12 및 Oracle Database 18c는 Amazon RDS에서 더 이상 지원되지 않는 레거시 버전입니다.

DB 인스턴스를 생성하기 전에 이 안내서의 Amazon RDS 환경 설정 단원에 있는 단계를 완료해야 합니다. 마스터 계정을 사용하여 DB 인스턴스를 생성하면 계정에 DBA 권한이 부여되며 몇 가지 제한이 있습니다. 추가 데이터베이스 계정 생성과 같은 관리 작업에 이 계정을 사용합니다. SYS, SYSTEM 또는 기타 Oracle에서 제공하는 관리 계정은 사용할 수 없습니다.

다음을 생성할 수 있습니다.

- DB 인스턴스
- DB 스냅샷
- 특정 시점 복원
- 자동 백업
- 수동 백업

VPC 내에서 Oracle Database를 실행하는 DB 인스턴스를 사용할 수 있습니다. 또한 Oracle Spatial 또는 Oracle Statspack과 같은 다양한 옵션을 활성화하여 DB 인스턴스에 기능을 추가할 수 있습니다. Amazon RDS는 고가용성 장애 조치 솔루션으로서 Oracle용 다중 AZ 배포를 지원합니다.

> **Important**
>
> 관리형 서비스 환경을 제공하기 위해 Amazon RDS는 DB 인스턴스에 대해 shell 액세스를 제공하지 않습니다. 또한 고급 권한이 필요한 특정 시스템 절차와 테이블에 대한 액세스를 제한합니다. Oracle SQL*Plus와 같은 표준 SQL 클라이언트를 사용하여 데이터베이스에 액세스할

수 있습니다. 그러나 Telnet 또는 SSH(Secure Shell)를 사용하여 호스트에 직접 액세스할 수는 없습니다.

주제

- Oracle on Amazon RDS 개요
- Oracle DB 인스턴스에 연결
- Oracle DB 인스턴스 연결 보안
- RDS for Oracle에서 CDB 작업
- RDS for Oracle DB 인스턴스 관리
- RDS for Oracle 고급 기능 구성
- Amazon RDS의 Oracle로 데이터 가져오기
- Amazon RDS의 Oracle의 읽기 전용 복제본 작업
- Oracle DB 인스턴스에 옵션 추가
- RDS for Oracle DB 엔진 업그레이드
- RDS for Oracle DB 인스턴스에 서드 파티 소프트웨어 사용
- Oracle 데이터베이스 엔진 릴리스 정보

Oracle on Amazon RDS 개요

다음 섹션을 읽으면 RDS for Oracle의 개요를 파악할 수 있습니다.

주제

- RDS for Oracle 기능
- RDS for Oracle 릴리스
- RDS for Oracle 라이선스 옵션
- RDS for Oracle 사용자 및 권한
- RDS for Oracle DB 인스턴스 클래스
- RDS for Oracle 데이터베이스 아키텍처
- RDS for Oracle 초기화 파라미터
- RDS for Oracle 문자 집합

- RDS for Oracle 제한 사항

RDS for Oracle 기능

Amazon RDS for Oracle은 Oracle 데이터베이스의 기능을 대부분 지원합니다. 일부 기능에는 제한된 지원 또는 제한된 권한이 있을 수 있습니다. 일부 기능은 Enterprise Edition에서만 사용할 수 있으며 일부 기능은 추가 라이선스가 필요합니다. 특정 Oracle 데이터베이스 버전의 Oracle Database 기능에 대한 자세한 내용은 사용 중인 버전의 Oracle 데이터베이스 라이선싱 정보 사용자 설명서를 참조하십시오.

주제

- RDS for Oracle의 새로운 기능
- RDS for Oracle에서 지원되는 기능
- RDS for Oracle에서 지원되지 않는 기능

RDS for Oracle의 새로운 기능

RDS for Oracle의 새로운 기능을 보려면 **`Oracle`** 키워드에서 문서 기록을 검색합니다.

RDS for Oracle에서 지원되는 기능

Amazon RDS for Oracle은 다음 Oracle 데이터베이스 기능을 지원합니다.

> **Note**
>
> 다음 목록이 전부는 아닙니다.

- 고급 압축
- Oracle Application Express(APEX)

  자세한 내용은 Oracle Application Express(APEX) 섹션을 참조하세요.
- 자동 메모리 관리
- 자동 실행 취소 관리
- Automatic Workload Repository(AWR)

자세한 내용은 Automatic Workload Repository(AWR)를 사용하여 성능 보고서 생성 섹션을 참조하세요.

- 동일한 AWS 리전 또는 AWS 리전 간에 최대 성능을 제공하는 활성 데이터 보호

  자세한 내용은 Amazon RDS의 Oracle의 읽기 전용 복제본 작업 단원을 참조하세요.
- 블록체인 테이블(Oracle Database 21c 이상)

  자세한 내용은 Oracle Database 설명서의 블록체인 테이블 관리를 참조하세요.
- 연속 쿼리 알림

  자세한 내용은 Oracle 설명서의 Using Continuous Query Notification(CQN)을 참조하세요.
- 데이터 개정
- 연속 쿼리 알림

  자세한 내용은 Oracle 설명서의 Database Change Notification을 참조하세요.
- 데이터베이스 인 메모리
- 분산 쿼리 및 트랜잭션
- 에디션 기반 재정의

  자세한 내용은 DB 인스턴스의 기본 에디션 설정 단원을 참조하세요.
- EM Express(12c 이상)

  자세한 내용은 Oracle Enterprise Manager을 참조하세요.
- 세분화된 감사
- 플래시백 테이블, 플래시백 쿼리, 플래시백 트랜잭션 쿼리
- 애플리케이션에 대한 점진적인 암호 롤오버(Oracle Database 21c 이상)

  자세한 내용은 Oracle Database 설명서의 Managing Gradual Database Password Rollover for Applications(애플리케이션에 대한 점진적인 데이터베이스 암호 롤오버 관리)를 참조하세요.
- HugePages

  자세한 내용은 RDS for Oracle 인스턴스에 HugePages 활성화 섹션을 참조하세요.
- 가져오기/내보내기(레거시 및 데이터 펌프) 및 SQL*Loader

  자세한 내용은 Amazon RDS의 Oracle로 데이터 가져오기 섹션을 참조하세요.
- Java 가상 머신(JVM)

자세한 내용은 Oracle Java 가상 머신을 참조하세요.

- JavaScript(Oracle Database 21c 이상)

  자세한 내용은 Oracle Database 설명서의 DBMS_MLE를 참조하세요.
- 레이블 보안

  자세한 내용은 Oracle 레이블 보안 섹션을 참조하세요.
- 로케이터

  자세한 내용은 Oracle Locator 섹션을 참조하세요.
- 구체화된 보기
- 멀티테넌트

  Oracle 멀티테넌트 아키텍처는 모든 Oracle Database 19c 이상 릴리스에서 지원됩니다. 자세한 내용은 RDS for Oracle에서 CDB 작업 섹션을 참조하세요.
- 네트워크 암호화

  자세한 내용은 Oracle 기본 네트워크 암호화 및 Oracle 보안 소켓 Layer 단원을 참조하십시오.
- 분할
- 실제 애플리케이션 테스트

  전체 캡처 및 재생 기능을 사용하려면 Amazon Elastic File System(Amazon EFS)을 사용하여 Oracle Real 애플리케이션 테스트에서 생성된 파일에 액세스해야 합니다. 자세한 내용은 Amazon EFS 통합 및 블로그 게시물 Amazon RDS for Oracle과 함께 Oracle Real 애플리케이션 테스트 기능 사용을 참조하세요.
- 애플리케이션 수준에서의 샤딩(Oracle 샤딩 기능 제외)
- 공간 및 그래프

  자세한 내용은 Oracle Spatial 섹션을 참조하세요.
- 스타 쿼리 최적화
- 스트림 및 고급 대기열
- 요약 관리 – 구체화된 보기 쿼리 다시 쓰기
- 텍스트(파일 및 URL 데이터 스토어 유형은 지원되지 않음)
- 토탈 리콜
- TDE(Transparent Data Encryption)

자세한 내용은 Oracle Transparent Data Encryption을 참조하세요.

- 통합 감사, 혼합 모드

  자세한 내용은 Oracle 설명서의 혼합 모드 감사를 참조하세요.

- XML DB(XML DB Protocol Server 사용 안 함)

  자세한 내용은 Oracle XML DB 섹션을 참조하세요.

- 가상 프라이빗 데이터베이스

RDS for Oracle에서 지원되지 않는 기능

Amazon RDS for Oracle은 다음 Oracle 데이터베이스 기능을 지원하지 않습니다.

> **Note**
>
> 다음 목록이 전부는 아닙니다.

- Automatic Storage Management(ASM)
- 데이터베이스 볼트
- 플래시백 데이터베이스

  > **Note**
  >
  > 대체 솔루션은 AWS 데이터베이스 블로그 항목인 Amazon RDS for Oracle에서 Oracle 플래시백 데이터베이스 기능의 대안을 참조하세요.

- FTP 및 SFTP
- 파티셔닝된 하이브리드 테이블
- 메시징 게이트웨이
- Oracle Enterprise Manager Cloud Control Management Repository
- 실제 애플리케이션 클러스터(Oracle RAC)
- Real Application Security(RAS)
- 통합 감사, 순수 모드

- 작업 영역 관리자(WMSYS) 스키마

> **Warning**
>
> 일반적으로 Amazon RDS에서는 지원되지 않는 기능에 대한 스키마를 생성하지 못하게 막지 않습니다. 그러나 SYSDBA 권한이 필요한 Oracle 기능 및 구성 요소에 대한 스키마를 생성하는 경우, 데이터 딕셔너리가 손상되어 DB 인스턴스의 가용성에 영향을 줄 수 있습니다. Oracle DB 인스턴스에 옵션 추가에서 사용할 수 있는 지원되는 기능 및 스키마만 사용하세요.

RDS for Oracle 릴리스

RDS for Oracle은 여러 Oracle Database 릴리스를 지원합니다.

> **Note**
>
> 릴리스 업그레이드에 대한 자세한 내용은 RDS for Oracle DB 엔진 업그레이드 단원을 참조하세요.

주제

- Amazon RDS 기반 Oracle Database 21c
- Amazon RDS 기반 Oracle Database 19c

Amazon RDS 기반 Oracle Database 21c

Amazon RDS는 Oracle Enterprise Edition과 Oracle Standard Edition 2를 포함하는 Oracle Database 21c를 지원합니다. Oracle Database 21c(21.0.0.0)에는 이전 버전에 비해 새로운 기능과 업데이트가 많이 포함되어 있습니다. 주요 변경 사항은 Oracle Database 21c가 멀티테넌트 아키텍처만 지원한다는 것입니다. 즉, 더 이상 기존의 비 CDB로 데이터베이스를 생성할 수 없습니다. CDB와 비 CDB 간의 차이점에 대한 자세한 내용은 RDS for Oracle CDB 제한 사항 단원을 참조하세요.

이 단원에서는 Amazon RDS에서 Oracle Database 21c(21.0.0.0)를 사용하는 데 중요한 기능과 변경 내용을 확인할 수 있습니다. 변경 내용의 전체 목록은 Oracle Database 21c 문서를 참조하세요. 각 Oracle Database 21c 에디션에서 지원하는 전체 기능 목록은 Oracle 설명서의 Permitted Features, Options, and Management Packs by Oracle Database Offering(Oracle Database 제품 및 서비스에서 허용되는 기능, 옵션 및 관리 팩)을 참조하세요.

Oracle Database 21c(21.0.0.0)의 Amazon RDS 파라미터 변경

Oracle Database 21c(21.0.0.0)에는 새로운 파라미터와 범위 및 기본값이 새로 변경된 파라미터가 다수 포함되어 있습니다.

주제

- 새 파라미터
- 호환되는 파라미터에 대한 변경 사항
- 제거된 파라미터

새 파라미터

다음 표에는 Oracle Database 21c(21.0.0.0)의 새로운 Amazon RDS 파라미터가 나와 있습니다.

명칭	값 범위	기본값	수정 가능	설명
blockchain_table_max_no_drop	NONE | 0	NONE	Y	블록체인 테이블을 생성할 때 지정 가능한 최대 유휴 시간을 제어할 수 있습니다.
dbnest_enable	NONE | CDB_RESOURCE_PDB_ALL	NONE	N	dbNest를 사용하거나 사용하지 않도록 설정할 수 있습니다. dbNest는 PDB를 위한 운영 체제 리소스 격리 및 관리, 파일 시스템 격리, 보안 컴퓨팅을 제공합니다.
dbnest_pdb_fs_conf	NONE | *pathname*	NONE	N	PDB에 대한 dbNest 파일 시스템 구성 파일을 지정합니다.
diagnostics_control	ERROR | WARNING | IGNORE	IGNORE	Y	잠재적으로 안전하지 않은 데이터베이스 진단 작업을 수행하는 사용자를 제어하고 모니터링할 수 있습니다.
drcp_dedicated_opt	YES | NO	YES	Y	DRCP(Database Resident Connection Pooling)에서 전용

명칭	값 범위	기본값	수정 가능	설명
				최적화를 사용하거나 사용하지 않도록 설정합니다.
enable_per_pdb_drcp	`true | false`	`true`	N	DRCP(Database Resident Connection Pooling)가 전체 CDB에 대해 하나의 연결 풀을 구성할지 아니면 각 PDB에 대해 하나의 격리된 연결 풀을 구성할지 제어합니다.
inmemory_deep_vectorization	`true | false`	`true`	Y	심층 벡터화 프레임워크를 사용하거나 사용하지 않도록 설정합니다.
mandatory_user_profile	*`profile_name`*	N/A	N	CDB 또는 PDB에 대한 필수 사용자 프로파일을 지정합니다.
optimizer_capture_sql_quarantine	`true | false`	`false`	Y	심층 벡터화 프레임워크를 사용하거나 사용하지 않도록 설정합니다.
optimizer_use_sql_quarantine	`true | false`	`false`	Y	SQL Quarantine 구성의 자동 생성을 사용하거나 사용하지 않도록 설정합니다.
result_cache_execution_threshold	`0~68719476736`	2	Y	결과가 결과 캐시에 저장되기 전에 PL/SQL 함수를 실행할 수 있는 최대 횟수를 지정합니다.
result_cache_max_temp_result	`0~100`	5	Y	캐시된 단일 쿼리 결과가 사용할 수 있는 `RESULT_CACHE_MAX_TEMP_SIZE` 의 백분율을 지정합니다.

명칭	값 범위	기본값	수정 가능	설명
result_cache_max_temp_size	0~2199023255552	RESULT_CACHE_SIZE * 10	Y	결과 캐시가 사용할 수 있는 임시 테이블스페이스의 최대 크기(바이트)를 지정합니다.
sga_min_size	0~2199023255552 (최대값은 sga_target 의 50%)	0	Y	플러그형 데이터베이스(PDB)의 SGA 사용량에 가능한 최소값을 나타냅니다.
tablespace_encryption_default_algorithm	GOST256 | SEED128 | ARIA256 | ARIA192 | ARIA128 | 3DES168 | AES256 | AES192 | AES128	AES128	Y	데이터베이스가 테이블스페이스를 암호화할 때 사용하는 기본 알고리즘을 지정합니다.

호환되는 파라미터에 대한 변경 사항

compatible 파라미터에 Amazon RDS 기반 Oracle Database 21c(21.0.0.0)의 새로운 최대값이 있습니다. 다음 표에는 새 기본값이 나와 있습니다.

파라미터 이름	Oracle Database 21c(21.0.0.0) 최대값
compatible	21.0.0

제거된 파라미터

Oracle Database 21c(21.0.0.0)에서는 다음 파라미터가 제거되었습니다.

- remote_os_authent

- sec_case_sensitive_logon
- unified_audit_sga_queue_size

Amazon RDS 기반 Oracle Database 19c

Amazon RDS는 Oracle Enterprise Edition과 Oracle Standard Edition Two를 포함하는 Oracle Database 19c를 지원합니다.

Oracle Database 19c(19.0.0.0)에는 이전 버전에 비해 새로운 기능과 업데이트가 많이 포함되어 있습니다. 이 단원에서는 Amazon RDS에서 Oracle Database 19c(19.0.0.0)를 사용하는 데 중요한 기능과 변경 내용을 확인할 수 있습니다. 변경 내용의 전체 목록은 Oracle Database 19c 문서를 참조하세요. 각 Oracle Database 19c 에디션에서 지원하는 전체 기능 목록은 Oracle 설명서의 Permitted Features, Options, and Management Packs by Oracle Database Offering을 참조하세요.

Oracle Database 19c(19.0.0.0)의 Amazon RDS 파라미터 변경

Oracle Database 19c(19.0.0.0)에는 새로운 파라미터와 범위 및 기본값이 새로 변경된 파라미터가 다수 포함되어 있습니다.

주제

- 새 파라미터
- 호환되는 파라미터에 대한 변경 사항
- 제거된 파라미터

새 파라미터

다음 표에는 Oracle Database 19c(19.0.0.0)의 새로운 Amazon RDS 파라미터가 나와 있습니다.

명칭	값	수정 가능	설명
lob_signature_enable	TRUE, FALSE(기본값)	Y	LOB 로케이터 서명 기능을 활성화하거나 비활성화합니다.
max_datapump_parallel_per_job	1 ~ 1024 또는 AUTO	Y	각 Oracle Data Pump 작업에 허용되는 최대 병렬 프로세스 수를 지정합니다.

호환되는 파라미터에 대한 변경 사항

`compatible` 파라미터에 Amazon RDS 기반 Oracle Database 19c(19.0.0.0)의 새로운 최대값이 있습니다. 다음 표에는 새 기본값이 나와 있습니다.

파라미터 이름	Oracle Database 19c(19.0.0.0) 최대값
compatible	19.0.0

제거된 파라미터

Oracle Database 19c(19.0.0.0)에서는 다음 파라미터가 제거되었습니다.

- `exafusion_enabled`
- `max_connections`
- `o7_dictionary_access`

RDS for Oracle 라이선스 옵션

Amazon RDS for Oracle에는 License Included(LI) 및 Bring Your Own License(BYOL)의 두 가지 라이선스 옵션이 있습니다. Amazon RDS에서 Oracle DB 인스턴스를 생성한 후 DB 인스턴스를 수정하여 라이선스 모델을 변경할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

> **Important**
>
> DB 인스턴스 클래스와 Oracle Database 에디션에 적합한 Oracle 데이터베이스 라이선스(소프트웨어 업데이트 라이선스 및 지원 포함)가 있어야 합니다. 또한 별도로 라이선스가 부여된 Oracle Database 기능에 대한 라이선스가 있는지 확인하세요.

주제

- SE2용 라이선스 포함 모델
- EE 및 SE2용 기존 보유 라이선스 사용(BYOL)
- Oracle 다중 AZ 배포 라이선스

SE2용 라이선스 포함 모델

License Included 모델에서는 Oracle 데이터베이스 라이선스를 별도로 구매할 필요가 없습니다. 즉 AWS에 Oracle 데이터베이스 소프트웨어 라이선스가 포함됩니다. 라이선스 포함 모델은 Amazon RDS for Oracle Database Standard Edition Two(SE2)에 대해 지원됩니다.

이 모델에서는 사례 지원이 포함된 AWS Support 계정이 있는 경우, Amazon RDS 및 Oracle 데이터베이스 서비스 요청은 모두 지원에 문의합니다. RDS for Oracle 사용 시 LI 옵션은 AWS서비스 약관의 섹션 10.3.1의 적용을 받습니다.

EE 및 SE2용 기존 보유 라이선스 사용(BYOL)

BYOL 모델에서는 기존의 Oracle 데이터베이스 라이선스를 사용하여 Amazon RDS에서 데이터베이스를 배포할 수 있습니다. Amazon RDS는 Oracle Database Enterprise Edition(EE) 및 Oracle Database Standard Edition 2(SE2)에 대해서만 BYOL 모델을 지원합니다.

실행할 DB 인스턴스 클래스와 Oracle Database 에디션에 적합한 Oracle 데이터베이스 라이선스(소프트웨어 업데이트 라이선스 및 지원 포함)가 있어야 합니다. 또한 클라우드 컴퓨팅 환경에 대한 Oracle의 Oracle Database 소프트웨어 라이선스 부여 정책을 따라야 합니다. Oracle의 Amazon EC2 라이선스 정책에 대한 자세한 내용은 Licensing Oracle Software in the Cloud Computing Environment를 참조하십시오.

이 모델에서는 활성 Oracle 지원 계정을 계속 사용할 수 있습니다. Oracle Database 서비스 요청은 Oracle에 직접 문의하십시오. AWS Support 계정에 사례 지원이 있는 경우 Amazon RDS 관련 문제는 지원에 문의합니다. Amazon Web Services 및 Oracle에는 두 조직의 지원이 필요한 사례에 대해 다중 벤더 지원 프로세스가 있습니다.

AWS License Manager와 통합

BYOL 모델에서 Oracle 라이선스 사용량을 보다 쉽게 모니터링하려면 AWS License Manager는 Amazon RDS for Oracle과 통합합니다. License Manager는 Oracle 엔진 버전에 대한 RDS 추적 및 vCPU(가상 코어)에 기반한 라이선스 팩을 지원합니다. 또한 License Manager를 AWS Organizations와 함께 사용하여 모든 조직 계정을 중앙에서 관리할 수도 있습니다.

다음 표에는 RDS for Oracle용 제품 정보 필터가 나와 있습니다.

Filter	이름	설명
엔진 에디션	`oracle-ee`	Oracle Database Enterprise Edition(EE)

Filter	이름	설명
	`oracle-se2`	Oracle Database Standard Edition 2(SE2)
라이선스 팩	`data guard`	Amazon RDS의 Oracle의 읽기 전용 복제본 작업 참조(Oracle Active Data Guard)
	`olap`	Oracle OLAP섹션을 참조하십시오.
	`ols`	Oracle 레이블 보안섹션을 참조하십시오.
	`diagnostic pack sqlt`	Oracle SQLT섹션을 참조하십시오.
	`tuning pack sqlt`	Oracle SQLT 부분 참조

Oracle DB 인스턴스의 라이선스 사용량을 추적하려면 AWS License Manager를 사용하여 자체 관리형 라이선스를 생성합니다. 이 경우 제품 정보 필터와 일치하는 RDS for Oracle 리소스가 자체 관리형 라이선스와 자동으로 연결됩니다. Oracle DB 인스턴스 검색에는 최대 24시간이 소요될 수 있습니다. AWS Resource Access Manager를 사용하여 여러 계정의 라이선스를 추적할 수도 있습니다.

콘솔

RDS for Oracle DB 인스턴스의 라이선스 사용량을 추적하기 위해 AWS License Manager에서 자체 관리형 라이선스를 생성하려면

1. https://console.aws.amazon.com/license-manager/로 이동합니다.
2. 자체 관리 라이선스 생성을 선택합니다.

   자세한 내용은 AWS License Manager 사용 설명서의 Create a self-managed license를 참조하세요.

   제품 정보 패널에서 RDS Product Information Filter(RDS 제품 정보 필터)에 대한 규칙을 추가합니다.

   자세한 내용은AWS License Manager API 참조의 ProductInformation을 참조하십시오.
3. (교차 계정 추적만 해당) AWS Resource Access Manager를 사용하여 AWS 계정 또는 AWS Organizations를 통해 자체 관리형 라이선스를 공유합니다. 자세한 내용은 AWS 리소스 공유를 참조하세요.

AWS CLI

AWS CLI를 사용하여 자체 관리형 라이선스를 생성하려면 create-license-configuration 명령을 직접 호출합니다. `--cli-input-json` 또는 `--cli-input-yaml` 파라미터를 사용하여 파라미터를 명령에 전달합니다.

Example

다음 예에서는 Oracle Enterprise Edition에 대한 자체 관리형 라이선스를 생성합니다.

```
aws license-manager create-license-configuration --cli-input-json file://rds-oracle-ee.json
```

다음은 예제에서 사용되는 샘플 `rds-oracle-ee.json` 파일입니다.

```
{
    "Name": "rds-oracle-ee",
    "Description": "RDS Oracle Enterprise Edition",
    "LicenseCountingType": "vCPU",
    "LicenseCountHardLimit": false,
    "ProductInformationList": [
        {
            "ResourceType": "RDS",
            "ProductInformationFilterList": [
                {
                    "ProductInformationFilterName": "Engine Edition",
                    "ProductInformationFilterValue": ["oracle-ee"],
                    "ProductInformationFilterComparator": "EQUALS"
                }
            ]
        }
    ]
}
```

제품 정보에 대한 자세한 내용은 AWS License Manager 사용 설명서의 리소스 인벤토리 자동 검색을 참조하십시오.

`--cli-input` 파라미터에 대한 자세한 내용은 AWS CLI 사용 설명서의 JSON 또는 YAML 입력 파일에서 AWS CLI 스켈레톤 및 입력 파라미터 생성을 참조하세요.

Oracle Database 버전 간 마이그레이션

실행하려는 DB 인스턴스의 버전과 클래스에 적합한 미사용 BYOL Oracle Database 라이선스가 있다고 가정하면, Standard Edition 2(SE2)에서 Enterprise Edition(EE)으로 마이그레이션할 수 있습니다. EE에서 다른 버전으로 마이그레이션할 수는 없습니다.

Oracle Database 버전을 변경하고 데이터를 유지하려면

1. DB 인스턴스의 스냅샷을 생성합니다.

   자세한 내용은 Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성 섹션을 참조하세요.
2. 스냅샷을 새 DB 인스턴스로 복원하고 사용하려는 Oracle 데이터베이스 에디션을 선택합니다.

   자세한 내용은 DB 인스턴스 복원 섹션을 참조하세요.
3. (선택 사항) 이전 DB 인스턴스를 계속 실행하고 적절한 Oracle 데이터베이스 라이선스를 갖고 싶지 않으면 이전 DB 인스턴스를 삭제하십시오.

   자세한 내용은 DB 인스턴스 삭제 섹션을 참조하세요.

Oracle 다중 AZ 배포 라이선스

Amazon RDS는 고가용성 장애 조치 솔루션으로서 Oracle용 다중 AZ 배포를 지원합니다. 프로덕션 워크로드에는 다중 AZ를 권장합니다. 자세한 내용은 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 섹션을 참조하세요.

기본 보유 라이선스 사용 모델을 사용하는 경우, 다중 AZ 배포에 기본 DB 인스턴스와 보조 DB 인스턴스 모두에 대한 라이선스가 있어야 합니다.

RDS for Oracle 사용자 및 권한

Amazon RDS for Oracle DB 인스턴스를 생성할 때 기본 마스터 사용자는 DB 인스턴스에 대한 최대 사용자 권한의 대부분을 갖게 됩니다. 마스터 사용자 계정을 사용하여 데이터베이스에서 추가 사용자 계정 생성과 같은 관리 작업을 수행합니다. RDS는 관리형 서비스이므로 SYS 및 SYSTEM으로 로그인할 수 없습니다. 따라서 SYSDBA 권한을 보유하지 않습니다.

주제

- Oracle DBA 권한에 대한 제한 사항
- SYS 객체에 대한 권한을 관리하는 방법

Oracle DBA 권한에 대한 제한 사항

데이터베이스에서 역할이란 사용자에 대해 부여하거나 취소할 수 있는 권한 모음입니다. Oracle 데이터베이스는 역할을 사용하여 보안을 제공합니다. 자세한 내용은 Oracle Database 설명서에서 권한 및 역할 권한 부여 구성을 참조하세요.

일반적으로 사전 정의된 역할 DBA는 Oracle 데이터베이스에 대한 모든 관리 권한을 허용합니다. DB 인스턴스를 생성하면 마스터 사용자 계정에 DBA 권한이 부여됩니다(일부 제한 사항 포함). 관리형 경험을 제공하기 위해 RDS for Oracle 데이터베이스는 DBA 역할에 다음 권한을 제공하지 않습니다.

- `ALTER DATABASE`
- `ALTER SYSTEM`
- `CREATE ANY DIRECTORY`
- `DROP ANY DIRECTORY`
- `GRANT ANY PRIVILEGE`
- `GRANT ANY ROLE`

RDS for Oracle 시스템 권한 및 역할에 대한 자세한 내용은 마스터 사용자 계정 권한 섹션을 참조하세요.

SYS 객체에 대한 권한을 관리하는 방법

`rdsadmin.rdsadmin_util` 패키지를 사용하여 SYS 객체에 대한 권한을 관리할 수 있습니다. 예를 들어, 데이터베이스 사용자 `myuser`를 생성하면 `rdsadmin.rdsadmin_util.grant_sys_object` 프로시저를 사용하여 `myuser`에게 `V_$SQLAREA`에 대한 `SELECT` 권한을 부여할 수 있습니다. 자세한 정보는 다음 주제를 참조하세요.

- SYS 객체에 대한 SELECT 또는 EXECUTE 권한 부여
- SYS 객체에 대한 SELECT 또는 EXECUTE 권한 취소
- 마스터가 아닌 사용자에게 권한 부여

RDS for Oracle DB 인스턴스 클래스

RDS for Oracle DB 인스턴스의 계산 및 메모리 용량은 인스턴스 클래스에 따라 결정됩니다. 필요한 DB 인스턴스 클래스는 DB 인스턴스의 처리력 및 메모리 요구 사항에 따라 다릅니다.

지원되는 RDS for Oracle DB 인스턴스 클래스

지원되는 RDS for Oracle 인스턴스 클래스는 RDS DB 인스턴스 클래스의 하위 집합입니다. RDS 인스턴스 클래스의 전체 목록은 DB 인스턴스 클래스 섹션을 참조하세요.

RDS for Oracle 사전 구성된 DB 인스턴스 클래스

RDS for Oracle은 vCPU당 추가 메모리, 스토리지 및 I/O가 필요한 워크로드에 사전 구성된 인스턴스 클래스도 제공합니다. 이러한 인스턴스 클래스는 다음 명명 규칙을 사용합니다.

```
db.r5b.instance_size.tpcthreads_per_core.memratio
db.r5.instance_size.tpcthreads_per_core.memratio
```

다음은 추가 메모리용으로 사전 구성된 인스턴스 클래스의 예입니다.

```
db.r5b.4xlarge.tpc2.mem2x
```

이전 인스턴스 클래스 이름의 구성 요소는 다음과 같습니다.

- `db.r5b.4xlarge` - 인스턴스 클래스의 이름입니다.
- `tpc2` - 코어당 스레드 수입니다. 값 2는 다중 스레딩이 켜져 있음을 의미합니다. 값 1은 다중 스레딩이 꺼져 있음을 의미합니다.
- `mem2x` - 인스턴스 클래스의 표준 메모리에 대한 추가 메모리의 비율입니다. 이 예에서 최적화는 표준 db.r5.4xlarge DB 인스턴스보다 두 배 많은 메모리를 제공합니다.

RDS for Oracle에서 지원되는 에디션, 인스턴스 클래스 및 라이선스 조합

RDS 콘솔을 사용하는 경우 데이터베이스 생성을 선택하고 다른 옵션을 지정하여 특정 에디션, 인스턴스 클래스 및 라이선스 조합이 지원되는지 여부를 확인할 수 있습니다. AWS CLI에서 다음 명령을 실행할 수 있습니다.

```
aws rds describe-orderable-db-instance-options --engine engine-type --license-
model license-type
```

다음 테이블에는 RDS for Oracle에 지원되는 모든 에디션, 인스턴스 클래스 및 라이선스 유형이 나와 있습니다. 각 유형의 메모리 특성에 대한 자세한 내용은 RDS for Oracle 인스턴스 유형을 참조하세요. 요금에 대한 자세한 내용은 Amazon RDS for Oracle 요금 모델을 참조하세요.

Oracle 버전	Oracle Database 19c 이상
Enterprise Edition(EE)	표준 인스턴스 클래스
기존 보유 라이선스 사용 (BYOL)	db.m7i.large–db.m7i.48xlarge db.m6id.large–db.m6id.32xlarge db.m6i.large–db.m6i.32xlarge db.m5d.large–db.m5d.24xlarge db.m5.large-db.m5.24xlarge
	메모리 최적화 인스턴스 클래스
	db.r7i.large–db.r7i.48xlarge db.r6id.large–db.r6id.32xlarge db.r6i.large–db.r6i.32xlarge db.r5d.large–db.r5d.24xlarge db.r5b.large–db.r5b.24xlarge db.r5.large-db.r5.24xlarge db.x2iedn.xlarge–db.x2iedn.32xlarge db.x2iezn.2xlarge–db.x2iezn.12xlarge db.x2idn.16xlarge–db.x2idn.32xlarge db.x1e.xlarge–db.x1e.32xlarge db.x1.16xlarge–db.x1.32xlarge[1] db.z1d.large–db.z1d.12xlarge
	사전 구성된 메모리 최적화 인스턴스 클래스
	db.r6i.8xlarge.tpc2.mem4x

Oracle 버전	Oracle Database 19c 이상
	db.r6i.8xlarge.tpc2.mem3x
	db.r6i.6xlarge.tpc2.mem4x
	db.r6i.4xlarge.tpc2.mem4x
	db.r6i.4xlarge.tpc2.mem3x
	db.r6i.4xlarge.tpc2.mem2x
	db.r6i.2xlarge.tpc2.mem8x
	db.r6i.2xlarge.tpc2.mem4x
	db.r6i.2xlarge.tpc1.mem2x
	db.r6i.xlarge.tpc2.mem4x
	db.r6i.xlarge.tpc2.mem2x
	db.r6i.large.tpc1.mem2x
	db.r5b.8xlarge.tpc2.mem3x
	db.r5b.6xlarge.tpc2.mem4x
	db.r5b.4xlarge.tpc2.mem4x
	db.r5b.4xlarge.tpc2.mem3x
	db.r5b.4xlarge.tpc2.mem2x
	db.r5b.2xlarge.tpc2.mem8x
	db.r5b.2xlarge.tpc2.mem4x
	db.r5b.2xlarge.tpc1.mem2x
	db.r5b.xlarge.tpc2.mem4x
	db.r5b.xlarge.tpc2.mem2x

Oracle 버전	Oracle Database 19c 이상
	db.r5b.large.tpc1.mem2x
	db.r5.12xlarge.tpc2.mem2x
	db.r5.8xlarge.tpc2.mem3x
	db.r5.6xlarge.tpc2.mem4x
	db.r5.4xlarge.tpc2.mem4x
	db.r5.4xlarge.tpc2.mem3x
	db.r5.4xlarge.tpc2.mem2x
	db.r5.2xlarge.tpc2.mem8x
	db.r5.2xlarge.tpc2.mem4x
	db.r5.2xlarge.tpc1.mem2x
	db.r5.xlarge.tpc2.mem4x
	db.r5.xlarge.tpc2.mem2x
	db.r5.large.tpc1.mem2x
	버스트 가능한 성능 인스턴스 클래스
	db.t3.small–db.t3.2xlarge
Standard Edition 2(SE2)	표준 인스턴스 클래스
기존 보유 라이선스 사용 (BYOL)	db.m7i.large–db.m7i.4xlarge
	db.m6id.large–db.m6id.4xlarge
	db.m6i.large–db.m6i.4xlarge
	db.m5d.large–db.m5d.4xlarge
	db.m5.large-db.m5.4xlarge

Oracle 버전	Oracle Database 19c 이상
	메모리 최적화 인스턴스 클래스
	db.r7i.large–db.r7i.4xlarge db.r6id.large–db.r6id.4xlarge db.r6i.large–db.r6i.4xlarge db.r5d.large–db.r5d.4xlarge db.r5b.large–db.r5b.4xlarge db.r5.large–db.r5.4xlarge db.x2iedn.xlarge–db.x2iedn.4xlarge db.x2iezn.2xlarge–db.x2iezn.4xlarge db.z1d.large–db.z1d.3xlarge
	사전 구성된 메모리 최적화 인스턴스 클래스

Oracle 버전	Oracle Database 19c 이상
	db.r6i.4xlarge.tpc2.mem4x
	db.r6i.4xlarge.tpc2.mem3x
	db.r6i.4xlarge.tpc2.mem2x
	db.r6i.2xlarge.tpc2.mem8x
	db.r6i.2xlarge.tpc2.mem4x
	db.r6i.2xlarge.tpc1.mem2x
	db.r6i.xlarge.tpc2.mem4x
	db.r6i.xlarge.tpc2.mem2x
	db.r6i.large.tpc1.mem2x
	db.r5b.4xlarge.tpc2.mem4x
	db.r5b.4xlarge.tpc2.mem3x
	db.r5b.4xlarge.tpc2.mem2x
	db.r5b.2xlarge.tpc2.mem8x
	db.r5b.2xlarge.tpc2.mem4x
	db.r5b.2xlarge.tpc1.mem2x
	db.r5b.xlarge.tpc2.mem4x
	db.r5b.xlarge.tpc2.mem2x
	db.r5b.large.tpc1.mem2x
	db.r5.4xlarge.tpc2.mem4x
	db.r5.4xlarge.tpc2.mem3x
	db.r5.4xlarge.tpc2.mem2x

Oracle 버전	Oracle Database 19c 이상
	db.r5.2xlarge.tpc2.mem8x
	db.r5.2xlarge.tpc2.mem4x
	db.r5.2xlarge.tpc1.mem2x
	db.r5.xlarge.tpc2.mem4x
	db.r5.xlarge.tpc2.mem2x
	db.r5.large.tpc1.mem2x
	버스트 가능한 성능 인스턴스 클래스
	db.t3.small–db.t3.2xlarge
Standard Edition 2(SE2)	표준 인스턴스 클래스
라이선스 포함	db.m5.large-db.m5.4xlarge
	메모리 최적화 인스턴스 클래스
	db.r6i.large–db.r6i.4xlarge
	db.r5.large–db.r5.4xlarge
	버스트 가능한 성능 인스턴스 클래스
	db.t3.small–db.t3.2xlarge

[1] 이제 X1.* 인스턴스 클래스 패밀리를 사용하여 RDS for Oracle DB 인스턴스를 만들 수 없습니다(X1e.* 인스턴스 클래스는 계속 지원됨). 현재 X1.* 클래스를 사용하는 경우 가능한 한 빨리 새로운 세대의 인스턴스 클래스로 전환하세요. 2025년 1월 22일부터 RDS는 정의된 유지 관리 기간에 자동 업그레이드를 시작합니다. 업그레이드 중에 RDS는 동등한 X2iedn 인스턴스 유형을 선택하여 업그레이드합니다. 자세한 내용은 Amazon RDS for Oracle is ending support for X1 Database Instances on January 22, 2025 re.Post 게시물을 참조하세요.

> **Note**
>
> 모든 BYOL 고객들은 라이선싱 계약을 참고하여 Amazon RDS for Oracle의 지원 중단이 미치는 영향을 평가할 것을 권장합니다. RDS for Oracle에서 지원되는 DB 인스턴스 클래스의 컴퓨팅 용량에 대한 자세한 내용은 DB 인스턴스 클래스 및 RDS for Oracle에서 DB 인스턴스 클래스의 프로세서 구성 단원을 참조하세요.

> **Note**
>
> 지원 중단된 DB 인스턴스 클래스를 사용한 DB 인스턴스의 DB 스냅샷이 있는 경우 DB 스냅샷을 복원할 때 만료되지 않은 DB 인스턴스 클래스를 선택할 수 있습니다. 자세한 내용은 DB 인스턴스 복원 섹션을 참조하세요.

RDS for Oracle DB 인스턴스 클래스 지원 중단

다음은 RDS for Oracle에 대해 지원 중단되는 DB 인스턴스 클래스입니다.

- db.m1, db.m2, db.m3, db.m4
- db.t1, db.t2
- db.r1, db.r2, db.r3, db.r4

이전 DB 인스턴스 클래스는 성능이 더 좋고 일반적으로 낮은 가격으로 구할 수 있는 DB 인스턴스로 교체되었습니다. 지원 중단되는 DB 인스턴스 클래스를 사용하는 DB 인스턴스를 보유한 경우 다음과 같은 옵션이 있습니다.

- 지원 중단되지 않은 유사한 DB 인스턴스 클래스를 사용하려면 Amazon RDS가 각 DB 인스턴스를 자동으로 수정하도록 허용합니다. 지원 중단 일정은 DB 인스턴스 클래스 유형 섹션을 참조하세요.
- DB 인스턴스를 수정하여 DB 인스턴스 클래스를 직접 변경합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

지원 중단된 DB 인스턴스 클래스를 사용한 DB 인스턴스의 DB 스냅샷이 있는 경우 DB 스냅샷을 복원할 때 만료되지 않은 DB 인스턴스 클래스를 선택할 수 있습니다. 자세한 정보는 DB 인스턴스 복원의 내용을 참조하세요.

RDS for Oracle 데이터베이스 아키텍처

Oracle 멀티테넌트 아키텍처(CDB 아키텍처)는 Oracle 데이터베이스가 멀티테넌트 컨테이너 데이터베이스(CDB) 기능을 하도록 합니다. CDB에는 고객이 생성한 플러그형 데이터베이스(PDB)가 포함될 수 있습니다. PDB를 포함할 수 없는 기존 아키텍처를 사용하는 Oracle 데이터베이스는 비 CDB입니다. 멀티테넌트 아키텍처에 대한 자세한 내용은 Oracle Multitenant Administrator's Guide를 참조하세요.

Oracle Database 19c 이상의 경우 CDB 아키텍처를 사용하는 RDS for Oracle DB 인스턴스를 생성할 수 있습니다. RDS for Oracle에서는 PDB를 테넌트 데이터베이스라고 합니다. 클라이언트 애플리케이션은 CDB 수준이 아닌 테넌트 데이터베이스(PDB) 수준에서 연결합니다. RDS for Oracle은 CDB 아키텍처의 다음 구성을 지원합니다.

다중 테넌트 구성

이 RDS 플랫폼은 RDS for Oracle CDB 인스턴스에 1~30개의 테넌트 데이터베이스를 포함하도록 허용합니다. 이 데이터베이스는 데이터베이스 에디션 및 필요한 옵션 라이선스에 따라 다릅니다. RDS API를 사용하여 테넌트 데이터베이스를 추가, 수정 및 제거할 수 있습니다. RDS for Oracle의 다중 테넌트 구성은 특별한 유형의 PDB인 애플리케이션 PDB 또는 프록시 PDB를 지원하지 않습니다. 애플리케이션 PDB 및 프록시 PDB에 대한 자세한 내용은 Oracle Database 설명서에 나와 있는 Types of PDBs를 참조하시기 바랍니다.

> **Note**
>
> Amazon RDS 구성은 Oracle DB 엔진뿐만 아니라 RDS 플랫폼의 기능이기 때문에 '멀티테넌트'가 아닌 '다중 테넌트'라고 합니다. 마찬가지로 RDS 용어 '테넌트'는 Oracle PDB뿐만 아니라 RDS 구성의 모든 테넌트를 나타냅니다. RDS 설명서에서 하이픈이 포함되지 않은 용어 'Oracle 멀티테넌트'는 온프레미스 및 RDS 배포 모두와 호환되는 Oracle 데이터베이스 CDB 아키텍처만을 의미합니다.

단일 테넌트 구성

이 RDS 플랫폼 기능은 RDS for Oracle CDB 인스턴스의 테넌트 데이터베이스(PDB)를 1개로 제한합니다. RDS API를 사용하여 PDB를 더 추가할 수는 없습니다. 단일 테넌트 구성은 비CDB 아키텍처와 동일한 RDS API를 사용합니다. 따라서 단일 테넌트 구성에서 CDB를 사용하는 경험은 비CDB를 사용한 작업과 거의 동일합니다.

단일 테넌트 구성을 사용하는 CDB를 다중 테넌트 구성으로 변환할 수 있으므로, PDB를 CDB에 추가할 수 있습니다. 이 아키텍처 변경은 영구적이며 되돌릴 수 없습니다. 자세한 내용은 단일 테넌트 구성을 다중 테넌트로 변환 단원을 참조하십시오.

> **Note**
>
> CDB 자체에 액세스할 수 없습니다.

Oracle Database 21c 이상에서는 모든 데이터베이스가 CDB입니다. 반면에 Oracle Database 19c DB 인스턴스는 CDB 또는 비CDB로 생성할 수 있습니다. 비CDB를 CDB로 업그레이드할 수는 없지만 Oracle Database 19c 비CDB를 CDB로 변환한 후 업그레이드할 수는 있습니다. CDB를 비CDB로 변환할 수 없습니다.

자세한 정보는 다음 자료를 참조하세요.

- RDS for Oracle에서 CDB 작업
- RDS for Oracle CDB 제한 사항
- Amazon RDS DB 인스턴스 생성

RDS for Oracle 초기화 파라미터

Amazon RDS에서는 파라미터 그룹을 사용하여 DB 파라미터를 관리합니다. 이 그룹을 사용하여 초기화 파라미터를 사용자 지정할 수 있습니다. 예를 들어 `sort_area_size`를 사용하여 정렬 영역 크기를 구성할 수 있습니다. 특정 DB 파라미터 그룹과 연결된 모든 RDS for Oracle DB 인스턴스는 동일한 파라미터 설정을 사용합니다. 자세한 내용은 Amazon RDS의 파라미터 그룹 섹션을 참조하세요.

RDS for Oracle에서 지원되는 초기화 파라미터

DB 인스턴스에 지원되는 파라미터는 Oracle Database 에디션 및 버전에 따라 다릅니다. 특정 Oracle Database 에디션 및 버전에 지원되는 초기화 파라미터를 보려면 AWS CLI 명령 `describe-engine-default-parameters`를 실행합니다. 예를 들어 Oracle Database 19c의 Enterprise Edition에 지원되는 초기화 파라미터를 보려면 다음 명령을 실행합니다(샘플 출력 포함).

```
aws rds describe-engine-default-parameters \
    --db-parameter-group-family oracle-ee-19 \
    --output json | jq -r '.EngineDefaults.Parameters[].ParameterName'
```

```
_add_col_optim_enabled
_adg_parselock_timeout
_allow_insert_with_update_check
_allow_level_without_connect_by
_always_semi_join
_autotask_max_window
_awr_disabled_flush_tables
_awr_mmon_cpuusage
_awr_mmon_deep_purge_all_expired
_b_tree_bitmap_plans
_bct_bitmaps_per_file
_bloom_filter_enabled
_buffered_publisher_flow_control_threshold
_bug29394014_allow_triggers_on_vpd_table
_cleanup_rollback_entries
_client_enable_auto_unregister
_clusterwide_global_transactions
_complex_view_merging
_connect_by_use_union_all
_cost_equality_semi_join
_cursor_features_enabled
_cursor_obsolete_threshold
_datafile_write_errors_crash_instance
_db_block_buffers
...
```

단일 초기화 파라미터를 설명하려면 다음 명령을 사용하고 sga_max_size를 해당 파라미터 이름으로 바꿉니다(샘플 출력 포함).

```
aws rds describe-engine-default-parameters \
    --db-parameter-group-family oracle-ee-19 \
    --query 'EngineDefaults.Parameters[?ParameterName==`sga_max_size`]' \
    --output json

[
    {
        "ParameterName": "sga_max_size",
        "Description": "max total SGA size",
        "Source": "engine-default",
        "ApplyType": "static",
        "DataType": "integer",
        "AllowedValues": "0-2199023255552",
        "IsModifiable": true
```

```
    }
]
```

Oracle 데이터베이스 초기화 파라미터에 대한 일반적인 설명서를 찾으려면 Oracle Database 설명서의 Initialization Parameters를 참조하세요. RDS for Oracle에서는 파라미터 ARCHIVE_LAG_TARGET에 특별한 고려 사항이 있다는 점에 유의하세요. 이 파라미터는 지정된 시간이 경과한 후 온라인 다시 실행 로그를 강제로 전환합니다. RDS for Oracle에서는 Recovery Point Objective(RPO)가 5분이기 때문에 ARCHIVE_LAG_TARGET이 300으로 설정됩니다. 이 목표를 달성하기 위해 RDS for Oracle은 5분마다 온라인 다시 실행 로그를 전환하여 Amazon S3 버킷에 저장합니다.

온라인 다시 실행 로그 전환 빈도로 인해 RDS for Oracle 데이터베이스의 성능이 저하되는 경우, 더 높은 IOPS와 처리량을 사용하도록 DB 인스턴스와 스토리지의 규모를 조정할 수 있습니다. 또는 RDS Custom for Oracle을 사용하거나 Amazon EC2에 Oracle 데이터베이스를 배포하는 경우 ARCHIVE_LAG_TARGET 초기화 파라미터의 설정을 조정할 수 있습니다.

RDS for Oracle의 유효한 파라미터 값

RDS for Oracle에서는 파라미터 값에 다음 문자만 유효합니다.

- 문자(A-Z 및 a-z)
- 숫자(0-9)
- 공백(공백, 탭 및 줄 바꿈)
- 다음 특수 문자: _ / . : + = ( ) ' * , % $ -(하이픈)

RDS for Oracle 문자 집합

RDS for Oracle은 DB 문자 집합과 국가별 문자 집합의 두 가지 유형의 문자 집합을 지원합니다.

DB 문자 집합

Oracle 데이터베이스 문자 집합은 CHAR, VARCHAR2 및 CLOB 데이터 유형에 사용됩니다. 데이터베이스는 테이블 이름, 열 이름 및 SQL 문과 같은 메타데이터에도 이 문자 집합을 사용합니다. Oracle 데이터베이스 문자 집합은 일반적으로 DB 문자 집합이라고합니다.

DB 인스턴스를 생성할 때 문자 집합을 설정합니다. 데이터베이스를 생성한 후에는 DB 문자 집합을 변경할 수 없습니다.

지원되는 DB 문자 집합

다음 표에는 Amazon RDS에서 지원되는 Oracle DB 데이터베이스 문자 집합이 나와 있습니다. AWS CLI create-db-instance 명령의 `--character-set-name` 파라미터 또는 Amazon RDS API CreateDBInstance 작업의 `CharacterSetName` 파라미터와 함께 이 표의 값을 사용할 수 있습니다.

> **Note**
>
> CDB의 문자 집합은 항상 AL32UTF8입니다. PDB에 대해서만 다른 문자 집합을 설정할 수 있습니다.

값	설명
AL32UTF8	유니코드 5.0 UTF-8 범용 문자 집합(기본값)
AR8ISO8859P6	ISO 8859-6 라틴어/아랍어
AR8MSWIN1256	Microsoft Windows 코드 페이지 1256 8비트 라틴어/아랍어
BLT8ISO8859P13	ISO 8859-13 발트어
BLT8MSWIN1257	Microsoft Windows 코드 페이지 1257 8비트 발트어
CL8ISO8859P5	ISO 88559-5 라틴어/키릴 자모
CL8MSWIN1251	Microsoft Windows 코드 페이지 1251 8비트 라틴어/키릴 자모
EE8ISO8859P2	ISO 8859-2 동유럽어
EL8ISO8859P7	ISO 8859-7 라틴어/그리스어
EE8MSWIN1250	Microsoft Windows 코드 페이지 1250 8비트 동유럽어
EL8MSWIN1253	Microsoft Windows 코드 페이지 1253 8비트 라틴어/그리스어

값	설명
IW8ISO8859P8	ISO 8859-8 라틴어/히브리어
IW8MSWIN1255	Microsoft Windows 코드 페이지 1255 8비트 라틴어/히브리어
JA16EUC	EUC 24비트 일본어
JA16EUCTILDE	유니코드와의 사이에서 물결표 매핑을 제외하면 JA16EUC와 동일
JA16SJIS	Shift-JIS 16비트 일본어
JA16SJISTILDE	유니코드와의 사이에서 물결표 매핑을 제외하면 JA16SJIS와 동일
KO16MSWIN949	Microsoft Windows 코드 페이지 949 한국어
NE8ISO8859P10	ISO 8859-10 북유럽어
NEE8ISO8859P4	ISO 8859-4 북유럽 및 북동 유럽어
TH8TISASCII	태국 산업 표준 620-2533-ASCII 8비트
TR8MSWIN1254	Microsoft Windows 코드 페이지 1254 8비트 터키어
US7ASCII	ASCII 7비트 영어
UTF8	유니코드 3.0 UTF-8 범용 문자 집합, CESU-8 준수
VN8MSWIN1258	Microsoft Windows 코드 페이지 1258 8비트 베트남어
WE8ISO8859P1	서유럽어 8비트 ISO 8859 Part 1
WE8ISO8859P15	ISO 8859-15 서유럽어
WE8ISO8859P9	ISO 8859-9 서유럽어 및 터키어

값	설명
WE8MSWIN1252	Microsoft Windows 코드 페이지 1252 8비트 서유럽어
ZHS16GBK	GBK 16비트 중국어 간체
ZHT16HKSCS	Microsoft Windows 코드 페이지 950, 홍콩 보조 문자 집합 HKSCS-2001 포함. 문자 집합 변환은 유니코드 3.0을 기반으로 합니다.
ZHT16MSWIN950	Microsoft Windows 코드 페이지 950 중국어 번체
ZHT32EUC	EUC 32비트 중국어 번체

NLS_LANG 환경 변수

로캘은 지정된 언어와 국가에 해당하는 언어 및 문화적 요구 사항을 해결하는 일련의 정보입니다. 클라이언트 환경에서 NLS_LANG 환경 변수를 설정하는 것이 가장 간단하게 Oracle에 대한 로캘 동작을 지정하는 방법입니다. 이 변수는 클라이언트 애플리케이션과 데이터베이스 서버에서 사용되는 언어와 지역을 설정합니다. 또한, 이 파라미터는 클라이언트 애플리케이션에서 입력되거나 표시되는 데이터에 대한 문자 집합에 해당하는 클라이언트의 문자 집합을 표시합니다. NLS_LANG 및 문자 집합에 대한 자세한 정보는 Oracle 설명서의 What is a Character set or Code?를 참조하십시오.

NLS 초기화 파라미터

Amazon RDS의 Oracle DB 인스턴스에 대한 인스턴스 수준에서 다음 National Language Support(NLS) 초기화 파라미터를 설정할 수도 있습니다.

- NLS_DATE_FORMAT
- NLS_LENGTH_SEMANTICS
- NLS_NCHAR_CONV_EXCP
- NLS_TIME_FORMAT
- NLS_TIME_TZ_FORMAT
- NLS_TIMESTAMP_FORMAT
- NLS_TIMESTAMP_TZ_FORMAT

인스턴스 파라미터 수정에 대한 자세한 정보는 Amazon RDS의 파라미터 그룹을 참조하십시오.

SQL 클라이언트에서 다른 NLS 초기화 파라미터를 설정할 수 있습니다. 예를 들어 다음 명령문은 Oracle DB 인스턴스에 연결된 SQL 클라이언트에서 NLS_LANGUAGE 초기화 파라미터를 GERMAN으로 설정합니다.

```
ALTER SESSION SET NLS_LANGUAGE=GERMAN;
```

SQL 클라이언트를 사용하여 Oracle DB 인스턴스에 연결하는 방법에 대한 자세한 정보는 Oracle DB 인스턴스에 연결을 참조하십시오.

국가별 문자 집합

국가별 문자 집합은 NCHAR, NVARCHAR2 및 NCLOB 데이터 유형에 사용됩니다. 국가별 문자 집합은 일반적으로 NCHAR 문자 집합이라고 합니다. DB 문자 집합과 달리 NCHAR 문자 집합은 데이터베이스 메타데이터에 영향을 주지 않습니다.

NCHAR 문자 집합은 다음 문자 집합을 지원합니다.

- AL16UTF16(기본값)
- UTF8

create-db-instance 명령의 `--nchar-character-set-name` 파라미터에 두 값 중 하나를 지정할 수 있습니다(AWS CLI 버전 2만 해당). Amazon RDS API를 사용하는 경우 CreateDBInstance 작업의 `NcharCharacterSetName` 파라미터를 지정합니다. 데이터베이스를 생성한 후에는 국가별 문자 집합을 변경할 수 없습니다.

Oracle 데이터베이스의 유니코드에 대한 자세한 내용은 Oracle 설명서의 Supporting Multilingual Databases with Unicode를 참조하세요.

RDS for Oracle 제한 사항

다음 섹션에서는 RDS for Oracle을 사용할 때 발생하는 중요한 제한 사항을 확인할 수 있습니다. CDB와 관련된 제한 사항은 RDS for Oracle CDB 제한 사항 섹션을 참조하세요.

> **Note**
>
> 단, 이 목록이 전부는 아닙니다.

주제

- Amazon RDS의 Oracle 파일 크기 제한
- RDS for Oracle의 블록 크기 한도
- Oracle에서 제공한 스키마에 대한 공개 동의어
- RDS for Oracle에서 지원되지 않는 기능에 대한 스키마
- RDS for Oracle의 DBA 권한에 대한 제한 사항
- RDS for Oracle의 TLS 1.0 및 1.1 Transport Layer Security 사용 중단

Amazon RDS의 Oracle 파일 크기 제한

RDS for Oracle DB 인스턴스에서 단일 파일의 최대 크기는 16TiB(테라바이트)입니다. 이 제한은 인스턴스에서 사용하는 ext4 파일 시스템에 의해 적용됩니다. 따라서 Oracle 빅파일 데이터 파일은 16TiB로 제한됩니다. 제한을 초과하는 값으로 빅파일 테이블스페이스의 데이터 파일 크기를 조정하려고 하면 다음과 같은 오류가 발생합니다.

```
ORA-01237: cannot extend datafile 6
ORA-01110: data file 6: '/rdsdbdata/db/mydir/datafile/myfile.dbf'
ORA-27059: could not reduce file size
Linux-x86_64 Error: 27: File too large
Additional information: 2
```

RDS for Oracle의 블록 크기 한도

RDS for Oracle DB 인스턴스에서 단일 블록의 최대 크기는 8KB입니다. 16KB의 블록 크기는 지원되지 않습니다.

Oracle에서 제공한 스키마에 대한 공개 동의어

SYS, SYSTEM 및 RDSADMIN을 포함하여 Oracle에서 제공하는 스키마에 대한 공개 동의어를 생성하거나 수정하지 마십시오. 그러면 핵심 데이터베이스 구성 요소가 무효화되고 DB 인스턴스의 가용성에 영향을 미칠 수 있습니다.

자체 스키마에서 객체를 참조하는 공개 동의어를 생성할 수 있습니다.

RDS for Oracle에서 지원되지 않는 기능에 대한 스키마

일반적으로 Amazon RDS에서는 지원되지 않는 기능에 대한 스키마를 생성하지 못하게 막지 않습니다. 그러나 SYS 권한이 필요한 Oracle 기능 및 구성 요소에 대한 스키마를 생성하는 경우, 데이터 딕셔

너리가 손상되어 인스턴스 가용성에 영향을 줄 수 있습니다. Oracle DB 인스턴스에 옵션 추가에서 사용할 수 있는 지원되는 기능 및 스키마만 사용하세요.

RDS for Oracle의 DBA 권한에 대한 제한 사항

데이터베이스에서 역할이란 사용자에 대해 부여하거나 취소할 수 있는 권한 모음입니다. Oracle 데이터베이스는 역할을 사용하여 보안을 제공합니다.

일반적으로 사전 정의된 역할 DBA는 Oracle 데이터베이스에 대한 모든 관리 권한을 허용합니다. DB 인스턴스를 생성하면 마스터 사용자 계정에 DBA 권한이 부여됩니다(일부 제한 사항 포함). 관리형 경험을 제공하기 위해 RDS for Oracle 데이터베이스는 DBA 역할에 다음 권한을 제공하지 않습니다.

- `ALTER DATABASE`
- `ALTER SYSTEM`
- `CREATE ANY DIRECTORY`
- `DROP ANY DIRECTORY`
- `GRANT ANY PRIVILEGE`
- `GRANT ANY ROLE`

마스터 사용자 계정을 사용하여 데이터베이스에서 추가 사용자 계정 생성과 같은 관리 작업을 수행합니다. SYS, SYSTEM 및 기타 Oracle에서 제공하는 관리 계정은 사용할 수 없습니다.

RDS for Oracle의 TLS 1.0 및 1.1 Transport Layer Security 사용 중단

전송 계층 보안 프로토콜 버전 1.0 및 1.1(TLS 1.0 및 TLS 1.1)은 더 이상 사용되지 않습니다. 보안 모범 사례에 따라 Oracle은 TLS 1.0 및 TLS 1.1을 더 이상 사용하지 않습니다. 보안 요구 사항을 충족하기 위해 TLS 1.2를 대신 사용할 것을 적극 권장합니다.

Oracle DB 인스턴스에 연결

Amazon RDS가 Oracle DB 인스턴스를 프로비저닝한 후에는 표준 SQL 클라이언트 애플리케이션을 사용해 DB 인스턴스에 로그인할 수 있습니다. RDS는 관리형 서비스이므로 SYS 또는 SYSTEM으로 로그인할 수 없습니다. 자세한 내용은 RDS for Oracle 사용자 및 권한 단원을 참조하십시오.

이 주제에서는 Oracle SQL Developer 또는 SQL*Plus를 사용하여 RDS for Oracle DB 인스턴스에 연결하는 방법을 알아봅니다. 사용자가 샘플 DB 인스턴스를 만들어 연결하는 절차를 실습하는 예제는 Oracle DB 인스턴스 생성 및 해당 인스턴스에 연결 단원을 참조하십시오.

주제

- RDS for Oracle DB 인스턴스의 엔드포인트 찾기
- Oracle SQL Developer를 사용하여 DB 인스턴스에 연결
- SQL*Plus를 사용하여 DB 인스턴스에 연결
- 보안 그룹에 대한 고려 사항
- 프로세스 아키텍처 고려 사항
- Oracle DB 인스턴스에 대한 연결 문제 해결
- sqlnet.ora 파라미터를 사용하여 연결 속성 수정

RDS for Oracle DB 인스턴스의 엔드포인트 찾기

각 Amazon RDS DB 인스턴스에는 엔드포인트가 있으며, 각 엔드포인트에는 DB 인스턴스의 DNS 이름과 포트 번호가 있습니다. SQL 클라이언트 애플리케이션을 사용해 DB 인스턴스에 연결하려면 DB 인스턴스에 연결할 수 있는 DNS 이름과 포트 번호가 필요합니다.

Amazon RDS 콘솔 또는 AWS CLI를 사용하여 DB 인스턴스의 엔드포인트를 찾을 수 있습니다.

> **Note**
>
> Kerberos 인증을 사용 중이라면 Kerberos 인증을 사용하여 Oracle에 연결 단원을 참조하십시오.

콘솔

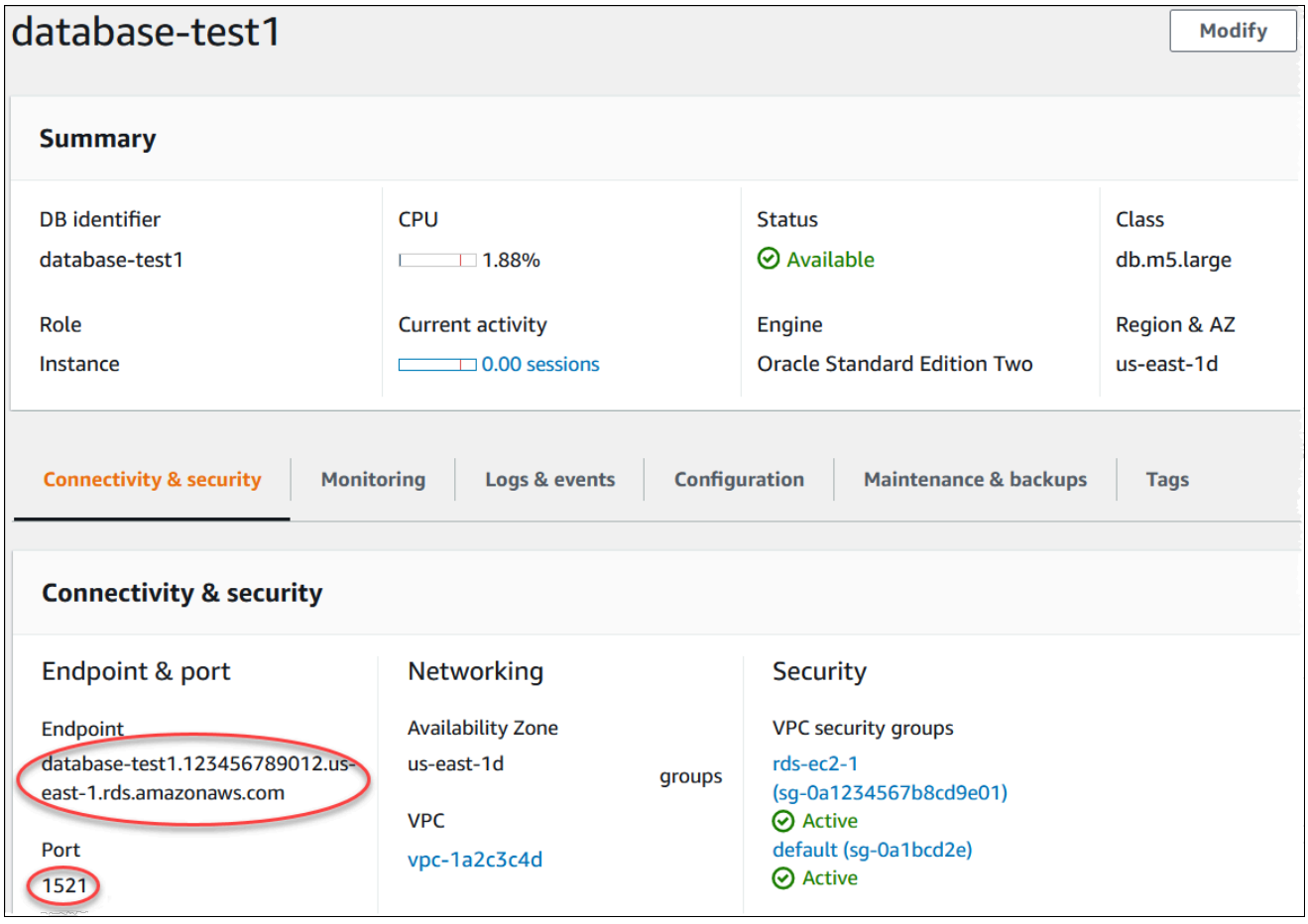
콘솔을 사용하여 엔드포인트를 찾으려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 콘솔의 오른쪽 상단에서 DB 인스턴스의 AWS 리전을 선택합니다.
3. DB 인스턴스에 대한 DNS 이름과 포트 번호를 찾습니다.

   a. DB 인스턴스 목록을 표시할 인스턴스를 선택합니다.

   b. 인스턴스 세부 정보를 표시할 Oracle DB 인스턴스 이름을 선택합니다.

   c. Connectivity & security(연결 및 보안) 탭에서 엔드포인트를 복사합니다. 또한 포트 번호를 적어 둡니다. DB 인스턴스에 연결하려면 엔드포인트와 포트 번호가 모두 필요합니다.



AWS CLI

AWS CLI를 사용하여 Oracle DB 인스턴스의 엔드포인트를 찾으려면 describe-db-instances 명령을 호출하십시오.

Example AWS CLI를 사용하여 엔드포인트를 찾으려면

```
aws rds describe-db-instances
```

출력에서 `Endpoint`를 검색하여 DB 인스턴스의 DNS 이름과 포트 번호를 찾습니다. DNS 이름은 출력의 `Address` 라인에 포함되어 있습니다. 다음은 JSON 엔드포인트 출력의 예입니다.

```
"Endpoint": {
    "HostedZoneId": "Z1PVIF0B656C1W",
    "Port": 3306,
    "Address": "myinstance.123456789012.us-west-2.rds.amazonaws.com"
},
```

> **Note**
>
> 출력에는 여러 DB 인스턴스 정보가 포함될 수 있습니다.

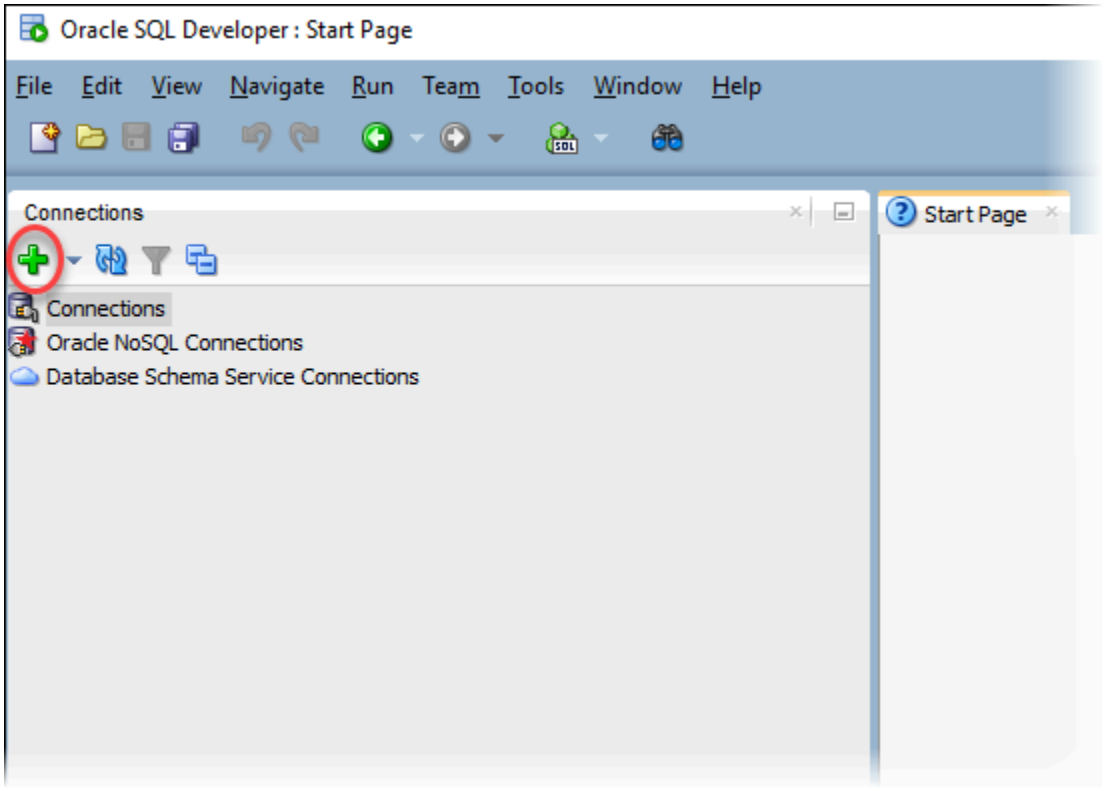
Oracle SQL Developer를 사용하여 DB 인스턴스에 연결

이 절차에서는 Oracle SQL Developer를 사용하여 DB 인스턴스에 연결합니다. 이 유틸리티의 독립 실행형 버전을 다운로드하려면 Oracle SQL Developer 다운로드 페이지를 참조하세요.

DB 인스턴스에 연결하려면 인스턴스의 DNS 이름과 포트 번호가 필요합니다. DB 인스턴스의 DNS 이름 및 포트 번호를 찾는 자세한 내용은 RDS for Oracle DB 인스턴스의 엔드포인트 찾기를 참조하십시오.

SQL Developer를 사용하여 DB 인스턴스에 연결하려면

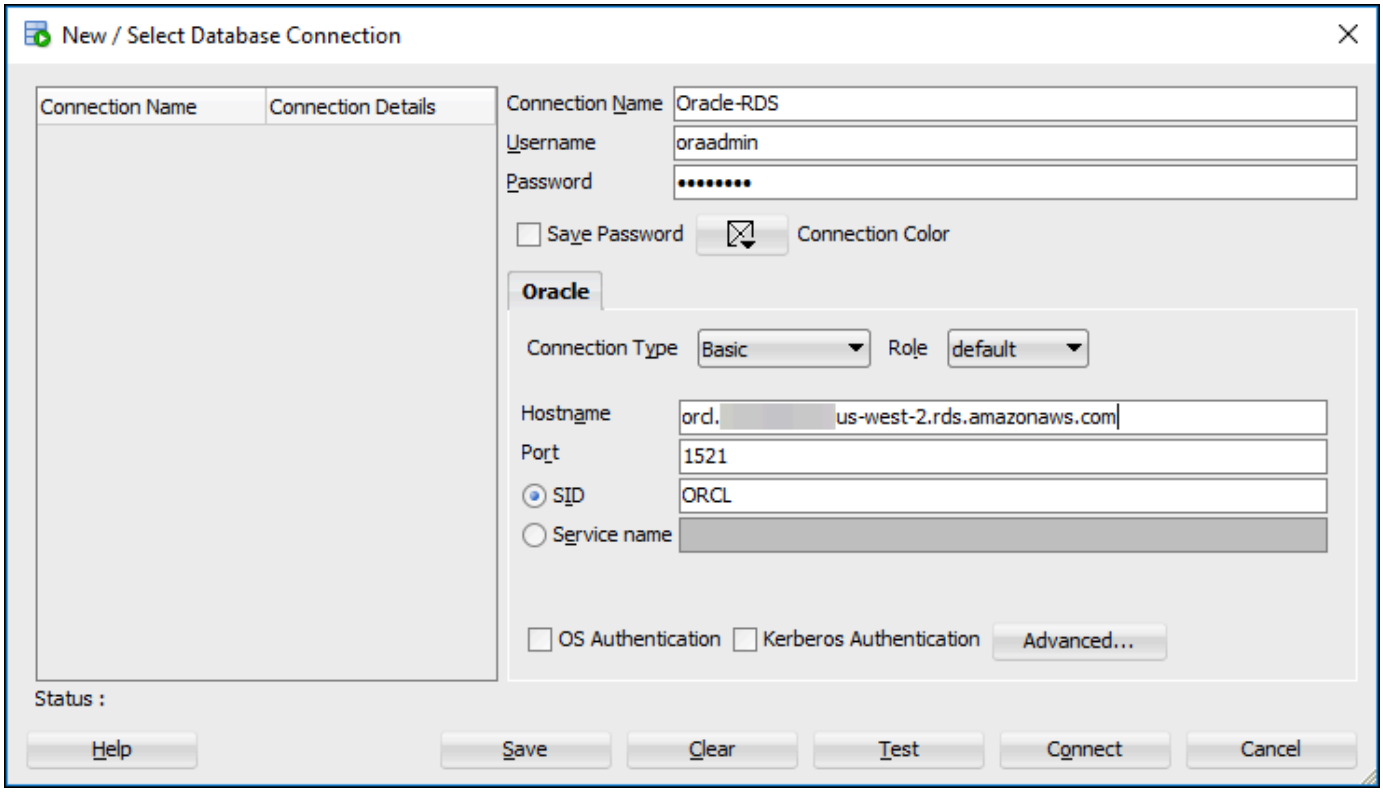
1. Oracle SQL Developer를 시작합니다.
2. [Connections] 탭에서 [add (+)] 아이콘을 선택합니다.



3. [New/Select Database Connection] 대화 상자에 다음과 같이 DB instance 정보를 입력합니다.

   - 연결 이름에 연결을 설명하는 이름(예: `Oracle-RDS`)을 입력합니다.
   - 사용자 이름에 DB 인스턴스의 데이터베이스 관리자 이름을 입력합니다.
   - 암호에 데이터베이스 관리자 암호를 입력합니다.
   - 호스트 이름에 DB 인스턴스의 DNS 이름을 입력하거나 붙여 넣습니다.
   - 포트에 포트 번호를 입력합니다.
   - SID의 경우 DB 이름을 입력합니다. DB 이름은 데이터베이스 세부 정보 페이지의 Configuration(구성) 탭에서 찾을 수 있습니다.

   정보가 모두 입력된 대화 상자는 다음과 비슷한 모습이 되어야 합니다.

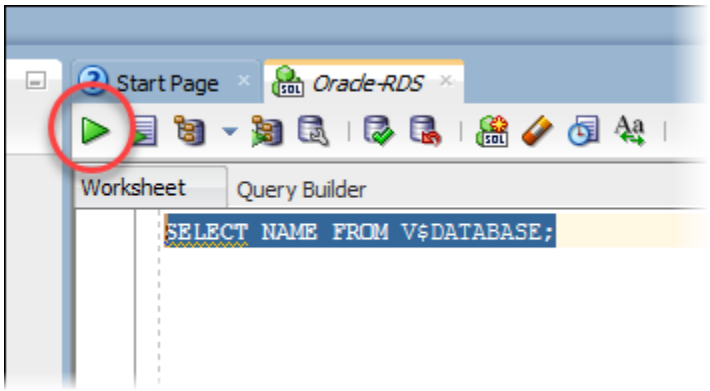


4. [Connect]를 선택합니다.
5. 이제 자체 데이터베이스 생성을 시작하고 평소대로 DB 인스턴스와 데이터베이스에 대한 쿼리 실행을 시작할 수 있습니다. DB 인스턴스에 대한 테스트 쿼리를 실행하려면 다음 중 하나를 수행합니다.

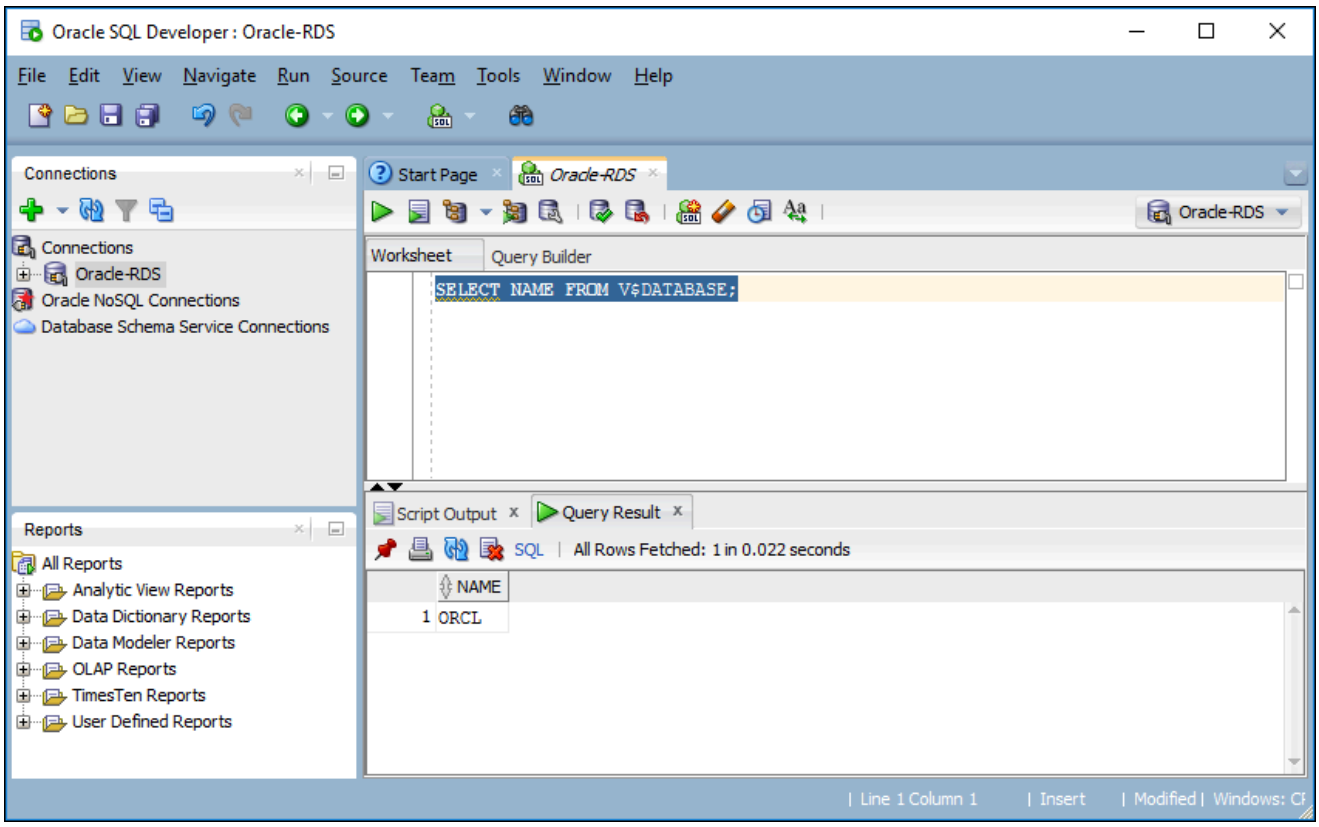
   a. 해당 연결의 워크시트 탭에 다음 SQL 쿼리를 입력합니다.

   ```
   SELECT NAME FROM V$DATABASE;
   ```

   b. 실행 아이콘을 클릭하여 쿼리를 실행합니다.

   SQL Developer가 데이터베이스 이름을 반환합니다.



SQL*Plus를 사용하여 DB 인스턴스에 연결

SQL*Plus 같은 유틸리티를 사용하면 Oracle을 실행하는 Amazon RDS DB 인스턴스에 연결할 수 있습니다. 독립형 버전의 SQL*Plus를 포함하는 Oracle 인스턴트 클라이언트를 다운로드하려면 Oracle 인스턴트 클라이언트 다운로드를 참조하세요.

DB 인스턴스에 연결하려면 인스턴스의 DNS 이름과 포트 번호가 필요합니다. DB 인스턴스의 DNS 이름 및 포트 번호를 찾는 자세한 내용은 RDS for Oracle DB 인스턴스의 엔드포인트 찾기를 참조하십시오.

Example SQL*Plus를 사용하여 Oracle DB 인스턴스에 연결하려면

다음 예제에서 DB 인스턴스 관리자의 사용자 이름을 대체합니다. 또한 DB 인스턴스를 DNS 이름으로 대체한 다음, 포트 번호와 Oracle SID를 포함시킵니다. SID 값은 DB 인스턴스의 이름이 아니라, DB 인스턴스를 생성할 때 지정한 DB 인스턴스의 데이터베이스 이름입니다.

대상 LinuxmacOS, 또는Unix:

```
sqlplus 'user_name@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=dns_name)(PORT=port))
(CONNECT_DATA=(SID=database_name)))'
```

Windows의 경우:

```
sqlplus user_name@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=dns_name)(PORT=port))
(CONNECT_DATA=(SID=database_name)))
```

다음과 유사한 출력 화면이 표시되어야 합니다.

```
SQL*Plus: Release 12.1.0.2.0 Production on Mon Aug 21 09:42:20 2017
```

사용자 암호를 입력하면 SQL 프롬프트가 표시됩니다.

```
SQL>
```

> **Note**
>
> `sqlplus USER/PASSWORD@longer-than-63-chars-rds-endpoint-here:1521/database-identifier`와 같은 짧은 형식의 연결 문자열(EZ Connect)이 최대 문자 제한에 걸릴 수 있으며, 이런 문자열을 연결할 때 사용하면 안 됩니다.

보안 그룹에 대한 고려 사항

DB 인스턴스에 연결하려면 DB 인스턴스가 필요한 IP 주소와 네트워크 구성이 할당되어 있는 보안 그룹과 연동되어야 합니다. DB 인스턴스가 기본 보안 그룹을 사용할 수 있습니다. DB 인스턴스를 생성할 때 구성할 필요가 없는 기본 보안 그룹을 할당한 경우에는 방화벽이 연결을 차단합니다. 새 보안 그룹 생성에 대한 자세한 내용은 보안 그룹을 통한 액세스 제어 단원을 참조하십시오.

새 보안 그룹을 생성하였으면 보안 그룹과 연동되도록 DB 인스턴스 설정을 변경합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

SSL을 사용하여 DB 인스턴스 연결을 암호화함으로써 보안을 강화할 수 있습니다. 자세한 내용은 Oracle 보안 소켓 Layer 단원을 참조하십시오.

프로세스 아키텍처 고려 사항

서버 프로세스는 Oracle DB 인스턴스에 대한 사용자 연결을 처리합니다. 기본적으로 Oracle DB 인스턴스는 전용 서버 프로세스를 사용합니다. 전용 서버 프로세스에서는 서버 프로세스 하나로 사용자 프로세스 하나만 처리합니다. 원한다면 공유 서버 프로세스를 구성할 수 있습니다. 공유 서버 프로세스에서는 서버 프로세스 하나로 사용자 프로세스를 여러 개 처리할 수 있습니다.

다수의 사용자 세션에서 서버 메모리를 지나치게 많이 사용한다면 공유 서버 프로세스를 사용해 보십시오. 또한 세션이 빈번하게 연결 및 연결 해제되어 성능 문제가 발생하는 경우에도 공유 서버 프로세스의 사용을 고려해 볼 수 있습니다. 그러나 공유 서버 프로세스 사용에는 단점도 있습니다. 예를 들어, 공유 서버 프로세스는 CPU 리소스를 과도하게 사용할 수 있고 구성 및 관리하기가 훨씬 복잡합니다.

전용 및 공유 서버 프로세스에 대한 자세한 내용은 Oracle 설명서의 About Dedicated and Shared Server Processes를 참조하십시오. RDS for Oracle DB 인스턴스에서 공유 서버 프로세스를 구성하는 방법에 대한 자세한 내용은 지식 센터에서 공유 서버에서 Amazon RDS for Oracle Database가 작동하도록 구성하려면 어떻게 해야 합니까?를 참조하세요.

Oracle DB 인스턴스에 대한 연결 문제 해결

다음은 Oracle DB 인스턴스 연결을 시도할 때 발생할 수 있는 문제입니다.

문제	문제 해결 제안
DB 인스턴스에 연결할 수 없습니다.	새로 생성한 DB 인스턴스의 경우, DB 인스턴스를 사용할 준비가 될 때까지 DB 인스턴스의 상태는 [creating]입니다. 상태가 available로 변경되면 DB 인스턴스에 연결할 수 있습니다. DB 인스턴스 클래스와 스토리지의 양에 따라 새 DB 인스턴스를 사용할 수 있을 때까지 최대 20분이 걸릴 수 있습니다.
DB 인스턴스에 연결할 수 없습니다.	DB 인스턴스를 만들 때 지정한 포트를 통해 통신을 보내거나 받을 수 없으면 DB 인스턴스에 연결할 수 없습니다. 네트워크 관리자에게 문의해 DB 인스턴스에 대해 지정한 포트가 인바운드 및 아웃바운드 통신을 허용하는지 확인하십시오.
DB 인스턴스에 연결할 수 없습니다.	로컬 방화벽에서 적용되는 액세스 규칙과 DB 인스턴스의 보안 그룹에 있는 DB 인스턴스에 액세스하기 위한 권한을 부여한 IP 주소가 일치하지 않을 수 있습니다. 방화벽의 인바운드 또는 아웃바운드 규칙에 문제가 있을 가능성이 높습니다. 보안 그룹에서 인바운드 규칙을 추가하거나 편집할 수 있습니다. 소스에서 내 IP를 선택합니다. 이렇게 하면 브라우저에서 감지된 IP 주소에서 DB 인스턴스에 액세스할 수 있습니다. 자세한 내용은 Amazon VPC 및 Amazon RDS 섹션을 참조하세요. 보안 그룹에 대한 자세한 내용은 보안 그룹을 통한 액세스 제어 단원을 참조하십시오.

문제	문제 해결 제안
	보안 그룹의 규칙 설정 절차에 대한 자세한 내용은 자습서: DB 인스턴스에 사용할 Amazon VPC 생성(IPv4 전용) 단원을 참조하십시오.
Connect failed because target host or object does not exist – Oracle, Error: ORA-12545(대상 호스트 또는 객체가 존재하지 않으므로 연결이 실패했습니다 - Oracle, 오류: ORA-12545)	서버 이름과 포트 번호를 정확하게 지정했는지 확인하십시오. 서버 이름에는 DNS 이름을 입력하거나 콘솔에서 붙여 넣습니다. DB 인스턴스의 DNS 이름 및 포트 번호를 찾는 자세한 내용은 RDS for Oracle DB 인스턴스의 엔드포인트 찾기를 참조하십시오.
Invalid username/ password; logon denied – Oracle, Error: ORA-01017 (잘못된 사용자 이름/암호이므로, 로그인이 거부되었습니다 - Oracle, 오류: ORA-01017)	DB 인스턴스에 연결할 수 있지만 연결이 거부되었습니다. 이 문제는 주로 사용자 이름이나 암호를 잘못 입력하면 발생합니다. 사용자 이름과 암호를 확인하고 다시 시도하십시오.
TNS:listener에서 현재 연결 설명자 - Oracle에 제공된 SID를 알 수 없습니다(오류: ORA-12505).	올바른 SID가 입력되었는지 확인합니다. SID는 DB 이름과 동일합니다. DB 이름은 인스턴스의 Databases(데이터베이스) 페이지에 있는 Configuration(구성) 탭에 나와 있습니다. AWS CLI를 사용하여 DB 이름도 찾을 수 있습니다. `aws rds describe-db-instances --query 'DBInstances[*].[DBInstanceIdentifier,DBName]' --output text`

연결 문제에 대한 자세한 내용은 Amazon RDS DB 인스턴스에 연결할 수 없음 단원을 참조하십시오.

sqlnet.ora 파라미터를 사용하여 연결 속성 수정

sqlnet.ora 파일에는 Oracle 데이터베이스 서버와 클라이언트에서 Oracle Net 기능을 구성하는 파라미터가 포함되어 있습니다. sqlnet.ora 파일에서 이 파라미터를 사용하면 데이터베이스 안팎으로 연결하는 속성을 수정할 수 있습니다.

sqlnet.ora 파라미터의 설정 이유에 대한 자세한 정보는 Oracle 설명서의 프로필 파라미터 구성을 참조하십시오.

sqlnet.ora 파라미터 설정

Amazon RDS for Oracle 파라미터 그룹에는 sqlnet.ora 파라미터의 하위 집합이 포함됩니다. 이것들은 다른 Oracle 파라미터와 같은 방법으로 설정해야 합니다. `sqlnetora.` 접두사는 어떤 파라미터가 sqlnet.ora 파라미터인지 식별합니다. 예를 들어 Amazon RDS의 Oracle 파라미터 그룹에서 `default_sdu_size` sqlnet.ora 파라미터는 `sqlnetora.default_sdu_size`입니다.

파라미터 그룹 관리 및 파라미터 값 설정에 대한 자세한 정보는 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.

지원되는 sqlnet.ora 파라미터

Amazon RDS에서는 다음 sqlnet.ora 파라미터를 지원합니다. 동적 sqlnet.ora 파라미터의 변경 내용은 즉시 적용됩니다.

파라미터	유효한 값	정적/동적	설명
`sqlnetora.default_sdu_size`	512~20	동적	바이트로 표기하는 세션 데이터 단위(SDU). SDU는 버퍼에 두었다가 한 번에 네트워크로 전송하는 데이터의 양입니다.
`sqlnetora.diag_adr_enabled`	ON, OFF	동적	ADR(Automatic Diagnostic Repository) 추적을 활성화하거나 비활성화하는 값. ON은 사용하는 ADR 파일 추적을 지정합니다. OFF는 사용하는 비ADR 파일 추적을 지정합니다.
`sqlnetora.recv_buf_size`	8192~2	동적	세션 작업 수신을 위한 버퍼 공간 제한. TCP/IP, SSL을 사

파라미터	유효한 값	정적/동적	설명
			용하는 TCP/IP, SDP 프로토콜이 지원합니다.
sqlnetora.send_buf_size	8192~2	동적	세션 작업 발송을 위한 버퍼 공간 제한. TCP/IP, SSL을 사용하는 TCP/IP, SDP 프로토콜이 지원합니다.
sqlnetora.sqlnet.allowed_logon_version_client	8, 10, 11, 12	동적	클라이언트 및 클라이언트 역할을 하는 서버가 Oracle DB 인스턴스에 연결하는 데 허용되는 최소 인증 프로토콜 버전입니다.
sqlnetora.sqlnet.allowed_logon_version_server	8, 9, 10, 11, 12, 12a	동적	Oracle DB 인스턴스에 연결할 수 있는 최소 인증 프로토콜 버전입니다.
sqlnetora.sqlnet.expire_time	0~1440	동적	클라이언트-서버 연결이 활성인지 확인하기 위해 점검을 보내는 시간 간격(분).
sqlnetora.sqlnet.inbound_connect_timeout	0 또는 10~720	동적	클라이언트가 데이터베이스 서버에 연결하고 필요한 인증 정보를 제공하는 시간(초).
sqlnetora.sqlnet.outbound_connect_timeout	0 또는 10~720	동적	클라이언트가 Oracle Net 연결을 DB 인스턴스에 수립하는 시간(초).
sqlnetora.sqlnet.recv_timeout	0 또는 10~720	동적	데이터베이스 서버가 연결 수립 후 클라이언트 데이터를 기다리는 시간(초).

파라미터	유효한 값	정적/동적	설명
sqlnetora.sqlnet.send_timeout	0 또는 10~720	동적	데이터베이스 서버가 연결 수립 후 클라이언트에 작업 전송을 완료하는 시간(초).
sqlnetora.tcp.connect_timeout	0 또는 10~720	동적	클라이언트가 TCP 연결을 데이터베이스 서버에 수립하는 시간(초).
sqlnetora.trace_level_server	0, 4, 10, 16, OFF, USER, ADMIN, SUPPOR	동적	비ADR 추적의 경우, 지정된 레벨에서 서버 추적을 켜거나 끄십시오.

지원되는 각 sqlnet.ora 파라미터의 기본값은 해당 릴리스에서 Oracle Database의 기본값입니다.

sqlnet.ora 파라미터 보기

AWS Management Console, AWS CLI, SQL 클라이언트를 사용하여 sqlnet.ora 파라미터와 그 설정을 볼 수 있습니다.

콘솔을 사용하여 sqlnet.ora 파라미터 보기

파라미터 그룹에서 파라미터 보기에 대한 자세한 정보는 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.

Oracle 파라미터 그룹에서 `sqlnetora.` 접두사는 어떤 파라미터가 sqlnet.ora 파라미터인지 식별합니다.

AWS CLI를 사용하여 sqlnet.ora 파라미터 보기

Oracle 파라미터 그룹에서 구성한 sqlnet.ora 파라미터를 보려면 AWS CLI describe-db-parameters 명령을 사용합니다.

Oracle DB 인스턴스의 sqlnet.ora 파라미터를 모두 보려면 AWS CLI download-db-log-file-portion 명령을 호출합니다. DB 인스턴스 식별자와 로그 파일 이름, 출력 유형을 지정합니다.

Example

다음 코드는 mydbinstance에 대한 모든 sqlnet.ora 파라미터를 나열합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds download-db-log-file-portion \
    --db-instance-identifier mydbinstance \
    --log-file-name trace/sqlnet-parameters \
    --output text
```

Windows의 경우:

```
aws rds download-db-log-file-portion ^
    --db-instance-identifier mydbinstance ^
    --log-file-name trace/sqlnet-parameters ^
    --output text
```

SQL 클라이언트를 사용하여 sqlnet.ora 파라미터 보기

SQL 클라이언트에서 Oracle DB 인스턴스에 연결하면 다음 쿼리가 sqlnet.ora 파라미터를 나열합니다.

```
SELECT * FROM TABLE
   (rdsadmin.rds_file_util.read_text_file(
        p_directory => 'BDUMP',
        p_filename  => 'sqlnet-parameters'));
```

SQL 클라이언트에서 Oracle DB 인스턴스에 연결하는 방법에 대한 자세한 정보는 Oracle DB 인스턴스에 연결을 참조하십시오.

Oracle DB 인스턴스 연결 보안

Amazon RDS for Oracle은 SSL/TLS 암호화 연결과 Oracle NNE(기본 네트워크 암호화) 옵션을 사용하여 애플리케이션과 Oracle DB 인스턴스 간의 연결을 암호화할 수 있습니다. Oracle 기본 네트워크 암호화 옵션에 대한 자세한 내용은 Oracle 기본 네트워크 암호화 단원을 참조하십시오.

주제

- RDS for Oracle DB 인스턴스에 SSL 사용
- 새 SSL/TLS 인증서를 사용해 Oracle DB에 연결할 애플리케이션을 업데이트
- RDS for Oracle DB 인스턴스에 기본 네트워크 암호화 사용
- Amazon RDS for Oracle에 대한 Kerberos 인증 구성
- 인증서 및 Oracle Wallet을 사용하여 UTL_HTTP 액세스 구성

RDS for Oracle DB 인스턴스에 SSL 사용

SSL(Secure Sockets Layer)은 클라이언트와 서버 간의 네트워크 연결을 보호하는 데 사용되는 업계 표준 프로토콜입니다. SSL 버전 3.0 이후에 이름이 전송 계층 보안(TLS)으로 변경되었지만 여전히 이 프로토콜을 SSL로 지칭하는 경우가 많습니다. Amazon RDS는 Oracle DB 인스턴스에 SSL 암호화를 지원합니다. SSL을 사용하여 애플리케이션 클라이언트와 Oracle DB 인스턴스 간의 연결을 암호화할 수 있습니다. 모든 Oracle용 AWS 리전에서 SSL 지원 기능을 사용할 수 있습니다.

Oracle DB 인스턴스에 대해 SSL 암호화를 활성화하려면 DB 인스턴스와 연결된 옵션 그룹에 Oracle SSL 옵션을 추가합니다. Amazon RDS는 Oracle에서 요구하는 대로 SSL 연결을 위해 두 번째 포트를 사용합니다. 이로 인해 DB 인스턴스와 Oracle 클라이언트 간에 클리어 텍스트 통신과 SSL로 암호화된 통신이 동시에 발생할 수 있습니다. 예를 들어 이 포트를 클리어 텍스트 통신에 사용하여 VPC 내의 다른 리소스와 통신하면서 동일한 포트를 SSL로 암호화된 통신에 사용하여 VPC 외부의 리소스와 통신할 수 있습니다.

자세한 내용은 Oracle 보안 소켓 Layer 섹션을 참조하세요.

> **Note**
>
> 동일한 DB 인스턴스에서는 SSL과 Oracle NNE(기본 네트워크 암호화)를 모두 사용할 수 없습니다. SSL 암호화를 사용하려면 먼저 다른 연결 암호화를 모두 비활성화해야 합니다.

새 SSL/TLS 인증서를 사용해 Oracle DB에 연결할 애플리케이션을 업데이트

2023년 1월 13일부터 Amazon RDS는 보안 소켓 계층(SSL) 또는 전송 계층 보안(TLS)을 사용해 RDS DB 인스턴스에 연결하기 위한 용도의 새 인증 기관(CA) 인증서를 게시하였습니다. 아래에서 새 인증서를 사용하기 위해 애플리케이션을 업데이트하는 방법에 관한 정보를 찾으실 수 있습니다.

이 주제는 클라이언트 애플리케이션에서 SSL/TLS를 사용해 DB 인스턴스에 연결하는지 여부를 판단하는 데 도움이 됩니다.

Important

Amazon RDS for Oracle DB 인스턴스에 대한 인증서를 변경하면 데이터베이스 리스너만 다시 시작됩니다. DB 인스턴스는 다시 시작되지 않았습니다. 기존 데이터베이스 연결은 영향을 받지 않지만, 리스너가 다시 시작되는 동안 잠시 새 접속에 오류가 발생합니다.
연결 오류를 방지하려면 Oracle DB를 다시 시작하는 것이 좋습니다.

Note

SSL/TLS를 사용하여 DB 인스턴스에 연결하는 클라이언트 애플리케이션의 경우, 새 CA 인증서를 포함하도록 클라이언트 애플리케이션 트러스트 스토어를 업데이트해야 합니다.

클라이언트 애플리케이션 트러스트 스토어에서 CA 인증서를 업데이트한 후에는 DB 인스턴스에서 인증서를 교환할 수 있습니다. 이 절차를 프로덕션 환경에서 구현하기 전에 개발 또는 스테이징 환경에서 테스트해볼 것을 적극 권장합니다.

인증서 교환에 대한 자세한 내용은 SSL/TLS 인증서 교체 단원을 참조하십시오. 인증서 다운로드에 대한 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 단원을 참조하십시오. Oracle DB 인스턴스에서 SSL/TLS를 사용하는 방법에 관한 자세한 내용은 Oracle 보안 소켓 Layer 단원을 참조하십시오.

주제

- 애플리케이션이 SSL을 사용하여 연결하는지 확인
- 애플리케이션 트러스트 스토어 업데이트
- SSL 연결 설정을 위한 Java 코드 예시

애플리케이션이 SSL을 사용하여 연결하는지 확인

Oracle DB 인스턴스가 SSL 옵션이 추가된 옵션 그룹을 사용하는 경우 SSL을 사용 중일 수 있습니다. 옵션 그룹의 옵션 및 옵션 설정 표시하기의 지침을 따라서 확인합니다. SSL 옵션에 대한 자세한 내용은 Oracle 보안 소켓 Layer 단원을 참조하십시오.

리스너 로그를 확인하여 SSL 연결이 있는지 확인하십시오. 다음은 리스너 로그의 출력 샘플입니다.

```
date time * (CONNECT_DATA=(CID=(PROGRAM=program)
(HOST=host)(USER=user))(SID=sid)) *
(ADDRESS=(PROTOCOL=tcps)(HOST=host)(PORT=port)) * establish * ORCL * 0
```

PROTOCOL의 한 항목 값이 tcps인 경우, SSL 연결이 표시됩니다. 그러나 HOST가 127.0.0.1인 경우 항목을 무시할 수 있습니다. 127.0.0.1로부터의 연결은 DB 인스턴스의 로컬 관리 에이전트입니다. 이 연결은 외부 SSL 연결이 아닙니다. 따라서 리스너 로그 항목에 PROTOCOL이 tcps로, HOST가 127.0.0.1이 아닌 것으로 표시되면 SSL을 사용하여 연결하는 애플리케이션이 있는 것입니다.

리스너 로그를 확인하려면 Amazon CloudWatch Logs에 로그를 게시하면 됩니다. 자세한 내용은 Amazon CloudWatch Logs에 Oracle 로그 게시 섹션을 참조하세요.

애플리케이션 트러스트 스토어 업데이트

SSL/TLS 연결을 위해 SQL*Plus 또는 JDBC를 사용하는 애플리케이션에 대해 트러스트 스토어를 업데이트할 수 있습니다.

SQL*Plus를 위한 애플리케이션 트러스트 스토어 업데이트

SSL/TLS 연결을 위해 SQL*Plus를 사용하는 애플리케이션에 대해 트러스트 스토어를 업데이트할 수 있습니다.

> **Note**
>
> 트러스트 스토어를 업데이트할 때 새 인증서를 추가할 뿐 아니라 이전 인증서를 유지할 수도 있습니다.

SQL*Plus 애플리케이션에 대해 트러스트 스토어를 업데이트하려면

1. 모든 AWS 리전에서 작동하는 새 루트 인증서를 다운로드하고 이 파일을 ssl_wallet 디렉터리에 저장하세요.

루트 인증서 다운로드에 대한 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 단원을 참조하십시오.

2. 다음 명령을 실행하여 Oracle wallet을 업데이트합니다.

```
prompt>orapki wallet add -wallet $ORACLE_HOME/ssl_wallet -trusted_cert -cert
      $ORACLE_HOME/ssl_wallet/ssl-cert.pem -auto_login_only
```

파일 이름을 다운로드한 파일 이름으로 바꿉니다.

3. wallet이 성공적으로 업데이트되었는지 확인하려면 다음 명령을 실행하십시오.

```
prompt>orapki wallet display -wallet $ORACLE_HOME/ssl_wallet
```

출력에 다음 사항이 포함되어 있어야 합니다.

```
Trusted Certificates:
Subject: CN=Amazon RDS Root 2019 CA,OU=Amazon RDS,O=Amazon Web Services\,
 Inc.,L=Seattle,ST=Washington,C=US
```

JDBC를 위한 애플리케이션 트러스트 스토어 업데이트

SSL/TLS 연결을 위해 JDBC를 사용하는 애플리케이션에 대해 트러스트 스토어를 업데이트할 수 있습니다.

루트 인증서 다운로드에 대한 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 단원을 참조하십시오.

인증서를 가져오는 샘플 스크립트는 트러스트 스토어로 인증서를 가져오기 위한 샘플 스크립트 섹션을 참조하세요.

SSL 연결 설정을 위한 Java 코드 예시

다음은 JDBC를 사용하여 SSL 연결을 설정하는 방법을 보여 주는 코드 예입니다.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;
```

```
public class OracleSslConnectionTest {
    private static final String DB_SERVER_NAME = "<dns-name-provided-by-amazon-rds>";
    private static final Integer SSL_PORT = "<ssl-option-port-configured-in-option-
group>";
    private static final String DB_SID = "<oracle-sid>";
    private static final String DB_USER = "<user name>";
    private static final String DB_PASSWORD = "<password>";
    // This key store has only the prod root ca.
    private static final String KEY_STORE_FILE_PATH = "<file-path-to-keystore>";
    private static final String KEY_STORE_PASS = "<keystore-password>";

    public static void main(String[] args) throws SQLException {
        final Properties properties = new Properties();
        final String connectionString = String.format(
                "jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)(HOST=%s)(PORT=
%d))(CONNECT_DATA=(SID=%s)))",
                DB_SERVER_NAME, SSL_PORT, DB_SID);
        properties.put("user", DB_USER);
        properties.put("password", DB_PASSWORD);
        properties.put("oracle.jdbc.J2EE13Compliant", "true");
        properties.put("javax.net.ssl.trustStore", KEY_STORE_FILE_PATH);
        properties.put("javax.net.ssl.trustStoreType", "JKS");
        properties.put("javax.net.ssl.trustStorePassword", KEY_STORE_PASS);
        final Connection connection = DriverManager.getConnection(connectionString,
 properties);
        // If no exception, that means handshake has passed, and an SSL connection can
 be opened
    }
}
```

Important

데이터베이스 연결에서 SSL/TLS를 사용함을 확인하고 애플리케이션 트러스트 스토어를 업데이트한 후에는 데이터베이스에서 rds-ca-rsa2048-g1 인증서를 사용하도록 업데이트할 수 있습니다. 지침은 DB 인스턴스 또는 클러스터를 수정하여 CA 인증서 업데이트의 3단계를 참조하십시오.

RDS for Oracle DB 인스턴스에 기본 네트워크 암호화 사용

Oracle Database는 네트워크를 통해 데이터를 암호화하는 두 가지 방법, 즉 기본 네트워크 암호화(NNE)와 전송 계층 보안(TLS)을 제공합니다. NNE는 Oracle 전용 보안 기능인 반면 TLS는 업계 표준입니다. RDS for Oracle은 Oracle Database의 모든 버전에 대해 NNE를 지원합니다.

NNE는 TLS에 비해 다음과 같은 장점이 있습니다.

- NNE 옵션의 설정을 사용하여 클라이언트 및 서버에서 NNE를 제어할 수 있습니다.
  - `SQLNET.ALLOW_WEAK_CRYPTO_CLIENTS` 및 `SQLNET.ALLOW_WEAK_CRYPTO`
  - `SQLNET.CRYPTO_CHECKSUM_CLIENT` 및 `SQLNET.CRYPTO_CHECKSUM_SERVER`
  - `SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT` 및 `SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER`
  - `SQLNET.ENCRYPTION_CLIENT` 및 `SQLNET.ENCRYPTION_SERVER`
  - `SQLNET.ENCRYPTION_TYPES_CLIENT` 및 `SQLNET.ENCRYPTION_TYPES_SERVER`
- 대부분의 경우 클라이언트나 서버를 구성할 필요가 없습니다. 반면 TLS에서는 클라이언트와 서버를 모두 구성해야 합니다.
- 인증서가 필요하지 않습니다. TLS에서는 서버에 인증서(결국 만료됨)가 필요하며 클라이언트에는 서버 인증서를 발급한 인증 기관의 신뢰할 수 있는 루트 인증서가 필요합니다.

Oracle DB 인스턴스에 대해 NNE 암호화를 활성화하려면 DB 인스턴스와 연결된 옵션 그룹에 Oracle NNE 옵션을 추가합니다. 자세한 내용은 Oracle 기본 네트워크 암호화 단원을 참조하십시오.

> **Note**
>
> 동일한 DB 인스턴스에서 NNE와 TLS를 함께 사용할 수 없습니다.

Amazon RDS for Oracle에 대한 Kerberos 인증 구성

사용자가 Amazon RDS for Oracle DB 인스턴스에 연결할 때 Kerberos 인증을 통해 사용자를 인증할 수 있습니다. 이 구성에서, DB 인스턴스는 AWS Directory Service for Microsoft Active Directory(이)라는 AWS Managed Microsoft AD과(와) 함께 작동합니다. 사용자가 신뢰하는 도메인에 속한 RDS for Oracle DB 인스턴스를 사용하여 인증할 경우 AWS Directory Service를 사용하여 만든 디렉터리에 인증 요청이 전달됩니다.

모든 자격 증명을 동일한 디렉터리에 보관하면 시간과 노력을 절약할 수 있습니다. 여러 데이터베이스 인스턴스에 대한 자격 증명을 보관하고 관리할 수 있는 중앙 집중식 공간이 있습니다. 디렉터리는 전체 보안 프로필을 향상시킬 수도 있습니다.

리전 및 버전 사용 가능 여부

기능 가용성 및 해당 지원은 각 데이터베이스 엔진의 특정 버전 및 AWS 리전에 따라 다릅니다. Kerberos 인증을 사용하는 RDS for Oracle의 버전 및 리전 가용성에 대한 자세한 내용은 Amazon RDS에서 Kerberos 인증을 지원하는 리전 및 DB 엔진 단원을 참조하세요.

> **Note**
>
> RDS for Oracle DB 인스턴스에서 사용 중단된 DB 인스턴스 클래스에는 Kerberos 인증이 지원되지 않습니다. 자세한 내용은 RDS for Oracle DB 인스턴스 클래스 단원을 참조하십시오.

주제

- Oracle DB 인스턴스에 대해 Kerberos 설정
- 도메인에서 DB 인스턴스 관리
- Kerberos 인증을 사용하여 Oracle에 연결

Oracle DB 인스턴스에 대해 Kerberos 설정

AWS Directory Service for Microsoft Active Directory라고도 하는 AWS Managed Microsoft AD을 사용하여 Oracle DB 인스턴스의 Kerberos 인증을 설정합니다. Kerberos 인증을 설정하려면 다음 단계를 완료하십시오.

- 1단계: AWS Managed Microsoft AD를 사용하여 디렉터리 만들기
- 2단계: 신뢰 생성
- 3단계: Amazon RDS에 대한 IAM 권한 구성
- 4단계: 사용자 생성 및 구성
- 5단계: 디렉터리와 DB 인스턴스 사이에 VPC 간 트래픽 활성화
- 6단계: Oracle DB 인스턴스 생성 또는 수정
- 7단계: Kerberos 인증 Oracle 로그인 생성
- 8단계: Oracle 클라이언트 구성

> **Note**
>
> 설치하는 동안 RDS는 `CREATE SESSION` 권한이 있는 *managed_service_user*@*example.com*이라는 Oracle 데이터베이스 사용자를 생성합니다. 여기서 *example.com*은 도메인 이름입니다. 이 사용자는 Directory Service가 Managed Active Directory 내에 생성하는 사용자에 해당합니다. 정기적으로, RDS는 Directory Service에서 제공하는 자격 증명을 사용하여 Oracle 데이터베이스에 로그인합니다. 그 후, RDS는 티켓 캐시를 즉시 삭제합니다.

1단계: AWS Managed Microsoft AD를 사용하여 디렉터리 만들기

AWS Directory Service는 AWS 클라우드에서 완전 관리형 Microsoft Active Directory를 생성합니다. AWS Managed Microsoft AD 디렉터리를 생성할 때 AWS Directory Service에서 두 개의 도메인 컨트롤러 및 Domain Name System(DNS) 서버가 자동으로 생성됩니다. 디렉터리 서버는 VPC 내 다른 서브넷에서 생성됩니다. 이러한 중복으로 인해 장애가 발생해도 디렉터리에 액세스할 수 있습니다.

AWS Managed Microsoft AD 디렉터리를 생성하는 경우 AWS Directory Service에서 다음 작업이 자동으로 수행됩니다.

- VPC 내에서 Active Directory를 설정합니다.
- 사용자 이름 Admin과 지정된 암호를 사용하여 디렉터리 관리자 계정을 생성합니다. 이 계정을 사용하여 디렉터리를 관리할 수 있습니다.

  > **Note**
  >
  > 이 암호를 저장하십시오. AWS Directory Service에서는 저장되지 않습니다. 재설정은 가능하지만 검색은 불가능합니다.

- 디렉터리 컨트롤러에 대한 보안 그룹을 만듭니다.

AWS Managed Microsoft AD를 시작하면 AWS에서 모든 디렉터리의 객체를 포함하는 OU(조직 단위)를 생성합니다. 이 OU는 디렉터리를 생성할 때 입력한 NetBIOS 이름을 가지고 있으며, 도메인 루트에 위치합니다. 도메인 루트는 AWS에서 소유하고 관리합니다.

AWS Managed Microsoft AD 디렉터리를 사용해 생성한 관리자 계정은 OU의 가장 일반적인 관리 활동에 대한 권한을 갖습니다.

- 사용자 생성, 업데이트 또는 삭제

- 도메인(예: 파일 또는 인쇄 서버)에 리소스를 추가한 다음 OU 내의 사용자에 해당 리소스에 대한 권한 할당
- 추가 OU 및 컨테이너 생성
- 권한 위임
- Active Directory 휴지통에서 삭제된 객체 복원
- Active Directory 웹 서비스에서 AD 및 DNS Windows PowerShell 모듈 실행

또한 admin 계정은 다음과 같은 도메인 차원 활동을 수행할 권한이 있습니다.

- DNS 구성 관리(레코드, 영역 및 전달자 추가, 제거 또는 업데이트)
- DNS 이벤트 로그 보기
- 보안 이벤트 로그 보기

디렉터리를 생성하려면 AWS Management Console, AWS CLI 또는 AWS Directory Service API를 사용합니다. 디렉터리가 Oracle DB 인스턴스와 통신할 수 있도록 디렉터리 보안 그룹에서 관련 아웃바운드 포트를 열어야 합니다.

AWS Managed Microsoft AD으로 디렉터리를 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/directoryservicev2/에서 AWS Directory Service 콘솔을 엽니다.
2. 탐색 창에서 디렉터리를 선택한 후 디렉터리 설정을 선택합니다.
3. AWS Managed Microsoft AD를 선택합니다. AWS Managed Microsoft AD는 현재 Amazon RDS와 함께 사용할 수 있는 유일한 옵션입니다.
4. 다음 정보를 입력합니다.

   디렉터리 DNS 이름

   디렉터리를 위한 정규화된 이름(예: **`corp.example.com`**)입니다.

   디렉터리 NetBIOS 이름

   디렉터리의 짧은 이름(예: **`CORP`**)입니다.

   디렉터리 설명

   (선택 사항) 디렉터리에 대한 설명입니다.

관리자 암호

디렉터리 관리자의 암호입니다. 디렉터리 생성 프로세스에서는 사용자 이름 Admin과 이 암호를 사용하여 관리자 계정을 생성합니다.

디렉터리 관리자 암호는 "admin"이라는 단어를 포함할 수 없습니다. 암호는 대소문자를 구분하며 길이가 8~64자 사이여야 합니다. 또한 다음 네 범주 중 세 개에 해당하는 문자를 1자 이상 포함해야 합니다.

- 소문자(a–z)
- 대문자(A–Z)
- 숫자(0–9)
- 영숫자 외의 특수 문자(~!@#$%^&*_-+=`|\(){}[]:;"'<>,.?/)

[Confirm Password]

관리자 암호를 다시 입력했습니다.

5. 다음을 선택합니다.
6. 네트워킹 섹션에 다음 정보를 입력하고 다음을 선택합니다.

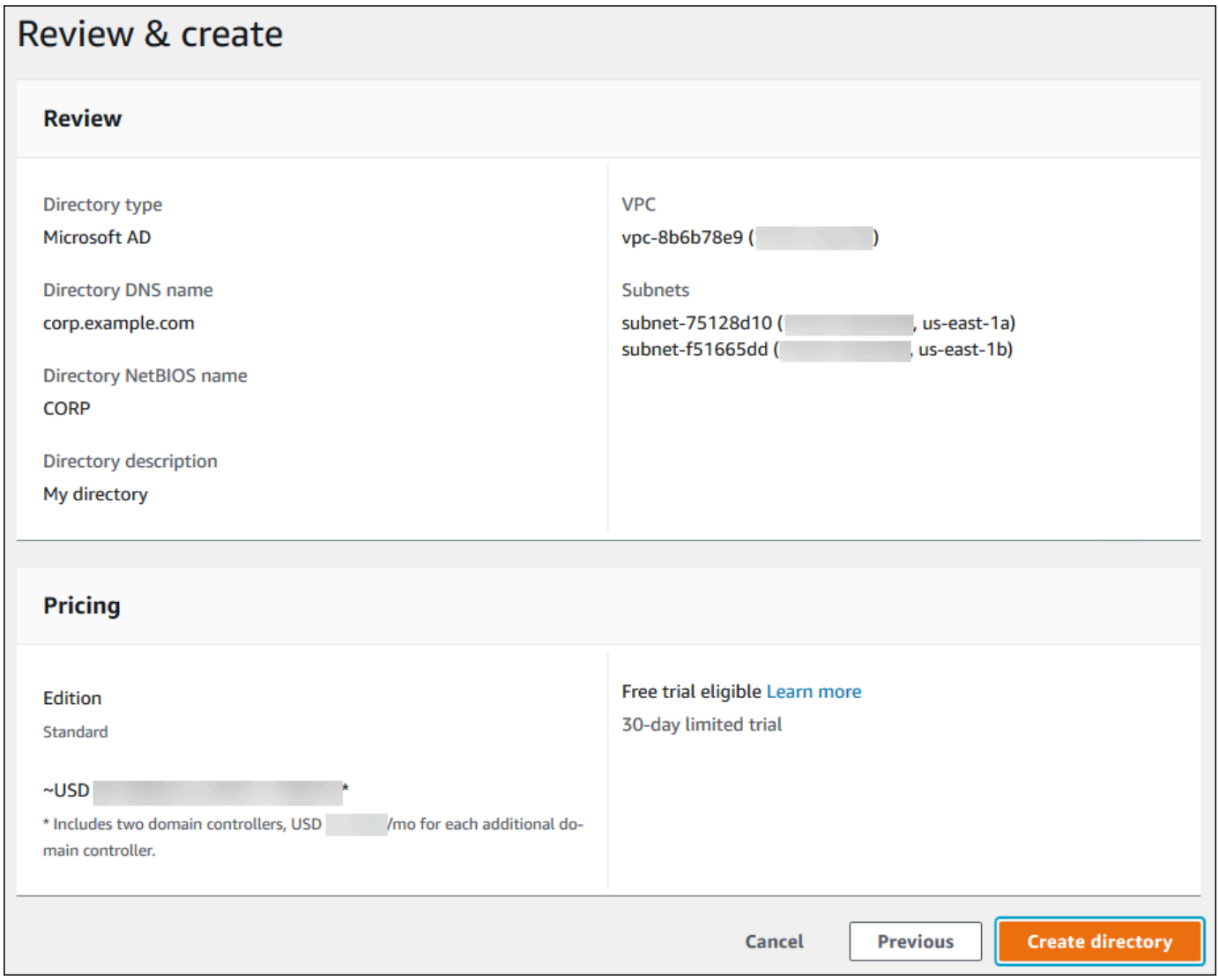
VPC

디렉터리에 대한 VPC입니다. 동일한 VPC에 Oracle DB 인스턴스를 생성합니다.

서브넷

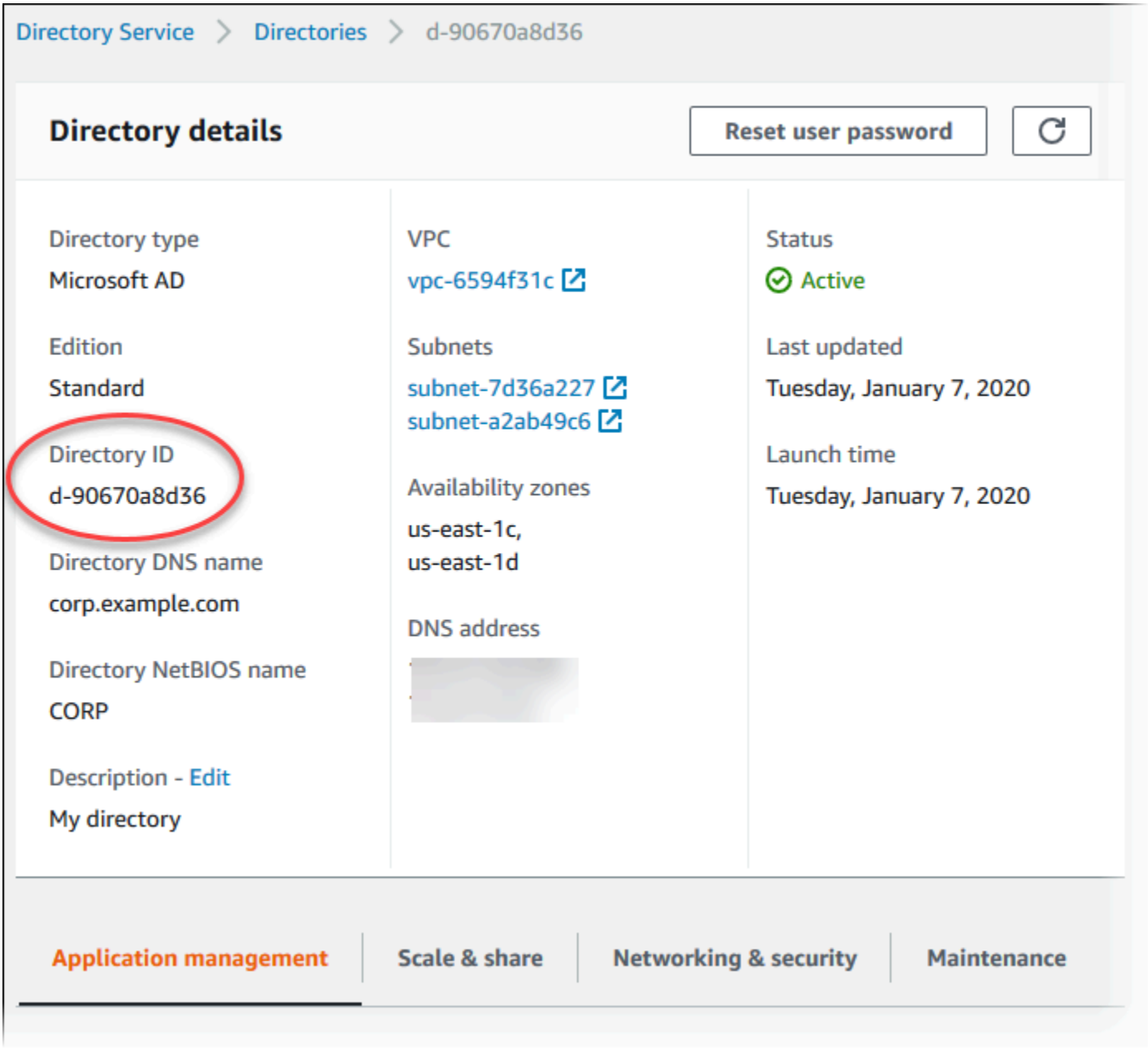
디렉터리 서버에 대한 서브넷입니다. 두 서브넷이 서로 다른 가용 영역에 있어야 합니다.

7. 디렉터리 정보를 검토하고 필요한 사항을 변경합니다. 정보가 올바르면 디렉터리 생성을 선택합니다.



디렉터리를 생성하는 데 몇 분 정도 걸립니다. 디렉터리가 성공적으로 생성되면 상태 값이 활성으로 변경됩니다.

디렉터리에 대한 정보를 보려면 디렉터리 목록에서 해당 디렉터리 이름을 선택합니다. Oracle DB 인스턴스를 생성하거나 수정할 때 이 값이 필요하므로 디렉터리 ID 값을 기록해 두십시오.



2단계: 신뢰 생성

AWS Managed Microsoft AD만 사용하려는 경우 3단계: Amazon RDS에 대한 IAM 권한 구성 섹션으로 넘어갑니다.

자체 관리형 Active Directory를 사용하여 Kerberos 인증을 활성화하려면 자체 관리형 Active Directory와 이전 단계에서 생성된 AWS Managed Microsoft AD 간에 포리스트 신뢰 관계를 생성해야 합니다. 트러스트는 AWS Managed Microsoft AD가 자체 관리형 Active Directory를 신뢰하는 단방향일 수 있습니다. 두 Active Directory가 서로를 신뢰하는 양방향일 수도 있습니다. AWS Directory Service를 사용

하여 포리스트 신뢰를 설정하는 방법에 대한 자세한 내용은 AWS Directory Service 관리 안내서의 신뢰 관계를 생성해야 하는 경우를 참조하세요.

3단계: Amazon RDS에 대한 IAM 권한 구성

AWS Directory Service를 호출하려면 Amazon RDS에 관리형 IAM 정책 `AmazonRDSDirectoryServiceAccess`를 사용하는 IAM 역할이 필요합니다. 이 역할을 사용하여 Amazon RDS에서 AWS Directory Service를 호출할 수 있습니다.

> **Note**
>
> 역할이 액세스를 허용하려면 AWS 리전에서 AWS 계정의 AWS Security Token Service(AWS STS) 엔드포인트를 활성화해야 합니다. AWS STS 엔드포인트는 기본적으로 모든 AWS 리전에서 활성화되어 있으므로 별도의 조치 없이 사용할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 리전에서 AWS STS 활성화 및 비활성화를 참조하세요.

IAM 역할 생성

AWS Management Console을 사용하여 DB 인스턴스를 생성할 때 콘솔 사용자에게 `iam:CreateRole` 권한이 있으면 콘솔에서 `rds-directoryservice-kerberos-access-role`을 자동으로 생성합니다. 그렇지 않으면 IAM 역할을 수동으로 생성해야 합니다. 이 IAM 역할을 생성할 때 `Directory Service`를 선택하고 여기에 AWS 관리형 정책인 `AmazonRDSDirectoryServiceAccess`를 연결합니다.

서비스에 대한 IAM 역할 생성에 대한 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.

> **Note**
>
> RDS for Microsoft SQL Server에 대한 Windows 인증에 사용되는 IAM 역할은 RDS for Oracle에 사용할 수 없습니다.

수동으로 IAM 신뢰 정책 만들기

선택 사항으로 관리형 IAM 정책인 `AmazonRDSDirectoryServiceAccess`를 사용하는 대신 필요한 권한으로 리소스 정책을 생성할 수 있습니다. `directoryservice.rds.amazonaws.com` 및 `rds.amazonaws.com` 둘 다 보안 주체로 지정합니다.

Amazon RDS가 다른 서비스에 제공하는 리소스에 대한 권한을 제한하려면 리소스 정책에서 aws:SourceArn 및 aws:SourceAccount 전역 조건 컨텍스트 키를 사용하는 것이 좋습니다. 혼동된 대리자 문제로부터 보호하는 가장 효과적인 방법은 Amazon RDS 리소스의 전체 ARN이 포함된 aws:SourceArn 글로벌 조건 컨텍스트 키를 사용하는 것입니다. 자세한 내용은 교차 서비스 혼동된 대리자 문제 방지 섹션을 참조하세요.

다음 예에서는 Amazon RDS에서 aws:SourceArn 및 aws:SourceAccount 전역 조건 컨텍스트 키를 사용하여 혼동된 대리자 문제를 방지하는 방법을 보여줍니다.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.amazonaws.com",
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:rds:us-east-1:123456789012:db:mydbinstance"
        },
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

또한 옵트인 리전의 경우 directoryservice.rds.*region_name*.amazonaws.com의 양식으로 해당 리전의 서비스 보안 주체를 포함해야 합니다. 예를 들어 아프리카(케이프타운) 리전에서 다음의 신뢰 정책을 사용합니다.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.amazonaws.com",
          "directoryservice.rds.af-south-1.amazonaws.com",
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:rds:af-south-1:123456789012:db:mydbinstance"
        },
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

또한 역할에는 다음과 같은 IAM 정책도 있어야 합니다.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ds:DescribeDirectories",
        "ds:AuthorizeApplication",
        "ds:UnauthorizeApplication",
        "ds:GetAuthorizedApplicationDetails"
```

```
      ],
    "Effect": "Allow",
    "Resource": "*"
    }
  ]
}
```

4단계: 사용자 생성 및 구성

Active Directory 도메인 서비스 및 Active Directory Lightweight Directory Services 도구의 일부인 Active Directory 사용자 및 컴퓨터 도구를 사용하여 사용자를 생성할 수 있습니다. 이 경우 사용자는 디렉터리에 액세스할 수 있는 개별 사용자 또는 개체입니다.

AWS Directory Service 디렉터리에서 사용자를 생성하려면 AWS Directory Service 디렉터리의 멤버인 Windows 기반 Amazon EC2 인스턴스에 연결되어 있어야 합니다. 이와 동시에 사용자를 생성할 권한이 있는 사용자로 로그인한 상태이어야 합니다. Microsoft Active Directory에서 사용자를 생성하는 방법에 대한 자세한 내용은 AWS Managed Microsoft AD 관리 안내서의 AWS Directory Service에서 사용자 및 그룹 관리를 참조하세요.

5단계: 디렉터리와 DB 인스턴스 사이에 VPC 간 트래픽 활성화

디렉터리와 DB 인스턴스를 동일한 VPC에 배치하려면 이 단계를 건너뛰고 6단계: Oracle DB 인스턴스 생성 또는 수정 섹션으로 이동하세요.

디렉터리와 DB 인스턴스를 서로 다른 AWS 계정 또는 VPC에 배치하려면 VPC 피어링 또는 AWS Transit Gateway를 사용하여 VPC 간 트래픽을 구성합니다. 다음 절차는 VPC 피어링을 사용하여 VPC 간 트래픽을 활성화합니다. Amazon Virtual Private Cloud 피어링 안내서의 VPC 피어링이란? 지침을 따르십시오.

VPC 피어링을 사용하여 VPC 간 트래픽을 활성화하려면

1. 네트워크 트래픽이 양방향으로 흐를 수 있도록 적절한 VPC 라우팅 규칙을 설정합니다.
2. DB 인스턴스의 보안 그룹이 디렉터리의 보안 그룹에서 인바운드 트래픽을 수신할 수 있는지 확인합니다. 자세한 내용은 AWS Managed Microsoft AD 관리 안내서에서 AWS Directory Service 모범 사례를 참조하세요.
3. 트래픽을 차단하는 네트워크 ACL(액세스 제어 목록) 규칙이 없어야 합니다.

다른 AWS 계정이 디렉터리를 소유하는 경우 디렉터리를 공유해야 합니다.

AWS 계정 간에 디렉터리를 공유하려면

1. AWS 관리 안내서의 자습서: 원활한 EC2 도메인 조인을 위해 AWS Managed Microsoft AD 디렉터리 공유에 있는 지침에 따라 DB 인스턴스가 생성될 AWS Directory Service 계정과 디렉터리를 공유하는 작업을 시작합니다.
2. DB 인스턴스용 계정을 사용하여 AWS Directory Service 콘솔에 로그인하고 계속하기 전에 도메인이 SHARED 상태가 되었는지 확인합니다.
3. DB 인스턴스용 계정을 사용하여 AWS Directory Service 콘솔에 로그인하는 동안 디렉터리 ID 값을 기록해 둡니다. 이 디렉터리 ID를 사용하여 DB 인스턴스를 도메인에 조인합니다.

6단계: Oracle DB 인스턴스 생성 또는 수정

디렉터리에서 사용할 Oracle DB 인스턴스를 생성하거나 수정합니다. 콘솔, CLI 또는 RDS API를 사용하여 DB 인스턴스를 디렉터리에 연결할 수 있습니다. 이 작업을 다음 중 한 가지 방법으로 수행할 수 있습니다.

- 콘솔, create-db-instance CLI 명령 또는 CreateDBInstance RDS API 작업을 사용하여 새 Oracle DB 인스턴스를 생성합니다.

  지침은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
- 콘솔, modify-db-instance CLI 명령 또는 ModifyDBInstance RDS API 작업을 사용하여 기존 Oracle DB 인스턴스를 수정합니다.

  지침은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.
- 콘솔, restore-db-instance-from-db-snapshot CLI 명령 또는 RestoreDBInstanceFromDBSnapshot RDS API 작업을 사용하여 DB 스냅샷에서 Oracle DB 인스턴스를 복원합니다.

  지침은 DB 인스턴스 복원 섹션을 참조하세요.
- 콘솔, restore-db-instance-to-point-in-time CLI 명령 또는 RestoreDBInstanceToPointInTime RDS API 작업을 사용하여 Oracle DB 인스턴스를 특정 시점으로 복구합니다.

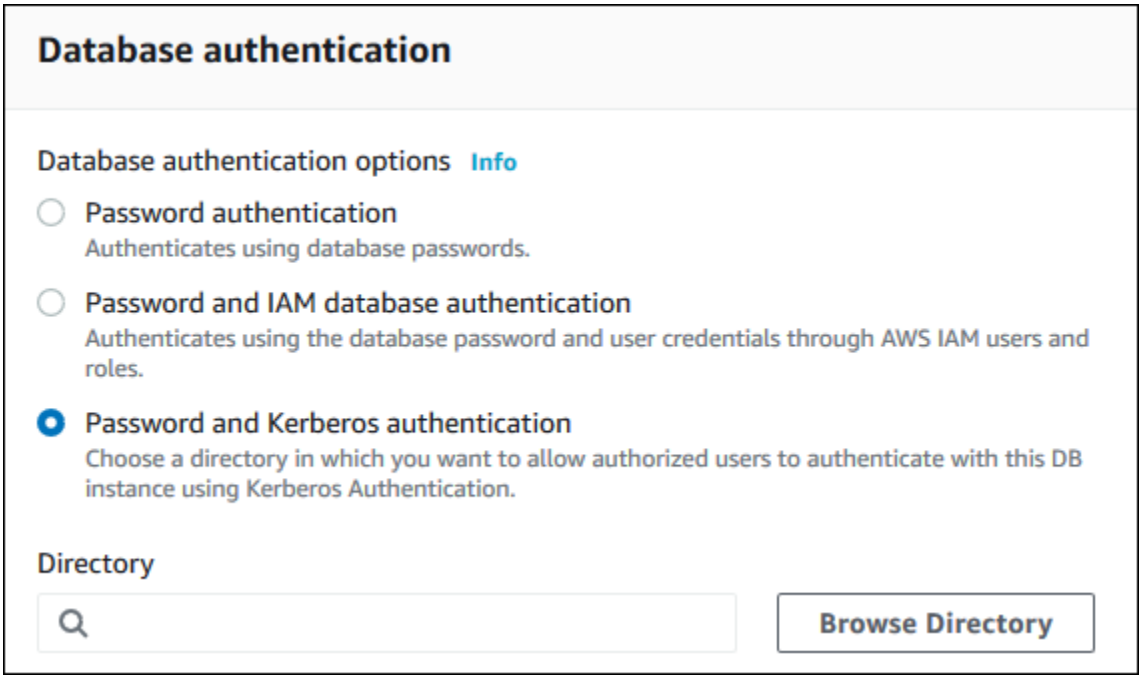
  지침은 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 섹션을 참조하세요.

Kerberos 인증은 VPC의 Oracle DB 인스턴스에 대해서만 지원됩니다. DB 인스턴스는 디렉터리와 동일한 VPC 또는 다른 VPC에 있을 수 있습니다. DB 인스턴스를 생성하거나 수정할 때 다음을 수행합니다.

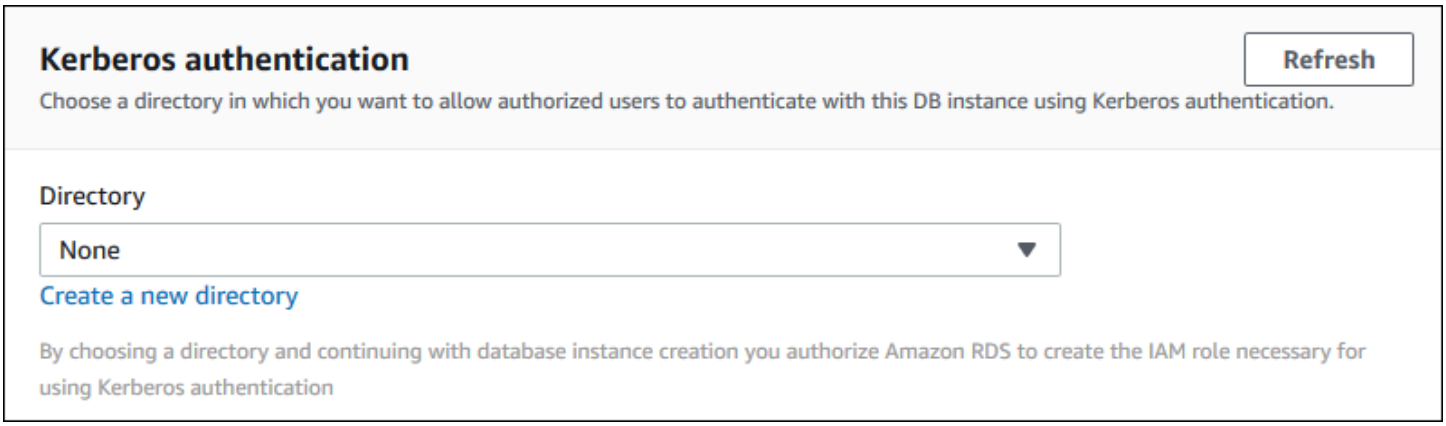
- 디렉터리를 만들 때 생성된 도메인 식별자(d-* 식별자)를 제공합니다.

- 생성한 IAM 역할의 이름을 제공합니다.
- DB 인스턴스 보안 그룹이 디렉터리 보안 그룹에서 인바운드 트래픽을 수신하고 디렉터리로 아웃바운드 트래픽을 전송할 수 있는지 확인합니다.

콘솔을 사용하여 DB 인스턴스를 생성하는 경우 Database authentication(데이터베이스 인증) 섹션에서 Password and Kerberos authentication(암호 및 Kerberos 인증)을 선택합니다. Browse Directory(디렉터리 찾아보기)를 선택한 다음 디렉터리를 선택하거나 새 디렉터리 생성을 선택합니다.



콘솔을 사용하여 DB 인스턴스를 수정하거나 복원할 때는 Kerberos authentication(Kerberos 인증) 섹션에서 디렉터리를 선택하거나 새 디렉터리 생성을 선택합니다.



AWS CLI를 사용하는 경우 생성한 도메인 디렉터리를 DB 인스턴스에서 사용하려면 다음과 같은 파라미터가 필요합니다.

- `--domain` 파라미터의 경우 디렉터리를 만들 때 생성된 도메인 식별자("d-*" 식별자)를 사용하십시오.
- `--domain-iam-role-name` 파라미터의 경우 귀하가 생성한, 관리형 IAM 정책 `AmazonRDSDirectoryServiceAccess`를 사용하는 역할을 사용하십시오.

예를 들어 다음 CLI 명령에서는 디렉터리를 사용하도록 DB 인스턴스를 수정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --domain d-ID \
    --domain-iam-role-name role-name
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --domain d-ID ^
    --domain-iam-role-name role-name
```

Important

DB 인스턴스를 수정하여 Kerberos 인증을 활성화하는 경우에는 변경 후 DB 인스턴스를 재부팅하세요.

Note

*MANAGED_SERVICE_USER*는 이름이 Directory Service for RDS에서 임의로 생성되는 서비스 계정입니다. Kerberos 인증을 설정하는 동안, RDS for Oracle은 동일한 이름을 가진 사용자를 생성하고 해당 사용자에게 `CREATE SESSION` 권한을 할당합니다. Oracle DB 사용자는 외부에서 *MANAGED_SERVICE_USER@EXAMPLE.COM*으로 식별됩니다. 여기서 *EXAMPLE.COM*은 도메인의 이름입니다. 정기적으로, RDS는 Directory Service에서 제공하는 자격 증명을 사용하여 Oracle 데이터베이스에 로그인합니다. 그 후, RDS는 티켓 캐시를 즉시 삭제합니다.

7단계: Kerberos 인증 Oracle 로그인 생성

다른 DB 인스턴스의 경우와 같은 방법으로 Amazon RDS 마스터 사용자 자격 증명을 사용하여 Oracle DB 인스턴스에 연결합니다. DB 인스턴스는 AWS Managed Microsoft AD 도메인에 조인됩니다. 따라서 도메인 내 Microsoft Active Directory 사용자에서 Oracle 로그인 및 사용자를 프로비저닝할 수 있습니다. 데이터베이스 권한을 관리하려면 이 로그인에 대해 표준 Oracle 권한을 부여하고 취소하십시오.

Microsoft Active Directory 사용자가 Oracle로 인증할 수 있게 허용하는 방법

1. Amazon RDS 마스터 사용자 자격 증명을 사용하여 Oracle DB 인스턴스에 연결합니다.
2. Oracle 데이터베이스에서 외부에서 인증된 사용자를 생성합니다.

   다음 예제에서는 *KRBUSER@CORP.EXAMPLE.COM*을(를) 사용자 이름 및 도메인 이름으로 교체합니다.

   ```
   CREATE USER "KRBUSER@CORP.EXAMPLE.COM" IDENTIFIED EXTERNALLY;
   GRANT CREATE SESSION TO "KRBUSER@CORP.EXAMPLE.COM";
   ```

   이제 도메인의 사용자(사람 및 애플리케이션)는 Kerberos 인증을 사용하여 도메인이 조인된 클라이언트 머신에서 Oracle DB 인스턴스에 연결할 수 있습니다.

8단계: Oracle 클라이언트 구성

Oracle 클라이언트를 구성하려면 다음 요구 사항을 충족하십시오.

- 도메인을 가리키도록 krb5.conf(Linux) 또는 krb5.ini(Windows)라는 구성 파일을 만듭니다. 이 구성 파일을 사용하도록 Oracle 클라이언트를 구성합니다.
- TCP/UDP를 사용하여 DNS 포트 53, TCP를 사용하여 Kerberos 포트(관리형 AWS Directory Service의 경우 88 및 464) 및 TCP를 사용하여 LDAP 포트 389를 통해 클라이언트 호스트와 AWS Directory Service 간에 트래픽이 흐를 수 있는지 확인합니다.
- 데이터베이스 포트를 통해 클라이언트 호스트와 DB 인스턴스 간에 트래픽이 흐를 수 있는지 확인합니다.

다음은 AWS Managed Microsoft AD의 샘플 콘텐츠입니다.

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
```

```
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
[domain_realm]
 .example.com = CORP.EXAMPLE.COM
 example.com = CORP.EXAMPLE.COM
```

다음은 온프레미스 Microsoft AD의 샘플 krb5.conf 콘텐츠입니다. krb5.conf 또는 krb5.ini 파일에서 *on-prem-ad-server-name*을 온프레미스 AD 서버의 이름으로 바꿉니다.

```
[libdefaults]
 default_realm = ONPREM.COM
[realms]
 AWSAD.COM = {
  kdc = awsad.com
  admin_server = awsad.com
 }
 ONPREM.COM = {
  kdc = on-prem-ad-server-name
  admin_server = on-prem-ad-server-name
 }
[domain_realm]
 .awsad.com = AWSAD.COM
 awsad.com= AWSAD.COM
 .onprem.com = ONPREM.COM
 onprem.com= ONPREM.COM
```

> **Note**
>
> krb5.ini 또는 krb5.conf 파일을 구성한 후에는 서버를 재부팅하는 것이 좋습니다.

다음은 SQL*Plus 구성에 대한 샘플 sqlnet.ora 콘텐츠입니다.

```
SQLNET.AUTHENTICATION_SERVICES=(KERBEROS5PRE,KERBEROS5)
SQLNET.KERBEROS5_CONF=path_to_krb5.conf_file
```

SQL Developer 구성의 예는 Oracle Support의 문서 1609359.1을 참조하세요.

도메인에서 DB 인스턴스 관리

콘솔, CLI 또는 RDS API를 사용하여 DB 인스턴스 및 DB 인스턴스와 Microsoft Active Directory의 관계를 관리할 수 있습니다. 예를 들어 Microsoft Active Directory를 연결하여 Kerberos 인증을 활성화할 수 있습니다. 또한 Microsoft Active Directory를 연결 해제하여 Kerberos 인증을 비활성화할 수 있습니다. 또한 DB 인스턴스를 이동하여 한 Microsoft Active Directory에서 다른 Microsoft Active Directory로 외부 인증하게 할 수 있습니다.

예를 들어 CLI를 사용하여 다음 작업을 수행할 수 있습니다.

- 실패한 멤버십에 대한 Kerberos 인증 활성화를 다시 시도하려면 modify-db-instance CLI 명령을 사용하여 `--domain` 옵션에 현재 멤버십의 디렉터리 ID를 지정합니다.
- DB 인스턴스에서 Kerberos 인증을 비활성화화려면 modify-db-instance CLI 명령을 사용하여 none 옵션에 대해 `--domain`을 지정합니다.
- 한 도메인에서 다른 도메인으로 DB 인스턴스를 이동하려면 modify-db-instance CLI 명령을 사용하여 `--domain` 옵션에 대해 새 도메인의 도메인 식별자를 지정합니다.

도메인 멤버십 상태 보기

DB 인스턴스를 생성하거나 수정하면 해당 DB 인스턴스는 도메인의 멤버가 됩니다. 콘솔에서 또는 describe-db-instances CLI 명령을 실행하여 DB 인스턴스에 대한 도메인 멤버십의 상태를 확인할 수 있습니다. DB 인스턴스의 상태는 다음 중 한 가지가 될 수 있습니다.

- `kerberos-enabled` - DB 인스턴스에 Kerberos 인증이 활성화되어 있습니다.
- `enabling-kerberos` - AWS에서 이 DB 인스턴스에 대한 Kerberos 인증 활성화를 진행 중입니다.
- `pending-enable-kerberos` - 이 DB 인스턴스에 대한 Kerberos 인증 활성화가 보류 중입니다.
- `pending-maintenance-enable-kerberos` - AWS에서 예약된 다음 유지 관리 기간에 DB 인스턴스에 대한 Kerberos 인증을 활성화하려 합니다.
- `pending-disable-kerberos` - 이 DB 인스턴스에 대한 Kerberos 인증 비활성화가 보류 중입니다.
- `pending-maintenance-disable-kerberos` - AWS에서 예약된 다음 유지 관리 기간에 DB 인스턴스에 대한 Kerberos 인증을 비활성화하려 합니다.
- `enable-kerberos-failed` - 구성 문제로 인해 AWS에서 DB 인스턴스에 대해 Kerberos 인증을 활성화하지 못했습니다. DB 인스턴스 수정 명령을 다시 실행하기 전에 구성 문제를 해결하십시오.
- `disabling-kerberos` - AWS에서 이 DB 인스턴스에 대한 Kerberos 인증 비활성화를 진행 중입니다.

네트워크 연결 문제 또는 잘못된 IAM 역할로 인해 Kerberos 인증 활성화 요청이 실패할 수 있습니다. DB 인스턴스를 생성하거나 수정할 때 Kerberos 인증 활성화 시도가 실패하면 올바른 IAM 역할을 사용하고 있는지 확인하세요. 그런 다음 DB 인스턴스를 수정하여 도메인에 조인하게 합니다.

> **Note**
>
> Amazon RDS for Oracle을 사용한 Kerberos 인증에서만 트래픽을 도메인의 DNS 서버로 전송합니다. 다른 모든 DNS 요청은 Oracle을 실행 중인 DB 인스턴스에서 아웃바운드 네트워크 액세스로 취급됩니다. Amazon RDS for Oracle을 사용한 아웃바운드 네트워크 액세스에 대한 자세한 내용은 사용자 지정 DNS 서버 설정 단원을 참조하십시오.

Kerberos 키 강제 교체

비밀 키는 AWS Managed Microsoft AD와 Amazon RDS for Oracle DB 인스턴스 간에 공유됩니다. 이 키는 45일마다 자동으로 교체됩니다. 다음 Amazon RDS 절차를 사용하여 이 키를 강제로 회전할 수 있습니다.

```
SELECT rdsadmin.rdsadmin_kerberos_auth_tasks.rotate_kerberos_keytab AS TASK_ID FROM
 DUAL;
```

> **Note**
>
> 읽기 전용 복제본 구성에서 이 절차는 읽기 전용 복제본이 아닌 원본 DB 인스턴스에서만 사용할 수 있습니다.

SELECT 문은 VARCHAR2 데이터 형식으로 작업 ID를 반환합니다. 진행 중인 작업의 상태를 bdump 파일에서 볼 수 있습니다. 이 bdump 파일은 `/rdsdbdata/log/trace` 디렉터리에 위치합니다. 각 bdump 파일 이름은 다음 형식으로 되어 있습니다.

```
dbtask-task-id.log
```

작업의 출력 파일을 표시하여 결과를 볼 수 있습니다.

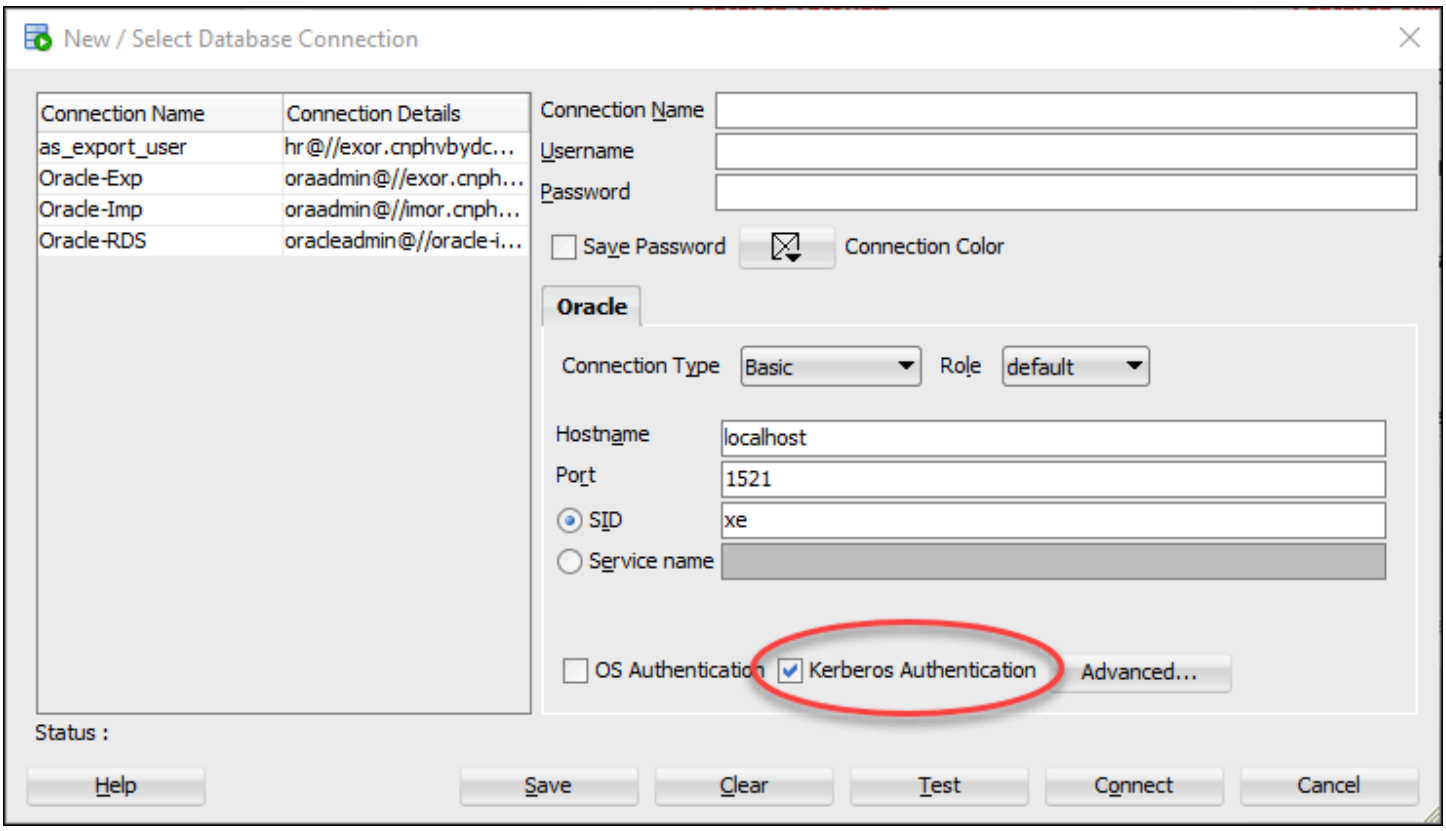
```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-task-
id.log'));
```

*task-id*를 절차에서 반환된 작업 ID로 대체합니다.

> **Note**
>
> 작업은 비동기식으로 실행됩니다.

Kerberos 인증을 사용하여 Oracle에 연결

이 섹션에서는 8단계: Oracle 클라이언트 구성에 설명된 대로 Oracle 클라이언트를 설정했다고 가정합니다. Kerberos 인증을 사용하여 Oracle DB에 연결하려면 Kerberos 인증 유형을 사용하여 로그인합니다. 예를 들어, 다음 예제에서와 같이 Oracle SQL Developer를 시작한 후 Kerberos 인증을 인증 유형으로 선택합니다.



SQL*Plus에서 Kerberos 인증을 사용하여 Oracle에 연결하려면

1. 명령 프롬프트에서 다음 명령을 실행합니다.

```
kinit username
```

   *username*을 사용자 이름으로 대체하고 프롬프트에서 Microsoft Active Directory에 저장된 사용자 암호를 입력합니다.

2. SQL*Plus를 열고 Oracle DB 인스턴스의 DNS 이름 및 포트 번호를 사용하여 연결합니다.

   SQL*Plus에서 Oracle DB 인스턴스에 연결하는 방법에 대한 자세한 내용은 SQL*Plus를 사용하여 DB 인스턴스에 연결 단원을 참조하십시오.

> **Tip**
>
> 네이티브 Windows 캐시를 사용하는 경우 sqlnet.ora 파일에서 `SQLNET.KERBEROS5_CC_NAME` 파라미터를 `OSMSFT://` 또는 `MSLSA`로 설정하여 Microsoft Active Directory에 저장된 자격 증명을 사용할 수도 있습니다.

인증서 및 Oracle Wallet을 사용하여 UTL_HTTP 액세스 구성

Amazon RDS는 RDS for Oracle DB 인스턴스에서 아웃바운드 네트워크 액세스를 지원합니다. DB 인스턴스를 네트워크에 연결하기 위해 다음 PL/SQL 패키지를 사용할 수 있습니다.

`UTL_HTTP`

이 패키지는 SQL 및 PL/SQL에서 HTTP를 호출합니다. HTTP를 통해 인터넷의 데이터에 액세스하는 데 사용할 수 있습니다. 자세한 내용은 Oracle 설명서의 UTL_HTTP를 참조하세요.

`UTL_TCP`

이 패키지는 PL/SQL에서 TCP/IP 클라이언트 측 액세스 기능을 제공합니다. 이 패키지는 인터넷 프로토콜과 이메일을 사용하는 PL/SQL 애플리케이션에 유용합니다. 자세한 내용은 Oracle 설명서의 UTL_TCP를 참조하세요.

`UTL_SMTP`

이 패키지는 클라이언트가 SMTP 서버로 이메일을 보낼 수 있도록 SMTP 명령에 대한 인터페이스를 제공합니다. 자세한 내용은 Oracle 설명서의 UTL_SMTP를 참조하세요.

다음 태스크를 완료하면 SSL 핸드셰이크 중에 클라이언트 인증 인증서가 필요한 웹 사이트에서 작동하도록 `UTL_HTTP.REQUEST`를 구성할 수 있습니다. Oracle Wallet 생성 명령과 `DBMS_NETWORK_ACL_ADMIN.APPEND_WALLET_ACE` 절차를 수정하여 웹 사이트에 대한 `UTL_HTTP` 액세스의 암호 인증을 구성할 수도 있습니다. 자세한 내용은 Oracle Database 설명서의 DBMS_NETWORK_ACL_ADMIN을 참조하세요.

> **Note**
>
> SSL/TLS(Amazon Simple Email Service 포함)를 통해 이메일을 보낼 수 있도록 하는 UTL_SMTP에 대해 다음 태스크를 적용할 수 있습니다.

주제

- UTL_HTTP 액세스 구성 구성 시 고려 사항
- 1단계: 웹 사이트의 루트 인증서 가져오기
- 2단계: Oracle Wallet 생성
- 3단계: RDS for Oracle 인스턴스로 Oracle Wallet 다운로드
- 4단계: Oracle Wallet에 대한 사용자 권한 부여
- 5단계: DB 인스턴스에서 웹 사이트에 대한 액세스 구성
- 6단계: DB 인스턴스에서 웹 사이트로의 연결 테스트

UTL_HTTP 액세스 구성 구성 시 고려 사항

액세스를 구성하기 전에 다음 사항을 고려하세요.

- SMTP를 UTL_MAIL 옵션으로 사용할 수 있습니다. 자세한 내용은 Oracle UTL_MAIL 단원을 참조하십시오.
- 원격 호스트의 DNS(Domain Name Server) 이름은 다음 중 어느 것이나 될 수 있습니다.
  - 공개적으로 확인할 수 있어야 함.
  - Amazon RDS DB 인스턴스의 엔드포인트.
  - 사용자 지정 DNS 서버를 통해 확인 가능 자세한 내용은 사용자 지정 DNS 서버 설정 섹션을 참조하세요.
  - 동일한 VPC 또는 피어링된 VPC에 있는 Amazon EC2 인스턴스의 프라이빗 DNS 이름. 이 경우, 사용자 지정 DNS 서버를 통해 이름을 확인할 수 있어야 합니다. 또는 Amazon이 제공하는 DNS를 사용하기 위해 VPC 설정에서 `enableDnsSupport` 속성을 활성화하고 VPC 피어링 연결에 대한 DNS 확인 지원을 활성화할 수 있습니다. 자세한 내용은 VPC에서 DNS 지원 및 VPC 피어링 연결 수정 단원을 참조하십시오.
  - 원격 SSL/TLS 리소스에 안전하게 연결하기 위해 사용자 지정 Oracle Wallet을 생성하고 업로드하는 것이 좋습니다. Amazon S3와 Amazon RDS for Oracle의 통합을 사용하여 Oracle DB 인스턴스

로 Amazon S3의 Wallet을 다운로드할 수 있습니다. Oracle에 대한 Amazon S3 통합에 대한 자세한 내용은 Amazon S3 통합 단원을 참조하십시오.

- 각 인스턴스에 Oracle SSL 옵션이 구성된 경우 SSL/TLS 엔드포인트를 통해 Oracle DB 인스턴스 사이에 데이터베이스 링크를 설정할 수 있습니다. 추가 구성이 필요하지 않습니다. 자세한 내용은 Oracle 보안 소켓 Layer 단원을 참조하십시오.

1단계: 웹 사이트의 루트 인증서 가져오기

RDS for Oracle DB 인스턴스가 웹 사이트에 대한 보안 연결을 설정하도록 루트 CA 인증서를 추가합니다. Amazon RDS는 루트 인증서를 사용하여 Oracle Wallet에 웹 사이트 인증서를 서명합니다.

다양한 방법으로 루트 인증서를 얻을 수 있습니다. 예를 들어 다음을 수행할 수 있습니다.

1. 웹 서버를 사용하여 인증서로 보호되는 웹 사이트를 방문합니다.
2. 서명에 사용된 루트 인증서를 다운로드합니다.

AWS 서비스의 경우 루트 인증서는 일반적으로 Amazon trust services 리포지토리에 있습니다.

2단계: Oracle Wallet 생성

웹 서버 인증서와 클라이언트 인증 인증서를 모두 포함하는 Oracle Wallet을 생성합니다. RDS Oracle 인스턴스는 웹 서버 인증서를 사용하여 웹 사이트에 대한 보안 연결을 설정합니다. 웹 사이트는 Oracle 데이터베이스 사용자를 인증하기 위해 클라이언트 인증서를 필요로 합니다.

인증에 클라이언트 인증서를 사용하지 않고 보안 연결을 구성할 수 있습니다. 이 경우 다음 절차에서 Java KeyStore 단계를 건너뛸 수 있습니다.

Oracle Wallet 생성

1. 루트 및 클라이언트 인증서를 단일 디렉터리에 넣은 다음 이 디렉터리로 변경합니다.
2. .p12 클라이언트 인증서를 Java KeyStore로 변환합니다.

> **Note**
>
> 인증에 클라이언트 인증서를 사용하지 않는 경우 이 단계를 건너뛸 수 있습니다.

다음 예에서는 *client_certificate.p12*라는 클라이언트 인증서를 *client_keystore.jks*라는 Java KeyStore로 변환합니다. 그러면 KeyStore가 Oracle Wallet에 포함됩니다. KeyStore 암호는 *P12PASSWORD*입니다.

```
orapki wallet pkcs12_to_jks -wallet ./client_certificate.p12 -
jksKeyStoreLoc ./client_keystore.jks -jksKeyStorepwd P12PASSWORD
```

3. 인증서 디렉터리와 다른 Oracle Wallet용 디렉터리를 생성합니다.

   다음 예에서는 `/tmp/wallet` 디렉터리를 생성합니다.

   ```
   mkdir -p /tmp/wallet
   ```

4. wallet 디렉터리에 Oracle Wallet을 생성합니다.

   다음 예에서는 Oracle Wallet 암호를 이전 단계에서 Java KeyStore에서 사용한 것과 동일한 암호인 *P12PASSWORD*로 설정합니다. 동일한 암호를 사용하면 편리하지만 필수는 아닙니다. `-auto_login` 파라미터가 자동 로그인 기능을 켜므로 액세스할 때마다 암호를 지정할 필요가 없습니다.

   > **Note**
   >
   > 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

   ```
   orapki wallet create -wallet /tmp/wallet -pwd P12PASSWORD -auto_login
   ```

5. Oracle Wallet에 Java KeyStore를 추가합니다.

   > **Note**
   >
   > 인증에 클라이언트 인증서를 사용하지 않는 경우 이 단계를 건너뛸 수 있습니다.

   다음 예에서는 */tmp/wallet*이라는 Oracle Wallet에 KeyStore *client_keystore.jks*를 추가합니다. 이 예에서는 Java KeyStore와 Oracle Wallet에 대해 동일한 암호를 지정합니다.

```
orapki wallet jks_to_pkcs12 -wallet /tmp/wallet -pwd P12PASSWORD -
keystore ./client_keystore.jks -jkspwd P12PASSWORD
```

6. 대상 웹 사이트의 루트 인증서를 Oracle Wallet에 추가합니다.

   다음 예에서는 *Root_CA.cer*이라는 인증서를 추가합니다.

   ```
   orapki wallet add -wallet /tmp/wallet -trusted_cert -cert ./Root_CA.cer -
   pwd P12PASSWORD
   ```

7. 중간 인증서를 추가합니다.

   다음 예에서는 *Intermediate.cer*이라는 인증서를 추가합니다. 모든 중간 인증서를 로드하는 데 필요한 만큼 이 단계를 반복합니다.

   ```
   orapki wallet add -wallet /tmp/wallet -trusted_cert -cert ./Intermediate.cer -
   pwd P12PASSWORD
   ```

8. 새로 생성된 Oracle Wallet에 필요한 인증서가 있는지 확인합니다.

   ```
   orapki wallet display -wallet /tmp/wallet -pwd P12PASSWORD
   ```

3단계: RDS for Oracle 인스턴스로 Oracle Wallet 다운로드

이 단계에서는 Oracle Wallet을 Amazon S3에 업로드한 다음 Amazon S3에서 RDS for Oracle 인스턴스로 Wallet을 다운로드합니다.

RDS for Oracle DB 인스턴스로 Oracle Wallet 다운로드

1. Oracle과 Amazon S3 통합을 위한 사전 조건을 만족시키고 `S3_INTEGRATION` 옵션을 Oracle DB 인스턴스에 추가하십시오. 옵션의 IAM 역할에 사용 중인 Amazon S3 버킷에 대한 액세스 권한이 있어야 합니다.

   자세한 내용은 Amazon S3 통합 단원을 참조하십시오.

2. DB 인스턴스에 마스터 사용자로 로그인한 다음 Oracle Wallet을 보관할 Oracle 디렉터리를 생성합니다.

   다음 예에서는 *WALLET_DIR*이라는 Oracle 디렉터리를 생성합니다.

```
EXEC rdsadmin.rdsadmin_util.create_directory('WALLET_DIR');
```

자세한 내용은 메인 데이터 스토리지 공간에서 디렉터리 생성 및 삭제 단원을 참조하십시오.

3. Amazon S3 버킷에 Oracle Wallet을 업로드합니다.

   지원되는 업로드 기술을 사용할 수 있습니다.

4. Oracle Wallet을 다시 업로드하는 경우 기존 Wallet을 삭제합니다. 그렇지 않은 경우 다음 단계로 건너뜁니다.

   다음 예에서는 *cwallet.sso*라는 기존 Wallet을 제거합니다.

```
EXEC UTL_FILE.FREMOVE ('WALLET_DIR','cwallet.sso');
```

5. Amazon S3 버킷에서 Oracle DB 인스턴스로 Oracle Wallet을 다운로드합니다.

   다음 예에서는 *my_s3_bucket*이라는 Amazon S3 버킷에서 *WALLET_DIR*이라는 DB 인스턴스 디렉터리로 *cwallet.sso*라는 Wallet을 다운로드합니다.

```
SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3(
      p_bucket_name    =>  'my_s3_bucket',
      p_s3_prefix      =>  'cwallet.sso',
      p_directory_name =>  'WALLET_DIR')
   AS TASK_ID FROM DUAL;
```

6. (선택 사항) 암호로 보호된 Oracle Wallet을 다운로드합니다.

   Wallet을 사용할 때마다 암호를 요구하려는 경우에만 이 Wallet을 다운로드합니다. 다음 예에서는 암호로 보호된 Wallet *ewallet.p12*를 다운로드합니다.

```
SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3(
      p_bucket_name    =>  'my_s3_bucket',
      p_s3_prefix      =>  'ewallet.p12',
      p_directory_name =>  'WALLET_DIR')
   AS TASK_ID FROM DUAL;
```

7. DB 태스크의 상태를 확인합니다.

   다음 예에서 *dbtask-1234567890123-4567.log*를 이전 단계에서 반환된 태스크 ID로 대체합니다.

```
SELECT TEXT FROM
 TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-1234567890123-4567.log'));
```

8. Oracle Wallet을 저장하는 데 사용하는 디렉터리의 내용을 확인합니다.

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.listdir(p_directory => 'WALLET_DIR'));
```

자세한 내용은 DB 인스턴스 디렉터리의 파일 목록 표시 단원을 참조하십시오.

4단계: Oracle Wallet에 대한 사용자 권한 부여

새 데이터베이스 사용자를 생성하거나 기존 사용자를 구성할 수 있습니다. 두 경우 모두 인증서를 사용한 클라이언트 인증 및 보안 연결을 위해 Oracle Wallet에 액세스하도록 사용자를 구성해야 합니다.

Oracle Wallet에 대한 사용자 권한 부여

1. RDS for Oracle DB 인스턴스에 마스터 사용자로 로그인합니다.
2. 기존 데이터베이스 사용자를 구성하지 않으려는 경우 새 사용자를 생성합니다. 그렇지 않은 경우 다음 단계로 건너뜁니다.

   다음 예에서는 *my-user*라는 데이터베이스 사용자를 생성합니다.

```
CREATE USER my-user IDENTIFIED BY my-user-pwd;
GRANT CONNECT TO my-user;
```

3. 데이터베이스 사용자에게 Oracle Wallet이 포함된 디렉터리에 대한 권한을 부여합니다.

   다음 예에서는 사용자 *my-user* 에게 디렉터리 *WALLET_DIR*에 대한 읽기 액세스 권한을 부여합니다.

```
GRANT READ ON DIRECTORY WALLET_DIR TO my-user;
```

4. 데이터베이스 사용자에게 UTL_HTTP 패키지를 사용할 수 있는 권한을 부여합니다.

   다음 PL/SQL 프로그램에서는 사용자 *my-user*에게 UTL_HTTP 액세스 권한을 부여합니다.

```
BEGIN
  rdsadmin.rdsadmin_util.grant_sys_object('UTL_HTTP', UPPER('my-user'));
  END;
```

```
/
```

5. 데이터베이스 사용자에게 UTL_FILE 패키지를 사용할 수 있는 권한을 부여합니다.

   다음 PL/SQL 프로그램에서는 사용자 *my-user*에게 UTL_FILE 액세스 권한을 부여합니다.

```
BEGIN
  rdsadmin.rdsadmin_util.grant_sys_object('UTL_FILE', UPPER('my-user'));
  END;
/
```

5단계: DB 인스턴스에서 웹 사이트에 대한 액세스 구성

이 단계에서는 UTL_HTTP, 업로드한 Oracle Wallet 및 클라이언트 인증서를 사용하여 대상 웹 사이트에 연결할 수 있도록 Oracle 데이터베이스 사용자를 구성합니다. 자세한 내용은 Oracle Database 설명서의 Configuring Access Control to an Oracle Wallet(Oracle Wallet에 대한 액세스 제어 구성)을 참조하세요.

RDS for Oracle DB 인스턴스에서 웹 사이트에 대한 액세스 구성

1. RDS for Oracle DB 인스턴스에 마스터 사용자로 로그인합니다.
2. 보안 포트에 사용자 및 대상 웹 사이트에 대한 ACE(Host Access Control Entry)를 생성합니다.

   다음 예에서는 보안 포트 443에서 *secret.encrypted-website.com*에 액세스하도록 *my-user*를 구성합니다.

```
BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
    host       => 'secret.encrypted-website.com',
    lower_port => 443,
    upper_port => 443,
    ace        => xs$ace_type(privilege_list => xs$name_list('http'),
                              principal_name => 'my-user',
                              principal_type => xs_acl.ptype_db));
                           -- If the program unit results in PLS-00201, set
                           -- the principal_type parameter to 2 as follows:
                           -- principal_type => 2));
END;
/
```

> **Important**
>
> 이전 프로그램 단위에서 다음 `PLS-00201: identifier 'XS_ACL' must be declared` 오류가 발생할 수 있습니다. 이 오류가 반환되면 값을 할당하는 줄을 다음 줄의 `principal_type`으로 바꾼 다음 프로그램 단위를 다시 실행하세요.
>
> ```
> principal_type => 2));
> ```
>
> PL/SQL 패키지 `XS_ACL`의 상수에 대한 자세한 내용은 Oracle Database 설명서의 Real Application Security 관리자 및 개발자 안내서를 참조하세요.

자세한 내용은 Oracle Database 설명서의 Configuring Access Control for External Network Services(외부 네트워크 서비스에 대한 액세스 제어 구성)를 참조하세요.

3. (선택 사항) 표준 포트에서 사용자 및 대상 웹 사이트에 대한 ACE를 만듭니다.

   일부 웹 페이지가 보안 포트(443) 대신 표준 웹 서버 포트(80)에서 제공되는 경우 표준 포트를 사용해야 할 수 있습니다.

   ```
   BEGIN
     DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
       host       => 'secret.encrypted-website.com',
       lower_port => 80,
       upper_port => 80,
       ace        => xs$ace_type(privilege_list => xs$name_list('http'),
                                 principal_name => 'my-user',
                                 principal_type => xs_acl.ptype_db));
                               -- If the program unit results in PLS-00201, set
                               -- the principal_type parameter to 2 as follows:
                               -- principal_type => 2));
   END;
   /
   ```

4. 액세스 제어 항목이 있는지 확인합니다.

   ```
   SET LINESIZE 150
   COLUMN HOST FORMAT A40
   COLUMN ACL FORMAT A50
   ```

```
SELECT HOST, LOWER_PORT, UPPER_PORT, ACL
  FROM DBA_NETWORK_ACLS
ORDER BY HOST;
```

5. 데이터베이스 사용자에게 UTL_HTTP 패키지를 사용할 수 있는 권한을 부여합니다.

   다음 PL/SQL 프로그램에서는 사용자 *my-user*에게 UTL_HTTP 액세스 권한을 부여합니다.

```
BEGIN
  rdsadmin.rdsadmin_util.grant_sys_object('UTL_HTTP', UPPER('my-user'));
  END;
/
```

6. 관련 액세스 제어 목록이 있는지 확인합니다.

```
SET LINESIZE 150
COLUMN ACL FORMAT A50
COLUMN PRINCIPAL FORMAT A20
COLUMN PRIVILEGE FORMAT A10

SELECT ACL, PRINCIPAL, PRIVILEGE, IS_GRANT,
       TO_CHAR(START_DATE, 'DD-MON-YYYY') AS START_DATE,
       TO_CHAR(END_DATE, 'DD-MON-YYYY') AS END_DATE
  FROM DBA_NETWORK_ACL_PRIVILEGES
ORDER BY ACL, PRINCIPAL, PRIVILEGE;
```

7. 데이터베이스 사용자에게 클라이언트 인증에 인증서를 사용하고 연결에 Oracle Wallet을 사용할 수 있는 권한을 부여합니다.

   > **Note**
   >
   > 인증에 클라이언트 인증서를 사용하지 않는 경우 이 단계를 건너뛸 수 있습니다.

```
DECLARE
  l_wallet_path all_directories.directory_path%type;
BEGIN
  SELECT DIRECTORY_PATH
    INTO l_wallet_path
    FROM ALL_DIRECTORIES
   WHERE UPPER(DIRECTORY_NAME)='WALLET_DIR';
  DBMS_NETWORK_ACL_ADMIN.APPEND_WALLET_ACE(
```

```
    wallet_path => 'file:/' || l_wallet_path,
    ace         =>  xs$ace_type(privilege_list => xs
$name_list('use_client_certificates'),
                                principal_name => 'my-user',
                                principal_type => xs_acl.ptype_db));
END;
/
```

6단계: DB 인스턴스에서 웹 사이트로의 연결 테스트

이 단계에서는 UTL_HTTP, 업로드한 Oracle Wallet 및 클라이언트 인증서를 사용하여 웹 사이트에 연결할 수 있도록 데이터베이스 사용자를 구성합니다.

RDS for Oracle DB 인스턴스에서 웹 사이트에 대한 액세스 구성

1. UTL_HTTP 권한이 있는 데이터베이스 사용자로 RDS for Oracle DB 인스턴스에 로그인합니다.
2. 대상 웹 사이트에 대한 연결이 호스트 주소를 확인할 수 있는지 확인합니다.

   다음 예에서는 *secret.encrypted-website.com*에서 호스트 주소를 가져옵니다.

   ```
   SELECT UTL_INADDR.GET_HOST_ADDRESS(host => 'secret.encrypted-website.com')
     FROM DUAL;
   ```

3. 실패한 연결을 테스트합니다.

   UTL_HTTP는 인증서가 있는 Oracle Wallet의 위치를 요구하기 때문에 다음 쿼리는 실패합니다.

   ```
   SELECT UTL_HTTP.REQUEST('secret.encrypted-website.com') FROM DUAL;
   ```

4. UTL_HTTP.SET_WALLET을 사용하고 DUAL에서 선택하여 웹 사이트 액세스를 테스트합니다.

   ```
   DECLARE
     l_wallet_path all_directories.directory_path%type;
   BEGIN
     SELECT DIRECTORY_PATH
       INTO l_wallet_path
       FROM ALL_DIRECTORIES
      WHERE UPPER(DIRECTORY_NAME)='WALLET_DIR';
     UTL_HTTP.SET_WALLET('file:/' || l_wallet_path);
   END;
   ```

```
/

SELECT UTL_HTTP.REQUEST('secret.encrypted-website.com') FROM DUAL;
```

5. (선택 사항) 쿼리를 변수에 저장하고 EXECUTE IMMEDIATE를 사용하여 웹 사이트 액세스를 테스트합니다.

```
DECLARE
  l_wallet_path all_directories.directory_path%type;
  v_webpage_sql VARCHAR2(1000);
  v_results     VARCHAR2(32767);
BEGIN
  SELECT DIRECTORY_PATH
    INTO l_wallet_path
    FROM ALL_DIRECTORIES
   WHERE UPPER(DIRECTORY_NAME)='WALLET_DIR';
  v_webpage_sql := 'SELECT UTL_HTTP.REQUEST(''secret.encrypted-website.com'', ''''',
 ''file:/' ||l_wallet_path||''''') FROM DUAL';
  DBMS_OUTPUT.PUT_LINE(v_webpage_sql);
  EXECUTE IMMEDIATE v_webpage_sql INTO v_results;
  DBMS_OUTPUT.PUT_LINE(v_results);
END;
/
```

6. (선택 사항) Oracle Wallet 디렉터리의 파일 시스템 위치를 찾습니다.

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.listdir(p_directory => 'WALLET_DIR'));
```

이전 명령의 출력을 사용하여 HTTP 요청을 만듭니다. 예를 들어 디렉터리가 *rdsdbdata/userdirs/01*인 경우 다음 쿼리를 실행합니다.

```
SELECT UTL_HTTP.REQUEST('https://secret.encrypted-website.com/', '',
 'file://rdsdbdata/userdirs/01')
FROM   DUAL;
```

RDS for Oracle에서 CDB 작업

Oracle 멀티테넌트 아키텍처에서 컨테이너 데이터베이스(CDB)에는 고객이 만든 플러그형 데이터베이스(PDB)가 포함될 수 있습니다. CDB에 대한 자세한 내용은 Oracle Database 설명서의 멀티테넌트 아키텍처 소개를 참조하세요.

주제

- RDS for Oracle CDB 개요
- RDS for Oracle CDB 구성
- CDB 백업 및 복원
- RDS for Oracle 비CDB를 CDB로 변환
- 단일 테넌트 구성을 다중 테넌트로 변환
- CDB 인스턴스에 RDS for Oracle 테넌트 데이터베이스 추가
- RDS for Oracle 테넌트 데이터베이스 수정
- CDB에서 RDS for Oracle 테넌트 데이터베이스 삭제
- 테넌트 데이터베이스 세부 정보 보기
- CDB 업그레이드

RDS for Oracle CDB 개요

Oracle Database 19c 이상을 실행할 때 RDS for Oracle DB 인스턴스를 컨테이너 데이터베이스(CDB)로 생성할 수 있습니다. Oracle Database 21c부터는 모든 데이터베이스가 CDB입니다. CDB는 RDS for Oracle에서 테넌트 데이터베이스라고 하는 플러그형 데이터베이스(PDB)를 포함할 수 있다는 점에서 비CDB와 다릅니다. PDB는 애플리케이션에 별도의 데이터베이스로 나타나는 스키마와 개체의 휴대용 모음입니다.

CDB 인스턴스를 생성할 때 초기 테넌트 데이터베이스(PDB)를 생성합니다. RDS for Oracle에서는 클라이언트 애플리케이션이 CDB가 아닌 PDB와 상호 작용합니다. PDB에서의 경험은 비CDB에서의 경험과 대부분 동일합니다.

주제

- CDB 아키텍처의 다중 테넌트 구성
- CDB 아키텍처의 단일 테넌트 구성
- CDB 생성 및 변환 옵션

- CDB의 사용자 계정 및 권한
- CDB의 파라미터 그룹 패밀리
- RDS for Oracle CDB 제한 사항

CDB 아키텍처의 다중 테넌트 구성

RDS for Oracle은 Oracle 멀티테넌트 아키텍처, 즉 CDB 아키텍처의 다중 테넌트 구성을 지원합니다. 이 구성에서 RDS for Oracle CDB 인스턴스에는 데이터베이스 에디션 및 필요한 옵션 라이선스에 따라 1~30개의 테넌트 데이터베이스가 포함될 수 있습니다. Oracle 데이터베이스의 경우 테넌트 데이터베이스는 PDB입니다. DB 인스턴스는 Oracle Database 릴리스 19.0.0.0.ru-2022-01.rur-2022.r1 이상을 사용해야 합니다.

> **Note**
>
> Amazon RDS 구성은 Oracle DB 엔진뿐만 아니라 RDS 플랫폼의 기능이기 때문에 '멀티테넌트'가 아닌 '다중 테넌트'라고 합니다. 마찬가지로 RDS 용어 '테넌트'는 Oracle PDB뿐만 아니라 RDS 구성의 모든 테넌트를 나타냅니다. RDS 설명서에서 하이픈이 포함되지 않은 용어 'Oracle 멀티테넌트'는 온프레미스 및 RDS 배포 모두와 호환되는 Oracle 데이터베이스 CDB 아키텍처만을 의미합니다.

다음과 같은 설정을 구성할 수 있습니다.

- 테넌트 데이터베이스 이름
- 테넌트 데이터베이스 마스터 사용자 이름
- 테넌트 데이터베이스 마스터 암호(선택적으로 Secrets Manager와 통합됨)
- 테넌트 데이터베이스 문자 집합
- 테넌트 데이터베이스 내셔널 문자 집합

테넌트 데이터베이스 문자 집합은 CDB 문자 집합과 다를 수 있습니다. 내셔널 문자 집합 역시 마찬가지입니다. 초기 테넌트 데이터베이스를 만든 후 RDS API를 사용하여 테넌트 데이터베이스를 생성, 수정 또는 삭제할 수 있습니다. CDB 이름은 기본값인 RDSCDB이며 변경할 수 없습니다. 자세한 내용은 DB 인스턴스에 대한 설정 및 RDS for Oracle 테넌트 데이터베이스 수정(을)를 참조하세요.

CDB 아키텍처의 단일 테넌트 구성

RDS for Oracle은 Oracle 멀티테넌트 아키텍처의 레거시 구성인 단일 테넌트 구성을 지원합니다. 이 구성에서 RDS for Oracle CDB 인스턴스에는 테넌트(PDB)가 하나만 포함될 수 있습니다. 나중에 더 많은 PDB를 만들 수 있습니다.

CDB 생성 및 변환 옵션

Oracle Database 21c는 CDB만 지원하지만 Oracle Database 19c는 CDB와 비CDB를 모두 지원합니다. 모든 RDS for Oracle CDB 인스턴스는 다중 테넌트 구성과 단일 테넌트 구성을 모두 지원합니다.

Oracle 데이터베이스 아키텍처를 위한 생성, 변환 및 업그레이드 옵션

다음 테이블에는 RDS for Oracle 데이터베이스를 생성하고 업그레이드하기 위한 다양한 아키텍처 옵션이 나와 있습니다.

릴리스	데이터베이스 생성 옵션	아키텍처 변환 옵션	메이저 버전 업그레이드 대상
Oracle Database 21c	CDB 아키텍처 전용	N/A	N/A
Oracle Database 19c	CDB 또는 비CDB 아키텍처	비CDB에서 CDB 아키텍처로 변환(2021년 4월 RU 이상)	Oracle Database 21c CDB

위 테이블에 표시된 것처럼 새 메이저 데이터베이스 버전에서는 비CDB를 CDB로 직접 업그레이드할 수 없습니다. 하지만 Oracle Database 19c 비CDB를 Oracle Database 19c CDB로 변환한 다음 Oracle Database 19c CDB를 Oracle Database 21c CDB로 업그레이드할 수 있습니다. 자세한 내용은 RDS for Oracle 비CDB를 CDB로 변환 섹션을 참조하세요.

CDB 아키텍처 구성을 위한 변환 옵션

다음 테이블에는 RDS for Oracle DB 인스턴스의 아키텍처 구성 변환을 위한 다양한 옵션이 나와 있습니다.

현재 아키텍처 및 구성	CDB 아키텍처의 단일 테넌트 구성으로 변환	CDB 아키텍처의 다중 테넌트 구성으로 변환	비CDB 아키텍처로 변환
비CDB	지원	지원*	N/A
단일 테넌트 구성을 사용하는 CDB	N/A	지원	지원되지 않음
다중 테넌트 구성을 사용하는 CDB	지원되지 않음	N/A	지원되지 않음

* 하나의 작업으로 비CDB를 다중 테넌트 구성으로 변환할 수 없습니다. 비CDB를 CDB로 변환할 경우 CDB는 단일 테넌트 구성입니다. 그런 다음 별도의 작업을 통해 단일 테넌트를 다중 테넌트 구성으로 변환할 수 있습니다.

CDB의 사용자 계정 및 권한

Oracle 멀티테넌트 아키텍처에서 모든 사용자 계정은 일반 사용자 또는 로컬 사용자입니다. CDB 공통 사용자는 단일 ID와 암호가 CDB 루트와 모든 기존 및 미래의 PDB에 알려진 데이터베이스 사용자입니다. 반면 로컬 사용자는 단일 PDB에만 존재합니다.

RDS 마스터 사용자는 PDB의 로컬 사용자 계정으로, DB 인스턴스를 만들 때 이름을 지정합니다. 새 사용자 계정을 생성하는 경우 새로 생성된 사용자는 PDB에 상주하는 로컬 사용자도 됩니다. 사용자 계정을 사용하여 새 PDB를 생성하거나 기존 PDB의 상태를 수정할 수는 없습니다.

`rdsadmin` 사용자는 공통 사용자 계정입니다. 이 계정에 있는 RDS for Oracle 패키지를 실행할 수 있지만 `rdsadmin`으로 로그인할 수는 없습니다. 자세한 내용은 Oracle 설명서에서 About Common Users and Local Users를 참조하세요.

다중 테넌트 구성과 단일 테넌트 구성의 마스터 사용자에 대해서는 자체 관리형 자격 증명 또는 AWS Secrets Manager에 의해 관리되는 자격 증명을 사용할 수 있습니다. 단일 테넌트 구성에서는 관리형 마스터 암호에 `create-db-instance`와 같은 인스턴스 수준 CLI 명령을 사용합니다. 다중 테넌트 구성에서는 관리형 마스터 암호에 `create-tenant-database`와 같은 테넌트 데이터베이스 명령을 사용합니다. Secrets Manager 통합에 대한 자세한 내용은 Secrets Manager를 사용하여 RDS for Oracle 테넌트 데이터베이스의 마스터 사용자 암호 관리 섹션을 참조하세요.

CDB의 파라미터 그룹 패밀리

CDB에는 자체 파라미터 패밀리 및 기본 파라미터 값이 있습니다. CDB 파라미터 그룹 패밀리는 다음과 같습니다.

- oracle-ee-cdb-21
- oracle-se2-cdb-21
- oracle-ee-cdb-19
- oracle-se2-cdb-19

RDS for Oracle CDB 제한 사항

RDS for Oracle은 온프레미스 CDB에서 사용할 수 있는 기능 중 일부를 지원합니다.

CDB 제한 사항

CBD 수준의 RDS for Oracle에는 다음과 같은 제한 사항이 적용됩니다.

- CDB에 연결할 수 없습니다. 항상 CDB가 아니라 테넌트 데이터베이스(PDB)에 연결합니다. 비 CDB의 경우와 마찬가지로 PDB의 엔드포인트를 지정합니다. 유일한 차이점은 데이터베이스 이름에 pdb_name을 지정한다는 것입니다. 여기서 pdb_name은 PDB에 대해 선택한 이름입니다.
- 다중 테넌트 구성의 CDB를 단일 테넌트 구성의 CDB로 변환할 수 없습니다. 다중 테넌트 구성으로의 변환은 단방향이며 되돌릴 수 없습니다.
- DB 인스턴스가 19.0.0.0.ru-2022-01.rur-2022.r1 미만의 Oracle 데이터베이스 릴리스를 사용하는 경우 다중 테넌트 구성을 지원하거나 다중 테넌트 구성으로 변환할 수 없습니다.
- ORDS 22 이상이 설치된 RDS for Oracle CDB는 사용할 수 없습니다. 해결 방법으로 ORDS의 이전 버전을 사용하거나 CDB가 아닌 Oracle Database 19c를 사용할 수 있습니다.
- 다중 테넌트 구성에서는 Oracle Data Guard를 사용할 수 없지만 단일 테넌트 구성에서는 사용할 수 있습니다.
- CDB에서는 데이터베이스 활동 스트림을 사용할 수 없습니다.
- `CDB$ROOT` 내부에서 감사를 활성화할 수 없습니다. 각 PDB 내에서 개별적으로 감사를 활성화해야 합니다.

테넌트 데이터베이스(PDB) 제한 사항

RDS for Oracle 다중 테넌트 구성의 테넌트 데이터베이스에는 다음과 같은 제한 사항이 적용됩니다.

- 테넌트 데이터베이스 작업을 유지 관리 기간으로 연기할 수 없습니다. 모든 변경 사항은 즉시 적용됩니다.
- 단일 테넌트 구성을 사용하는 CDB에는 테넌트 데이터베이스를 추가할 수 없습니다.
- 한 번의 작업으로 여러 테넌트 데이터베이스를 추가하거나 수정할 수 없습니다. 한 번에 하나만 추가 또는 수정할 수 있습니다.
- 테넌트 데이터베이스의 이름을 CDB$ROOT 또는 PDB$SEED로 수정할 수 없습니다.
- CDB의 유일한 테넌트인 테넌트 데이터베이스는 삭제할 수 없습니다.
- RDS for Oracle CDB 인스턴스에서 모든 DB 인스턴스 클래스 유형이 여러 PDB를 지원하기에 충분한 리소스를 보유한 것은 아닙니다. PDB 수가 증가하면 상대적으로 작은 인스턴스 클래스의 성능과 안정성에 영향을 미치고 대부분의 인스턴스 수준 작업(예: 데이터베이스 업그레이드)의 시간이 늘어납니다.
- 동일한 CDB에 여러 개의 AWS 계정을 사용하여 PDB를 만들 수 없습니다. PDB는 PDB가 호스팅되는 DB 인스턴스와 동일한 계정에서 소유해야 합니다.
- CDB의 모든 PDB는 동일한 엔드포인트와 데이터베이스 리스너를 사용합니다.
- 다음 작업은 PDB 수준에서는 지원되지 않지만 CDB 수준에서는 지원됩니다.
  - 백업 및 복구
  - 데이터베이스 업그레이드
  - 유지 관리 작업
- 다음 기능은 PDB 수준에서는 지원되지 않지만 CDB 수준에서는 지원됩니다.
  - 옵션 그룹(옵션은 CDB 인스턴스의 모든 PDB에 설치됨)
  - 파라미터 그룹(모든 파라미터는 CDB 인스턴스와 연결된 파라미터 그룹에서 파생됨)
- 온프레미스 CDB 아키텍처에서는 지원되지만 RSD for Oracle CDB에서는 지원되지 않는 PDB 수준 작업은 다음과 같습니다.

> **Note**
>
> 다음 목록이 전부는 아닙니다.

  - 애플리케이션 PDB
  - 프록시 PDB
  - PDB 시작 및 중지
  - PDB 연결 해제 및 연결

데이터를 CDB 안팎으로 이전하려면 비CDB와 동일한 방법을 사용하세요. 데이터 마이그레이션에 대한 자세한 내용은 Amazon RDS의 Oracle로 데이터 가져오기 섹션을 참조하세요.

- PDB 수준에서 옵션 설정

  PDB는 CDB 옵션 그룹의 옵션 설정을 상속합니다. 옵션 설정에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 섹션을 참조하세요. 모범 사례는 DB 파라미터 그룹 작업 섹션을 참조하세요.

- PDB의 파라미터 구성

  PDB는 CDB의 파라미터 설정을 상속합니다. 옵션 설정에 대한 자세한 내용은 Oracle DB 인스턴스에 옵션 추가 섹션을 참조하세요.

- 동일한 CDB의 PDB에 대해 서로 다른 리스너 구성
- Oracle Flashback 기능

RDS for Oracle CDB 구성

CDB를 구성하는 방법은 비CDB를 구성하는 것과 비슷합니다.

주제

- RDSfor Oracle CDB 인스턴스 생성
- RDS for Oracle CDB의 PDB에 연결

RDSfor Oracle CDB 인스턴스 생성

RDS for Oracle에서 CDB를 만드는 방법은 비CDB 인스턴스를 만드는 방법과 거의 동일합니다. 차이점은 DB 인스턴스를 만들 때 Oracle 멀티테넌트 아키텍처를 선택하며 아키텍처 구성(다중 테넌트 또는 단일 테넌트)도 선택한다는 것입니다. 다중 테넌트 구성에서 CDB를 생성할 때 태그를 생성하면 RDS가 태그를 초기 테넌트 데이터베이스로 전파합니다. CDB를 만들려면 AWS Management Console, AWS CLI 또는 RDS API를 사용합니다.

콘솔

CDB 인스턴스를 생성하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.

2. Amazon RDS 콘솔의 오른쪽 상단에서 CDB 인스턴스를 생성하려는 AWS 리전을 선택합니다.
3. 탐색 창에서 데이터베이스를 선택합니다.
4. 데이터베이스 생성을 선택합니다.
5. Choose a database creation method(데이터베이스 생성 방법 선택)에서 Standard Create(표준 생성)를 선택합니다.
6. Engine options(엔진 옵션)에서 Oracle을 선택합니다.
7. 데이터베이스 관리 유형에서 Amazon RDS를 선택합니다.
8. 아키텍처 설정에서 Oracle 멀티테넌트 아키텍처를 선택합니다.
9. 아키텍처 구성에서 다음 중 하나를 수행합니다.

   - 다중 테넌트 구성을 선택하고 다음 단계로 진행합니다.
   - 단일 테넌트 구성을 선택하고 11단계로 건너뜁니다.

10. (다중 테넌트 구성) 테넌트 데이터베이스 설정에서 다음과 같이 변경합니다.

    - 테넌트 데이터베이스 이름에 초기 PDB의 이름을 입력합니다. PDB 이름은 CDB 이름(기본값 RDSCDB)과 달라야 합니다.
    - 테넌트 데이터베이스 마스터 사용자 이름에 PDB의 마스터 사용자 이름을 입력합니다. 테넌트 데이터베이스 마스터 사용자 이름을 사용하여 CDB 자체에 로그인할 수 없습니다.
    - 자격 증명 관리에서 다음 자격 증명 관리 옵션 중 하나를 선택합니다.
      - AWS Secrets Manager에서 관리됨

        관리형 암호는 인스턴스가 아닌 초기 테넌트 데이터베이스용입니다. 암호화 키 선택에서 Secrets Manager가 만드는 KMS 키 또는 사용자가 만든 키를 선택합니다.

        > **Note**
        >
        > 자격 증명을 관리하기 위한 가장 안전한 기법으로 AWS Secrets Manager를 사용하는 것이 좋습니다. 추가 요금이 발생합니다. 자세한 내용은 Amazon RDS 및 AWS Secrets Manager를 통한 암호 관리 섹션을 참조하세요.

      - 자체 관리형

        암호를 지정하려면 암호 자동 생성 확인란을 선택 취소합니다(선택되어 있는 경우). 마스터 암호 및 마스터 암호 확인에 동일한 암호를 입력합니다.
    - 테넌트 데이터베이스 문자 집합에서 PDB의 문자 집합을 선택합니다. CDB 문자 집합과 다른 테넌트 데이터베이스 문자 집합을 선택할 수 있습니다.

기본 PDB 문자 집합은 AL32UTF8입니다. 기본이 아닌 PDB 문자 집합을 선택하면 CDB 생성 속도가 느려질 수 있습니다.

> **Note**
>
> 만들기 작업에서는 다중 테넌트 데이터베이스를 지정할 수 없습니다. CDB는 만들어질 때 하나의 PDB를 가지고 있습니다. 별도의 작업으로 기존 CBD에 PDB를 추가할 수 있습니다.

11. (단일 테넌트 구성) DB 인스턴스에 대한 설정에 나열된 옵션에 따라 원하는 설정을 선택합니다.

    - Settings(설정) 섹션에서 Credential Settings(보안 인증 정보 설정)를 엽니다. 뒤이어 다음과 같이 하세요.

      i. 마스터 사용자 이름에 PDB의 로컬 사용자 이름을 입력합니다. 마스터 사용자 이름을 사용하여 CDB 루트에 로그인할 수 없습니다.

      ii. 자격 증명 관리에서 다음 자격 증명 관리 옵션 중 하나를 선택합니다.

      - AWS Secrets Manager에서 관리됨

        암호화 키 선택에서 Secrets Manager가 만드는 KMS 키 또는 사용자가 만든 키를 선택합니다.

        > **Note**
        >
        > 자격 증명을 관리하기 위한 가장 안전한 기법으로 AWS Secrets Manager를 사용하는 것이 좋습니다. 추가 요금이 발생합니다. 자세한 내용은 Amazon RDS 및 AWS Secrets Manager를 통한 암호 관리 섹션을 참조하세요.

      - 자체 관리형

        암호를 지정하려면 암호 자동 생성 확인란을 선택 취소합니다(선택되어 있는 경우). 마스터 암호 및 마스터 암호 확인에 동일한 암호를 입력합니다.

12. 나머지 섹션에서 DB 인스턴스 설정을 지정합니다. 각 설정에 대한 자세한 내용은 DB 인스턴스 설정을 참조하세요.
13. 데이터베이스 생성를 선택합니다.

AWS CLI

다중 테넌트 구성에서 CDB를 생성하려면 다음 파라미터와 함께 create-db-instance 명령을 사용합니다.

- `--db-instance-identifier`
- `--db-instance-class`
- `--engine { oracle-ee-cdb | oracle-se2-cdb }`
- `--master-username`
- `--master-user-password` 또는 `--manage-master-user-password`
- `--multi-tenant`(단일 테넌트 구성의 경우 `multi-tenant`를 지정하지 말거나 `--no-multi-tenant`를 지정하세요.)
- `--allocated-storage`
- `--backup-retention-period`

각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하세요.

다음 예에서는 다중 테넌트 구성에서 *my-cdb-inst*라는 RDS for Oracle DB 인스턴스를 생성합니다. `--no-multi-tenant`를 지정하거나 `--multi-tenant`를 지정하지 않는 경우 기본 CDB 구성은 단일 테넌트입니다. 엔진은 `oracle-ee`를 지정하는 `oracle-ee-cdb` 명령이며 `--multi-tenant`는 오류와 함께 실패합니다. 초기 테넌트 데이터베이스의 이름은 *mypdb*입니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance \
    --engine oracle-ee-cdb \
    --db-instance-identifier my-cdb-inst \
    --multi-tenant \
    --db-name mypdb \
    --allocated-storage 250 \
    --db-instance-class db.t3.large \
    --master-username pdb_admin \
    --manage-master-user-password \
    --backup-retention-period 3
```

Windows의 경우:

```
aws rds create-db-instance ^
    --engine oracle-ee-cdb ^
    --db-instance-identifier my-cdb-inst ^
    --multi-tenant ^
    --db-name mypdb ^
    --allocated-storage 250 ^
    --db-instance-class db.t3.large ^
    --master-username pdb_admin ^
    --manage-master-user-password \ ^
    --backup-retention-period 3
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

다음과 비슷한 출력이 생성됩니다. 데이터베이스 이름, 문자 집합, 내셔널 문자 집합 및 마스터 사용자 및 마스터 사용자 보안 암호는 출력에 포함되지 않습니다. `describe-tenant-databases` CLI 명령을 사용하여 이 정보를 볼 수 있습니다.

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "my-cdb-inst",
        "DBInstanceClass": "db.t3.large",
        "MultiTenant": true,
        "Engine": "oracle-ee-cdb",
        "DBResourceId": "db-ABCDEFGJIJKLMNOPQRSTUVWXYZ",
        "DBInstanceStatus": "creating",
        "AllocatedStorage": 250,
        "PreferredBackupWindow": "04:59-05:29",
        "BackupRetentionPeriod": 3,
        "DBSecurityGroups": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-0a1bcd2e",
                "Status": "active"
            }
        ],
        "DBParameterGroups": [
            {
                "DBParameterGroupName": "default.oracle-ee-cdb-19",
                "ParameterApplyStatus": "in-sync"
```

```
            }
        ],
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-1234567a",
            "SubnetGroupStatus": "Complete",
            ...
```

RDS API

Amazon RDS API를 사용하여 DB 인스턴스를 생성하려면 CreateDBInstance 작업을 호출합니다.

각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하십시오.

RDS for Oracle CDB의 PDB에 연결

SQL*Plus와 같은 유틸리티를 사용하여 PDB에 연결할 수 있습니다. 독립형 버전의 SQL*Plus를 포함하는 Oracle 인스턴트 클라이언트를 다운로드하려면 Oracle 인스턴트 클라이언트 다운로드를 참조하세요.

PDB에 SQL*Plus를 연결하려면 다음 정보를 제공해야 합니다.

- PDB 이름
- 데이터베이스 사용자 이름 및 암호
- DB 인스턴스의 엔드포인트
- 포트 번호

이전 정보를 찾는 방법에 대한 자세한 내용은 RDS for Oracle DB 인스턴스의 엔드포인트 찾기 섹션을 참조하세요.

Example SQL*Plus를 사용하여 PDB에 연결하는 방법

다음 예에서는 마스터 사용자를 *master_user_name* 대신 사용하세요. 또한 DB 인스턴스를 엔드포인트로 대체한 다음, 포트 번호와 Oracle SID를 포함합니다. SID 값은 DB 인스턴스 식별자가 아니라 DB 인스턴스를 만들 때 지정한 PDB의 이름입니다.

대상 LinuxmacOS, 또는Unix:

```
sqlplus 'master_user_name@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=endpoint)
(PORT=port))(CONNECT_DATA=(SID=pdb_name)))'
```

Windows의 경우:

```
sqlplus master_user_name@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=endpoint)
(PORT=port))(CONNECT_DATA=(SID=pdb_name)))
```

다음과 유사한 출력 화면이 표시되어야 합니다.

```
SQL*Plus: Release 19.0.0.0.0 Production on Mon Aug 21 09:42:20 2021
```

사용자 암호를 입력하면 SQL 프롬프트가 표시됩니다.

```
SQL>
```

> **Note**
>
> `sqlplus username/password@LONGER-THAN-63-CHARS-RDS-ENDPOINT-HERE:1521/database-identifier`와 같은 짧은 형식의 연결 문자열(Easy Connect 또는 EZCONNECT)이 최대 문자 제한에 걸릴 수 있으며, 이런 문자열을 연결할 때 사용하면 안 됩니다.

CDB 백업 및 복원

RDS DB 스냅샷 또는 Recovery Manager(RMAN)를 사용하여 CDB를 백업하고 복원할 수 있습니다.

DB 스냅샷을 사용한 CDB 백업 및 복원

DB 스냅샷은 CDB와 비CDB 아키텍처에서 비슷하게 작동합니다. 주요 차이점은 다음과 같습니다.

- CDB의 DB 스냅샷을 복원할 때는 CDB의 이름을 바꿀 수 없습니다. CDB 이름은 기본값인 RDSCDB이며 변경할 수 없습니다.
- CDB의 DB 스냅샷을 복원할 때는 PDB의 이름을 바꿀 수 없습니다. modify-tenant-database 명령을 사용하여 PDB 이름을 수정할 수 있습니다.
- 스냅샷에서 테넌트 데이터베이스를 찾으려면 CLI 명령 describe-db-snapshot-tenant-databases를 사용합니다.
- 다중 테넌트 아키텍처 구성을 사용하는 CDB 스냅샷에서는 테넌트 데이터베이스와 직접 상호 작용할 수 없습니다. DB 스냅샷을 복원하면 해당하는 모든 테넌트 데이터베이스가 복원됩니다.

- RDS for Oracle은 테넌트 데이터베이스의 태그를 DB 스냅샷의 테넌트 데이터베이스에 암시적으로 복사합니다. 테넌트 데이터베이스를 복원하면 복원된 데이터베이스에 태그가 나타납니다.
- DB 스냅샷을 복원하고 `--tags` 파라미터를 사용하여 새 태그를 지정하는 경우 새 태그가 기존 태그를 모두 덮어씁니다.
- 태그가 있는 CDB 인스턴스의 DB 스냅샷을 만들고 `--copy-tags-to-snapshot`을 지정하는 경우 RDS for Oracle은 테넌트 데이터베이스의 태그를 스냅샷의 테넌트 데이터베이스로 복사합니다.

자세한 내용은 Oracle Database 고려 사항 단원을 참조하십시오.

RMAN을 사용한 CDB 백업 및 복원

RMAN을 사용하여 CDB 또는 개별 테넌트 데이터베이스를 백업하고 복원하는 방법에 대한 자세한 내용은 Oracle DB 인스턴스에 대한 공통 RMAN 작업 수행 섹션을 참조하세요.

RDS for Oracle 비CDB를 CDB로 변환

`modify-db-instance` 명령을 통해 Oracle 데이터베이스의 아키텍처를 비CDB 아키텍처에서 Oracle 멀티테넌트 아키텍처(CDB 아키텍처)로 변경할 수 있습니다. 대부분의 경우 새 CDB를 만들고 데이터를 가져오는 것보다 이 방법을 사용하는 것이 좋습니다. 변환 작업 시 다운타임이 발생합니다.

데이터베이스 엔진 버전을 업그레이드할 때 동일한 작업에서 데이터베이스 아키텍처를 변경할 수 없습니다. 따라서 Oracle Database 19c 비CDB를 Oracle Database 21c CDB로 업그레이드하려면 먼저 한 단계에서 비CDB를 CDB로 변환한 다음, 별도의 단계에서 19c CDB를 21c CDB로 업그레이드해야 합니다.

비CDB 변환 작업에는 다음과 같은 요구 사항이 있습니다.

- DB 엔진 유형에는 `oracle-ee-cdb` 또는 `oracle-se2-cdb`를 지정해야 합니다. 이 값만 지원됩니다.
- DB 엔진은 2021년 4월 이후 릴리스 업데이트(RU)가 포함된 Oracle Database 19c를 사용해야 합니다.

이 작업에는 다음과 같은 제한 사항이 있습니다.

- CDB를 비CDB로 변환할 수 없습니다. 비CDB를 CDB로만 변환할 수 있습니다.
- 하나의 `modify-db-instance` 호출로 비CDB를 다중 테넌트 구성으로 변환할 수 없습니다. 비CDB를 CDB로 변환한 후에 CDB는 단일 테넌트 구성이 됩니다. 단일 테넌트 구성을 다중 테넌트 구

성으로 변환하려면 `modify-db-instance`를 다시 실행하세요. 자세한 내용은 단일 테넌트 구성을 다중 테넌트로 변환 섹션을 참조하세요.

- Oracle Data Guard가 사용 설정된 기본 또는 복제본 데이터베이스는 변환할 수 없습니다. 읽기 전용 복제본이 있는 비CDB를 변환하려면 먼저 읽기 전용 복제본을 모두 삭제하세요.
- 동일한 작업에서 DB 엔진 버전을 업그레이드하고 비CDB를 CDB로 변환할 수 없습니다.

비CDB를 전환하기 전에 다음 사항을 고려하세요.

- 옵션 및 파라미터 그룹에 대한 고려 사항은 DB 엔진 업그레이드와 동일합니다. 자세한 내용은 Oracle 데이터베이스 업그레이드에 대한 고려 사항 섹션을 참조하세요.
- 관리형 마스터 암호를 사용하는 기존 비CDB 인스턴스를 단일 작업을 사용하여 단일 테넌트 인스턴스로 변환할 수 있습니다. 단일 테넌트 인스턴스는 관리형 암호를 상속합니다.
- DB 인스턴스에 `OEMAGENT` 옵션이 설치되어 있는 경우 비CDB로 전환하기 전에 이 옵션을 제거하는 것이 가장 좋습니다. 비CDB를 CDB로 전환한 후 옵션을 다시 설치하세요. 자세한 내용은 Oracle Management Agent for Enterprise Manager Cloud Control 섹션을 참조하세요.
- 변환 프로세스 중에 RDS는 온라인 다시 실행 로그 크기를 기본값 128M으로 재설정합니다.

콘솔

비CDB를 CDB로 변환하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 콘솔의 오른쪽 상단에서 DB 인스턴스가 상주하는 AWS 리전을 선택합니다.
3. 탐색 창에서 데이터베이스를 선택한 후 CDB 인스턴스로 변환하려는 비CDB 인스턴스를 선택합니다.
4. 수정을 선택합니다.
5. 아키텍처 설정에서 Oracle 멀티테넌트 아키텍처를 선택합니다. 변환 후 CDB는 단일 테넌트 구성이 됩니다.
6. (선택 사항) DB 파라미터 그룹에서 CDB 인스턴스의 새 파라미터 그룹을 선택합니다. DB 인스턴스를 변환할 때는 DB 인스턴스를 업그레이드할 때와 동일한 파라미터 그룹 고려 사항이 적용됩니다. 자세한 내용은 파라미터 그룹 고려 사항 섹션을 참조하세요.
7. (선택 사항) 옵션 그룹에서 CDB 인스턴스의 새로운 옵션 그룹을 선택합니다. DB 인스턴스를 변환할 때는 DB 인스턴스를 업그레이드할 때와 동일한 옵션 그룹 고려 사항이 적용됩니다. 자세한 내용은 옵션 그룹 고려 사항 섹션을 참조하세요.

8. (선택 사항) 자격 증명 관리에서 AWS Secrets Manager에서 관리 또는 자체 관리를 선택합니다. 자세한 내용은 Secrets Manager를 통해 DB 인스턴스의 마스터 사용자 암호 관리 섹션을 참조하세요.
9. 원하는 대로 모두 변경되었으면 [Continue]를 선택하고 수정 사항 요약을 확인합니다.
10. (선택 사항) 즉시 적용을 선택하여 변경 내용을 즉시 적용합니다. 일부의 경우 이 옵션을 선택하면 가동 중지 시간이 발생할 수 있습니다. 자세한 내용은 수정 예약 설정 사용 섹션을 참조하세요.
11. 확인 페이지에서 변경 내용을 검토합니다. 내용이 정확할 경우 DB 인스턴스 수정을 선택합니다.

    또는 뒤로를 선택하여 변경 내용을 편집하거나 취소를 선택하여 변경 내용을 취소합니다.

AWS CLI

DB 인스턴스의 비CDB를 단일 테넌트 구성의 CDB로 변환하려면 modify-db-instance AWS CLI 명령에서 `--engine`을 `oracle-ee-cdb` 또는 `oracle-se2-cdb`로 설정합니다. 자세한 내용은 DB 인스턴스에 대한 설정 섹션을 참조하세요.

다음 예에서는 *my-non-cdb*라는 이름의 DB 인스턴스를 변환하고 사용자 지정 옵션 그룹과 파라미터 그룹을 지정합니다. 명령은 Secrets Manager를 사용한 암호 관리도 활성화합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier my-non-cdb \
    --engine oracle-ee-cdb \
    --option-group-name custom-option-group \
    --db-parameter-group-name custom-parameter-group \
    --manage-master-user-password
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-non-cdb ^
    --engine oracle-ee-cdb ^
    --option-group-name custom-option-group ^
    --db-parameter-group-name custom-parameter-group ^
    --manage-master-user-password
```

RDS API

비CDB를 CDB로 변환하려면 ModifyDBInstance RDS API 작업에서 `Engine`을 지정합니다.

단일 테넌트 구성을 다중 테넌트로 변환

RDS for Oracle CDB의 아키텍처를 단일 테넌트 구성에서 다중 테넌트 구성으로 수정할 수 있습니다. 변환 전과 후에는 CDB에 하나의 단일 테넌트 데이터베이스(PDB)가 포함됩니다. DB 인스턴스의 태그는 변환 중에 만들어진 초기 테넌트 데이터베이스에 전파됩니다.

시작하기 전에 IAM 정책에 테넌트 데이터베이스를 만들 수 있는 권한이 있는지 확인합니다. 변환 중에 RDS for Oracle은 다음 메타데이터를 새 테넌트 데이터베이스로 마이그레이션합니다.

- 마스터 사용자 이름
- 관리형 마스터 암호(소스 CDB가 Secrets Manager와 통합되는 경우)
- 데이터베이스 이름
- 문자 집합
- 내셔널 문자 집합

변환 전에 `describe-db-instances` 명령을 사용하여 위의 정보를 볼 수 있습니다. 변환 전에 `describe-tenant-database` 명령을 사용하여 정보를 볼 수 있습니다.

단일 테넌트에서 다중 테넌트로의 변환에는 다음과 같은 제한 사항이 있습니다.

- 다중 테넌트 구성으로 변환한 후에는 단일 테넌트 구성으로 다시 변환할 수 없습니다. 변환은 되돌릴 수 없습니다.
- Oracle Data Guard가 사용 설정된 기본 또는 복제본 데이터베이스는 변환할 수 없습니다.
- 동일한 작업에서 DB 엔진 버전을 업그레이드하고 다중 테넌트 구성으로 변환할 수 없습니다.
- 변환 중에는 관리형 마스터 사용자 암호를 활성화하거나 비활성화할 수 없습니다.

콘솔

단일 테넌트 구성을 사용하는 CDB를 다중 테넌트 구성으로 변환하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 콘솔의 오른쪽 상단에서 DB 인스턴스가 상주하는 AWS 리전을 선택합니다.

3. 탐색 창에서 데이터베이스를 선택한 후 CDB 인스턴스로 변환하려는 비CDB 인스턴스를 선택합니다.
4. 수정을 선택합니다.
5. 아키텍처 설정에서 Oracle 멀티테넌트 아키텍처를 선택합니다.
6. 아키텍처 구성에서 다중 테넌트 구성을 선택합니다.
7. (선택 사항) DB 파라미터 그룹에서 CDB 인스턴스의 새 파라미터 그룹을 선택합니다. DB 인스턴스를 변환할 때는 DB 인스턴스를 업그레이드할 때와 동일한 파라미터 그룹 고려 사항이 적용됩니다.
8. (선택 사항) 옵션 그룹에서 CDB 인스턴스의 새로운 옵션 그룹을 선택합니다. DB 인스턴스를 변환할 때는 DB 인스턴스를 업그레이드할 때와 동일한 옵션 그룹 고려 사항이 적용됩니다.
9. 원하는 대로 모두 변경되었으면 [Continue]를 선택하고 수정 사항 요약을 확인합니다.
10. 즉시 적용을 선택합니다. 이 옵션은 다중 테넌트 구성으로 전환할 때 필요합니다. 경우에 따라 이 옵션을 선택하면 가동 중지가 발생할 수 있습니다.
11. 확인 페이지에서 변경 내용을 검토합니다. 내용이 정확할 경우 DB 인스턴스 수정을 선택합니다.

    또는 뒤로를 선택하여 변경 내용을 편집하거나 취소를 선택하여 변경 내용을 취소합니다.

AWS CLI

단일 테넌트 구성을 사용하는 CDB를 다중 테넌트 구성으로 변환하려면 modify-db-instance AWS CLI 명령에 `--multi-tenant`를 지정하세요.

다음 예시는 `my-st-cdb`라는 DB 인스턴스를 단일 테넌트 구성에서 다중 테넌트 구성으로 변환합니다. `--apply-immediately` 옵션은 필수입니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance --region us-east-1\
    --db-instance-identifier my-st-cdb \
    --multi-tenant \
    --apply-immediately
```

Windows의 경우:

```
aws rds modify-db-instance --region us-east-1 ^
```

```
    --db-instance-identifier my-st-cdb ^
    --multi-tenant ^
    --apply-immediately
```

출력은 다음과 같습니다.

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "my-st-cdb",
        "DBInstanceClass": "db.r5.large",
        "MultiTenant": false,
        "Engine": "oracle-ee-cdb",
        "DBResourceId": "db-AB1CDE2FGHIJK34LMNOPRLXTXU",
        "DBInstanceStatus": "modifying",
        "MasterUsername": "admin",
        "DBName": "ORCL",
        ...
        "EngineVersion": "19.0.0.0.ru-2022-01.rur-2022-01.r1",
        "AutoMinorVersionUpgrade": true,
        "ReadReplicaDBInstanceIdentifiers": [],
        "LicenseModel": "bring-your-own-license",
        "OptionGroupMemberships": [
            {
                "OptionGroupName": "default:oracle-ee-cdb-19",
                "Status": "in-sync"
            }
        ],
        ...
        "PendingModifiedValues": {
            "MultiTenant": "true"
        }
    }
}
```

CDB 인스턴스에 RDS for Oracle 테넌트 데이터베이스 추가

RDS for Oracle 다중 테넌트 구성에서 테넌트 데이터베이스는 PDB입니다. 테넌트 데이터베이스를 추가하려면 다음 필수 조건을 충족해야 합니다.

- CDB에 다중 테넌트 구성이 활성화되어 있습니다. 자세한 내용은 CDB 아키텍처의 다중 테넌트 구성 섹션을 참조하세요.
- 테넌트 데이터베이스를 생성할 IAM 권한이 있습니다.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 테넌트 데이터베이스를 추가할 수 있습니다. 한 번의 작업으로 여러 테넌트 데이터베이스를 추가할 수 없습니다. 한 번에 하나씩 추가해야 합니다. CDB에 백업 보존이 활성화된 경우 Amazon RDS는 새 테넌트 데이터베이스를 추가하기 전과 후에 DB 인스턴스를 백업합니다.

콘솔

DB 인스턴스에 테넌트 데이터베이스를 추가하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 콘솔의 오른쪽 상단에서 테넌트 데이터베이스를 생성하려는 AWS 리전을 선택합니다.
3. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
4. 테넌트 데이터베이스를 추가할 CDB 인스턴스를 선택합니다. DB 인스턴스는 CDB 아키텍처의 다중 테넌트 구성을 사용해야 합니다.
5. 작업을 선택한 다음 테넌트 데이터베이스 추가를 선택합니다.
6. 테넌트 데이터베이스 설정에서 다음을 수행합니다.
   - 테넌트 데이터베이스 이름에 새 PDB의 이름을 입력합니다.
   - 테넌트 데이터베이스 마스터 사용자 이름에 PDB의 마스터 사용자 이름을 입력합니다.
   - 다음 자격 증명 관리 옵션 중 하나를 선택합니다.
     - AWS Secrets Manager에서 관리됨

       암호화 키 선택에서 Secrets Manager가 만드는 KMS 키 또는 사용자가 만든 키를 선택합니다.

       > **Note**
       >
       > 자격 증명을 관리하기 위한 가장 안전한 기법으로 AWS Secrets Manager를 사용하는 것이 좋습니다. 추가 요금이 발생합니다. 자세한 내용은 Amazon RDS 및 AWS Secrets Manager를 통한 암호 관리 섹션을 참조하세요.

     - 자체 관리형

       암호를 지정하려면 암호 자동 생성 확인란을 선택 취소합니다(선택되어 있는 경우). 마스터 암호 및 마스터 암호 확인에 동일한 암호를 입력합니다.

- 추가 구성에서 초기 데이터베이스 이름에 대한 PDB의 이름을 입력합니다. 이름이 기본값 RDSCDB인 CDB에는 이름을 지정할 수 없습니다.
- 테넌트 데이터베이스 문자 집합에서 PDB의 문자 집합을 선택합니다. 기본값은 AL32UTF8입니다. CDB 문자 집합과 다른 PDB 문자 집합을 선택할 수 있습니다.
- 테넌트 데이터베이스 내셔널 문자 집합에서 PDB의 내셔널 문자 집합을 선택합니다. 기본값은 AL32UTF8입니다. 내셔널 문자 집합은 NCHAR 데이터 유형(NCHAR, NVARCHAR2 및NCLOB)을 사용하는 열의 인코딩만 지정하고 데이터베이스 메타데이터에는 영향을 주지 않습니다.

위 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 섹션을 참조하세요.

7. 테넌트 추가를 선택합니다.

AWS CLI

AWS CLI를 사용하여 CDB에 테넌트 데이터베이스를 추가하려면 다음 필수 파라미터와 함께 create-tenant-database 명령을 사용합니다.

- `--db-instance-identifier`
- `--tenant-db-name`
- `--master-username`
- `--master-user-password`

다음 예에서는 *my-cdb-inst*라는 RDS for Oracle CDB 인스턴스에 *mypdb2*라는 테넌트 데이터베이스를 생성합니다. PDB 문자 집합은 UTF-16입니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-tenant-database --region us-east-1 \
    --db-instance-identifier my-cdb-inst \
    --tenant-db-name mypdb2 \
    --master-username mypdb2-admin \
    --master-user-password mypdb2-pwd \
    --character-set-name UTF-16
```

Windows의 경우:

```
aws rds create-tenant-database --region us-east-1 \
    --db-instance-identifier my-cdb-inst ^
    --tenant-db-name mypdb2 ^
    --master-username mypdb2-admin ^
    --master-user-password mypdb2-pwd ^
    --character-set-name UTF-16
```

출력 결과는 다음과 비슷합니다.

```
...}
    "TenantDatabase" :
         {
            "DbiResourceId" : "db-abc123",
            "TenantDatabaseResourceId" : "tdb-bac567",
            "TenantDatabaseArn" : "arn:aws:rds:us-east-1:123456789012:db:my-cdb-
inst:mypdb2",
            "DBInstanceIdentifier" : "my-cdb-inst",
            "TenantDBName" : "mypdb2",
            "Status" : "creating",
            "MasterUsername" : "mypdb2",
            "CharacterSetName" : "UTF-16",
            ...
        }
}...
```

RDS for Oracle 테넌트 데이터베이스 수정

CDB에 있는 테넌트 데이터베이스의 PDB 이름과 마스터 사용자 암호만 수정할 수 있습니다. 다음과 같은 요구 사항 및 제한 사항에 유의하세요.

- DB 인스턴스의 테넌트 데이터베이스 설정을 수정하려면 테넌트 데이터베이스가 있어야 합니다.
- 한 번의 작업으로 여러 테넌트 데이터베이스를 수정할 수 없습니다. 한 번에 하나의 테넌트 데이터베이스만 수정할 수 있습니다.
- 테넌트 데이터베이스의 이름을 CDB$ROOT 또는 PDB$SEED로 변경할 수 없습니다.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 PDB를 수정할 수 있습니다.

콘솔

테넌트 데이터베이스의 PDB 이름 또는 마스터 암호를 수정하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 콘솔의 오른쪽 상단에서 테넌트 데이터베이스를 생성하려는 AWS 리전을 선택합니다.
3. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
4. 데이터베이스 이름 또는 마스터 사용자 암호를 수정하려는 테넌트 데이터베이스를 선택합니다.
5. 수정을 선택합니다.
6. 테넌트 데이터베이스 설정에서 다음 중 원하는 작업을 수행합니다.
   - 테넌트 데이터베이스 이름에 새 PDB의 새 이름을 입력합니다.
   - 테넌트 데이터베이스 마스터 암호에 새 암호를 입력합니다.
7. 테넌트 수정을 선택합니다.

AWS CLI

AWS CLI를 사용하여 테넌트 데이터베이스를 수정하려면 다음 파라미터를 사용하여 modify-tenant-database 명령을 호출합니다.

- `--db-instance-identifier #`
- `--tenant-db-name value`
- `[--new-tenant-db-name value]`
- `[--master-user-password value]`

다음 예에서는 테넌트 데이터베이스의 이름을 pdb1에서 my-cdb-inst DB 인스턴스의 pdb-hr로 변경합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-tenant-database --region us-east-1 \
    --db-instance-identifier my-cdb-inst \
    --tenant-db-name pdb1 \
```

```
    --new-tenant-db-name pdb-hr
```

Windows의 경우:

```
aws rds modify-tenant-database --region us-east-1 ^
    --db-instance-identifier my-cdb-inst ^
    --tenant-db-name pdb1 ^
    --new-tenant-db-name pdb-hr
```

다음과 비슷한 출력이 생성됩니다.

```
{
    "TenantDatabase" : {
        "DbiResourceId" : "db-abc123",
        "TenantDatabaseResourceId" : "tdb-bac567",
        "TenantDatabaseArn" : "arn:aws:rds:us-east-1:123456789012:db:my-cdb-inst:pdb1",
        "DBInstanceIdentifier" : "my-cdb-inst",
        "TenantDBName" : "pdb1",
        "Status" : "modifying",
        "MasterUsername" : "tenant-admin-user"
        "Port" : "6555",
        "CharacterSetName" : "UTF-16",
        "MaxAllocatedStorage" : "1000",
        "ParameterGroups": [
            {
                "ParameterGroupName": "pdb1-params",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        "OptionGroupMemberships": [
            {
                "OptionGroupName": "pdb1-options",
                "Status": "in-sync"
            }
        ],
        "PendingModifiedValues": {
            "TenantDBName": "pdb-hr"
        }
    }
}
```

CDB에서 RDS for Oracle 테넌트 데이터베이스 삭제

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 테넌트 데이터베이스(PDB)를 삭제할 수 있습니다. 다음 사전 조건 및 제한 사항을 고려하세요.

- 테넌트 데이터베이스와 DB 인스턴스가 있어야 합니다.
- 삭제에 성공하려면 다음 상황 중 하나가 있어야 합니다.
  - 테넌트 데이터베이스와 DB 인스턴스를 사용할 수 있어야 합니다.

    > **Note**
    >
    > `delete-tenant-database` 명령을 실행하기 전에 테넌트 데이터베이스와 DB 인스턴스가 사용 가능한 상태였던 경우에만 최종 스냅샷을 만들 수 있습니다.

  - 테넌트 데이터베이스를 생성하고 있습니다.
  - DB 인스턴스가 테넌트 데이터베이스를 수정하고 있습니다.
- 한 번의 작업으로 여러 테넌트 데이터베이스를 삭제할 수 없습니다.
- CDB의 유일한 테넌트인 테넌트 데이터베이스는 삭제할 수 없습니다.

콘솔

테넌트 데이터베이스를 삭제하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 다음 삭제하려는 테넌트 데이터베이스를 선택합니다.
3. [Actions]에 대해 [Delete]를 선택합니다.
4. DB 인스턴스의 최종 DB 스냅샷을 생성하려면 최종 스냅샷 생성 여부를 선택합니다.
5. 최종 스냅샷을 생성하도록 선택한 경우 최종 스냅샷 이름을 입력합니다.
6. 상자에 **`delete me`**를 입력합니다.
7. 삭제를 선택합니다.

AWS CLI

AWS CLI를 사용하여 테넌트 데이터베이스를 삭제하려면 다음 파라미터를 사용하여 delete-tenant-database 명령을 호출합니다.

- `--db-instance-identifier value`
- `--tenant-db-name value`
- `[--skip-final-snapshot | --no-skip-final-snapshot]`
- `[--final-snapshot-identifier value]`

다음 예에서는 *my-cdb-inst*라는 CDB에서 *pdb-test*라는 테넌트 데이터베이스를 삭제합니다. 기본적으로 이 작업은 최종 스냅샷을 생성합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds delete-tenant-database --region us-east-1 \
    --db-instance-identifier my-cdb-inst \
    --tenant-db-name pdb-test \
    --final-snapshot-identifier final-snap-pdb-test
```

Windows의 경우:

```
aws rds delete-tenant-database --region us-east-1 ^
    --db-instance-identifier my-cdb-inst ^
    --tenant-db-name pdb-test ^
    --final-snapshot-identifier final-snap-pdb-test
```

다음과 비슷한 출력이 생성됩니다.

```
{
    "TenantDatabase" : {
        "DbiResourceId" : "db-abc123",
        "TenantDatabaseResourceId" : "tdb-bac456",
        "TenantDatabaseArn" : "arn:aws:rds:us-east-1:123456789012:db:my-cdb-inst:pdb-
test",
        "DBInstanceIdentifier" : "my-cdb-inst",
        "TenantDBName" : "pdb-test",
        "Status" : "deleting",
        "MasterUsername" : "pdb-test-admin"
        "Port" : "6555",
        "CharacterSetName" : "UTF-16",
        "MaxAllocatedStorage" : "1000",
```

```
        "ParameterGroups": [
            {
                "ParameterGroupName": "tenant-1-params",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        "OptionGroupMemberships": [
            {
                "OptionGroupName": "tenant-1-options",
                "Status": "in-sync"
            }
        ]
    }
}
```

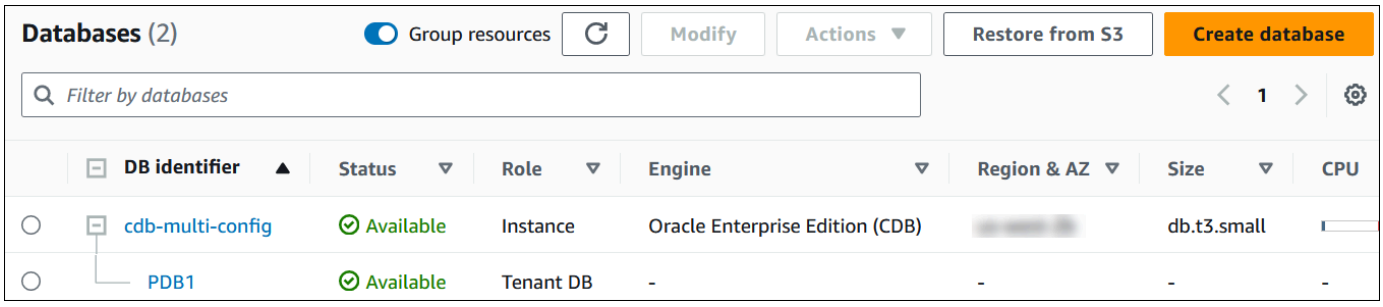
테넌트 데이터베이스 세부 정보 보기

비CDB 또는 CDB의 경우와 동일한 방식으로 테넌트 데이터베이스에 대한 세부 정보를 볼 수 있습니다.

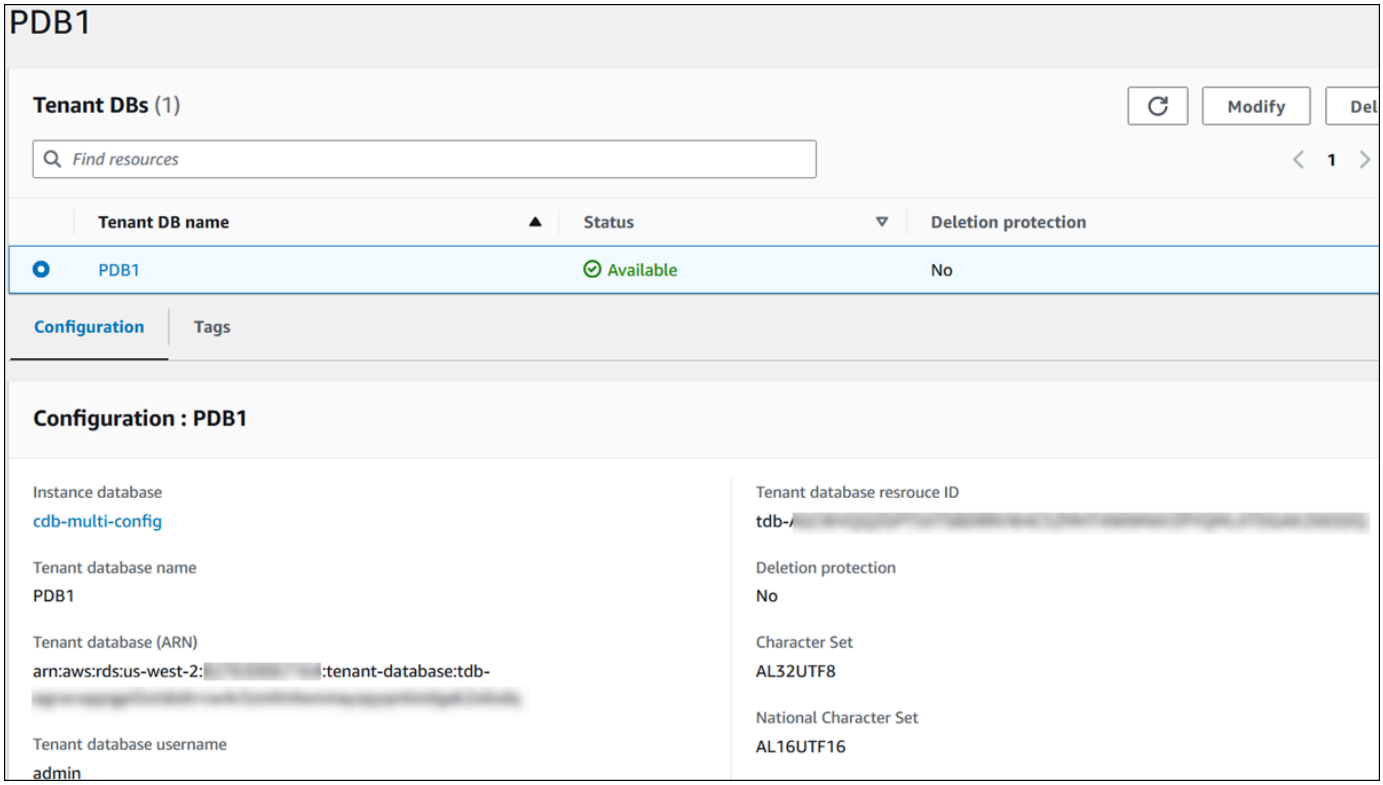
콘솔

테넌트 데이터베이스에 대한 세부 정보를 보는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 콘솔의 오른쪽 상단에서 DB 인스턴스가 상주하는 AWS 리전을 선택합니다.
3. 탐색 창에서 Databases(데이터베이스)를 선택합니다.



   위 이미지에서 단독 테넌트 데이터베이스(PDB)는 DB 인스턴스의 하위 데이터베이스로 나타납니다.
4. 데이터베이스의 이름을 선택합니다.



AWS CLI

PDB에 대한 세부 정보를 보려면 describe-tenant-databases 명령을 사용하세요.

이 예에서는 지정된 리전의 모든 테넌트 데이터베이스를 설명합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-tenant-databases --region us-east-1
```

Windows의 경우:

```
aws rds describe-tenant-databases --region us-east-1
```

다음과 비슷한 출력이 생성됩니다.

```
    "TenantDatabases" : [
        {
           "DBInstanceIdentifier" : "my-cdb-inst",
           "TenantDBName" : "pdb-test",
```

```
            "Status" : "available",
            "MasterUsername" : "pdb-test-admin",
            "DbiResourceId" : "db-abc123",
            "TenantDatabaseResourceId" : "tdb-bac456",
            "TenantDatabaseArn" : "arn:aws:rds:us-east-1:123456789012:db:my-cdb-
inst:pdb-test",
            "CharacterSetName": "AL32UTF8",
            "NcharCharacterSetName": "AL16UTF16",
            "DeletionProtection": false,
            "PendingModifiedValues": {
                 "MasterUserPassword": "****"
            },
            "TagList": []
         },
         {

            "DBInstanceIdentifier" : "my-cdb-inst2",
            "TenantDBName" : "pdb-dev",
            "Status" : "modifying",
            "MasterUsername" : "masterrdsuser"
            "DbiResourceId" : "db-xyz789",
            "TenantDatabaseResourceId" : "tdb-ghp890",
            "TenantDatabaseArn" : "arn:aws:rds:us-east-1:123456789012:db:my-cdb-
inst2:pdb-dev",
            "CharacterSetName": "AL32UTF8",
            "NcharCharacterSetName": "AL16UTF16",
            "DeletionProtection": false,
            "PendingModifiedValues": {
                 "MasterUserPassword": "****"
            },
            "TagList": []
         },
         ... other truncated data
```

다음 예에서는 지정된 리전의 `my-cdb-inst` DB 인스턴스에 있는 테넌트 데이터베이스를 설명합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-tenant-databases --region us-east-1 \
    --db-instance-identifier my-cdb-inst
```

Windows의 경우:

```
aws rds describe-tenant-databases --region us-east-1 ^
    --db-instance-identifier my-cdb-inst
```

다음과 비슷한 출력이 생성됩니다.

```
{
    "TenantDatabase": {
        "TenantDatabaseCreateTime": "2023-10-19T23:55:30.046Z",
        "DBInstanceIdentifier": "my-cdb-inst",
        "TenantDBName": "pdb-hr",
        "Status": "creating",
        "MasterUsername": "tenant-admin-user",
        "DbiResourceId": "db-abc123",
        "TenantDatabaseResourceId": "tdb-bac567",
        "TenantDatabaseARN": "arn:aws:rds:us-west-2:579508833180:pdb-hr:tdb-
abcdefghi1jklmno2p3qrst4uvw5xy6zabc7defghi8jklmn90op",
        "CharacterSetName": "AL32UTF8",
        "NcharCharacterSetName": "AL16UTF16",
        "DeletionProtection": false,
        "PendingModifiedValues": {
            "MasterUserPassword": "****"
        },
        "TagList": [
            {
                "Key": "TEST",
                "Value": "testValue"
            }
        ]
    }
}
```

다음 예에서는 미국 동부(버지니아 북부) 리전의 `my-cdb-inst` DB 인스턴스에 있는 `pdb1` 테넌트 데이터베이스를 설명합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-tenant-databases --region us-east-1 \
--db-instance-identifier my-cdb-inst \
```

```
--tenant-db-name pdb1
```

Windows의 경우:

```
aws rds describe-tenant-databases --region us-east-1 ^
--db-instance-identifier my-cdb-inst ^
--tenant-db-name pdb1
```

다음과 비슷한 출력이 생성됩니다.

```
{
    "TenantDatabases" : [
        {
            "DbiResourceId" : "db-abc123",
            "TenantDatabaseResourceId" : "tdb-bac567",
            "TenantDatabaseArn" : "arn:aws:rds:us-east-1:123456789012:db:my-cdb-
inst:pdb1"
            "DBInstanceIdentifier" : "my-cdb-inst",
            "TenantDBName" : "pdb1",
            "Status" : "ACTIVE",
            "MasterUsername" : "masterawsuser"
            "Port" : "1234",
            "CharacterSetName": "UTF-8",
            "ParameterGroups": [
                {
                    "ParameterGroupName": "tenant-custom-pg",
                    "ParameterApplyStatus": "in-sync"
                }
            ],
            {
            "OptionGroupMemberships": [
                {
                    "OptionGroupName": "tenant-custom-og",
                    "Status": "in-sync"
                }
            ]
         }
    ]
}
```

CDB 업그레이드

CDB를 다른 Oracle Database 릴리스로 업그레이드할 수 있습니다. 예를 들어 Oracle Database 19c CDB를 Oracle Database 21c CDB로 업그레이드할 수 있습니다. 업그레이드 중에는 데이터베이스 아키텍처를 변경할 수 없습니다. 따라서 비CDB를 CDB로 업그레이드하거나 CDB를 비CDB로 업그레이드할 수 없습니다.

CDB를 CDB로 업그레이드하는 절차는 비CDB를 비CDB로 업그레이드하는 절차와 동일합니다. 자세한 내용은 RDS for Oracle DB 엔진 업그레이드 단원을 참조하십시오.

RDS for Oracle DB 인스턴스 관리

다음은 RDS for Oracle DB 인스턴스와 관련하여 수행하는 일반적인 관리 작업입니다. 일부 작업은 모든 RDS DB 인스턴스에서 동일합니다. 다른 작업은 RDS for Oracle에만 해당됩니다.

다음 작업은 모든 RDS 데이터베이스에 공통적으로 적용되지만 Oracle 데이터베이스의 경우 특별한 고려 사항이 있습니다. 예를 들어, Oracle 클라이언트 SQL*Plus 및 SQL Developer를 사용하여 Oracle 데이터베이스에 연결합니다.

작업 영역	관련 설명서
인스턴스 클래스, 스토리지 및 PIOPS 프로덕션 인스턴스를 생성하는 경우 Amazon RDS에서 인스턴스 클래스, 스토리지 유형 및 프로비저닝된 IOPS가 작동하는 방식을 알아봅니다.	RDS for Oracle DB 인스턴스 클래스 Amazon RDS 스토리지 유형
다중 AZ 배포 프로덕션 DB 인스턴스에서는 다중 AZ 배포를 사용해야 합니다. 다중 AZ 배포는 DB 인스턴스를 위해 향상된 가용성, 데이터 내구성 및 내결함성을 제공합니다.	Amazon RDS에 대한 다중 AZ 배포 구성 및 관리
Amazon VPC AWS 계정에 기본 Virtual Private Cloud(VPC)가 있는 경우에는 DB 인스턴스가 기본 VPC 내부에 자동으로 생성됩니다. 계정에 기본 VPC가 없는데 VPC에 DB 인스턴스를 생성하려면 VPC와 서브넷 그룹을 생성한 후 인스턴스를 생성합니다.	VPC에서 DB 인스턴스를 사용한 작업
보안 그룹 기본적으로 DB 인스턴스는 액세스를 막는 방화벽을 사용합니다. 따라서 DB 인스턴스에 액세스하기 위한 알맞은 IP 주소와 네트워크 구성으로 보안 그룹을 생성해야 합니다.	보안 그룹을 통한 액세스 제어
파라미터 그룹 수(Parameter groups) DB 인스턴스에 특정 데이터베이스 파라미터가 필요할 경우, 파라미터 그룹을 만든 후 DB 인스턴스를 만듭니다.	Amazon RDS의 파라미터 그룹

작업 영역	관련 설명서
옵션 그룹 수 DB 인스턴스에 특정 데이터베이스 옵션이 필요할 경우, 옵션 그룹을 생성한 후 DB 인스턴스를 생성합니다.	Oracle DB 인스턴스에 옵션 추가
DB 인스턴스에 연결 보안 그룹을 생성하고 이를 DB 인스턴스에 연결한 후, Oracle SQL*Plus와 같은 표준 SQL 클라이언트 애플리케이션을 사용하여 DB 인스턴스에 연결할 수 있습니다.	Oracle DB 인스턴스에 연결
백업 및 복원 DB 인스턴스를 구성하여 자동 백업을 생성하거나 수동 스냅샷을 생성한 다음 백업 또는 스냅샷에서 인스턴스를 복원할 수 있습니다.	데이터 백업, 복원 및 내보내기
모니터링(Monitoring) CloudWatch Amazon RDS 측정치, 이벤트 및 향상된 모니터링 기능을 통해 Oracle DB 인스턴스를 모니터링할 수 있습니다.	Amazon RDS 콘솔에서 지표 보기 Amazon RDS 이벤트 보기
로그 파일 Oracle DB 인스턴스의 로그 파일에 액세스할 수 있습니다.	Amazon RDS 로그 파일 모니터링

다음에는 RDS Oracle에 대한 일반적인 DBA 태스크의 Amazon RDS 특정 구현에 대한 설명을 확인할 수 있습니다. 관리형 서비스 환경을 제공하기 위해 Amazon RDS는 DB 인스턴스에 대해 shell 액세스를 제공하지 않습니다. 또한 RDS는 고급 권한이 필요한 특정 시스템 절차와 테이블에 대한 액세스를 제한합니다. 많은 태스크에서는 Amazon RDS 전용 도구인 `rdsadmin` 패키지를 실행하여 데이터베이스를 관리할 수 있습니다.

다음은 Oracle을 실행 중인 DB 인스턴스에 대한 공통 DBA 작업입니다.

- 시스템 작업

세션 분리	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.disconnect`

	Oracle 메서드: `alter system disconnect session`
세션 종료	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.kill` Oracle 메서드: `alter system kill session`
세션에서 SQL 문 취소	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.cancel` Oracle 메서드: `alter system cancel sql`
제한 세션 활성화 및 비활성화	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.restricted_session` Oracle 메서드: `alter system enable restricted session`
공유 풀 플러시	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.flush_shared_pool` Oracle 메서드: `alter system flush shared_pool`
버퍼 캐시 플러시	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.flush_buffer_cache` Oracle 메서드: `alter system flush buffer_cache`
SYS 객체에 대한 SELECT 또는 EXECUTE 권한 부여	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.grant_sys_object` Oracle 메서드: `grant`
SYS 객체에 대한 SELECT 또는 EXECUTE 권한 취소	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.revoke_sys_object` Oracle 메서드: `revoke`
Oracle DB 인스턴스의 RDS_X$ 뷰 관리	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.create_sys_x$_view` Oracle 메서드: `CREATE VIEW`

마스터가 아닌 사용자에게 권한 부여	Amazon RDS 메서드: `grant`
사용자 지정 암호 확인 함수 생성	Amazon RDS 메서드: `rdsadmin.rdsadmin_password_verify.create_verify_function` Amazon RDS 메서드: `rdsadmin.rdsadmin_password_verify.create_passthrough_verify_fcn`
사용자 지정 DNS 서버 설정	—
허용되는 시스템 진단 이벤트 나열	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.list_allowed_system_events` Oracle 메서드: –
시스템 진단 이벤트 설정	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.set_allowed_system_events` Oracle 메서드: `ALTER SYSTEM SET EVENTS 'set_event_clause'`
설정된 시스템 진단 이벤트 나열	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.list_set_system_events` Oracle 메서드: `ALTER SESSION SET EVENTS 'IMMEDIATE EVENTDUMP(SYSTEM)'`
시스템 진단 이벤트 설정 해제	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.unset_system_event` Oracle 메서드: `ALTER SYSTEM SET EVENTS 'unset_event_clause'`

- 데이터베이스 작업

데이터베이스의 전역 이름 변경	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.rename_global_name` Oracle 메서드: `alter database rename`
테이블스페이스 생성과 크기 조정	Amazon RDS 메서드: `create tablespace` Oracle 메서드: `alter database`
기본 테이블스페이스 설정	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.alter_default_tablespace` Oracle 메서드: `alter database default tablespace`
기본 임시 테이블스페이스 설정	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.alter_default_temp_tablespace` Oracle 메서드: `alter database default temporary tablespace`
인스턴스 스토어에 임시 테이블스페이스 생성	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.create_inst_store_tmp_tblspace` Oracle 메서드: `create temporary tablespace`
데이터베이스 체크포인트	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.checkpoint` Oracle 메서드: `alter system checkpoint`
분산 복구 설정	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.enable_distr_recovery` Oracle 메서드: `alter system enable distributed recovery`

데이터베이스 시간대 설정	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.alter_db_time_zone` Oracle 메서드: `alter database set time_zone`
Oracle 외부 테이블 작업	—
Automatic Workload Repository(AWR)를 사용하여 성능 보고서 생성	Amazon RDS 메서드: `rdsadmin.rdsadmin_diagnostic_util` 프로시저 Oracle 메서드: `dbms_workload_repository` 패키지
VPC의 DB 인스턴스에 사용하기 위한 데이터베이스 링크 조정	—
DB 인스턴스의 기본 에디션 설정	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.alter_default_edition` Oracle 메서드: `alter database default edition`
SYS.AUD$ 테이블에 대한 감사 활성화	Amazon RDS 메서드: `rdsadmin.rdsadmin_master_util.audit_all_sys_aud_table` Oracle 메서드: `audit`
SYS.AUD$ 테이블에 대한 감사 비활성화	Amazon RDS 메서드: `rdsadmin.rdsadmin_master_util.noaudit_all_sys_aud_table` Oracle 메서드: `noaudit`
중단된 온라인 인덱스 빌드 정리	Amazon RDS 메서드: `rdsadmin.rdsadmin_dbms_repair.online_index_clean` Oracle 메서드: `dbms_repair.online_index_clean`
손상된 블록 건너뛰기	Amazon RDS 메서드: 몇 가지 `rdsadmin.rdsadmin_dbms_repair` 절차 Oracle 메서드: `dbms_repair` 패키지

테이블스페이스, 데이터 파일, 임시 파일 크기 조정	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.resize_temp_tablespace` , `rdsadmin.rdsadmin_util.resize_tempfile` 또는 `rdsadmin.rdsadmin_util.autoextend_tempfile` 프로시저 `rdsadmin.rdsadmin_util.resize_datafile` 또는 `rdsadmin.rdsadmin_util.autoextend_datafile` 프로시저 Oracle 메서드: –
휴지통 비우기	Amazon RDS 메서드: `EXEC rdsadmin.rdsadmin_util.purge_dba_recyclebin` Oracle 메서드: `purge dba_recyclebin`
전체 수정을 위한 기본 표시 값 설정	Amazon RDS 메서드: `EXEC rdsadmin.rdsadmin_util.dbms_redact_upd_full_rdct_val` Oracle 메서드: `exec dbms_redact.UPDATE_FULL_REDACTION_VALUES`

- 로그 작업

강제 로깅 설정	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.force_logging` Oracle 메서드: `alter database force logging`

보충 로깅 설정	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.alter_supplemental_logging` Oracle 메서드: `alter database add supplemental log`
온라인 로그 파일 전환	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.switch_logfile` Oracle 메서드: `alter system switch logfile`
온라인 다시 실행 로그 추가	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.add_logfile`
온라인 다시 실행 로그 드롭	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.drop_logfile`
온라인 다시 실행 로그 크기 조절	—
보관된 다시 실행 로그 보존	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.set_configuration`

Amazon S3에서 아카이빙된 다시 실행 로그 다운로드	Amazon RDS 메서드: `rdsadmin.rdsadmin_archive_log_download.download_log_with_seqnum` Amazon RDS 메서드: `rdsadmin.rdsadmin_archive_log_download.download_logs_in_seqnum_range`
온라인 및 아카이빙된 다시 실행 로그 액세스	Amazon RDS 메서드: `rdsadmin.rdsadmin_master_util.create_archivelog_dir` Amazon RDS 메서드: `rdsadmin.rdsadmin_master_util.create_onlinelog_dir`

- RMAN 작업

RDS for Oracle DB에서 데이터베이스 파일 검증	Amazon RDS 메서드: `rdsadmin_rman_util. *procedure*` Oracle 메서드: `RMAN VALIDATE`

블록 변경 추적 활성화 및 비활성화	Amazon RDS 메서드: `rdsadmin_rman_util . procedure` Oracle 메서드: `ALTER DATABASE`
보관된 재실행 로그 대조 확인	Amazon RDS 메서드: `rdsadmin_rman_util .crosscheck_archivelog` Oracle 메서드: `RMAN BACKUP`
보관된 재실행 로그 파일 백업	Amazon RDS 메서드: `rdsadmin_rman_util . procedure` Oracle 메서드: `RMAN BACKUP`
전체 데이터베이스 백업 수행	Amazon RDS 메서드: `rdsadmin_rman_util .backup_database_full` Oracle 메서드: `RMAN BACKUP`
증분 데이터베이스 백업 수행	Amazon RDS 메서드: `rdsadmin_rman_util .backup_database_incremental` Oracle 메서드: `RMAN BACKUP`

테이블스페이스 백업	Amazon RDS 메서드: `rdsadmin_rman_util.backup_database_tablespace` Oracle 메서드: `RMAN BACKUP`

- Oracle Scheduler 작업

DBMS_SCHEDULER 작업 변경	Amazon RDS 메서드: `dbms_scheduler.set_attribute` Oracle 메서드: `dbms_scheduler.set_attribute`
AutoTask 유지 관리 기간 수정	Amazon RDS 메서드: `dbms_scheduler.set_attribute` Oracle 메서드: `dbms_scheduler.set_attribute`
Oracle Scheduler 작업의 시간대 설정	Amazon RDS 메서드: `dbms_scheduler.set_scheduler_attribute` Oracle 메서드: `dbms_scheduler.set_scheduler_attribute`

SYS가 소유한 Oracle Scheduler 작업 해제	Amazon RDS 메서드: `rdsadmin.rdsadmin_dbms_scheduler.disable` Oracle 메서드: `dbms_scheduler.disable`
SYS가 소유한 Oracle Scheduler 작업 켜기	Amazon RDS 메서드: `rdsadmin.rdsadmin_dbms_scheduler.enable` Oracle 메서드: `dbms_scheduler.enable`
CALENDAR 형식 작업에 대한 Oracle Scheduler 반복 간격 수정	Amazon RDS 메서드: `rdsadmin.rdsadmin_dbms_scheduler.set_attribute` Oracle 메서드: `dbms_scheduler.set_attribute`
NAMED 형식 작업에 대한 Oracle Scheduler 반복 간격 수정	Amazon RDS 메서드: `rdsadmin.rdsadmin_dbms_scheduler.set_attribute` Oracle 메서드: `dbms_scheduler.set_attribute`

Oracle Scheduler 작업 생성을 위한 자동 커밋 해제	Amazon RDS 메서드: `rdsadmin.rdsadmin_dbms_scheduler.set_no_commit_flag` Oracle 메서드: `dbms_isched.set_no_commit_flag`

- 문제 진단

인시던트 나열	Amazon RDS 메서드: `rdsadmin.rdsadmin_adrci_util.list_adrci_incidents` Oracle 메서드: ADRCI 명령 `show incident`
문제 나열	Amazon RDS 메서드: `rdsadmin.rdsadmin_adrci_util.list_adrci_problem` Oracle 메서드: ADRCI 명령 `show problem`
인시던트 패키지 생성	Amazon RDS 메서드: `rdsadmin.rdsadmin_adrci_util.create_adrci_package` Oracle 메서드: ADRCI 명령 `ips create package`

추적 파일 표시	Amazon RDS 메서드: `rdsadmin.rdsadmin_adrci_util.show_adrci_tracefile` Oracle 메서드: ADRCI 명령 `show tracefile`

- 기타 작업

메인 데이터 스토리지 공간에서 디렉터리 생성 및 삭제	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.create_directory` Oracle 메서드: `CREATE DIRECTORY` Amazon RDS 메서드: `rdsadmin.rdsadmin_util.drop_directory` Oracle 메서드: `DROP DIRECTORY`
DB 인스턴스 디렉터리의 파일 목록 표시	Amazon RDS 메서드: `rdsadmin.rds_file_util.listdir` Oracle 메서드: –
DB 인스턴스 디렉터리의 파일 목록 읽기	Amazon RDS 메서드: `rdsadmin.rds_file_util.read_text_file` Oracle 메서드: –

Opatch 파일 액세스	Amazon RDS 메서드: `rdsadmin.rds_file_util.read_text_file` 또는 `rdsadmin.tracefile_listing` Oracle 메서드: `opatch`
Advisor 작업에 대한 파라미터 설정	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.advisor_task_set_parameter` Oracle 메서드: 다양한 저장 패키지 프로시저
AUTO_STATS_ADVISOR_TASK 비활성화	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.advisor_task_drop` Oracle 메서드: –
AUTO_STATS_ADVISOR_TASK 다시 활성화	Amazon RDS 메서드: `rdsadmin.rdsadmin_util.dbms_stats_init` Oracle 메서드: –

Amazon S3과 Oracle의 통합 및 OEM 관리 에이전트 데이터베이스 작업 실행에 Amazon RDS 프로시저를 사용할 수도 있습니다. 자세한 내용은 Amazon S3 통합 및 Management Agent를 사용하여 데이터베이스 작업 수행 단원을 참조하세요.

Oracle DB 인스턴스에 대한 공통 시스템 작업 수행

그 다음에는 Oracle을 실행하는 Amazon RDS DB 인스턴스에서 시스템과 관련된 특정 공통 DBA 작업을 수행하는 방법을 알아봅니다. 관리형 서비스 환경을 제공하기 위해 Amazon RDS는 DB 인스턴스에

대해 shell 액세스를 제공하지 않으며, 고급 권한이 필요한 특정 시스템 절차와 테이블에 대한 액세스를 제한합니다.

주제

- 세션 분리
- 세션 종료
- 세션에서 SQL 문 취소
- 제한 세션 활성화 및 비활성화
- 공유 풀 플러시
- SYS 객체에 대한 SELECT 또는 EXECUTE 권한 부여
- SYS 객체에 대한 SELECT 또는 EXECUTE 권한 취소
- Oracle DB 인스턴스의 RDS_X$ 뷰 관리
- 마스터가 아닌 사용자에게 권한 부여
- 사용자 지정 암호 확인 함수 생성
- 사용자 지정 DNS 서버 설정
- 시스템 진단 이벤트 설정 및 설정 해제

세션 분리

전용 서버 프로세스를 종료하여 현재 세션을 분리하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.disconnect`를 사용합니다. `disconnect` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
`sid`	숫자	—	예	세션 식별자입니다.
`serial`	숫자	—	예	세션 일련번호입니다.
`method`	varchar	`'IMMEDIATE'`	아니요	유효한 값은 `'IMMEDIATE'` 또는 `'POST_TRANSACTION'` 입니다.

다음 예제에서는 세션을 분리합니다.

```
begin
    rdsadmin.rdsadmin_util.disconnect(
        sid    => sid,
        serial => serial_number);
end;
/
```

세션 식별자와 세션 일련번호를 얻으려면, V$SESSION 뷰를 쿼리하세요. 다음 예제에서는 사용자 AWSUSER의 모든 세션을 확보합니다.

```
SELECT SID, SERIAL#, STATUS FROM V$SESSION WHERE USERNAME = 'AWSUSER';
```

이 메서드를 사용하려면 데이터베이스가 열려 있어야 합니다. 세션 분리에 대한 자세한 내용은 Oracle 문서에서 ALTER SYSTEM 단원을 참조하세요.

세션 종료

세션을 종료하려면 Amazon RDS 프로시저 rdsadmin.rdsadmin_util.kill을 사용합니다. kill 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
sid	숫자	—	예	세션 식별자입니다.
serial	숫자	—	예	세션 일련번호입니다.
method	varchar	null	아니요	유효한 값은 'IMMEDIATE' 또는 'PROCESS' 입니다. IMMEDIATE 를 지정하면 다음 문을 실행하는 것과 같은 효과가 있습니다. `ALTER SYSTEM KILL SESSION 'sid,serial#' IMMEDIATE`

파라미터 이름	데이터 형식	기본값	필수	설명
				PROCESS를 지정하면 세션과 연결된 프로세스를 종료합니다. IMMEDIATE를 사용하여 세션이 종료되지 못한 경우에만 PROCESS를 지정하세요.

세션 식별자와 세션 일련번호를 얻으려면, V$SESSION 뷰를 쿼리하세요. 다음 예제에서는 사용자 *AWSUSER*의 모든 세션을 확보합니다.

```
SELECT SID, SERIAL#, STATUS FROM V$SESSION WHERE USERNAME = 'AWSUSER';
```

다음 예제에서는 세션을 종료합니다.

```
BEGIN
    rdsadmin.rdsadmin_util.kill(
        sid    => sid,
        serial => serial_number,
        method => 'IMMEDIATE');
END;
/
```

다음 예제에서는 세션과 연결된 프로세스를 종료합니다.

```
BEGIN
    rdsadmin.rdsadmin_util.kill(
        sid    => sid,
        serial => serial_number,
        method => 'PROCESS');
END;
/
```

세션에서 SQL 문 취소

세션에서 SQL 문을 취소하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.cancel`을 사용합니다.

> **Note**
>
> 이 절차는 Oracle Database 19c(19.0.0)와 RDS for Oracle의 모든 상위 메이저 및 마이너 버전에서 지원됩니다.

`cancel` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
`sid`	숫자	—	예	세션 식별자입니다.
`serial`	숫자	—	예	세션 일련번호입니다.
`sql_id`	varchar2	null	아니요	SQL 문의 SQL 식별자입니다.

다음 예제는 세션에서 SQL 문을 취소합니다.

```
begin
    rdsadmin.rdsadmin_util.cancel(
        sid    => sid,
        serial => serial_number,
        sql_id => sql_id);
end;
/
```

세션 식별자, 세션 일련 번호 및 SQL의 SQL 식별자를 가져오려면 `V$SESSION` 보기를 쿼리합니다. 다음 예제에서는 사용자 `AWSUSER`의 모든 세션 및 SQL 식별자를 가져옵니다.

```
select SID, SERIAL#, SQL_ID, STATUS from V$SESSION where USERNAME = 'AWSUSER';
```

제한 세션 활성화 및 비활성화

제한 세션을 활성화 또는 비활성화하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.restricted_session`을 사용합니다. `restricted_session` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	예	설명
p_enable	부울	true	아니요	true는 제한 세션을 활성화하고, false는 제한 세션을 비활성화하도록 설정됩니다.

다음은 제한 세션을 활성화 및 비활성화하는 방법을 나타낸 예제입니다.

```
/* Verify that the database is currently unrestricted. */

SELECT LOGINS FROM V$INSTANCE;

LOGINS
-------
ALLOWED

/* Enable restricted sessions */

EXEC rdsadmin.rdsadmin_util.restricted_session(p_enable => true);


/* Verify that the database is now restricted. */

SELECT LOGINS FROM V$INSTANCE;

LOGINS
----------
RESTRICTED


/* Disable restricted sessions */

EXEC rdsadmin.rdsadmin_util.restricted_session(p_enable => false);


/* Verify that the database is now unrestricted again. */

SELECT LOGINS FROM V$INSTANCE;
```

```
LOGINS
-------
ALLOWED
```

공유 풀 플러시

공유 풀을 플러시하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.flush_shared_pool`을 사용합니다. `flush_shared_pool` 프로시저에는 파라미터가 없습니다.

다음 예제는 공유 풀을 플러시합니다.

```
EXEC rdsadmin.rdsadmin_util.flush_shared_pool;
```

버퍼 캐시 플러시

버퍼 캐시를 플러시하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.flush_buffer_cache`를 사용합니다. `flush_buffer_cache` 프로시저에는 파라미터가 없습니다.

다음 예제는 버퍼 캐시를 플러시합니다.

```
EXEC rdsadmin.rdsadmin_util.flush_buffer_cache;
```

데이터베이스 스마트 플래시 캐시 플러시

데이터베이스 스마트 캐시를 플러시하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.flush_flash_cache`를 사용합니다. `flush_flash_cache` 프로시저에는 파라미터가 없습니다. 다음 예는 데이터베이스 스마트 플래시 캐시를 플러시합니다.

```
EXEC rdsadmin.rdsadmin_util.flush_flash_cache;
```

RDS for Oracle로 데이터베이스 스마트 플래시 캐시를 사용하는 방법에 대한 자세한 내용은 RDS for Oracle 인스턴스 스토어에 임시 데이터 저장 섹션을 참조하세요.

SYS 객체에 대한 SELECT 또는 EXECUTE 권한 부여

일반적으로 여러 객체가 포함되어 있을 수 있는 역할을 사용하여 권한을 이전합니다. 단일 객체에 권한을 부여하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.grant_sys_object`를 사용합니다. 이 프로시저는 마스터 사용자가 역할이나 직접 부여를 통해 이미 부여받은 권한만 부여합니다.

grant_sys_object 프로시저에는 다음과 같은 파라미터가 있습니다.

> **Important**
>
> 대소문자를 구분하는 식별자로 사용자를 생성하지 않는 한 모든 파라미터 값에 대문자를 사용합니다. 예를 들어 CREATE USER myuser 또는 CREATE USER MYUSER를 실행하는 경우 데이터 딕셔너리에 MYUSER가 저장됩니다. 그러나 CREATE USER "MyUser"에서 큰 따옴표를 사용하는 경우 데이터 딕셔너리를 MyUser에 저장합니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_obj_name	varchar2	—	예	권한을 부여할 객체의 이름입니다. 디렉터리, 기능, 패키지, 프로시저, 시퀀스, 테이블, 뷰가 객체가 될 수 있습니다. 객체 이름은 DBA_OBJECTS 에 표시된 대로 정확하게 입력해야 합니다. 대부분의 시스템 객체는 대문자로 정의되므로 먼저 대문자로 시도하는 것이 좋습니다.
p_grantee	varchar2	—	예	권한을 부여할 객체의 이름입니다. 스키마나 역할이 객체가 될 수 있습니다.
p_privilege	varchar2	null	예	—
p_grant_option	boolean	false	아니요	true는 부여 옵션과 함께 사용하도록 설정됩니다.

다음 예에서는 V_$SESSION이라는 객체에 대한 선택 권한을 USER1이라는 사용자에게 부여합니다.

```
begin
```

```
    rdsadmin.rdsadmin_util.grant_sys_object(
        p_obj_name  => 'V_$SESSION',
        p_grantee   => 'USER1',
        p_privilege => 'SELECT');
end;
/
```

다음 예에서는 V_$SESSION 이라는 객체에 대한 선택 권한을 USER1이라는 사용자에게 부여 옵션과 함께 부여합니다.

```
begin
    rdsadmin.rdsadmin_util.grant_sys_object(
        p_obj_name     => 'V_$SESSION',
        p_grantee      => 'USER1',
        p_privilege    => 'SELECT',
        p_grant_option => true);
end;
/
```

객체에 권한을 부여할 수 있으려면 해당 권한을 부여 옵션을 통해 계정에 직접 부여하거나 `with admin option`을 사용해 부여된 역할을 통해 계정에 부여해야 합니다. 대부분 이미 SELECT 역할에 부여된 DBA 뷰에 SELECT_CATALOG_ROLE 권한을 부여하는 경우가 많습니다. 해당 역할이 아직 `with admin option`을 통해 사용자에게 직접 부여되지 않았다면 권한을 양도할 수 없습니다. DBA 권한이 있다면 역할을 직접 다른 사용자에게 부여할 수 있습니다.

다음은 SELECT_CATALOG_ROLE과 EXECUTE_CATALOG_ROLE을 USER1에게 부여하는 예제입니다. `with admin option`을 사용했기 때문에, USER1은 이제 SELECT_CATALOG_ROLE에 부여된 SYS 객체 액세스 권한을 부여할 수 있습니다.

```
GRANT SELECT_CATALOG_ROLE TO USER1 WITH ADMIN OPTION;
GRANT EXECUTE_CATALOG_ROLE to USER1 WITH ADMIN OPTION;
```

PUBLIC에 이미 부여된 객체는 다시 부여할 필요가 없습니다. `grant_sys_object` 프로시저를 사용하여 액세스 권한을 다시 부여한다면, 프로시저는 성공을 호출합니다.

SYS 객체에 대한 SELECT 또는 EXECUTE 권한 취소

단일 객체에 대한 권한을 취소하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.revoke_sys_object`를 사용합니다. 이 프로시저는 마스터 계정이 역할 또는 직접 부여를 통해 이미 부여받은 권한만 취소합니다.

revoke_sys_object 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_obj_name	varchar2	—	예	권한을 취소할 객체의 이름입니다. 디렉터리, 기능, 패키지, 프로시저, 시퀀스, 테이블, 뷰가 객체가 될 수 있습니다. 객체 이름은 DBA_OBJECTS 에 표시된 대로 정확하게 입력해야 합니다. 대부분의 시스템 객체는 대문자로 정의되므로 먼저 대문자로 시도해 보는 것이 좋습니다.
p_revokee	varchar2	—	예	권한을 취소할 객체의 이름입니다. 스키마나 역할이 객체가 될 수 있습니다.
p_privilege	varchar2	null	예	—

다음 예에서는 V_$SESSION이라는 객체에 대한 선택 권한을 USER1이라는 사용자에게서 취소합니다.

```
begin
    rdsadmin.rdsadmin_util.revoke_sys_object(
        p_obj_name  => 'V_$SESSION',
        p_revokee   => 'USER1',
        p_privilege => 'SELECT');
end;
/
```

Oracle DB 인스턴스의 RDS_X$ 뷰 관리

SYS를 통해서만 액세스할 수 있는 SYS.X$ 고정 테이블에 액세스해야 할 수도 있습니다. 대상이 되는 X$ 표에 SYS.RDS_X$ 뷰를 만들려면 rdsadmin.rdsadmin_util 패키지의 프로시저를 사용합니다.

마스터 사용자에게 RDS_X$ 뷰에 대한 SELECT … WITH GRANT OPTION 권한이 자동으로 부여됩니다.

rdsadmin.rdsadmin_util 프로시저는 다음 데이터베이스 엔진 버전에서 사용할 수 있습니다.

- Oracle Database 21c 버전 21.0.0.0.ru-2023-10.rur-2023-10.r1 이상
- Oracle Database 19c 버전 19.0.0.0.ru-2023-10.rur-2023-10.r1 이상

> **Important**
>
> rdsadmin.rdsadmin_util 패키지는 내부적으로 X$ 표에 뷰를 생성합니다. X$ 표는 Oracle Database 설명서에 설명되어 있지 않은 내부 시스템 객체입니다. 비 프로덕션 데이터베이스에서 특정 뷰를 테스트하고 Oracle Support의 안내에 따라 프로덕션 데이터베이스에서만 뷰를 생성하는 것이 좋습니다.

RDS_X$ 뷰에서 사용할 수 있는 X$ 고정 표 나열

RDS_X$ 뷰에 사용할 수 있는 X$ 표를 나열하려면 RDS 프로시저 rdsadmin.rdsadmin_util.list_allowed_sys_x$_views를 사용합니다. 이 프로시저는 파라미터를 받지 않습니다. 다음 문에는 해당하는 모든 X$ 표가 나열되어 있습니다(샘플 출력 포함).

```
SQL> SET SERVEROUTPUT ON
SQL> SELECT * FROM TABLE(rdsadmin.rdsadmin_util.list_allowed_sys_x$_views);

'X$BH'
'X$K2GTE'
'X$KCBWBPD'
'X$KCBWDS'
'X$KGLLK'
'X$KGLOB'
'X$KGLPN'
'X$KSLHOT'
'X$KSMSP'
'X$KSPPCV'
'X$KSPPI'
'X$KSPPSV'
'X$KSQEQ'
'X$KSQRS'
'X$KTUXE'
```

```
'X$KQRFP'
```

대상 X$ 표 목록은 시간이 지남에 따라 변경될 수 있습니다. 대상 X$ 고정 표 목록을 최신으로 유지하려면 주기적으로 `list_allowed_sys_x$_views`를 다시 실행하세요.

SYS.RDS_X$ 뷰 생성

대상이 되는 X$ 표에 RDS_X$ 뷰를 만들려면 RDS 프로시저 `rdsadmin.rdsadmin_util.create_sys_x$_view`를 사용합니다. `rdsadmin.rdsadmin_util.list_allowed_sys_x$_views`의 출력에 나열된 표에 대한 뷰만 생성할 수 있습니다. `create_sys_x$_view` 프로시저는 다음 파라미터를 수용합니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_x$_tbl	varchar2	Null	예	유효한 X$ 표 이름입니다. 이 값은 `list_allowed_sys_x$_views` 에서 보고한 X$ 표 중 하나여야 합니다.
p_force_creation	불린(Boolean)	FALSE	아니요	X$ 표에 이미 존재하는 RDS_X$ 뷰를 강제로 생성할지 여부를 나타내는 값입니다. 기본적으로 RDS는 뷰가 이미 있는 경우 뷰를 만들지 않습니다. 강제로 생성하려면 이 파라미터를 TRUE로 설정합니다.

다음 예제에서는 X$KGLOB 표에 SYS.RDS_X$KGLOB 뷰를 생성합니다. 뷰 이름 형식은 RDS_*X$tablename*입니다.

```
SQL> SET SERVEROUTPUT ON
SQL> EXEC rdsadmin.rdsadmin_util.create_sys_x$_view('X$KGLOB');

PL/SQL procedure successfully completed.
```

다음 데이터 사전 쿼리는 SYS.RDS_X$KGLOB 뷰를 나열하고 상태를 보여줍니다. 마스터 사용자에게 이 뷰에 대한 SELECT ... WITH GRANT OPTION 권한이 자동으로 부여됩니다.

```
SQL> SET SERVEROUTPUT ON
SQL> COL OWNER FORMAT A30
SQL> COL OBJECT_NAME FORMAT A30
SQL> COL STATUS FORMAT A30
SQL> SET LINESIZE 200
SQL> SELECT OWNER, OBJECT_NAME, STATUS
FROM DBA_OBJECTS
WHERE OWNER = 'SYS' AND OBJECT_NAME = 'RDS_X$KGLOB';

OWNER                          OBJECT_NAME                    STATUS
------------------------------ ------------------------------
 ------------------------------
SYS                            RDS_X$KGLOB                    VALID
```

> **Important**
>
> 업그레이드 전과 후에 X$ 표가 동일하게 유지된다고 보장할 수는 없습니다. RDS for Oracle은 엔진 업그레이드 중에 X$ 표에서 RDS_X$ 뷰를 삭제하고 다시 생성합니다. 그런 다음 마스터 사용자에게 SELECT ... WITH GRANT OPTION 권한을 부여합니다. 업그레이드 후에는 해당 RDS_X$ 뷰에서 필요에 따라 데이터베이스 사용자에게 권한을 부여합니다.

SYS.RDS_X$ 뷰 목록

기존 RDS_X$ 뷰를 나열하려면 RDS 프로시저 rdsadmin.rdsadmin_util.list_created_sys_x$_views를 사용합니다. 프로시저에는 create_sys_x$_view 프로시저로 만든 뷰만 나열됩니다. 다음 예제에서는 해당하는 RDS_X$ 뷰(샘플 출력 포함)가 있는 X$ 표를 나열합니다.

```
SQL> SET SERVEROUTPUT ON
SQL> COL XD_TBL_NAME FORMAT A30
SQL> COL STATUS FORMAT A30
SQL> SET LINESIZE 200
SQL> SELECT * FROM TABLE(rdsadmin.rdsadmin_util.list_created_sys_x$_views);

XD_TBL_NAME                    STATUS
------------------------------ ------------------------------
X$BH                           VALID
X$K2GTE                        VALID
```

```
X$KCBWBPD                      VALID

3 rows selected.
```

RDS_X$ 뷰 삭제

SYS.RDS_X$ 뷰를 삭제하려면 RDS 프로시저 rdsadmin.rdsadmin_util.drop_sys_x$_view를 사용합니다. rdsadmin.rdsadmin_util.list_allowed_sys_x$_views의 출력에 나열된 뷰만 삭제할 수 있습니다. drop_sys_x$_view 프로시저는 다음 파라미터를 받습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_x$_tbl	varchar2	Null	예	유효한 X$ 고정 표 이름입니다. 이 값은 list_created_sys_x$_views 에서 보고한 X$ 고정 표 중 하나여야 합니다.

다음 예제에서는 X$KGLOB 표에 만든 RDS_X$KGLOB 뷰를 삭제합니다.

```
SQL> SET SERVEROUTPUT ON
SQL> EXEC rdsadmin.rdsadmin_util.drop_sys_x$_view('X$KGLOB');

PL/SQL procedure successfully completed.
```

다음 예제에서는 SYS.RDS_X$KGLOB 뷰가 삭제되었음을 보여줍니다(샘플 출력 포함).

```
SQL> SET SERVEROUTPUT ON
SQL> COL OWNER FORMAT A30
SQL> COL OBJECT_NAME FORMAT A30
SQL> COL STATUS FORMAT A30
SQL> SET LINESIZE 200
SQL> SELECT OWNER, OBJECT_NAME, STATUS
FROM DBA_OBJECTS
WHERE OWNER = 'SYS' AND OBJECT_NAME = 'RDS_X$KGLOB';

no rows selected
```

마스터가 아닌 사용자에게 권한 부여

SYS 역할을 사용하여 SELECT_CATALOG_ROLE 스키마에 있는 여러 객체에 선택 권한을 부여할 수 있습니다. SELECT_CATALOG_ROLE 역할은 사용자에게 데이터 딕셔너리 뷰에 대한 SELECT 권한을 부여합니다. 다음은 역할 SELECT_CATALOG_ROLE을 사용자 user1에 부여하는 예제입니다.

```
GRANT SELECT_CATALOG_ROLE TO user1;
```

EXECUTE 역할을 사용하여 SYS 스키마에 있는 여러 객체에 EXECUTE_CATALOG_ROLE 권한을 부여할 수 있습니다. EXECUTE_CATALOG_ROLE 역할은 사용자에게 데이터 딕셔너리의 패키지와 프로시저에 대한 EXECUTE 권한을 부여합니다. 다음 예에서는 EXECUTE_CATALOG_ROLE이라는 역할을 user1이라는 사용자에게 부여합니다.

```
GRANT EXECUTE_CATALOG_ROLE TO user1;
```

다음 예에서는 SELECT_CATALOG_ROLE 및 EXECUTE_CATALOG_ROLE이라는 역할에서 허용하는 권한을 얻습니다.

```
  SELECT *
    FROM ROLE_TAB_PRIVS
   WHERE ROLE IN ('SELECT_CATALOG_ROLE','EXECUTE_CATALOG_ROLE')
ORDER BY ROLE, TABLE_NAME ASC;
```

다음 예에서는 sh.sales라는 데이터베이스에서 마스터 사용자가 아닌 user1이라는 사용자를 생성한 후 CREATE SESSION 권한과 SELECT 권한을 부여합니다.

```
CREATE USER user1 IDENTIFIED BY PASSWORD;
GRANT CREATE SESSION TO user1;
GRANT SELECT ON sh.sales TO user1;
```

사용자 지정 암호 확인 함수 생성

사용자 지정 암호 확인 함수는 다음과 같은 방법으로 생성할 수 있습니다.

- 표준 확인 로직을 사용하고 SYS 스키마에 함수를 저장하려면, create_verify_function 프로시저를 사용합니다.
- 사용자 지정 확인 로직을 사용하거나 SYS 스키마에 함수를 저장하지 않으려면, create_passthrough_verify_fcn 프로시저를 사용합니다.

create_verify_function 프로시저

Amazon RDS 프로시저 `rdsadmin.rdsadmin_password_verify.create_verify_function`을 사용하여 사용자 지정 암호 확인 함수를 만들 수 있습니다. 이 `create_verify_function` 프로시저는 RDS for Oracle의 모든 버전에 대해 지원됩니다.

`create_verify_function` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
`p_verify_function_name`	varchar2	—	예	사용자 정의 함수 이름입니다. 이 함수는 SYS 스키마에서 자동으로 만들어집니다. 이 함수를 사용자 프로필에 할당합니다.
`p_min_length`	숫자	8	아니요	필요한 최소 문자 수입니다.
`p_max_length`	숫자	256	아니요	허용되는 최대 문자 수입니다.
`p_min_letters`	숫자	1	아니요	필요한 최소 글자 수입니다.
`p_min_uppercase`	숫자	0	아니요	필요한 최소 대문자 수입니다.
`p_min_lowercase`	숫자	0	아니요	필요한 최소 소문자 수입니다.
`p_min_digits`	숫자	1	아니요	필요한 최소 자릿수 숫자입니다.
`p_min_special`	숫자	0	아니요	필요한 최소 특수문자 수입니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_min_different_chars	숫자	3	아니요	이전 암호와 새 암호 간에 필요한 서로 다른 문자의 최소 개수입니다.
p_disallow_username	부울	true	아니요	true로 설정되어 암호에 사용자 이름을 사용할 수 없습니다.
p_disallow_reverse	부울	true	아니요	true로 설정되어 암호에 사용자 이름 반전을 사용할 수 없습니다.
p_disallow_db_name	부울	true	아니요	true로 설정되어 암호에 데이터베이스나 서버 이름을 사용할 수 없습니다.
p_disallow_simple_strings	부울	true	아니요	true로 설정되어 단순 문자열을 암호로 사용할 수 없습니다.
p_disallow_whitespace	boolean	false	아니요	true로 설정되어 암호에 공백 문자를 사용할 수 없습니다.
p_disallow_at_sign	boolean	false	아니요	true로 설정되어 암호에 @ 문자를 사용할 수 없습니다.

암호 확인 기능을 여러 개 만들 수 있습니다.

사용자 지정 기능의 이름에 대한 제한 사항이 있습니다. 사용자 지정 함수는 기존 시스템 객체와 이름이 같을 수 없고, 30자를 넘어서는 안 됩니다. 또한 PASSWORD, VERIFY, COMPLEXITY, ENFORCE, STRENGTH 중 한 가지 문자열을 포함해야 합니다.

다음 예시에서는 CUSTOM_PASSWORD_FUNCTION이라는 기능을 생성합니다. 이 함수에는 최소한 문자 12개, 대문자 2개, 자릿수 1개, 특수 문자 1개가 필요하며 암호에는 @ 문자를 사용할 수 없습니다.

```
begin
    rdsadmin.rdsadmin_password_verify.create_verify_function(
        p_verify_function_name => 'CUSTOM_PASSWORD_FUNCTION',
        p_min_length           => 12,
        p_min_uppercase        => 2,
        p_min_digits           => 1,
        p_min_special          => 1,
        p_disallow_at_sign     => true);
end;
/
```

확인 함수의 텍스트를 보려면 DBA_SOURCE를 쿼리하세요. 다음은 CUSTOM_PASSWORD_FUNCTION이라는 이름의 사용자 지정 암호 기능의 텍스트를 얻는 예제입니다.

```
COL TEXT FORMAT a150

  SELECT TEXT
    FROM DBA_SOURCE
   WHERE OWNER = 'SYS'
     AND NAME = 'CUSTOM_PASSWORD_FUNCTION'
ORDER BY LINE;
```

확인 함수를 사용자 프로파일과 연결하려면 `alter profile`을 사용하세요. 다음은 확인 함수를 DEFAULT 사용자 프로필과 연결하는 예제입니다.

```
ALTER PROFILE DEFAULT LIMIT PASSWORD_VERIFY_FUNCTION CUSTOM_PASSWORD_FUNCTION;
```

어떤 사용자 프로파일이 어떤 확인 기능과 연결되어 있는지 보려면 DBA_PROFILES를 쿼리하세요. 다음은 CUSTOM_PASSWORD_FUNCTION이라는 이름의 사용자 지정 확인 기능과 연결된 프로필을 얻는 예제입니다.

```
SELECT * FROM DBA_PROFILES WHERE RESOURCE_NAME = 'PASSWORD' AND LIMIT =
 'CUSTOM_PASSWORD_FUNCTION';


PROFILE                    RESOURCE_NAME                    RESOURCE  LIMIT
-------------------------  -------------------------------  --------
 ------------------------
```

```
DEFAULT                       PASSWORD_VERIFY_FUNCTION          PASSWORD
 CUSTOM_PASSWORD_FUNCTION
```

다음은 모든 프로필과 프로필에 연결된 암호 확인 기능을 얻는 예제입니다.

```
SELECT * FROM DBA_PROFILES WHERE RESOURCE_NAME = 'PASSWORD_VERIFY_FUNCTION';

PROFILE                       RESOURCE_NAME                     RESOURCE  LIMIT
------------------------  --------------------------------  --------
 ------------------------
DEFAULT                       PASSWORD_VERIFY_FUNCTION          PASSWORD
 CUSTOM_PASSWORD_FUNCTION
RDSADMIN                      PASSWORD_VERIFY_FUNCTION          PASSWORD  NULL
```

create_passthrough_verify_fcn 프로시저

이 `create_passthrough_verify_fcn` 프로시저는 RDS for Oracle의 모든 버전에 대해 지원됩니다.

Amazon RDS 프로시저 `rdsadmin.rdsadmin_password_verify.create_passthrough_verify_fcn`을 사용하여 사용자 지정 암호 확인 함수를 만들 수 있습니다. `create_passthrough_verify_fcn` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
`p_verify_function_name`	varchar2	—	예	사용자 정의 확인 함수의 이름입니다. SYS 스키마에서 자동으로 생성되는 래퍼 함수로서 확인 로직을 포함하지 않습니다. 이 함수를 사용자 프로필에 할당합니다.
`p_target_owner`	varchar2	—	예	사용자 지정 확인 함수의 스키마 소유자입니다.
`p_target_function_name`	varchar2	—	예	확인 로직을 포함하는 기존 사용자 지정 함수의 이

파라미터 이름	데이터 형식	기본값	필수	설명
				름입니다. 사용자 정의 함수는 부울 값을 반환합니다. 사용자의 함수는 암호가 유효할 경우 `true`를 반환하고 암호가 유효하지 않을 경우 `false`를 반환해야 합니다.

다음은 `PASSWORD_LOGIC_EXTRA_STRONG`이라는 함수 로직을 사용하는 암호 확인 함수를 생성하는 예제입니다.

```
begin
    rdsadmin.rdsadmin_password_verify.create_passthrough_verify_fcn(
        p_verify_function_name => 'CUSTOM_PASSWORD_FUNCTION',
        p_target_owner         => 'TEST_USER',
        p_target_function_name => 'PASSWORD_LOGIC_EXTRA_STRONG');
end;
/
```

확인 기능을 사용자 프로파일과 연결하려면 `alter profile`를 사용하세요. 다음은 확인 함수를 `DEFAULT` 사용자 프로파일과 연결하는 예제입니다.

```
ALTER PROFILE DEFAULT LIMIT PASSWORD_VERIFY_FUNCTION CUSTOM_PASSWORD_FUNCTION;
```

사용자 지정 DNS 서버 설정

Amazon RDS는 Oracle을 실행하는 DB 인스턴스에서 아웃바운드 네트워크 액세스를 지원합니다. 사전 요구 사항을 포함한 아웃바운드 네트워크 액세스에 대한 자세한 내용은 인증서 및 Oracle Wallet을 사용하여 UTL_HTTP 액세스 구성 섹션을 참조하세요.

Amazon RDS Oracle에서는 고객이 소유한 사용자 지정 DNS 서버에서의 DNS(Domain Name Service) 확인이 가능합니다. 사용자 지정 DNS 서버를 통해 Amazon RDS DB 인스턴스에서 전체 주소 도메인 이름만을 확인할 수 있습니다.

사용자 지정 DNS 이름 서버를 설정한 후 변경 사항이 DB 인스턴스에 전파되는 데 최대 30분이 걸립니다. 변경 사항이 DB 인스턴스에 전파된 후 DNS 조회를 필요로 하는 모든 아웃바운드 네트워크 트래픽은 포트 53을 통해 DNS 서버를 쿼리합니다.

Amazon RDS for Oracle DB 인스턴스의 사용자 지정 DNS 서버를 설정하려면 다음과 같이 하세요.

- Virtual Private Cloud(VPC)에 연결된 DHCP 옵션 세트에서 DNS 이름 서버의 IP 주소에 대해 `domain-name-servers` 옵션을 설정합니다. 자세한 내용은 DHCP 옵션 세트 단원을 참조하세요.

  > **Note**
  >
  > `domain-name-servers` 옵션은 최대 4개의 값을 받아들이지만 Amazon RDS DB 인스턴스는 첫 번째 값만을 사용합니다.

- DNS 서버가 DNS 이름, Amazon EC2 프라이빗 DNS 이름, 고객별 DNS 이름을 비롯한 모든 조회 쿼리를 확인할 수 있는지 확인합니다. 아웃바운드 네트워크 트래픽에 DNS 서버가 처리할 수 없는 DNS 조회가 포함된 경우, DNS 서버에 적절한 업스트림 DNS 공급자가 구성되어 있어야 합니다.
- 512바이트 이하의 UDP(User Datagram Protocol) 응답을 생성하도록 DNS 서버를 구성하세요.
- 1,024바이트 이하의 TCP(Transmission Control Protocol) 응답을 생성하도록 DNS 서버를 구성하세요.
- 포트 53을 통한 Amazon RDS DB 인스턴스로부터의 인바운드 트래픽을 허용하도록 DNS 서버를 구성하세요. DNS 서버가 Amazon VPC에 있는 경우, VPC에는 포트 53에서 UDP 및 TCP 트래픽을 허용하는 인바운드 규칙이 포함된 보안 그룹이 있어야 합니다. DNS 서버가 Amazon VPC에 없는 경우, 포트 53에서 UDP 및 TCP 인바운드 트래픽을 허용하는 적절한 방화벽 허용 목록이 있어야 합니다.

  자세한 내용은 VPC의 보안 그룹 및 규칙 추가 및 제거 단원을 참조하세요.
- 포트 53을 통한 아웃바운드 트래픽을 허용하도록 Amazon RDS DB 인스턴스의 VPC를 구성하세요. VPC에는 포트 53에서 UDP 및 TCP 트래픽을 허용하는 아웃바운드 규칙이 포함된 보안 그룹이 있어야 합니다.

  자세한 내용은 VPC의 보안 그룹 및 규칙 추가 및 제거 단원을 참조하세요.
- Amazon RDS DB 인스턴스와 DNS 서버 간 라우팅 경로가 DNS 트래픽을 허용하도록 올바로 구성되어야 합니다.
  - Amazon RDS DB 인스턴스와 DNS 서버가 같은 VPC에 있지 않은 경우, 그 사이에 피어링 연결을 구축해야 합니다. 자세한 내용은 VPC 피어링이란?을 참조하세요.

시스템 진단 이벤트 설정 및 설정 해제

세션 수준에서 진단 이벤트를 설정하고 설정 해제하려면 `ALTER SESSION SET EVENTS`라는 Oracle SQL 문을 사용하면 됩니다. 하지만 시스템 수준에서 이벤트를 설정하는 데에는 Oracle SQL을 사용할 수 없습니다. 대신 `rdsadmin.rdsadmin_util` 패키지의 시스템 이벤트 프로시저를 사용합니다. 시스템 이벤트 프로시저는 다음 엔진 버전에서 사용할 수 있습니다.

- 모든 Oracle Database 21c 버전
- 19.0.0.0.ru-2020-10.rur-2020-10.r1 이상의 Oracle Database 19c 버전

  자세한 내용은 Amazon RDS for Oracle 릴리스 정보의 버전 19.0.0.0.ru-2020-10.rur-2020-10.r1을 참조하세요.

> **Important**
>
> 내부적으로, `rdsadmin.rdsadmin_util` 패키지는 `ALTER SYSTEM SET EVENTS` 문을 사용하여 이벤트를 설정합니다. 이 `ALTER SYSTEM` 문은 Oracle 데이터베이스 설명서에 설명되어 있지 않습니다. 일부 시스템 진단 이벤트는 많은 양의 추적 정보를 생성하거나 경합을 일으키거나 데이터베이스 가용성에 영향을 줄 수 있습니다. Oracle Support의 지침에 따라 비 프로덕션 데이터베이스에서 특정 진단 이벤트를 테스트하고 프로덕션 데이터베이스에서만 이벤트를 설정하는 것이 좋습니다.

허용되는 시스템 진단 이벤트 나열

설정 가능한 시스템 이벤트를 나열하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.list_allowed_system_events`를 사용합니다. 이 프로시저는 파라미터를 받지 않습니다.

다음 예에서는 설정 가능한 모든 시스템 이벤트를 나열합니다.

```
SET SERVEROUTPUT ON
EXEC rdsadmin.rdsadmin_util.list_allowed_system_events;
```

다음 샘플 출력에는 이벤트 번호와 해당 설명이 나열됩니다. Amazon RDS 프로시저 `set_system_event`를 사용하여 이러한 이벤트를 설정하고 `unset_system_event`를 사용하여 설정 해제합니다.

```
604    - error occurred at recursive SQL level
```

```
942   - table or view does not exist
1401  - inserted value too large for column
1403  - no data found
1410  - invalid ROWID
1422  - exact fetch returns more than requested number of rows
1426  - numeric overflow
1427  - single-row subquery returns more than one row
1476  - divisor is equal to zero
1483  - invalid length for DATE or NUMBER bind variable
1489  - result of string concatenation is too long
1652  - unable to extend temp segment by  in tablespace
1858  - a non-numeric character was found where a numeric was expected
4031  - unable to allocate  bytes of shared memory ("","","","")
6502  - PL/SQL: numeric or value error
10027 - Specify Deadlock Trace Information to be Dumped
10046 - enable SQL statement timing
10053 - CBO Enable optimizer trace
10173 - Dynamic Sampling time-out error
10442 - enable trace of kst for ORA-01555 diagnostics
12008 - error in materialized view refresh path
12012 - error on auto execute of job
12504 - TNS:listener was not given the SERVICE_NAME in CONNECT_DATA
14400 - inserted partition key does not map to any partition
31693 - Table data object  failed to load/unload and is being skipped due to error:
```

> **Note**
>
> 허용되는 시스템 이벤트 목록은 시간이 지남에 따라 변경될 수 있습니다. 적격 이벤트의 최신 목록이 있는지 확인하려면 `rdsadmin.rdsadmin_util.list_allowed_system_events`를 사용합니다.

시스템 진단 이벤트 설정

시스템 이벤트를 설정하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.set_system_event`를 사용합니다. `rdsadmin.rdsadmin_util.list_allowed_system_events`의 출력에 나열된 이벤트만 설정할 수 있습니다. `set_system_event` 프로시저는 다음 파라미터를 수용합니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_event	숫자	—	예	시스템 이벤트 번호입니다. 이 값은 list_allowed_system_events 에 의해 보고된 이벤트 번호 중 하나여야 합니다.
p_level	숫자	—	예	이벤트 수준입니다. 다양한 수준 값에 대한 설명은 Oracle 데이터베이스 설명서를 참조하거나 Oracle Support에 문의하세요.

set_system_event 프로시저는 다음 원칙에 따라 필요한 ALTER SYSTEM SET EVENTS 문을 구성하고 실행합니다.

- 이벤트 유형(context 또는 errorstack)은 자동으로 결정됩니다.
- ALTER SYSTEM SET EVENTS '*event* LEVEL *event_level*' 형식의 문이 컨텍스트 이벤트를 설정합니다. 이 표기법은 ALTER SYSTEM SET EVENTS '*event* TRACE NAME CONTEXT FOREVER, LEVEL *event_level*'과 동일합니다.
- ALTER SYSTEM SET EVENTS '*event* ERRORSTACK (*event_level*)' 형식의 문이 오류 스택 이벤트를 설정합니다. 이 표기법은 ALTER SYSTEM SET EVENTS '*event* TRACE NAME ERRORSTACK LEVEL *event_level*'과 동일합니다.

다음 예에서는 이벤트 942를 수준 3에 설정하고 이벤트 10442를 수준 10에 설정합니다. 샘플 출력이 포함되어 있습니다.

```
SQL> SET SERVEROUTPUT ON
SQL> EXEC rdsadmin.rdsadmin_util.set_system_event(942,3);
Setting system event 942 with: alter system set events '942 errorstack (3)'

PL/SQL procedure successfully completed.

SQL> EXEC rdsadmin.rdsadmin_util.set_system_event(10442,10);
```

```
Setting system event 10442 with: alter system set events '10442 level 10'

PL/SQL procedure successfully completed.
```

설정된 시스템 진단 이벤트 나열

현재 설정된 시스템 이벤트를 나열하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.list_set_system_events`를 사용합니다. 이 프로시저는 `set_system_event`에 의해 시스템 수준에 설정된 이벤트만 보고합니다.

다음 예에서는 활성 시스템 이벤트를 나열합니다.

```
SET SERVEROUTPUT ON
EXEC rdsadmin.rdsadmin_util.list_set_system_events;
```

다음 샘플 출력에서는 이벤트 목록, 이벤트 유형, 이벤트가 현재 설정된 레벨 및 이벤트가 설정된 시간을 보여 줍니다.

```
942 errorstack (3) - set at 2020-11-03 11:42:27
10442 level 10 - set at 2020-11-03 11:42:41

PL/SQL procedure successfully completed.
```

시스템 진단 이벤트 설정 해제

시스템 이벤트를 설정 해제하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.unset_system_event`를 사용합니다. `rdsadmin.rdsadmin_util.list_allowed_system_events`의 출력에 나열된 이벤트만 설정 해제할 수 있습니다. `unset_system_event` 프로시저는 다음 파라미터를 받습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_event	숫자	—	예	시스템 이벤트 번호입니다. 이 값은 `list_allowed_system_events` 에 의해 보고된 이벤트 번호 중 하나여야 합니다.

다음 예에서는 이벤트 942 및 10442를 설정 해제합니다. 샘플 출력이 포함되어 있습니다.

```
SQL> SET SERVEROUTPUT ON
SQL> EXEC rdsadmin.rdsadmin_util.unset_system_event(942);
Unsetting system event 942 with: alter system set events '942 off'

PL/SQL procedure successfully completed.

SQL> EXEC rdsadmin.rdsadmin_util.unset_system_event(10442);
Unsetting system event 10442 with: alter system set events '10442 off'

PL/SQL procedure successfully completed.
```

Oracle DB 인스턴스에 대한 공통 데이터베이스 작업 수행

그 다음에는 Oracle을 실행하는 Amazon RDS DB 인스턴스에서 데이터베이스와 관련된 특정 공통 DBA 작업을 수행하는 방법을 알아봅니다. 관리형 서비스 환경을 제공하기 위해 Amazon RDS는 DB 인스턴스에 대해 shell 액세스를 제공하지 않습니다. 또한 Amazon RDS는 고급 권한이 필요한 특정 시스템 절차와 테이블에 대한 액세스를 제한합니다.

주제

- 데이터베이스의 전역 이름 변경
- 테이블스페이스 사용
- tempfiles 사용
- 데이터베이스 체크포인트
- 분산 복구 설정
- 데이터베이스 시간대 설정
- Oracle 외부 테이블 작업
- Automatic Workload Repository(AWR)를 사용하여 성능 보고서 생성
- VPC의 DB 인스턴스에 사용하기 위한 데이터베이스 링크 조정
- DB 인스턴스의 기본 에디션 설정
- SYS.AUD$ 테이블에 대한 감사 활성화
- SYS.AUD$ 테이블에 대한 감사 비활성화
- 중단된 온라인 인덱스 빌드 정리
- 손상된 블록 건너뛰기

- 테이블스페이스, 데이터 파일, 임시 파일 크기 조정
- 휴지통 비우기
- 전체 수정을 위한 기본 표시 값 설정

데이터베이스의 전역 이름 변경

데이터베이스의 전역 이름을 변경하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.rename_global_name`을 사용합니다. `rename_global_name` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
`p_new_global_name`	varchar2	—	예	데이터베이스의 새로운 전역 이름입니다.

이름 변경이 적용될 수 있도록 데이터베이스가 열려 있어야 합니다. 데이터베이스의 전역 이름 변경에 대한 자세한 내용은 Oracle 문서의 ALTER DATABASE를 참조하세요.

다음은 데이터베이스의 전역 이름을 `new_global_name`으로 변경하는 예제입니다.

```
EXEC rdsadmin.rdsadmin_util.rename_global_name(p_new_global_name => 'new_global_name');
```

테이블스페이스 사용

RDS for Oracle에서 테이블스페이스를 사용할 수 있습니다. 이 테이블스페이스는 논리적 스토리지 단위이며 데이터베이스의 데이터를 저장합니다.

주제

- 테이블스페이스 생성과 크기 조정
- 기본 테이블스페이스 설정
- 기본 임시 테이블스페이스 설정
- 인스턴스 스토어에 임시 테이블스페이스 생성

테이블스페이스 생성과 크기 조정

Amazon RDS는 데이터 파일, 로그 파일 및 제어 파일에 대해 Oracle Managed Files(OMF)만 지원합니다. 데이터 파일과 로그 파일을 생성할 때는 물리적인 파일 이름을 지정할 수 없습니다.

기본적으로 데이터 파일 크기를 지정하지 않으면 테이블스페이스는 기본값이 AUTOEXTEND ON으로 생성되며, 최대 크기가 없습니다. 다음 예제에서 테이블스페이스 *users1*은 자동 확장이 가능합니다.

```
CREATE TABLESPACE users1;
```

이러한 기본 설정 때문에, 테이블스페이스가 할당된 모든 스토리지를 차지할 때까지 확장되기도 합니다. 되도록 영구 및 임시 테이블스페이스에 적절한 최대 크기를 지정하고, 공간 사용량을 자세히 살펴보십시오.

다음은 시작 크기가 1GB인 *users2*라는 이름의 테이블스페이스를 생성하는 예제입니다. 데이터 파일 크기가 지정되었지만 AUTOEXTEND ON이 지정되지 않았으므로 테이블스페이스를 자동 확장할 수 없습니다.

```
CREATE TABLESPACE users2 DATAFILE SIZE 1G;
```

다음은 시작 크기가 1GB이고 최대 크기가 10GB인 *users3*라는 이름의 테이블스페이스를 생성하는 예제입니다.

```
CREATE TABLESPACE users3 DATAFILE SIZE 1G AUTOEXTEND ON MAXSIZE 10G;
```

다음은 *temp01*이라는 이름의 임시 테이블스페이스를 생성하는 예제입니다.

```
CREATE TEMPORARY TABLESPACE temp01;
```

ALTER TABLESPACE를 사용하여 빅파일 테이블스페이스 크기를 조정할 수 있습니다. 크기는 킬로바이트(K), 메가바이트(M), 기가바이트(G)나 테라바이트(T)로 설정할 수 있습니다. 다음 예제에서는 *users_bf*라는 빅파일 테이블스페이스의 크기를 200MB로 조정합니다.

```
ALTER TABLESPACE users_bf RESIZE 200M;
```

다음 예에서는 *users_sf*라는 스몰파일 테이블스페이스에 데이터 파일을 추가합니다.

```
ALTER TABLESPACE users_sf ADD DATAFILE SIZE 100000M AUTOEXTEND ON NEXT 250m
 MAXSIZE UNLIMITED;
```

기본 테이블스페이스 설정

기본 테이블스페이스를 설정하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.alter_default_tablespace`를 사용합니다. `alter_default_tablespace` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
`tablespace_name`	varchar	—	예	기본 테이블스페이스의 이름입니다.

다음은 기본 테이블스페이스를 *`users2`*로 설정하는 예제입니다.

```
EXEC rdsadmin.rdsadmin_util.alter_default_tablespace(tablespace_name => 'users2');
```

기본 임시 테이블스페이스 설정

기본 임시 테이블스페이스를 설정하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.alter_default_temp_tablespace`를 사용합니다. `alter_default_temp_tablespace` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
`tablespace_name`	varchar	—	예	기본 임시 테이블스페이스의 이름입니다.

다음 예에서는 기본 임시 테이블스페이스를 *`temp01`*으로 설정합니다.

```
EXEC rdsadmin.rdsadmin_util.alter_default_temp_tablespace(tablespace_name => 'temp01');
```

인스턴스 스토어에 임시 테이블스페이스 생성

인스턴스 스토어에 임시 테이블스페이스를 생성하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.create_inst_store_tmp_tblspace`를 사용합니다. `create_inst_store_tmp_tblspace` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_tablespace_name	varchar	—	예	기본 임시 테이블스페이스의 이름입니다.

다음 예에서는 인스턴스 스토어에 임시 테이블스페이스 *temp01*을 생성합니다.

```
EXEC rdsadmin.rdsadmin_util.create_inst_store_tmp_tblspace(p_tablespace_name =>
 'temp01');
```

> **Important**
>
> rdsadmin_util.create_inst_store_tmp_tblspace를 실행할 때 새로 생성된 임시 테이블스페이스는 자동으로 기본 임시 테이블스페이스로 설정되지 않습니다. 기본값으로 설정하려면 기본 임시 테이블스페이스 설정를 참조하세요.

자세한 내용은 RDS for Oracle 인스턴스 스토어에 임시 데이터 저장 단원을 참조하십시오.

tempfiles 사용

읽기 전용 복제본의 인스턴스 스토어에 임시 파일 추가

기본 DB 인스턴스에서 임시 테이블스페이스를 생성하면 읽기 전용 복제본으로 임시 파일이 생성되지 않습니다. 다음 이유 중 하나로 인해 읽기 전용 복제본에 빈 임시 테이블스페이스가 있다고 가정해 보겠습니다.

- 읽기 전용 복제본의 테이블스페이스에서 임시 파일을 삭제했습니다. 자세한 내용은 읽기 전용 복제본에서 임시 파일 삭제 단원을 참조하십시오.
- 기본 DB 인스턴스에 새 임시 테이블스페이스를 생성했습니다. 이 경우 RDS for Oracle은 읽기 전용 복제본에 메타데이터를 동기화합니다.

빈 임시 테이블스페이스에 임시 파일을 추가하고 인스턴스 스토어에 임시 파일을 저장할 수 있습니다. 인스턴스 스토어에 임시 파일을 생성하려면 Amazon RDS 프로시저 rdsadmin.rdsadmin_util.add_inst_store_tempfile을 사용합니다. 이 프로시저는 읽기 전용 복제본에서만 사용할 수 있습니다. 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_tablespace_name	varchar	—	예	읽기 전용 복제본에 있는 임시 테이블스페이스의 이름입니다.

다음 예에서는 빈 임시 테이블스페이스 *temp01*이 읽기 전용 복제본에 있습니다. 다음 명령을 실행하여 이 테이블스페이스에 대한 임시 파일을 생성하고 인스턴스 스토어에 저장합니다.

```
EXEC rdsadmin.rdsadmin_util.add_inst_store_tempfile(p_tablespace_name => 'temp01');
```

자세한 내용은 RDS for Oracle 인스턴스 스토어에 임시 데이터 저장 단원을 참조하십시오.

읽기 전용 복제본에서 임시 파일 삭제

읽기 전용 복제본에서 기존 임시 테이블스페이스를 삭제할 수 없습니다. 읽기 전용 복제본의 임시 파일 스토리지를 Amazon EBS에서 인스턴스 스토어로 또는 인스턴스 스토어에서 Amazon EBS로 변경할 수 있습니다. 이를 위해 다음을 수행합니다.

1. 현재 읽기 전용 복제본의 임시 테이블스페이스에 있는 임시 파일을 삭제합니다.
2. 다른 스토리지에 새 임시 파일을 생성합니다.

임시 파일을 삭제하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.drop_replica_tempfiles`을 사용합니다. 이 절차는 읽기 전용 복제본에서만 사용할 수 있습니다. `drop_replica_tempfiles` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_tablespace_name	varchar	—	예	읽기 전용 복제본에 있는 임시 테이블스페이스의 이름입니다.

*temp01*이라는 임시 테이블스페이스가 읽기 전용 복제본의 인스턴스 스토어에 있다고 가정해 보겠습니다. 다음 명령을 실행하여 이 테이블스페이스의 모든 임시 파일을 삭제합니다.

```
EXEC rdsadmin.rdsadmin_util.drop_replica_tempfiles(p_tablespace_name => 'temp01');
```

자세한 내용은 RDS for Oracle 인스턴스 스토어에 임시 데이터 저장 단원을 참조하십시오.

데이터베이스 체크포인트

데이터베이스에 체크포인트를 만들려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.checkpoint`를 사용합니다. `checkpoint` 프로시저에는 파라미터가 없습니다.

다음 예에서는 데이터베이스에 체크포인트를 만듭니다.

```
EXEC rdsadmin.rdsadmin_util.checkpoint;
```

분산 복구 설정

분산 복구를 설정하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.enable_distr_recovery` 및 `disable_distr_recovery`를 사용합니다. 프로시저에는 파라미터가 없습니다.

다음 예에서는 분산 복구를 활성화합니다.

```
EXEC rdsadmin.rdsadmin_util.enable_distr_recovery;
```

다음 예에서는 분산 복구를 비활성화합니다.

```
EXEC rdsadmin.rdsadmin_util.disable_distr_recovery;
```

데이터베이스 시간대 설정

다음과 같은 방법으로 Amazon RDS Oracle 데이터베이스의 시간대를 설정할 수 있습니다.

- `Timezone` 옵션

  `Timezone` 옵션은 호스트 수준에서 시간대를 변경하여 `SYSDATE`를 포함한 모든 데이터 열과 값에 영향을 끼칩니다. 자세한 내용은 Oracle 시간대 섹션을 참조하세요.
- Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.alter_db_time_zone`

  `alter_db_time_zone` 절차는 특정 데이터 유형의 시간대만 변경하며 `SYSDATE`는 변경하지 않습니다. 시간대 설정에 대한 자세한 제한 사항은 Oracle 문서를 참조하세요.

> **Note**
>
> Oracle Scheduler의 기본 시간대를 설정할 수도 있습니다. 자세한 내용은 Oracle Scheduler 작업의 시간대 설정 섹션을 참조하세요.

`alter_db_time_zone` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_new_tz	varchar2	—	예	지정된 리전이나 협정 세계시(UTC)의 절대 오프셋으로 지정된 새로운 표준 시간대입니다. 사용할 수 있는 오프셋은 -12:00~+14:00입니다.

다음 예제에서는 시간대를 UTC+3시간으로 변경합니다.

```
EXEC rdsadmin.rdsadmin_util.alter_db_time_zone(p_new_tz => '+3:00');
```

다음 예제에서는 시간대를 아프리카/알제 시간대로 변경합니다.

```
EXEC rdsadmin.rdsadmin_util.alter_db_time_zone(p_new_tz => 'Africa/Algiers');
```

`alter_db_time_zone` 프로시저를 사용하여 시간대를 변경한 후에는 DB 인스턴스를 재부팅해야만 변경 사항이 적용됩니다. 자세한 내용은 DB 인스턴스 재부팅 섹션을 참조하세요. 시간대 업그레이드에 대한 자세한 내용은 시간대 고려 사항 단원을 참조하세요.

Oracle 외부 테이블 작업

Oracle 외부 테이블이란 데이터가 데이터베이스에 저장되어 있지 않은 테이블을 말합니다. 오히려 데이터베이스가 액세스할 수 있는 외부 파일에 데이터가 저장되어 있습니다. 외부 테이블을 사용하면 데이터베이스에 테이블을 로드하지 않고도 데이터에 액세스할 수 있습니다. 외부 테이블에 대한 자세한 내용은 Oracle 설명서에서 Managing External Tables를 참조하세요.

Amazon RDS에서는 외부 테이블 파일을 디렉터리 객체에 저장할 수 있습니다. 디렉터리 객체는 생성할 수도 있지만, DATA_PUMP_DIR 디렉터리처럼 Oracle 데이터베이스에 사전 정의되어 있는 객체를 사용할 수도 있습니다. 디렉터리 객체 생성에 대한 자세한 내용은 메인 데이터 스토리지 공간에서 디렉터리 생성 및 삭제 단원을 참조하세요. ALL_DIRECTORIES 뷰에 대한 쿼리를 실행하여 Amazon RDS Oracle DB 인스턴스의 디렉터리 객체 목록을 조회할 수 있습니다.

> **Note**
>
> 디렉터리 객체는 인스턴스에서 사용하는 메인 데이터 스토리지 공간(Amazon EBS 볼륨)을 가리킵니다. 데이터 파일, 재실행 로그, 감사, 트레이스 및 기타 파일 등에 사용되는 공간은 할당된 공간에서 차감됩니다.

외부 데이터 파일은 DBMS_FILE_TRANSFER 패키지 또는 UTL_FILE 패키지를 사용하여 Oracle 데이터베이스에서 다른 Oracle 데이터베이스로 이동시킬 수 있습니다. 그러면 원본 데이터베이스의 디렉터리에서 대상 데이터베이스의 특정 디렉터리로 외부 데이터 파일이 이동합니다. DBMS_FILE_TRANSFER 사용에 대한 자세한 내용은 Oracle Data Pump를 사용한 가져오기 단원을 참조하세요.

외부 데이터 파일을 이동시켰으면 이제 이 파일을 이용해 외부 테이블을 생성할 수 있습니다. 다음 예에서는 USER_DIR1 디렉터리의 `emp_xt_file1.txt` 파일을 사용하는 외부 테이블을 생성합니다.

```
CREATE TABLE emp_xt (
  emp_id      NUMBER,
  first_name  VARCHAR2(50),
  last_name   VARCHAR2(50),
  user_name   VARCHAR2(20)
)
ORGANIZATION EXTERNAL (
  TYPE ORACLE_LOADER
  DEFAULT DIRECTORY USER_DIR1
  ACCESS PARAMETERS (
    RECORDS DELIMITED BY NEWLINE
    FIELDS TERMINATED BY ','
    MISSING FIELD VALUES ARE NULL
    (emp_id,first_name,last_name,user_name)
  )
  LOCATION ('emp_xt_file1.txt')
)
PARALLEL
```

```
REJECT LIMIT UNLIMITED;
```

예를 들어 Amazon RDS Oracle DB 인스턴스에 저장된 데이터를 외부 데이터 파일로 이동시킨다고 가정하겠습니다. 이때는 외부 테이블을 생성한 후 데이터베이스 테이블에서 데이터를 선택하여 외부 데이터 파일을 채울 수 있습니다. 다음은 데이터베이스의 `orders_xt` 테이블에 대해 쿼리를 실행하여 `orders` 외부 테이블을 생성하는 SQL 문입니다.

```
CREATE TABLE orders_xt
  ORGANIZATION EXTERNAL
   (
     TYPE ORACLE_DATAPUMP
     DEFAULT DIRECTORY DATA_PUMP_DIR
     LOCATION ('orders_xt.dmp')
   )
   AS SELECT * FROM orders;
```

위 예제에서는 DATA_PUMP_DIR 디렉터리의 `orders_xt.dmp` 파일이 데이터로 채워집니다.

Automatic Workload Repository(AWR)를 사용하여 성능 보고서 생성

Oracle은 성능 데이터 수집과 보고서 생성을 위해 Automatic Workload Repository(AWR)를 사용할 것을 권장합니다. AWR에는 Oracle Database Enterprise Edition과 Diagnostics and Tuning 팩용 라이선스가 필요합니다. AWR을 활성화하려면 `CONTROL_MANAGEMENT_PACK_ACCESS` 초기화 파라미터를 `DIAGNOSTIC` 또는 `DIAGNOSTIC+TUNING`으로 설정합니다.

RDS에서 AWR 보고서 작업

`awrrpt.sql`과 같은 스크립트를 실행하면 AWR 보고서를 생성할 수 있습니다. 이러한 스크립트는 데이터베이스 호스트 서버에 설치됩니다. Amazon RDS에서는 호스트에 직접 액세스할 수 없습니다. 그러나 설치된 다른 Oracle Database에서 SQL 스크립트의 복사본을 가져올 수 있습니다.

`SYS.DBMS_WORKLOAD_REPOSITORY` PL/SQL 패키지에서 프로시저를 실행하여 AWR을 사용할 수도 있습니다. 이 패키지를 사용하여 기준 요소 및 스냅샷을 관리하고 ASH 및 AWR 보고서를 표시할 수 있습니다. 예를 들어 텍스트 형식으로 AWR 보고서를 생성하려면 `DBMS_WORKLOAD_REPOSITORY.AWR_REPORT_TEXT` 프로시저를 실행합니다. 그러나 AWS Management Console에서는 이러한 AWR 보고서에 연결할 수 없습니다.

AWR로 작업할 때 `rdsadmin.rdsadmin_diagnostic_util` 프로시저를 사용하는 것이 좋습니다. 이러한 프로시저를 사용하여 다음을 생성할 수 있습니다.

- AWR 보고서
- 활성 세션 기록(ASH) 보고서
- 자동 데이터베이스 진단 모니터(ADDM) 보고서
- AWR 데이터의 Oracle Data Pump Export 덤프 파일

`rdsadmin_diagnostic_util` 프로시저는 보고서를 DB 인스턴스 파일 시스템에 저장합니다. 콘솔에서 이러한 보고서에 액세스할 수 있습니다. 또한 `rdsadmin.rds_file_util` 프로시저를 사용하여 보고서에 액세스할 수 있고, S3 통합 옵션을 사용하여 Amazon S3에 복사된 보고서에 액세스할 수 있습니다. 자세한 내용은 DB 인스턴스 디렉터리의 파일 목록 읽기 및 Amazon S3 통합 단원을 참조하세요.

다음 Amazon RDS for Oracle DB 엔진 버전에서 `rdsadmin_diagnostic_util` 프로시저를 사용할 수 있습니다.

- 모든 Oracle Database 21c 버전
- 19.0.0.0.ru-2020-04.rur-2020-04.r1 이상의 Oracle Database 19c 버전

복제 시나리오에서 진단 보고서를 사용하는 방법을 설명하는 블로그를 보려면 Amazon RDS for Oracle 읽기 전용 복제본에 대한 AWR 보고서 생성을 참조하세요.

진단 유틸리티 패키지의 공통 파라미터

`rdsadmin_diagnostic_util` 패키지로 AWR 및 ADDM을 관리할 때는 일반적으로 다음 파라미터를 사용합니다.

파라미터	데이터 형식	기본값	필수	설명
`begin_snap_id`	NUMBEF	—	예	시작 스냅샷의 ID입니다.
`end_snap_id`	NUMBEF	—	예	종료 스냅샷의 ID입니다.
`dump_directory`	VARCHA	BDUMF	아니요	보고서를 작성하거나 파일을 내보낼 디렉터리입니다. 기본이 아닌 디렉터리를 지정하는 경우 `rdsadmin_`

파라미터	데이터 형식	기본값	필수	설명
				`diagnostic_util` 프로시저를 실행하는 사용자에게 디렉터리에 대한 쓰기 권한이 있어야 합니다.
`p_tag`	VARCHA	—	아니요	`incremental` 또는 `daily`와 같이 백업의 목적이나 용도를 나타내기 위해 백업을 구별하는 데 사용할 수 있는 문자열입니다. 최대 30자까지 지정할 수 있습니다. 유효한 문자는 `a-z`, `A-Z`, `0-9`, 밑줄(_), 대시(-), 마침표(.)입니다. 태그는 대/소문자를 구분하지 않습니다. RMAN은 태그를 입력할 때 사용된 대/소문자에 관계없이 항상 대문자로 태그를 저장합니다. 태그는 고유할 필요가 없으므로 여러 백업이 동일한 태그를 가질 수 있습니다. 태그를 지정하지 않으면 RMAN은 TAG*YYYYMMDDTHHMMSS* 형식을 사용하여 기본 태그를 자동으로 할당합니다. 여기서 *YYYY*는 연도, *MM*은 월, *DD*는 일, *HH*는 시간(24시간 형식), *MM*은 분, *SS*는 초입니다. 날짜 및 시간은 RMAN이 백업을 시작한 때를 나타냅니다. 예를 들어 기본 태그 `TAG20190927T214517` 이 있는 백업은 2019-09-27 21:45:17에 시작된 백업을 나타냅니다. `p_tag` 파라미터는 다음 RDS for Oracle DB 엔진 버전에서 지원됩니다. • Oracle Database 21c(21.0.0) • Oracle Database 19c(19.0.0), 19.0.0.0.ru-2021-10.rur-2021-10.r1 이상 사용
`report_type`	VARCHA	`HTML`	아니요	보고서의 형식입니다. 유효 값은 `TEXT` 및 `HTML`입니다.

파라미터	데이터 형식	기본값	필수	설명
dbid	NUMBEF	—	아니요	Oracle에 대한 DBA_HIST_DATABASE_INSTANCE 보기에 표시되는 유효한 데이터베이스 식별자(DBID)입니다. 이 파라미터를 지정하지 않으면 RDS가 V$DATABASE.DBID 보기에 표시된 현재 DBID를 사용합니다.

rdsadmin_diagnostic_util 패키지로 ASH를 관리할 때는 일반적으로 다음 파라미터를 사용합니다.

파라미터	데이터 형식	기본값	필수	설명
begin_time	DATE	—	예	ASH 분석의 시작 시간입니다.
end_time	DATE	—	예	ASH 분석의 종료 시간입니다.
slot_width	NUMBER	0	아니요	ASH 보고서의 “Top Activity” 섹션에 사용되는 슬롯의 지속 시간(초)입니다. 이 파라미터를 지정하지 않으면 begin_time 과 end_time 사이의 시간 간격이 10개 이하의 슬롯을 사용합니다.
sid	NUMBER	Null	아니요	세션 ID입니다.
sql_id	VARCHAR2	Null	아니요	SQL ID입니다.
wait_class	VARCHAR2	Null	아니요	대기 클래스 이름입니다.
service_hash	NUMBER	Null	아니요	서비스 이름 해시입니다.
module_name	VARCHAR2	Null	아니요	모듈 이름입니다.

파라미터	데이터 형식	기본값	필수	설명
action_name	VARCHAR2	Null	아니요	작업 이름입니다.
client_id	VARCHAR2	Null	아니요	데이터베이스 세션의 애플리케이션별 ID입니다.
plsql_entry	VARCHAR2	Null	아니요	PL/SQL 진입점입니다.

AWR 보고서 생성

AWR 보고서를 생성하려면 rdsadmin.rdsadmin_diagnostic_util.awr_report 프로시저를 사용합니다.

다음 예에서는 스냅샷 범위 101–106에 대한 AWR 보고서를 생성합니다. 출력 텍스트 파일의 이름은 awrrpt_101_106.txt입니다. AWS Management Console에서 이 보고서에 액세스할 수 있습니다.

```
EXEC rdsadmin.rdsadmin_diagnostic_util.awr_report(101,106,'TEXT');
```

다음 예에서는 스냅샷 범위 63–65에 대한 HTML 보고서를 생성합니다. 출력 HTML 파일의 이름은 awrrpt_63_65.html입니다. 이 프로시저는 기본이 아닌 데이터베이스 디렉터리 AWR_RPT_DUMP에 보고서를 작성합니다.

```
EXEC rdsadmin.rdsadmin_diagnostic_util.awr_report(63,65,'HTML','AWR_RPT_DUMP');
```

AWR 데이터를 덤프 파일로 추출

AWR 데이터를 덤프 파일로 추출하려면 rdsadmin.rdsadmin_diagnostic_util.awr_extract 프로시저를 사용합니다.

다음 예에서는 스냅샷 범위 101–106을 추출합니다. 출력 덤프 파일의 이름은 awrextract_101_106.dmp입니다. 콘솔을 통해 이 파일에 액세스할 수 있습니다.

```
EXEC rdsadmin.rdsadmin_diagnostic_util.awr_extract(101,106);
```

다음 예에서는 스냅샷 범위 63–65를 추출합니다. 출력 덤프 파일의 이름은 awrextract_63_65.dmp입니다. 이 파일은 기본이 아닌 데이터베이스 디렉터리 AWR_RPT_DUMP에 저장됩니다.

```
EXEC rdsadmin.rdsadmin_diagnostic_util.awr_extract(63,65,'AWR_RPT_DUMP');
```

ADDM 보고서 생성

ADDM 보고서를 생성하려면 rdsadmin.rdsadmin_diagnostic_util.addm_report 프로시저를 사용합니다.

다음 예에서는 스냅샷 범위 101–106에 대한 ADDM 보고서를 생성합니다. 출력 텍스트 파일의 이름은 addmrpt_101_106.txt입니다. 콘솔을 통해 보고서에 액세스할 수 있습니다.

```
EXEC rdsadmin.rdsadmin_diagnostic_util.addm_report(101,106);
```

다음 예에서는 스냅샷 범위 63–65에 대한 ADDM 보고서를 생성합니다. 출력 텍스트 파일의 이름은 addmrpt_63_65.txt입니다. 이 파일은 기본이 아닌 데이터베이스 디렉터리 ADDM_RPT_DUMP에 저장됩니다.

```
EXEC rdsadmin.rdsadmin_diagnostic_util.addm_report(63,65,'ADDM_RPT_DUMP');
```

ASH 보고서 생성

ASH 보고서를 생성하려면 rdsadmin.rdsadmin_diagnostic_util.ash_report 프로시저를 사용합니다.

다음 예에서는 14분 전부터 현재 시간까지의 데이터가 포함된 ASH 보고서를 생성합니다. 출력 파일의 이름은 ashrpt*begin_time*_*end_time*.txt 형식을 사용하며, 여기서 *begin_time* 및 *end_time*은 YYYYMMDDHH24MISS 형식을 사용합니다. 콘솔을 통해 파일에 액세스할 수 있습니다.

```
BEGIN
    rdsadmin.rdsadmin_diagnostic_util.ash_report(
        begin_time      =>      SYSDATE-14/1440,
        end_time        =>      SYSDATE,
        report_type     =>      'TEXT');
END;
/
```

다음 예에서는 2019년 11월 18일 오후 6시 7분부터 2019년 11월 18일 오후 6시 15분까지의 데이터가 포함된 ASH 보고서를 생성합니다. 출력 HTML 보고서의 이름은 `ashrpt_20190918180700_20190918181500.html`입니다. 보고서는 기본이 아닌 데이터베이스 디렉터리 `AWR_RPT_DUMP`에 저장됩니다.

```
BEGIN
    rdsadmin.rdsadmin_diagnostic_util.ash_report(
        begin_time     =>    TO_DATE('2019-09-18 18:07:00', 'YYYY-MM-DD HH24:MI:SS'),
        end_time       =>    TO_DATE('2019-09-18 18:15:00', 'YYYY-MM-DD HH24:MI:SS'),
        report_type    =>    'html',
        dump_directory =>    'AWR_RPT_DUMP');
END;
/
```

콘솔 또는 CLI에서 AWR 보고서 액세스

AWS Management Console 또는 AWS CLI를 사용하여 AWR 보고서에 액세스하거나 덤프 파일을 내보낼 수 있습니다. 자세한 내용은 데이터베이스 로그 파일 다운로드 단원을 참조하십시오.

VPC의 DB 인스턴스에 사용하기 위한 데이터베이스 링크 조정

동일한 Virtual Private Cloud(VPC) 또는 피어링된 VPC 내에서 Amazon RDS DB 인스턴스 간에 Oracle 데이터베이스 링크를 사용하려면 두 DB 인스턴스에 서로에게 이르는 유효한 경로가 있어야 합니다. VPC 라우팅 테이블과 네트워크 ACL(액세스 제어 목록)을 사용하여 DB 인스턴스 간 유효 경로를 확인합니다.

각 DB 인스턴스의 보안 그룹은 다른 DB 인스턴스로(부터)의 수신 및 발신을 허용해야 합니다. 인바운드 및 아웃바운드 규칙은 동일한 VPC 또는 피어링된 VPC에서 보안 그룹을 참조할 수 있습니다. 자세한 내용은 피어링된 VPC 보안 그룹을 참조하도록 보안 그룹 업데이트 단원을 참조하세요.

VPC에서 DHCP 옵션 세트를 이용해 사용자 지정 DNS 서버를 구성했다면, 사용자 지정 DNS 서버가 데이터베이스 링크 타깃의 이름을 확인할 수 있어야 합니다. 자세한 내용은 사용자 지정 DNS 서버 설정 섹션을 참조하세요.

Oracle Data Pump로 데이터베이스 링크를 사용하는 방법에 대한 자세한 내용은 Oracle Data Pump를 사용한 가져오기 단원을 참조하세요.

DB 인스턴스의 기본 에디션 설정

데이터베이스 객체는 에디션이라고 하는 프라이빗 환경에서 재정의할 수 있습니다. 이러한 에디션 기반 재정의를 통해 서비스 중단을 최소화하면서 애플리케이션의 데이터베이스 객체를 업그레이드할 수 있습니다.

Amazon RDS Oracle DB 인스턴스의 기본 에디션은 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.alter_default_edition`을 사용하여 설정할 수 있습니다.

다음은 Amazon RDS Oracle DB 인스턴스의 기본 에디션을 `RELEASE_V1`으로 설정하는 예제입니다.

```
EXEC rdsadmin.rdsadmin_util.alter_default_edition('RELEASE_V1');
```

다음은 Amazon RDS Oracle DB 인스턴스의 기본 에디션을 다시 Oracle 기본값으로 설정하는 예제입니다.

```
EXEC rdsadmin.rdsadmin_util.alter_default_edition('ORA$BASE');
```

Oracle 에디션 기반 재정의에 대한 자세한 내용은 Oracle 설명서에서 About Editions and Edition-Based Redefinition을 참조하세요.

SYS.AUD$ 테이블에 대한 감사 활성화

데이터베이스 감사 추적 테이블 `SYS.AUD$`에서 감사를 활성화하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_master_util.audit_all_sys_aud_table`을 사용합니다. 지원되는 유일한 감사 속성은 `ALL`입니다. 개별 문 또는 연산을 감사하거나 감사하지 않을 수 있습니다.

감사 활성화는 다음 버전을 실행하는 Oracle DB 인스턴스에 대해 지원됩니다.

- Oracle Database 21c(21.0.0)
- Oracle Database 19c(19.0.0)

`audit_all_sys_aud_table` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
`p_by_access`	부울	true	아니요	`true`로 설정하여 `BY ACCESS`를 감사합니다.

파라미터 이름	데이터 형식	기본값	필수	설명
				false로 설정하여 BY SESSION를 감사합니다.

다음 쿼리는 데이터베이스의 SYS.AUD$에 대한 현재 감사 구성을 반환합니다.

```
SELECT * FROM DBA_OBJ_AUDIT_OPTS WHERE OWNER='SYS' AND OBJECT_NAME='AUD$';
```

다음 명령은 ALL SYS.AUD$의 BY ACCESS에 대한 감사를 활성화합니다.

```
EXEC rdsadmin.rdsadmin_master_util.audit_all_sys_aud_table;

EXEC rdsadmin.rdsadmin_master_util.audit_all_sys_aud_table(p_by_access => true);
```

다음 명령은 ALL SYS.AUD$의 BY SESSION에 대한 감사를 활성화합니다.

```
EXEC rdsadmin.rdsadmin_master_util.audit_all_sys_aud_table(p_by_access => false);
```

자세한 내용은 Oracle 설명서의 AUDIT(기존 감사)를 참조하세요.

SYS.AUD$ 테이블에 대한 감사 비활성화

데이터베이스 감사 추적 테이블 SYS.AUD$에서 감사를 비활성화하려면 Amazon RDS 프로시저 rdsadmin.rdsadmin_master_util.noaudit_all_sys_aud_table을 사용합니다. 이 프로시저에는 파라미터가 없습니다.

다음 쿼리는 데이터베이스의 SYS.AUD$에 대한 현재 감사 구성을 반환합니다.

```
SELECT * FROM DBA_OBJ_AUDIT_OPTS WHERE OWNER='SYS' AND OBJECT_NAME='AUD$';
```

다음 명령은 ALL에 대한 SYS.AUD$의 감사를 비활성화합니다.

```
EXEC rdsadmin.rdsadmin_master_util.noaudit_all_sys_aud_table;
```

자세한 내용은 Oracle 설명서의 NOAUDIT(기존 감사)를 참조하세요.

중단된 온라인 인덱스 빌드 정리

실패한 온라인 인덱스 빌드를 정리하려면 Amazon RDS 절차 `rdsadmin.rdsadmin_dbms_repair.online_index_clean`을 사용하세요.

`online_index_clean` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
`object_id`	binary_integer	`ALL_INDEX_ID`	아니요	인덱스의 객체 ID입니다. 일반적으로 ORA-08104 오류 텍스트의 객체 ID를 사용할 수 있습니다.
`wait_for_lock`	binary_integer	`rdsadmin.rdsadmin_dbms_repair.lock_wait`	아니요	기본값인 `rdsadmin.rdsadmin_dbms_repair.lock_wait` 를 지정하여 기본 객체에 대한 잠금을 가져오세요. 잠금이 실패하면 내부 한도에 이를 때까지 재시도하세요. `rdsadmin.rdsadmin_dbms_repair.lock_nowait` 를 지정하여 기본 객체에 대한 잠금을 가져오되 잠금이 실패하면 재시도하지 마십시오.

다음 예에서는 실패한 온라인 인젝스 빌드를 정리합니다.

```
declare
  is_clean boolean;
begin
```

```
  is_clean := rdsadmin.rdsadmin_dbms_repair.online_index_clean(
    object_id     => 1234567890,
    wait_for_lock => rdsadmin.rdsadmin_dbms_repair.lock_nowait
  );
end;
/
```

자세한 내용은 Oracle 문서의 ONLINE_INDEX_CLEAN Function을 참조하세요.

손상된 블록 건너뛰기

인덱스 및 테이블 스캔 중에 손상된 블록을 건너뛰려면 `rdsadmin.rdsadmin_dbms_repair` 패키지를 사용하세요.

다음 절차에서는 `sys.dbms_repair.admin_table` 절차의 기능을 래핑하고 파라미터를 받아들이지 않습니다.

- `rdsadmin.rdsadmin_dbms_repair.create_repair_table`
- `rdsadmin.rdsadmin_dbms_repair.create_orphan_keys_table`
- `rdsadmin.rdsadmin_dbms_repair.drop_repair_table`
- `rdsadmin.rdsadmin_dbms_repair.drop_orphan_keys_table`
- `rdsadmin.rdsadmin_dbms_repair.purge_repair_table`
- `rdsadmin.rdsadmin_dbms_repair.purge_orphan_keys_table`

다음 절차에서는 Oracle 데이터베이스에 대해 `DBMS_REPAIR` 패키지에서 이에 상응하는 것으로 동일한 파라미터를 받아들입니다.

- `rdsadmin.rdsadmin_dbms_repair.check_object`
- `rdsadmin.rdsadmin_dbms_repair.dump_orphan_keys`
- `rdsadmin.rdsadmin_dbms_repair.fix_corrupt_blocks`
- `rdsadmin.rdsadmin_dbms_repair.rebuild_freelists`
- `rdsadmin.rdsadmin_dbms_repair.segment_fix_status`
- `rdsadmin.rdsadmin_dbms_repair.skip_corrupt_blocks`

데이터베이스 손상 처리에 대한 자세한 내용은 Oracle 설명서의 DBMS_REPAIR를 참조하세요.

Example 손상된 블록에 응답

이 예제에서는 손상된 블록에 응답하기 위한 기본 워크플로우를 보여줍니다. 단계는 블록 손상의 위치와 특성에 따라 달라집니다.

> **Important**
>
> 손상된 블록을 복구하기 전에 DBMS_REPAIRE 설명서를 주의 깊게 검토하세요.

인덱스 및 테이블 스캔 중에 손상된 블록을 건너뛰려면

1. 다음 프로시저를 실행하여 복구 테이블을 생성합니다(아직 없는 경우).

```
EXEC rdsadmin.rdsadmin_dbms_repair.create_repair_table;
EXEC rdsadmin.rdsadmin_dbms_repair.create_orphan_keys_table;
```

2. 다음 절차를 실행하여 기존 레코드를 확인하고 필요에 따라 제거하세요.

```
SELECT COUNT(*) FROM SYS.REPAIR_TABLE;
SELECT COUNT(*) FROM SYS.ORPHAN_KEY_TABLE;
SELECT COUNT(*) FROM SYS.DBA_REPAIR_TABLE;
SELECT COUNT(*) FROM SYS.DBA_ORPHAN_KEY_TABLE;

EXEC rdsadmin.rdsadmin_dbms_repair.purge_repair_table;
EXEC rdsadmin.rdsadmin_dbms_repair.purge_orphan_keys_table;
```

3. 다음 절차를 실행하여 손상된 블록을 확인하세요.

```
SET SERVEROUTPUT ON
DECLARE v_num_corrupt INT;
BEGIN
  v_num_corrupt := 0;
  rdsadmin.rdsadmin_dbms_repair.check_object (
    schema_name => '&corruptionOwner',
    object_name => '&corruptionTable',
    corrupt_count =>  v_num_corrupt
  );
  dbms_output.put_line('number corrupt: '||to_char(v_num_corrupt));
END;
/
```

```
COL CORRUPT_DESCRIPTION FORMAT a30
COL REPAIR_DESCRIPTION FORMAT a30

SELECT OBJECT_NAME, BLOCK_ID, CORRUPT_TYPE, MARKED_CORRUPT,
       CORRUPT_DESCRIPTION, REPAIR_DESCRIPTION
FROM   SYS.REPAIR_TABLE;

SELECT SKIP_CORRUPT
FROM   DBA_TABLES
WHERE  OWNER = '&corruptionOwner'
AND    TABLE_NAME = '&corruptionTable';
```

4. skip_corrupt_blocks 프로시저를 사용하여 영향을 받은 테이블에 대해 손상 건너뛰기를 활성화 또는 비활성화하세요. 상황에 따라 새 테이블에 데이터를 추출한 다음 손상된 블록이 포함된 테이블을 삭제해야 할 수도 있습니다.

   다음 절차를 실행하여 영향을 받은 테이블에 대해 손상 건너뛰기를 활성화하세요.

   ```
   begin
     rdsadmin.rdsadmin_dbms_repair.skip_corrupt_blocks (
       schema_name => '&corruptionOwner',
       object_name => '&corruptionTable',
       object_type => rdsadmin.rdsadmin_dbms_repair.table_object,
       flags => rdsadmin.rdsadmin_dbms_repair.skip_flag);
   end;
   /
   select skip_corrupt from dba_tables where owner = '&corruptionOwner' and table_name
    = '&corruptionTable';
   ```

   다음 절차를 실행하여 손상 건너뛰기를 비활성화하세요.

   ```
   begin
     rdsadmin.rdsadmin_dbms_repair.skip_corrupt_blocks (
       schema_name => '&corruptionOwner',
       object_name => '&corruptionTable',
       object_type => rdsadmin.rdsadmin_dbms_repair.table_object,
       flags => rdsadmin.rdsadmin_dbms_repair.noskip_flag);
   end;
   /

   select skip_corrupt from dba_tables where owner = '&corruptionOwner' and table_name
    = '&corruptionTable';
   ```

5. 모든 복구 작업을 완료했으면 다음 프로시저를 실행하여 복구 테이블을 삭제합니다.

```
EXEC rdsadmin.rdsadmin_dbms_repair.drop_repair_table;
EXEC rdsadmin.rdsadmin_dbms_repair.drop_orphan_keys_table;
```

테이블스페이스, 데이터 파일, 임시 파일 크기 조정

기본적으로 Oracle 테이블스페이스는 자동 확장이 켜진 상태로 생성되며 최대 크기는 지정되지 않습니다. 이러한 기본 설정 때문에 테이블스페이스가 너무 크게 확장될 때가 있습니다. 되도록 영구 및 임시 테이블스페이스에 적절한 최대 크기를 지정하고, 공간 사용량을 자세히 살펴보십시오.

영구 테이블스페이스 크기 조정

RDS for Oracle DB 인스턴스에서 영구 테이블스페이스의 크기를 조정하려면 다음 Amazon RDS 프로시저 중 하나를 사용하세요.

- `rdsadmin.rdsadmin_util.resize_datafile`
- `rdsadmin.rdsadmin_util.autoextend_datafile`

`resize_datafile` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
`p_data_file_id`	숫자	—	예	크기를 조정할 데이터 파일의 식별자.
`p_size`	varchar2	—	예	데이터 파일의 크기. 크기는 바이트(기본값), 킬로바이트(K), 메가바이트(M) 또는 기가바이트(G)로 지정합니다.

`autoextend_datafile` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_data_file_id	숫자	—	예	크기를 조정할 데이터 파일의 식별자.
p_autoextend_state	varchar2	—	예	자동 확장 기능의 상태. 데이터 파일을 자동으로 확장하려면 ON으로 지정하고, 자동 확장을 끄려면 OFF로 지정합니다.
p_next	varchar2	—	아니요	다음 데이터 파일 증분의 크기. 크기는 바이트(기본값), 킬로바이트(K), 메가바이트(M) 또는 기가바이트(G)로 지정합니다.
p_maxsize	varchar2	—	아니요	자동 확장에 허용되는 최대 디스크 공간. 크기는 바이트(기본값), 킬로바이트(K), 메가바이트(M) 또는 기가바이트(G)로 지정합니다. UNLIMITED 를 지정하여 파일 크기 제한을 없앨 수 있습니다.

다음 예제에서는 데이터 파일 크기를 4~500MB로 조정합니다.

```
EXEC rdsadmin.rdsadmin_util.resize_datafile(4,'500M');
```

다음 예제에서는 데이터 파일 4의 자동 확장을 끕니다. 또한 데이터 파일 5는 자동 확장을 켜고, 최대 크기 없이 128MB씩 증분하는 걸로 설정합니다.

```
EXEC rdsadmin.rdsadmin_util.autoextend_datafile(4,'OFF');
EXEC rdsadmin.rdsadmin_util.autoextend_datafile(5,'ON','128M','UNLIMITED');
```

임시 테이블스페이스의 크기 조정

RDS for Oracle DB 인스턴스에서 임시 테이블스페이스의 크기를 조정하려면 다음 Amazon RDS 프로시저 중 하나를 사용하세요.

- `rdsadmin.rdsadmin_util.resize_temp_tablespace`
- `rdsadmin.rdsadmin_util.resize_tempfile`
- `rdsadmin.rdsadmin_util.autoextend_tempfile`

`resize_temp_tablespace` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
`p_temp_tablespace_name`	varchar2	—	예	크기를 조정할 임시 테이블스페이스의 이름입니다.
`p_size`	varchar2	—	예	테이블스페이스의 크기. 크기는 바이트(기본값), 킬로바이트(K), 메가바이트(M) 또는 기가바이트(G)로 지정합니다.

`resize_tempfile` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
`p_temp_file_id`	숫자	—	예	크기를 조정할 임시 테이블스페이스의 파일 식별자.
`p_size`	varchar2	—	예	임시 파일의 크기. 크기는 바이트(기본값), 킬로바이트(K), 메가바이트(M) 또

파라미터 이름	데이터 형식	기본값	필수	설명
				는 기가바이트(G)로 지정합니다.

autoextend_tempfile 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_temp_file_id	숫자	—	예	크기를 조정할 임시 테이블스페이스의 파일 식별자.
p_autoextend_state	varchar2	—	예	자동 확장 기능의 상태. 임시 파일을 자동으로 확장하려면 ON으로 지정하고, 자동 확장을 끄려면 OFF로 지정합니다.
p_next	varchar2	—	아니요	다음 임시 파일 증분의 크기. 크기는 바이트(기본값), 킬로바이트(K), 메가바이트(M) 또는 기가바이트(G)로 지정합니다.
p_maxsize	varchar2	—	아니요	자동 확장에 허용되는 최대 디스크 공간. 크기는 바이트(기본값), 킬로바이트(K), 메가바이트(M) 또는 기가바이트(G)로 지정합니다. UNLIMITED 를 지정하여 파일 크기 제한을 없앨 수 있습니다.

다음 예제에서는 TEMP라는 이름의 임시 테이블스페이스의 크기를 4GB로 조정합니다.

```
EXEC rdsadmin.rdsadmin_util.resize_temp_tablespace('TEMP','4G');
```

```
EXEC rdsadmin.rdsadmin_util.resize_temp_tablespace('TEMP','4096000000');
```

다음 예제에서는 파일 식별자가 1인 임시 파일을 기준으로 하는 임시 테이블스페이스의 크기를 2MB로 조정합니다.

```
EXEC rdsadmin.rdsadmin_util.resize_tempfile(1,'2M');
```

다음 예제에서는 임시 파일 1의 자동 확장을 끕니다. 또한 임시 파일의 최대 자동 확장 크기를 2GB에서 10GB로 설정하며, 100MB씩 증분합니다.

```
EXEC rdsadmin.rdsadmin_util.autoextend_tempfile(1,'OFF');
EXEC rdsadmin.rdsadmin_util.autoextend_tempfile(2,'ON','100M','10G');
```

Oracle DB 인스턴스의 읽기 전용 복제본에 대한 자세한 내용은 Amazon RDS의 Oracle의 읽기 전용 복제본 작업 섹션을 참조하세요.

휴지통 비우기

테이블을 삭제해도 Oracle 데이터베이스가 해당 스토리지 공간을 즉시 확보하지 않습니다. 데이터베이스는 삭제된 테이블의 이름을 바꾸고 테이블 및 연결된 객체를 휴지통에 넣습니다. 휴지통을 비우면 이러한 항목이 제거되고 스토리지 공간이 확보됩니다.

전체 휴지통을 비우려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.purge_dba_recyclebin`을 사용합니다. 그러나 이 프로시저는 SYS 및 RDSADMIN 객체의 휴지통은 비울 수 없습니다. 이러한 객체를 삭제해야 하는 경우 AWS Support에 문의하세요.

다음 예에서는 전체 휴지통을 비웁니다.

```
EXEC rdsadmin.rdsadmin_util.purge_dba_recyclebin;
```

전체 수정을 위한 기본 표시 값 설정

Amazon RDS Oracle 인스턴스에서 전체 수정을 위한 기본 표시 값을 변경하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.dbms_redact_upd_full_rdct_val`을 사용합니다. 참고로 수

정 정책은 Oracle 데이터베이스 설명서에 설명된 것처럼 DBMS_REDACT PL/SQL 패키지를 사용하여 생성합니다. dbms_redact_upd_full_rdct_val 프로시저는 기존 정책의 영향을 받는 여러 데이터 유형에 대해 표시할 문자를 지정합니다.

dbms_redact_upd_full_rdct_val 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_number_val	number	Null	아니요	NUMBER 데이터 유형의 열에 대한 기본값을 수정합니다.
p_binfloat_val	binary_float	Null	아니요	BINARY_FLOAT 데이터 유형의 열에 대한 기본값을 수정합니다.
p_bindouble_val	binary_double	Null	아니요	BINARY_DOUBLE 데이터 유형의 열에 대한 기본값을 수정합니다.
p_char_val	char	Null	아니요	CHAR 데이터 유형의 열에 대한 기본값을 수정합니다.
p_varchar_val	varchar2	Null	아니요	VARCHAR2 데이터 유형의 열에 대한 기본값을 수정합니다.
p_nchar_val	nchar	Null	아니요	NCHAR 데이터 유형의 열에 대한 기본값을 수정합니다.
p_nvarchar_val	nvarchar2	Null	아니요	NVARCHAR2 데이터 유형의 열에 대한 기본값을 수정합니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_date_val	date	Null	아니요	DATE 데이터 유형의 열에 대한 기본값을 수정합니다.
p_ts_val	타임스탬프	Null	아니요	TIMESTAMP 데이터 유형의 열에 대한 기본값을 수정합니다.
p_tswtz_val	시간대가 있는 타임스탬프	Null	아니요	TIMESTAMP WITH TIME ZONE 데이터 유형의 열에 대한 기본값을 수정합니다.
p_blob_val	blob	Null	아니요	BLOB 데이터 유형의 열에 대한 기본값을 수정합니다.
p_clob_val	clob	Null	아니요	CLOB 데이터 유형의 열에 대한 기본값을 수정합니다.
p_nclob_val	nclob	Null	아니요	NCLOB 데이터 유형의 열에 대한 기본값을 수정합니다.

다음 예에서는 CHAR 데이터 유형에 대한 기본 수정 값을 *로 변경합니다.

```
EXEC rdsadmin.rdsadmin_util.dbms_redact_upd_full_rdct_val(p_char_val => '*');
```

다음 예에서는 NUMBER, DATE, CHAR 데이터 유형에 대한 기본 수정 값을 변경합니다.

```
BEGIN
rdsadmin.rdsadmin_util.dbms_redact_upd_full_rdct_val(
    p_number_val=>1,
    p_date_val=>to_date('1900-01-01','YYYY-MM-DD'),
```

```
    p_varchar_val=>'X');
END;
/
```

dbms_redact_upd_full_rdct_val 프로시저를 사용하여 전체 수정의 기본값을 변경한 후 DB 인스턴스를 재부팅하여 변경 사항을 적용합니다. 자세한 내용은 DB 인스턴스 재부팅 단원을 참조하십시오.

Oracle DB 인스턴스에 대한 공통 로그 관련 작업 수행

그 다음에는 Oracle을 실행하는 Amazon RDS DB 인스턴스에서 로깅과 관련된 특정 공통 DBA 작업을 수행하는 방법을 알아봅니다. 관리형 서비스 환경을 제공하기 위해 Amazon RDS는 DB 인스턴스에 대해 shell 액세스를 제공하지 않으며, 고급 권한이 필요한 특정 시스템 절차와 테이블에 대한 액세스를 제한합니다.

자세한 내용은 Amazon RDS for Oracle 데이터베이스 로그 파일 섹션을 참조하세요.

주제

- 강제 로깅 설정
- 보충 로깅 설정
- 온라인 로그 파일 전환
- 온라인 다시 실행 로그 추가
- 온라인 다시 실행 로그 드롭
- 온라인 다시 실행 로그 크기 조절
- 보관된 다시 실행 로그 보존
- 온라인 및 아카이빙된 다시 실행 로그 액세스
- Amazon S3에서 아카이빙된 다시 실행 로그 다운로드

강제 로깅 설정

강제 로깅 모드에서는 Oracle이 임시 테이블스페이스와 임시 세그먼트의 변경 사항을 제외하고 데이터베이스의 모든 변경 사항을 기록합니다(NOLOGGING 절은 무시됩니다). 자세한 내용은 Oracle 문서의 Specifying FORCE LOGGING Mode 단원을 참조하세요.

강제 로깅을 설정하려면 Amazon RDS 프로시저 rdsadmin.rdsadmin_util.force_logging을 사용합니다. force_logging 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	예	설명
p_enable	부울	true	아니요	true로 설정하면 데이터베이스를 강제 로깅 모드로 설정하고, false로 설정하면 데이터베이스를 강제 로깅 모드에서 해제합니다.

다음은 데이터베이스를 강제 로깅 모드로 설정하는 예제입니다.

```
EXEC rdsadmin.rdsadmin_util.force_logging(p_enable => true);
```

보충 로깅 설정

보충 로깅을 활성화하면 LogMiner가 연결된 행과 클러스터링된 테이블을 지원하는 데 필요한 정보를 갖게 됩니다. 자세한 내용은 Oracle 문서의 Supplemental Logging을 참조하세요.

Oracle 데이터베이스는 기본적으로 보충 로깅이 활성화되어 있지 않습니다. 보충 로깅을 활성화 또는 비활성화하려면 Amazon RDS 프로시저 rdsadmin.rdsadmin_util.alter_supplemental_logging을 사용합니다. Amazon RDS가 Oracle DB 인스턴스를 위해 보관된 다시 실행 로그 보존을 관리하는 방법에 대한 자세한 내용은 보관된 다시 실행 로그 보존 단원을 참조하세요.

alter_supplemental_logging 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_action	varchar2	—	예	'ADD'는 보충 로깅을 추가하며, 'DROP'은 보충 로깅을 드롭합니다.
p_type	varchar2	null	아니요	보충 로깅 유형입니다. 유효한 값은 'ALL', 'FOREIGN

파라미터 이름	데이터 형식	기본값	필수	설명
				KEY', 'PRIMARY KEY', 'UNIQUE' 또는 PROCEDURAL 입니다.

다음 예에서는 보충 로깅을 활성화합니다.

```
begin
    rdsadmin.rdsadmin_util.alter_supplemental_logging(
        p_action => 'ADD');
end;
/
```

다음 예에서는 고정 길이 최대 크기 열 전체에 대해 보충 로깅을 활성화합니다.

```
begin
    rdsadmin.rdsadmin_util.alter_supplemental_logging(
        p_action => 'ADD',
        p_type   => 'ALL');
end;
/
```

다음 예에서는 기본 키 열에 대한 보충 로깅을 활성화합니다.

```
begin
    rdsadmin.rdsadmin_util.alter_supplemental_logging(
        p_action => 'ADD',
        p_type   => 'PRIMARY KEY');
end;
/
```

온라인 로그 파일 전환

로그 파일을 전환하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.switch_logfile`을 사용합니다. `switch_logfile` 프로시저에는 파라미터가 없습니다.

다음은 로그 파일을 바꾸는 예제입니다.

```
EXEC rdsadmin.rdsadmin_util.switch_logfile;
```

온라인 다시 실행 로그 추가

Oracle을 실행하는 Amazon RDS DB 인스턴스는 각각 128MB인 온라인 다시 실행 로그 4개로 시작합니다. 다른 다시 실행 로그를 추가하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.add_logfile`을 사용합니다.

`add_logfile` 프로시저에는 다음과 같은 파라미터가 있습니다.

> **Note**
>
> 파라미터는 함께 사용할 수 없습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
`bytes`	positive	null	아니요	로그 파일 크기(바이트)입니다. 로그 크기가 2147483648바이트(2GiB) 미만인 경우에만 이 파라미터를 사용합니다. 그렇지 않으면 RDS에서 오류가 발생합니다. 이 바이트 값을 초과하는 로그 크기의 경우 `p_size` 파라미터를 대신 사용합니다.
`p_size`	varchar2	—	예	로그 파일의 크기는 킬로바이트(K), 메가바이트(M) 또는 기가바이트(G)입니다.

다음 명령을 실행하면 100MB 로그 파일이 추가됩니다.

```
EXEC rdsadmin.rdsadmin_util.add_logfile(p_size => '100M');
```

온라인 다시 실행 로그 드롭

다시 실행 로그를 삭제하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.drop_logfile`을 사용합니다. `drop_logfile` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
`grp`	positive	—	예	로그의 그룹 번호입니다.

다음 예에서는 그룹 번호가 3인 로그를 드롭합니다.

```
EXEC rdsadmin.rdsadmin_util.drop_logfile(grp => 3);
```

상태가 미사용이거나 비활성인 로그만 드롭할 수 있습니다. 다음 예에서는 로그의 상태를 가져옵니다.

```
SELECT GROUP#, STATUS FROM V$LOG;

GROUP#     STATUS
---------- ----------------
1          CURRENT
2          INACTIVE
3          INACTIVE
4          UNUSED
```

온라인 다시 실행 로그 크기 조절

Oracle을 실행하는 Amazon RDS DB 인스턴스는 각각 128MB인 온라인 다시 실행 로그 4개로 시작합니다. 다음은 Amazon RDS 프로시저를 사용하여 로그를 각각 128MB에서 512MB로 조정하는 예제입니다.

```
/* Query V$LOG to see the logs.          */
/* You start with 4 logs of 128 MB each. */

SELECT GROUP#, BYTES, STATUS FROM V$LOG;
```

```
GROUP#     BYTES      STATUS
---------- ---------- ----------------
1          134217728  INACTIVE
2          134217728  CURRENT
3          134217728  INACTIVE
4          134217728  INACTIVE


/* Add four new logs that are each 512 MB */

EXEC rdsadmin.rdsadmin_util.add_logfile(bytes => 536870912);
EXEC rdsadmin.rdsadmin_util.add_logfile(bytes => 536870912);
EXEC rdsadmin.rdsadmin_util.add_logfile(bytes => 536870912);
EXEC rdsadmin.rdsadmin_util.add_logfile(bytes => 536870912);


/* Query V$LOG to see the logs. */
/* Now there are 8 logs.        */

SELECT GROUP#, BYTES, STATUS FROM V$LOG;

GROUP#     BYTES      STATUS
---------- ---------- ----------------
1          134217728  INACTIVE
2          134217728  CURRENT
3          134217728  INACTIVE
4          134217728  INACTIVE
5          536870912  UNUSED
6          536870912  UNUSED
7          536870912  UNUSED
8          536870912  UNUSED


/* Drop each inactive log using the group number. */

EXEC rdsadmin.rdsadmin_util.drop_logfile(grp => 1);
EXEC rdsadmin.rdsadmin_util.drop_logfile(grp => 3);
EXEC rdsadmin.rdsadmin_util.drop_logfile(grp => 4);


/* Query V$LOG to see the logs. */
/* Now there are 5 logs.        */

select GROUP#, BYTES, STATUS from V$LOG;
```

```
GROUP#     BYTES      STATUS
---------- ---------- ----------------
2          134217728  CURRENT
5          536870912  UNUSED
6          536870912  UNUSED
7          536870912  UNUSED
8          536870912  UNUSED


/* Switch logs so that group 2 is no longer current. */

EXEC rdsadmin.rdsadmin_util.switch_logfile;


/* Query V$LOG to see the logs.        */
/* Now one of the new logs is current. */

SQL>SELECT GROUP#, BYTES, STATUS FROM V$LOG;

GROUP#     BYTES      STATUS
---------- ---------- ----------------
2          134217728  ACTIVE
5          536870912  CURRENT
6          536870912  UNUSED
7          536870912  UNUSED
8          536870912  UNUSED


/* If the status of log 2 is still "ACTIVE", issue a checkpoint to clear it to
 "INACTIVE".  */

EXEC rdsadmin.rdsadmin_util.checkpoint;


/* Query V$LOG to see the logs.            */
/* Now the final original log is inactive. */

select GROUP#, BYTES, STATUS from V$LOG;

GROUP#     BYTES      STATUS
---------- ---------- ----------------
2          134217728  INACTIVE
5          536870912  CURRENT
```

```
6          536870912  UNUSED
7          536870912  UNUSED
8          536870912  UNUSED


# Drop the final inactive log.

EXEC rdsadmin.rdsadmin_util.drop_logfile(grp => 2);


/* Query V$LOG to see the logs.    */
/* Now there are four 512 MB logs. */

SELECT GROUP#, BYTES, STATUS FROM V$LOG;

GROUP#     BYTES      STATUS
---------- ---------- ----------------
5          536870912  CURRENT
6          536870912  UNUSED
7          536870912  UNUSED
8          536870912  UNUSED
```

보관된 다시 실행 로그 보존

보관된 다시 실행 로그는 Oracle LogMiner(DBMS_LOGMNR) 같은 제품에서 사용할 수 있도록 DB 인스턴스에 로컬 보존할 수 있습니다. 다시 실행 로그를 보존하면 LogMiner를 사용하여 로그를 분석할 수 있습니다. 자세한 내용은 Oracle 문서의 Using LogMiner to Analyze Redo Log Files 단원을 참조하세요.

보관된 다시 실행 로그를 보존하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.set_configuration`을 사용합니다. Oracle Data Guard의 기본 인스턴스에서 이 프로시저를 사용하는 경우 RDS는 기본 인스턴스와 열린 읽기 전용 복제본에서 아카이브 로그 유지 설정을 변경하지만 탑재된 복제본에서는 변경하지 않습니다. RDS는 탑재된 복제본에 대한 최신 아카이브 다시 실행 로그를 짧은 기간 동안 유지합니다. RDS는 탑재된 복제본에 다운로드된 이전 로그를 자동으로 삭제합니다.

`set_configuration` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
name	varchar	—	예	업데이트할 구성의 이름입니다. 아카이브된 다시 실행 로그 유지 시간을 변경하려면 이름을 archivelog retention hours로 설정합니다.
value	varchar	—	예	구성 값입니다. 로그를 유지할 시간 값을 설정합니다.

다음 예에서는 24시간 동안 다시 실행 로그를 보존합니다.

```
begin
    rdsadmin.rdsadmin_util.set_configuration(
        name  => 'archivelog retention hours',
        value => '24');
end;
/
commit;
```

> **Note**
>
> 변경 사항을 적용하려면 커밋해야 합니다.

DB 인스턴스 다시 실행 로그의 보관 기간을 보려면 Amazon RDS 프로시저 rdsadmin.rdsadmin_util.show_configuration을 사용합니다.

다음 예에서는 로그 보관 시간을 보여줍니다.

```
set serveroutput on
EXEC rdsadmin.rdsadmin_util.show_configuration;
```

`archivelog retention hours`의 현재 설정이 출력됩니다. 다음 출력은 보관된 다시 실행 로그가 48시간 동안 보존됨을 보여줍니다.

```
NAME:archivelog retention hours
VALUE:48
DESCRIPTION:ArchiveLog expiration specifies the duration in hours before archive/redo
 log files are automatically deleted.
```

보관된 다시 실행 로그는 DB 인스턴스에 보존되므로 보존된 로그를 수용하기에 충분한 스토리지가 DB 인스턴스에 할당되어 있는지 확인해야 합니다. 지난 X 시간 동안 DB 인스턴스가 사용한 공간을 측정하려면 X 자리에 사용 시간을 대입하여 다음 쿼리를 실행합니다.

```
SELECT SUM(BLOCKS * BLOCK_SIZE) bytes
  FROM V$ARCHIVED_LOG
 WHERE FIRST_TIME >= SYSDATE-(X/24) AND DEST_ID=1;
```

RDS for Oracle은 DB 인스턴스의 백업 보존 기간이 0보다 큰 경우에만 아카이빙된 다시 실행 로그를 생성합니다. 기본적으로 백업 보존 기간은 0보다 큽니다.

아카이빙된 로그 보존 기간이 만료되면 RDS for Oracle은 DB 인스턴스에서 아카이빙된 다시 실행 로그를 제거합니다. DB 인스턴스를 특정 시점으로 복원하려면 Amazon RDS는 백업 보존 기간에 따라 아카이빙된 다시 실행 로그를 DB 인스턴스 외부에 보관합니다. 백업 보존 기간을 수정하려면 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

> **Note**
>
> 경우에 따라 Linux에서 JDBC를 사용하여 아카이브된 다시 실행 로그를 다운로드하고 지연 시간이 길어지고 연결이 재설정될 수 있습니다. 그러한 경우 Java 클라이언트의 기본 난수 생성기 설정이 문제의 원인일 수 있습니다. 비차단형 난수 생성기를 사용하도록 JDBC 드라이버를 설정하는 것이 좋습니다.

온라인 및 아카이빙된 다시 실행 로그 액세스

GoldenGate, Attunity, Informatica 등과 같은 외부 도구로 마이닝하려면 온라인 및 아카이브된 다시 실행 로그 파일에 액세스해야 합니다. 이러한 파일에 액세스하려면 다음을 수행합니다.

1. 물리적 파일 경로에 읽기 전용으로 액세스할 수 있는 디렉터리 객체를 만듭니다.

`rdsadmin.rdsadmin_master_util.create_archivelog_dir` 및 `rdsadmin.rdsadmin_master_util.create_onlinelog_dir` 사용

2. PL/SQL을 사용하여 파일을 읽습니다.

   PL/SQL을 사용하여 파일을 읽을 수 있습니다. 디렉터리 객체의 파일을 읽는 방법에 대한 자세한 내용은 DB 인스턴스 디렉터리의 파일 목록 표시 및 DB 인스턴스 디렉터리의 파일 목록 읽기 단원을 참조하세요.

다음 릴리스에서 트랜잭션 로그 액세스를 지원합니다.

- Oracle Database 21c
- Oracle Database 19c

다음은 온라인 및 아카이브된 다시 실행 로그 파일에 읽기 전용으로 액세스할 수 있는 디렉터리를 생성하는 코드입니다.

> **Important**
>
> 이 코드는 `DROP ANY DIRECTORY` 권한도 취소합니다.

```
EXEC rdsadmin.rdsadmin_master_util.create_archivelog_dir;
EXEC rdsadmin.rdsadmin_master_util.create_onlinelog_dir;
```

다음은 온라인 및 보관된 재실행 로그 파일용 디렉터리를 삭제하는 코드입니다.

```
EXEC rdsadmin.rdsadmin_master_util.drop_archivelog_dir;
EXEC rdsadmin.rdsadmin_master_util.drop_onlinelog_dir;
```

다음은 `DROP ANY DIRECTORY` 권한을 부여하고 취소하는 코드입니다.

```
EXEC rdsadmin.rdsadmin_master_util.revoke_drop_any_directory;
EXEC rdsadmin.rdsadmin_master_util.grant_drop_any_directory;
```

Amazon S3에서 아카이빙된 다시 실행 로그 다운로드

`rdsadmin.rdsadmin_archive_log_download` 패키지를 사용하여 DB 인스턴스에서 아카이빙된 다시 실행 로그를 다운로드할 수 있습니다. 아카이빙된 다시 실행 로그가 더 이상 DB 인스턴스에 없는 경우 Amazon S3에서 다시 다운로드할 수 있습니다. 그런 다음 로그를 마이닝하거나 로그를 사용하여 데이터베이스를 복구하거나 복제할 수 있습니다.

> **Note**
>
> 읽기 전용 복제본 인스턴스에서는 아카이브된 다시 실행 로그를 다운로드할 수 없습니다.

아카이빙된 다시 실행 로그 다운로드: 기본 단계

아카이빙된 다시 실행 로그의 가용성은 다음 보존 정책에 따라 다릅니다.

- 백업 보존 정책 - 이 정책 내의 로그는 Amazon S3에서 사용할 수 있습니다. 이 정책 외부의 로그는 제거됩니다.
- 아카이빙된 로그 보존 정책 - 이 정책 내의 로그를 DB 인스턴스에서 사용할 수 있습니다. 이 정책 외부의 로그는 제거됩니다.

로그가 인스턴스에 없지만 백업 보존 기간으로 보호되는 경우 `rdsadmin.rdsadmin_archive_log_download`을 사용하여 다시 다운로드합니다. RDS for Oracle은 DB 인스턴스의 `/rdsdbdata/log/arch` 디렉터리에 로그를 저장합니다.

Amazon S3에서 아카이빙된 다시 실행 로그를 다운로드하려면

1. 다운로드한 아카이브된 재실행 로그가 필요한 기간 동안 보존되도록 보존 기간을 구성합니다. `COMMIT`에 대한 변경을 확인하세요.

   RDS는 로그 다운로드 시점부터 보관된 로그 보존 정책에 따라 다운로드한 로그를 보존합니다. 보존 정책을 설정하는 방법에 대한 자세한 내용은 보관된 다시 실행 로그 보존 섹션을 참조하세요.
2. 아키이빙된 로그 보존 정책 변경이 적용되려면 최대 5분을 기다립니다.
3. `rdsadmin.rdsadmin_archive_log_download`을 사용하여 Amazon S3 에서 아카이빙된 다시 실행 로그를 다운로드합니다.

   자세한 내용은 아카이빙된 다시 실행 로그 다운로드 및 아카이빙된 다시 실행 로그 시리즈 다운로드 단원을 참조하세요.

> **Note**
>
> RDS는 다운로드하기 전에 사용 가능한 스토리지를 자동으로 확인합니다. 요청된 로그가 많은 공간을 사용하는 경우 경고가 표시됩니다.

4. 로그가 Amazon S3 에서 성공적으로 다운로드되었는지 확인합니다.

   다운로드 태스크의 상태를 bdump 파일에서 볼 수 있습니다. bdump 파일에는 `/rdsdbdata/log/trace/dbtask-task-id.log` 경로 이름이 있습니다. 이전 다운로드 단계에서 VARCHAR2 데이터 유형의 태스크 ID를 반환하는 SELECT 문을 실행합니다. 자세한 내용은 파일 전송 상태 모니터링에서 유사한 예를 참조하세요.

아카이빙된 다시 실행 로그 다운로드

하나의 아카이빙된 다시 실행 로그를 `/rdsdbdata/log/arch` 디렉터리에 다운로드하려면 `rdsadmin.rdsadmin_archive_log_download.download_log_with_seqnum`을 사용하세요. 이 프로시저에는 다음 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
seqnum	숫자	—	예	아카이빙된 다시 실행 로그의 시퀀스 번호입니다.

다음 예에서는 시퀀스 번호가 20인 로그를 다운로드합니다.

```
SELECT rdsadmin.rdsadmin_archive_log_download.download_log_with_seqnum(seqnum => 20)
       AS TASK_ID
FROM   DUAL;
```

아카이빙된 다시 실행 로그 시리즈 다운로드

아카이빙된 다시 실행 로그 시리즈를 `/rdsdbdata/log/arch` 디렉터리에 다운로드하려면 `download_logs_in_seqnum_range`을 사용하세요. 다운로드는 요청당 300개의 로그로 제한됩니다. `download_logs_in_seqnum_range` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
start_seq	숫자	—	예	시리즈의 시작 시퀀스 번호입니다.
end_seq	숫자	—	예	시리즈의 종료 시퀀스 번호입니다.

다음 예제에서는 시퀀스 50에서 100의 로그를 다운로드합니다.

```
SELECT rdsadmin.rdsadmin_archive_log_download.download_logs_in_seqnum_range(start_seq
 => 50, end_seq => 100)
       AS TASK_ID
FROM   DUAL;
```

Oracle DB 인스턴스에 대한 공통 RMAN 작업 수행

아래 단원에서는 Oracle을 실행하는 Amazon RDS DB 인스턴스에서 Oracle RMAN(Recovery Manager) DBA 작업을 수행하는 방식을 확인하실 수 있습니다. 관리형 서비스 환경을 제공하기 위해 Amazon RDS는 DB 인스턴스에 대해 shell 액세스를 제공하지 않습니다. 또한 고급 권한이 필요한 특정 시스템 절차와 테이블에 대한 액세스를 제한합니다.

Amazon RDS 패키지인 `rdsadmin.rdsadmin_rman_util`을 사용하여 Amazon RDS for Oracle 데이터베이스의 RMAN 백업을 디스크에 수행할 수 있습니다. `rdsadmin.rdsadmin_rman_util` 패키지에서는 전체 및 증분 데이터베이스 파일 백업, 테이블스페이스 백업 및 아카이브된 재실행 로그 백업을 지원합니다.

RMAN 백업이 종료된 후에는 Amazon RDS for Oracle DB 인스턴스 호스트에 대해 백업 파일을 복사할 수 있습니다. 이 작업은 RDS가 아닌 호스트 또는 백업 장기 보관에 대해 복원 목적으로 수행할 수 있습니다. 예를 들어 백업 파일을 Amazon S3 버킷에 복사할 수 있습니다. 자세한 내용은 Amazon S3 통합 사용 단원을 참조하세요.

RMAN 백업을 위한 백업 파일은 수동으로 제거하기 전에는 Amazon RDS DB 인스턴스에 남아 있습니다. `UTL_FILE.FREMOVE` Oracle 프로시저를 사용하여 디렉터리에서 파일을 제거할 수 있습니다. 자세한 내용은 Oracle Database 설명서의 FREMOVE 프로시저를 참조하세요.

RMAN을 사용하여 RDS for Oracle DB 인스턴스를 복원할 수는 없습니다. 하지만 RMAN을 사용하여 온프레미스 또는 Amazon EC2 인스턴스에 백업을 복원할 수 있습니다. 자세한 내용은 Amazon RDS for Oracle 인스턴스를 자체 관리형 인스턴스로 복원 블로그 문서를 참조하세요.

> **Note**
>
> 또 하나의 Amazon RDS for Oracle DB 인스턴스에 대한 백업 및 복원을 위해서는 Amazon RDS 백업 및 복원 기능을 계속 사용할 수 있습니다. 자세한 내용은 데이터 백업, 복원 및 내보내기 단원을 참조하십시오.

주제

- RMAN 백업을 위한 사전 요구 사항
- RMAN 프로시저용 공통 파라미터
- RDS for Oracle DB에서 데이터베이스 파일 검증
- 블록 변경 추적 활성화 및 비활성화
- 보관된 재실행 로그 대조 확인
- 보관된 재실행 로그 파일 백업
- 전체 데이터베이스 백업 수행
- 테넌트 데이터베이스의 전체 백업 수행
- 증분 데이터베이스 백업 수행
- 테넌트 데이터베이스의 증분 백업 수행
- 테이블스페이스 백업
- Backing up a control file
- 블록 미디어 복구 수행

RMAN 백업을 위한 사전 요구 사항

`rdsadmin.rdsadmin_rman_util` 패키지를 사용하여 데이터베이스를 백업하기 전에 다음 사전 요구 사항을 충족하는지 확인하세요.

- RDS for Oracle 데이터베이스 `ARCHIVELOG` 모드여야 합니다. 이 모드를 활성화하려면 백업 보존 기간을 `0`이 아닌 값으로 설정하세요.
- 아카이브된 재실행 로그를 백업하거나 아카이브된 재실행 로그가 포함된 전체 또는 증분 백업을 수행할 때, 그리고 데이터베이스를 백업할 때는 재실행 로그 보존을 `0`이 아닌 값으로 설정해야 합니다.

복구 중에 데이터베이스 파일의 일관성을 유지하려면 아카이브된 재실행 로그가 필요합니다. 자세한 내용은 보관된 다시 실행 로그 보존 단원을 참조하십시오.

- DB 인스턴스에 백업을 보관할 수 있는 여유 공간이 충분한지 확인합니다. 데이터베이스를 백업할 때 Oracle 디렉터리 객체를 프로시저 호출의 파라미터로 지정합니다. RMAN은 파일을 지정된 디렉터리에 배치합니다. `DATA_PUMP_DIR`과 같은 기본 디렉터리를 사용하거나 새 디렉터리를 생성할 수 있습니다. 자세한 내용은 메인 데이터 스토리지 공간에서 디렉터리 생성 및 삭제 단원을 참조하십시오.

  CloudWatch 지표 `FreeStorageSpace`를 사용하여 RDS for Oracle 인스턴스의 현재 여유 공간을 모니터링할 수 있습니다. RMAN은 형식이 지정된 블록만 백업하고 압축을 지원하지만 여유 공간이 현재 데이터베이스 크기를 초과하는 것이 좋습니다.

RMAN 프로시저용 공통 파라미터

Amazon RDS 패키지인 `rdsadmin.rdsadmin_rman_util`에서 프로시저를 사용해 RMAN으로 작업을 수행할 수 있습니다. 이 패키지에서 몇 가지 파라미터는 프로시저에 공통됩니다. 이 패키지에는 다음과 같은 공통 파라미터가 있습니다.

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
`p_directory_name`	varchar	유효한 데이터베이스 디렉터리 이름입니다.	—	예	백업 파일을 담을 디렉터리의 이름입니다.
`p_label`	varchar	`a-z`, `A-Z`, `0-9`, `'_'`, `'-'`, `'.'`	—	아니요	백업 파일 이름에 포함된 고유 문자열입니다. **Note** 제한은 30자입니다.
`p_owner`	varchar	`p_directory_name`에 지정된 디	—	예	백업 파일을 담을 디렉터리의 소유자입니다.

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
		렉터리의 유효한 소유자입니다.			

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
p_tag	varchar	a-z, A-Z, 0-9, '_', '-', '.'	NULL	아니요	매일, 매주 또는 증분 수준 백업과 같은 백업의 목적이나 용도를 나타내기 위해 백업을 구별하는 데 사용할 수 있는 문자열입니다. 제한은 30자입니다. 태그는 대/소문자를 구분하지 않습니다. 태그를 입력할 때 사용된 대소문자에 관계없이 항상 대문자로 태그가 저장됩니다. 태그는 고유할 필요가 없으므로 여러 백업이 동일한 태그를 가질 수 있습니다. 태그를 지정하지 않으면 RMAN은 TAG*YYYYMMDDTHHMMSS* 형식을 사용하여 기본 태그를 자동으로 할당합니다. 여기서 *YYYY*는 연도, *MM*은 월, *DD*는 일, *HH*는 시간(24시간 형식), *MM*은 분, *SS*는 초입니다. 날짜 및 시간은 RMAN이 백업을 시작한 때를 나타냅니다. 예를 들어 백업은 2019-09-27 21:45:17에 시작된 백업에 대해 TAG20190927T214517 태그를 수신할 수 있습니다. p_tag 파라미터는 다음 Amazon RDS for Oracle DB 엔진 버전에서 지원됩니다. • Oracle Database 21c(21.0.0) • Oracle Database 19c(19.0.0), 19.0.0.0.ru-2021-10.rur-2021-10.r1 이상 사용

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
p_compress	boolean	TRUE, FALSE	FALSE	아니요	TRUE를 지정하여 기본 백업 압축을 활성화합니다. FALSE를 지정하여 기본 백업 압축을 비활성화합니다.
p_include_archive_logs	부울	TRUE, FALSE	FALSE	아니요	TRUE를 지정하여 보관된 재실행 로그를 백업에 포함합니다. FALSE를 지정하여 보관된 재실행 로그를 백업에서 제외합니다. 보관된 재실행 로그를 백업에 포함하는 경우 rdsadmin.rdsadmin_util.set_configuration 프로시저를 사용해 보존 기간을 1시간 이상으로 설정하세요. 또한 백업을 실행하기 전에 rdsadmin.rdsadmin_rman_util.crosscheck_archivelog 프로시저를 즉시 호출하세요. 이렇게 하지 않으면 Amazon RDS 관리 프로시저에서 삭제한 보관된 재실행 로그 파일이 누락되어 백업이 실패할 수 있습니다.
p_include_controlfile	부울	TRUE, FALSE	FALSE	아니요	TRUE를 지정하여 백업에 제어 파일을 포함합니다. FALSE를 지정하여 백업에서 제어 파일을 제외합니다.

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
p_optimize	부울	TRUE, FALSE	TRUE	아니요	보관된 재실행 로그가 포함된 경우 백업 크기를 줄이기 위해 TRUE를 지정하여 백업 최적화를 활성화합니다. FALSE를 지정하여 백업 최적화를 비활성화합니다.
p_parallel	숫자	Oracle Database Enterprise Edition(EE)에서 1과 254 사이의 유효한 정수입니다. 1다른 Oracle Database 에디션일 때는 입니다.	1	아니요	채널 수입니다.
p_rman_to_dbms_output	부울	TRUE, FALSE	FALSE	아니요	TRUE일 때는 RMAN 출력이 DBMS_OUTPUT 패키지를 비롯해 BDUMP 디렉터리의 파일로 전송됩니다. SQL*Plus에서 SET SERVEROUTPUT ON을 사용하여 출력을 확인합니다. FALSE일 때는 RMAN 출력이 BDUMP 디렉터리의 파일로만 전송됩니다.

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
p_section_size_mb	숫자	유효한 정수입니다.	NULL	아니요	섹션 크기(MB)입니다. 각 파일을 지정된 섹션 크기로 나누어서 함께 확인합니다. NULL일 때는 파라미터를 무시합니다.
p_validation_type	varchar	'PHYSICAL', 'PHYSICAL+LOGICAL'	'PHYS'	아니요	손상 탐지 수준입니다. 물리적 손상 여부를 알아보고 싶다면 'PHYSICAL' 을 지정합니다. 물리적 손상을 예로 들면 헤더와 푸터가 서로 일치하는 않는 블록이 있습니다. 물리적 손상 외에 논리적 불일치 여부도 알아보고 싶다면 'PHYSICAL+LOGICAL' 을 지정합니다. 논리적 손상의 예로는 잘못된 블록을 들 수 있습니다.

RDS for Oracle DB에서 데이터베이스 파일 검증

데이터 파일, 테이블스페이스, 제어 파일, 서버 파라미터 파일(SPFILE) 같은 Amazon RDS for Oracle 데이터베이스 파일은 Amazon RDS 패키지 `rdsadmin.rdsadmin_rman_util`을 사용하여 검증할 수 있습니다.

RMAN 확인에 대한 자세한 내용은 Oracle 설명서에서 Validating Database Files and Backups 및 VALIDATE를 참조하세요.

주제

- 데이터베이스 검증
- 테넌트 데이터베이스 검증
- 테이블 스페이스 확인
- 제어 파일 확인

- SPFILE 확인
- Oracle 데이터 파일 검증

데이터베이스 검증

RDS for Oracle에서 Oracle 데이터베이스가 사용하는 모든 관련 파일을 검증하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_rman_util.validate_database`를 사용합니다.

이 프로시저에서는 RMAN 작업을 위해 다음과 같은 공통 파라미터를 사용합니다.

- `p_validation_type`
- `p_parallel`
- `p_section_size_mb`
- `p_rman_to_dbms_output`

자세한 내용은 RMAN 프로시저용 공통 파라미터 단원을 참조하십시오.

다음 예에서는 파라미터의 기본값을 사용하여 데이터베이스를 검증합니다.

```
EXEC rdsadmin.rdsadmin_rman_util.validate_database;
```

다음 예에서는 파라미터의 지정된 값을 사용하여 데이터베이스를 검증합니다.

```
BEGIN
    rdsadmin.rdsadmin_rman_util.validate_database(
        p_validation_type     => 'PHYSICAL+LOGICAL',
        p_parallel            => 4,
        p_section_size_mb     => 10,
        p_rman_to_dbms_output => FALSE);
END;
/
```

`p_rman_to_dbms_output` 파라미터가 `FALSE`로 설정되면 RMAN 출력이 `BDUMP` 디렉터리의 파일로 작성됩니다.

`BDUMP` 디렉터리의 파일을 보려면 다음과 같이 `SELECT` 문을 실행합니다.

```
SELECT * FROM table(rdsadmin.rds_file_util.listdir('BDUMP')) order by mtime;
```

BDUMP 디렉터리의 파일 내용을 보려면 다음과 같이 SELECT 문을 실행합니다.

```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','rds-rman-validate-nnn.txt'));
```

파일 이름을 보려고 하는 파일 이름으로 변경합니다.

테넌트 데이터베이스 검증

컨테이너 데이터베이스(CDB)에 있는 테넌트 데이터베이스의 데이터 파일을 검증하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_rman_util.validate_tenant`를 사용합니다.

이 프로시저는 현재 테넌트 데이터베이스에만 적용되며 RMAN 작업을 위해 다음과 같은 공통 파라미터를 사용합니다.

- `p_validation_type`
- `p_parallel`
- `p_section_size_mb`
- `p_rman_to_dbms_output`

자세한 내용은 RMAN 프로시저용 공통 파라미터 단원을 참조하십시오. 이 프로시저는 다음 DB 엔진 버전에서 지원됩니다.

- Oracle Database 21c(21.0.0) CDB
- Oracle Database 19c(19.0.0) CDB

다음 예에서는 파라미터의 기본값을 사용하여 현재 테넌트 데이터베이스를 검증합니다.

```
EXEC rdsadmin.rdsadmin_rman_util.validate_tenant;
```

다음 예에서는 파라미터의 지정된 값을 사용하여 현재 테넌트 데이터베이스를 검증합니다.

```
BEGIN
    rdsadmin.rdsadmin_rman_util.validate_tenant(
        p_validation_type     => 'PHYSICAL+LOGICAL',
        p_parallel            => 4,
        p_section_size_mb     => 10,
        p_rman_to_dbms_output => FALSE);
END;
```

```
/
```

p_rman_to_dbms_output 파라미터가 FALSE로 설정되면 RMAN 출력이 BDUMP 디렉터리의 파일로 작성됩니다.

BDUMP 디렉터리의 파일을 보려면 다음과 같이 SELECT 문을 실행합니다.

```
SELECT * FROM table(rdsadmin.rds_file_util.listdir('BDUMP')) order by mtime;
```

BDUMP 디렉터리의 파일 내용을 보려면 다음과 같이 SELECT 문을 실행합니다.

```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','rds-rman-
validate-nnn.txt'));
```

파일 이름을 보려고 하는 파일 이름으로 변경합니다.

테이블 스페이스 확인

테이블 스페이스와 연결된 파일을 확인하려면 Amazon RDS 프로시저 rdsadmin.rdsadmin_rman_util.validate_tablespace를 사용합니다.

이 프로시저에서는 RMAN 작업을 위해 다음과 같은 공통 파라미터를 사용합니다.

- p_validation_type
- p_parallel
- p_section_size_mb
- p_rman_to_dbms_output

자세한 내용은 RMAN 프로시저용 공통 파라미터 섹션을 참조하세요.

이 프로시저에서는 다음과 같은 추가 파라미터도 사용합니다.

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
p_tablespace_name	varchar2	유효한 테이블 스페이스 이름	—	예	테이블 스페이스 이름입니다.

제어 파일 확인

Amazon RDS Oracle DB 인스턴스에서 사용되는 제어 파일을 확인하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_rman_util.validate_current_controlfile`을 사용합니다.

이 프로시저에서는 RMAN 작업을 위해 다음과 같은 공통 파라미터를 사용합니다.

- `p_validation_type`
- `p_rman_to_dbms_output`

자세한 내용은 RMAN 프로시저용 공통 파라미터 섹션을 참조하세요.

SPFILE 확인

Amazon RDS Oracle DB 인스턴스에서 사용되는 서버 파라미터 파일(SPFILE)만 확인하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_rman_util.validate_spfile`을 사용합니다.

이 프로시저에서는 RMAN 작업을 위해 다음과 같은 공통 파라미터를 사용합니다.

- `p_validation_type`
- `p_rman_to_dbms_output`

자세한 내용은 RMAN 프로시저용 공통 파라미터 단원을 참조하십시오.

Oracle 데이터 파일 검증

데이터 파일을 확인하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_rman_util.validate_datafile`을 사용합니다.

이 프로시저에서는 RMAN 작업을 위해 다음과 같은 공통 파라미터를 사용합니다.

- `p_validation_type`
- `p_parallel`
- `p_section_size_mb`
- `p_rman_to_dbms_output`

자세한 내용은 RMAN 프로시저용 공통 파라미터 섹션을 참조하세요.

이 프로시저에서는 다음과 같은 추가 파라미터도 사용합니다.

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
p_datafile	varchar2	유효한 데이터파일 ID 번호 또는 유효한 데이터 파일 이름(전체 경로 포함)	—	예	데이터파일 ID 번호(v$datafile.file#) 또는 경로가 포함된 전체 데이터파일 이름(v$datafile.name) 입니다.
p_from_block	숫자	유효한 정수입니다.	NULL	아니요	데이터 파일 내에서 확인이 시작되는 블록 번호입니다. 이 값이 NULL인 경우 1이 사용됩니다.
p_to_block	숫자	유효한 정수입니다.	NULL	아니요	데이터 파일 내에서 확인이 끝나는 블록 번호입니다. 이 값이 NULL일 때는 데이터 파일에서 최대 블록이 사용됩니다.

블록 변경 추적 활성화 및 비활성화

블록 변경 내용 추적 기능은 추적 파일에 변경된 블록을 기록합니다. 이 기법을 사용하면 RMAN 증분 백업의 성능을 개선할 수 있습니다. 자세한 내용은 Oracle 데이터베이스 설명서의 블록 변경 내용 추적을 사용하여 증분 백업 성능 향상 섹션을 참조하세요.

RMAN 기능은 읽기 전용 복제본에서는 지원되지 않습니다. 하지만 고가용성 전략의 일환으로 프로시저 rdsadmin.rdsadmin_rman_util.enable_block_change_tracking을 사용하여 읽기 전용 복제본에서 블록 추적을 활성화하도록 선택할 수 있습니다. 이 읽기 전용 복제본을 소스 DB 인스턴스로 승격시키면 새 소스 인스턴스에서 블록 변경 추적 기능이 활성화됩니다. 따라서 인스턴스를 통해 빠른 증분 백업의 이점을 누릴 수 있습니다.

블록 변경 사항 추적 프로시저는 다음 DB 엔진 버전의 Enterprise Edition에만 지원됩니다.

- Oracle Database 21c(21.0.0)
- Oracle Database 19c(19.0.0)

> **Note**
>
> 단일 테넌트 CDB에서는 다음 작업이 작동하지만 고객에게 표시되는 메커니즘으로 작업의 현재 상태를 감지할 수 없습니다. 또한 RDS for Oracle CDB 제한 사항 단원도 참조하세요.

DB 인스턴스에 대한 블록 변경 내용 추적을 활성화하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_rman_util.enable_block_change_tracking`을 사용합니다. 블록 변경 내용 추적을 비활성화하려면 `disable_block_change_tracking`을 사용합니다. 이 프로시저에는 파라미터가 없습니다.

DB 인스턴스에 대해 블록 변경 추적이 활성화되어 있는지 확인하려면 다음 쿼리를 실행하세요.

```
SELECT STATUS, FILENAME FROM V$BLOCK_CHANGE_TRACKING;
```

다음 예에서는 DB 인스턴스에 대한 블록 변경 추적을 활성화합니다.

```
EXEC rdsadmin.rdsadmin_rman_util.enable_block_change_tracking;
```

다음 예에서는 DB 인스턴스에 대한 블록 변경 추적을 비활성화합니다.

```
EXEC rdsadmin.rdsadmin_rman_util.disable_block_change_tracking;
```

보관된 재실행 로그 대조 확인

보관된 재실행 로그를 Amazon RDS 프로시저 `rdsadmin.rdsadmin_rman_util.crosscheck_archivelog`를 사용해 대조 확인할 수 있습니다.

이 프로시저를 사용하여 제어 파일에 등록된 아카이브된 다시 실행 로그를 대조 확인하고 선택 사항으로 만료된 로그 레코드를 삭제할 수 있습니다. RMAN은 백업을 수행할 때 제어 파일에 레코드를 생성합니다. 시간이 지남에 따라 이 레코드로 인해 제어 파일의 크기가 증가합니다. 만료된 레코드는 주기적으로 제거하는 것이 좋습니다.

> **Note**
>
> 표준 Amazon RDS 백업은 RMAN을 사용하지 않으므로 제어 파일에 레코드를 생성하지 않습니다.

이 프로시저에서는 RMAN 작업을 위해 공통 파라미터인 `p_rman_to_dbms_output`을 사용합니다.

자세한 내용은 RMAN 프로시저용 공통 파라미터 섹션을 참조하세요.

이 프로시저에서는 다음과 같은 추가 파라미터도 사용합니다.

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
`p_delete_expired`	부울	`TRUE`, `FALSE`	`TRUE`	아니요	`TRUE`인 경우 제어 파일에서 만료된 아카이브된 다시 실행 로그 레코드를 삭제합니다. `FALSE`인 경우 제어 파일에 만료된 아카이브된 다시 실행 로그 레코드를 보관합니다.

이 프로시저는 다음 Amazon RDS for Oracle 엔진 버전에서 지원됩니다.

- Oracle Database 21c(21.0.0)
- Oracle Database 19c(19.0.0)

다음 예에서는 제어 파일에서 아카이브된 다시 실행 로그 레코드를 만료됨으로 표시하지만 레코드를 삭제하지는 않습니다.

```
BEGIN
    rdsadmin.rdsadmin_rman_util.crosscheck_archivelog(
        p_delete_expired      => FALSE,
        p_rman_to_dbms_output => FALSE);
END;
```

```
/
```

다음 예에서는 제어 파일에서 만료된 아카이브된 다시 실행 로그를 삭제합니다.

```
BEGIN
    rdsadmin.rdsadmin_rman_util.crosscheck_archivelog(
        p_delete_expired      => TRUE,
        p_rman_to_dbms_output => FALSE);
END;
/
```

보관된 재실행 로그 파일 백업

Amazon RDS 패키지인 `rdsadmin.rdsadmin_rman_util`을(를) 사용해 Amazon RDS Oracle DB 인스턴스에 대해 보관된 재실행 로그를 백업할 수 있습니다.

보관된 재실행 로그 백업 프로시저는 다음과 같은 Amazon RDS for Oracle 엔진 버전에서 지원됩니다.

- Oracle Database 21c(21.0.0)
- Oracle Database 19c(19.0.0)

주제

- 보관된 재실행 로그 전체 백업
- 날짜 범위에서 보관된 재실행 로그 백업
- SCN 범위에서 보관된 재실행 로그 백업
- 시퀀스 번호 범위에서 보관된 재실행 로그 백업

보관된 재실행 로그 전체 백업

Amazon RDS Oracle DB 인스턴스에 대해 보관된 다시 실행 로그를 모두 백업하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_rman_util.backup_archivelog_all`을 사용합니다.

이 프로시저에서는 RMAN 작업을 위해 다음과 같은 공통 파라미터를 사용합니다.

- `p_owner`
- `p_directory_name`
- `p_label`
- `p_parallel`

- p_compress
- p_rman_to_dbms_output
- p_tag

자세한 내용은 RMAN 프로시저용 공통 파라미터 섹션을 참조하세요.

다음 예에서는 DB 인스턴스에 대해 모든 보관된 재실행 로그를 백업합니다.

```
BEGIN
    rdsadmin.rdsadmin_rman_util.backup_archivelog_all(
        p_owner               => 'SYS',
        p_directory_name      => 'MYDIRECTORY',
        p_parallel            => 4,
        p_tag                 => 'MY_LOG_BACKUP',
        p_rman_to_dbms_output => FALSE);
END;
/
```

날짜 범위에서 보관된 재실행 로그 백업

날짜 범위를 지정하여 Amazon RDS Oracle DB 인스턴스에 대해 보관된 특정 다시 실행 로그를 백업하려면 Amazon RDS 프로시저 rdsadmin.rdsadmin_rman_util.backup_archivelog_date를 사용합니다. 날짜 범위에서는 백업할 보관된 재실행 로그를 지정합니다.

이 프로시저에서는 RMAN 작업을 위해 다음과 같은 공통 파라미터를 사용합니다.

- p_owner
- p_directory_name
- p_label
- p_parallel
- p_compress
- p_rman_to_dbms_output
- p_tag

자세한 내용은 RMAN 프로시저용 공통 파라미터 섹션을 참조하세요.

이 프로시저에서는 다음과 같은 추가 파라미터도 사용합니다.

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
p_from_date	날짜	디스크에 있는 보관된 재실행 로그의 start_date 및 next_date 사이의 날짜입니다. 이 값은 p_to_date 에 대해 지정된 값과 같거나 이보다 작아야 합니다.	—	예	보관된 로그 백업의 시작 날짜입니다.
p_to_date	날짜	디스크에 있는 보관된 재실행 로그의 start_date 및 next_date 사이의 날짜입니다. 이 값은 p_from_date 에 대	—	예	보관된 로그 백업의 종료 날짜입니다.

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
		해 지정된 값과 같거나 이보다 커야 합니다.			

다음 예에서는 DB 인스턴스의 날짜 범위에서 보관된 재실행 로그를 백업합니다.

```
BEGIN
    rdsadmin.rdsadmin_rman_util.backup_archivelog_date(
        p_owner                => 'SYS',
        p_directory_name       => 'MYDIRECTORY',
        p_from_date            => '03/01/2019 00:00:00',
        p_to_date              => '03/02/2019 00:00:00',
        p_parallel             => 4,
        p_tag                  => 'MY_LOG_BACKUP',
        p_rman_to_dbms_output => FALSE);
END;
/
```

SCN 범위에서 보관된 재실행 로그 백업

시스템 변경 번호(SCN) 범위를 지정하여 Amazon RDS Oracle DB 인스턴스에 대해 보관된 특정 다시 실행 로그를 백업하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_rman_util.backup_archivelog_scn`을 사용합니다. SCN 범위에서는 백업할 보관된 재실행 로그를 지정합니다.

이 프로시저에서는 RMAN 작업을 위해 다음과 같은 공통 파라미터를 사용합니다.

- `p_owner`
- `p_directory_name`
- `p_label`
- `p_parallel`
- `p_compress`

- p_rman_to_dbms_output
- p_tag

자세한 내용은 RMAN 프로시저용 공통 파라미터 섹션을 참조하세요.

이 프로시저에서는 다음과 같은 추가 파라미터도 사용합니다.

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
p_from_scn	숫자	디스크에 있는 보관된 재실행 로그의 SCN입니다. 이 값은 p_to_scn에 대해 지정된 값과 같거나 이보다 작아야 합니다.	—	예	보관된 로그 백업의 시작 SCN입니다.
p_to_scn	숫자	디스크에 있는 보관된 재실행 로그의 SCN입니다. 이 값은 p_from_scn 에 대해 지정된 값과 같거나 이보다	—	예	보관된 로그 백업의 종료 SCN입니다.

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
		커야 합니다.			

다음 예에서는 DB 인스턴스의 SCN 범위에서 보관된 재실행 로그를 백업합니다.

```
BEGIN
    rdsadmin.rdsadmin_rman_util.backup_archivelog_scn(
        p_owner                => 'SYS',
        p_directory_name       => 'MYDIRECTORY',
        p_from_scn             => 1533835,
        p_to_scn               => 1892447,
        p_parallel             => 4,
        p_tag                  => 'MY_LOG_BACKUP',
        p_rman_to_dbms_output => FALSE);
END;
/
```

시퀀스 번호 범위에서 보관된 재실행 로그 백업

시퀀스 번호 범위를 지정하여 Amazon RDS Oracle DB 인스턴스에 대해 보관된 특정 다시 실행 로그를 백업하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_rman_util.backup_archivelog_sequence`를 사용합니다. 시퀀스 번호 범위에서는 백업할 보관된 재실행 로그를 지정합니다.

이 프로시저에서는 RMAN 작업을 위해 다음과 같은 공통 파라미터를 사용합니다.

- `p_owner`
- `p_directory_name`
- `p_label`
- `p_parallel`
- `p_compress`
- `p_rman_to_dbms_output`
- `p_tag`

자세한 내용은 RMAN 프로시저용 공통 파라미터 섹션을 참조하세요.

이 프로시저에서는 다음과 같은 추가 파라미터도 사용합니다.

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
p_from_sequence	숫자	디스크에 있는 보관된 재실행 로그의 시퀀스 번호입니다. 이 값은 p_to_sequence 에 대해 지정된 값과 같거나 이보다 작아야 합니다.	—	예	보관된 로그 백업의 시작 시퀀스 번호입니다.
p_to_sequence	숫자	디스크에 있는 보관된 재실행 로그의 시퀀스 번호입니다. 이 값은 p_from_sequence 에 대해 지정된 값과 같거나 이보다 커야 합니다.	—	예	보관된 로그 백업의 종료 시퀀스 번호입니다.

다음 예에서는 DB 인스턴스의 시퀀스 번호 범위 범위에서 보관된 재실행 로그를 백업합니다.

```
BEGIN
    rdsadmin.rdsadmin_rman_util.backup_archivelog_sequence(
        p_owner               => 'SYS',
        p_directory_name      => 'MYDIRECTORY',
        p_from_sequence       => 11160,
        p_to_sequence         => 11160,
        p_parallel            => 4,
        p_tag                 => 'MY_LOG_BACKUP',
        p_rman_to_dbms_output => FALSE);
END;
/
```

전체 데이터베이스 백업 수행

Amazon RDS 프로시저인 `rdsadmin.rdsadmin_rman_util.backup_database_full`을 사용해 백업에 포함된 데이터 파일의 모든 블록을 백업할 수 있습니다.

이 프로시저에서는 RMAN 작업을 위해 다음과 같은 공통 파라미터를 사용합니다.

- `p_owner`
- `p_directory_name`
- `p_label`
- `p_parallel`
- `p_section_size_mb`
- `p_include_archive_logs`
- `p_optimize`
- `p_compress`
- `p_rman_to_dbms_output`
- `p_tag`

자세한 내용은 RMAN 프로시저용 공통 파라미터 섹션을 참조하세요.

이 프로시저는 다음 Amazon RDS for Oracle 엔진 버전에서 지원됩니다.

- Oracle Database 21c(21.0.0)
- Oracle Database 19c(19.0.0)

다음 예에서는 파라미터에 지정된 값을 사용하여 완전한 DB 인스턴스 백업을 수행합니다.

```
BEGIN
    rdsadmin.rdsadmin_rman_util.backup_database_full(
        p_owner               => 'SYS',
        p_directory_name      => 'MYDIRECTORY',
        p_parallel            => 4,
        p_section_size_mb     => 10,
        p_tag                 => 'FULL_DB_BACKUP',
        p_rman_to_dbms_output => FALSE);
END;
/
```

테넌트 데이터베이스의 전체 백업 수행

컨테이너 데이터베이스(CDB)에 테넌트 데이터베이스를 포함한 데이터 블록을 모두 백업할 수 있습니다. Amazon RDS 프로시저 `rdsadmin.rdsadmin_rman_util.backup_tenant_full`을 사용합니다. 이 프로시저는 현재 데이터베이스 백업에만 적용되며 RMAN 작업을 위해 다음과 같은 공통 파라미터를 사용합니다.

- `p_owner`
- `p_directory_name`
- `p_label`
- `p_parallel`
- `p_section_size_mb`
- `p_include_archive_logs`
- `p_optimize`
- `p_compress`
- `p_rman_to_dbms_output`
- `p_tag`

자세한 내용은 RMAN 프로시저용 공통 파라미터 단원을 참조하십시오.

`rdsadmin_rman_util.backup_tenant_full` 프로시저는 다음 RDS for Oracle DB 엔진 버전에서 지원됩니다.

- Oracle Database 21c(21.0.0) CDB

- Oracle Database 19c(19.0.0) CDB

다음 예에서는 파라미터에 지정된 값을 사용하여 현재 테넌트 데이터베이스에 전체 백업을 수행합니다.

```
BEGIN
    rdsadmin.rdsadmin_rman_util.backup_tenant_full(
        p_owner               => 'SYS',
        p_directory_name      => 'MYDIRECTORY',
        p_parallel            => 4,
        p_section_size_mb     => 10,
        p_tag                 => 'FULL_TENANT_DB_BACKUP',
        p_rman_to_dbms_output => FALSE);
END;
/
```

증분 데이터베이스 백업 수행

Amazon RDS 프로시저인 `rdsadmin.rdsadmin_rman_util.backup_database_incremental`을 사용해 DB 인스턴스에 대한 증분 백업을 수행할 수 있습니다.

증분 백업에 대한 자세한 내용은 Oracle 설명서의 증분 백업을 참조하세요.

이 프로시저에서는 RMAN 작업을 위해 다음과 같은 공통 파라미터를 사용합니다.

- `p_owner`
- `p_directory_name`
- `p_label`
- `p_parallel`
- `p_section_size_mb`
- `p_include_archive_logs`
- `p_include_controlfile`
- `p_optimize`
- `p_compress`
- `p_rman_to_dbms_output`
- `p_tag`

자세한 내용은 RMAN 프로시저용 공통 파라미터 섹션을 참조하세요.

이 프로시저는 다음 Amazon RDS for Oracle 엔진 버전에서 지원됩니다.

- Oracle Database 21c(21.0.0)
- Oracle Database 19c(19.0.0)

이 프로시저에서는 다음과 같은 추가 파라미터도 사용합니다.

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
p_level	숫자	0, 1	0	아니요	0을 지정하여 전체 증분 백업을 활성화합니다. 1을 지정하여 비누적 증분 백업을 활성화합니다.

다음 예에서는 지정된 값을 파라미터에 사용하여 DB 인스턴스에 대한 증분 백업을 수행합니다.

```
BEGIN
    rdsadmin.rdsadmin_rman_util.backup_database_incremental(
        p_owner                => 'SYS',
        p_directory_name       => 'MYDIRECTORY',
        p_level                => 1,
        p_parallel             => 4,
        p_section_size_mb      => 10,
        p_tag                  => 'MY_INCREMENTAL_BACKUP',
        p_rman_to_dbms_output => FALSE);
END;
/
```

테넌트 데이터베이스의 증분 백업 수행

CDB에서 현재 테넌트 데이터베이스의 증분 백업을 수행할 수 있습니다. Amazon RDS 프로시저 `rdsadmin.rdsadmin_rman_util.backup_tenant_incremental`을 사용합니다.

증분 백업에 대한 자세한 내용은 Oracle 데이터베이스 설명서의 증분 백업을 참조하세요.

이 프로시저는 현재 테넌트 데이터베이스에만 적용되며 RMAN 작업을 위해 다음과 같은 공통 파라미터를 사용합니다.

- p_owner
- p_directory_name
- p_label
- p_parallel
- p_section_size_mb
- p_include_archive_logs
- p_include_controlfile
- p_optimize
- p_compress
- p_rman_to_dbms_output
- p_tag

자세한 내용은 RMAN 프로시저용 공통 파라미터 단원을 참조하십시오.

이 프로시저는 다음 Amazon RDS for Oracle 엔진 버전에서 지원됩니다.

- Oracle Database 21c(21.0.0) CDB
- Oracle Database 19c(19.0.0) CDB

이 프로시저에서는 다음과 같은 추가 파라미터도 사용합니다.

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
p_level	숫자	0, 1	0	아니요	0을 지정하여 전체 증분 백업을 활성화합니다.

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
					1을 지정하여 비누적 증분 백업을 활성화합니다.

다음 예에서는 파라미터에 지정된 값을 사용하여 현재 테넌트 데이터베이스에 증분 백업을 수행합니다.

```
BEGIN
    rdsadmin.rdsadmin_rman_util.backup_tenant_incremental(
        p_owner               => 'SYS',
        p_directory_name      => 'MYDIRECTORY',
        p_level               => 1,
        p_parallel            => 4,
        p_section_size_mb     => 10,
        p_tag                 => 'MY_INCREMENTAL_BACKUP',
        p_rman_to_dbms_output => FALSE);
END;
/
```

테이블스페이스 백업

Amazon RDS 프로시저 `rdsadmin.rdsadmin_rman_util.backup_tablespace`를 사용하여 테이블스페이스를 백업할 수 있습니다.

이 프로시저에서는 RMAN 작업을 위해 다음과 같은 공통 파라미터를 사용합니다.

- `p_owner`
- `p_directory_name`
- `p_label`
- `p_parallel`
- `p_section_size_mb`
- `p_include_archive_logs`
- `p_include_controlfile`
- `p_optimize`

- p_compress
- p_rman_to_dbms_output
- p_tag

자세한 내용은 RMAN 프로시저용 공통 파라미터 섹션을 참조하세요.

이 프로시저에서는 다음과 같은 추가 파라미터도 사용합니다.

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
p_tablespace_name	varchar2	유효한 테이블스페이스 이름입니다.	—	예	백업할 테이블스페이스의 이름입니다.

이 프로시저는 다음 Amazon RDS for Oracle 엔진 버전에서 지원됩니다.

- Oracle Database 21c(21.0.0)
- Oracle Database 19c(19.0.0)

다음 예에서는 지정된 값을 파라미터에 사용하여 테이블스페이스 백업을 수행합니다.

```
BEGIN
    rdsadmin.rdsadmin_rman_util.backup_tablespace(
        p_owner               => 'SYS',
        p_directory_name      => 'MYDIRECTORY',
        p_tablespace_name     => 'MYTABLESPACE',
        p_parallel            => 4,
        p_section_size_mb     => 10,
        p_tag                 => 'MYTABLESPACE_BACKUP',
        p_rman_to_dbms_output => FALSE);
END;
/
```

Backing up a control file

Amazon RDS 프로시저 `rdsadmin.rdsadmin_rman_util.backup_current_controlfile`를 사용하여 제어 파일을 백업할 수 있습니다.

이 프로시저에서는 RMAN 작업을 위해 다음과 같은 공통 파라미터를 사용합니다.

- `p_owner`
- `p_directory_name`
- `p_label`
- `p_compress`
- `p_rman_to_dbms_output`
- `p_tag`

자세한 내용은 RMAN 프로시저용 공통 파라미터 섹션을 참조하세요.

이 프로시저는 다음 Amazon RDS for Oracle 엔진 버전에서 지원됩니다.

- Oracle Database 21c(21.0.0)
- Oracle Database 19c(19.0.0)

다음 예에서는 지정된 값을 파라미터에 사용하여 제어 파일을 백업합니다.

```
BEGIN
    rdsadmin.rdsadmin_rman_util.backup_current_controlfile(
        p_owner               => 'SYS',
        p_directory_name      => 'MYDIRECTORY',
        p_tag                 => 'CONTROL_FILE_BACKUP',
        p_rman_to_dbms_output => FALSE);
END;
/
```

블록 미디어 복구 수행

Amazon RDS 프로시저 `rdsadmin.rdsadmin_rman_util.recover_datafile_block`을 사용하여 블록 미디어 복구라고 하는 개별 데이터 블록을 복구할 수 있습니다. 이 오버로드된 절차를 사용하여 개별 데이터 블록 또는 일정 범위의 데이터 블록을 복구할 수 있습니다.

이 프로시저에서는 RMAN 작업을 위해 다음과 같은 공통 파라미터를 사용합니다.

- p_rman_to_dbms_output

자세한 내용은 RMAN 프로시저용 공통 파라미터 단원을 참조하십시오.

이 프로시저에서는 다음과 같은 추가 파라미터를 사용합니다.

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
p_datafile	NUMBER	유효한 데이터 파일 ID 번호.	—	예	손상된 블록을 포함하는 데이터 파일. 다음 방법 중 하나를 사용하여 데이터 파일을 지정합니다. • V$DATAFILE.FILE# 에 있는 데이터 파일 ID 번호. • 경로를 포함하여 V$DATAFILE.NAME 에 있는 전체 데이터 파일 이름.
p_block	NUMBER	유효한 정수.	—	예	복구할 개별 블록의 수. 다음 파라미터는 함께 사용할 수 없습니다. • p_block • p_from_block 및 p_to_block
p_from_block	NUMBER	유효한 정수.	—	예	복구할 블록 범위 중 첫 번째 블록 번호. 다음 파라미터는 함께 사용할 수 없습니다. • p_block

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
					• p_from_block 및 p_to_block
p_to_block	NUMBER	유효한 정수.	—	예	복구할 블록 범위 중 마지막 블록 번호. 다음 파라미터는 함께 사용할 수 없습니다. • p_block • p_from_block 및 p_to_block

이 프로시저는 다음 Amazon RDS for Oracle 엔진 버전에서 지원됩니다.

- Oracle Database 21c(21.0.0)
- Oracle Database 19c(19.0.0)

다음 예제에서는 데이터 파일 5의 블록 100을 복구합니다.

```
BEGIN
    rdsadmin.rdsadmin_rman_util.recover_datafile_block(
        p_datafile            => 5,
        p_block               => 100,
        p_rman_to_dbms_output => TRUE);
END;
/
```

다음 예제에서는 데이터 파일 5의 블록 100~150을 복구합니다.

```
BEGIN
    rdsadmin.rdsadmin_rman_util.recover_datafile_block(
        p_datafile            => 5,
        p_from_block          => 100,
        p_to_block            => 150,
        p_rman_to_dbms_output => TRUE);
```

```
END;
/
```

Oracle DB 인스턴스에 대한 공통 스케줄링 작업 수행

일부 SYS 소유 스케줄러 작업이 일반 데이터베이스 작업을 방해할 수 있습니다. 이러한 경우 Oracle Support는 일정을 수정할 것을 권장합니다. SYS 작업을 활성화 또는 비활성화해야 하는 경우 프로덕션 환경에서 구현하기 전에 테스트 환경에서 예약된 작업에 대해 작업을 테스트합니다. SYS 소유 Oracle Scheduler 작업을 위한 태스크를 수행하려면 Amazon RDS 패키지 `rdsadmin.rdsadmin_dbms_scheduler`를 사용합니다.

`rdsadmin.rdsadmin_dbms_scheduler` 프로시저는 다음 표에 나오는 Amazon RDS for Oracle DB 엔진 버전에서 지원됩니다. 이 패키지를 사용할 때 표에 나열된 SYS 작업을 지정할 수 있습니다.

데이터베이스 릴리스	기본적으로 활성화되는 작업	기본적으로 비활성화되는 작업
Oracle Database 19c	BSLN_MAINTAIN_STATS_JOB CLEANUP_NON_EXIST_OBJ CLEANUP_ONLINE_IND_BUILD CLEANUP_ONLINE_PMO CLEANUP_TAB_IOT_PMO CLEANUP_TRANSIENT_PKG CLEANUP_TRANSIENT_TYPE DRA_REEVALUATE_OPEN_FAILU RES FILE_SIZE_UPD ORA$AUTOTASK_CLEAN PMO_DEFERRED_GIDX_MAINT_JO B PURGE_LOG RSE$CLEAN_RECOVERABLE_SC RIPT SM$CLEAN_AUTO_SPLIT_MERGE	FGR$AUTOPURGE_JOB FILE_WATCHER HM_CREATE_OFFLINE_DICTIONARY LOAD_OPATCH_INVENTORY ORA$PREPLUGIN_BACKUP_JOB XMLDB_NFS_CLEANUP_JOB
Oracle Database 21c	BSLN_MAINTAIN_STATS_JOB CLEANUP_NON_EXIST_OBJ CLEANUP_ONLINE_IND_BUILD CLEANUP_ONLINE_PMO CLEANUP_TAB_IOT_PMO	FGR$AUTOPURGE_JOB FILE_WATCHER HM_CREATE_OFFLINE_DICTIONARY LOAD_OPATCH_INVENTORY ORA$PREPLUGIN_BACKUP_JOB

데이터베이스 릴리스	기본적으로 활성화되는 작업	기본적으로 비활성화되는 작업
	CLEANUP_TRANSIENT_PKG CLEANUP_TRANSIENT_TYPE DRA_REEVALUATE_OPEN_FAILURES FILE_SIZE_UPD ORA$AUTOTASK_CLEAN PMO_DEFERRED_GIDX_MAINT_JOB PURGE_LOG	ORA$_ATSK_AUTOSTS XMLDB_NFS_CLEANUP_JOB

Oracle Scheduler 프로시저용 공통 파라미터

Oracle Scheduler를 사용하여 작업을 수행하려면 Amazon RDS 패키지 `rdsadmin.rdsadmin_dbms_scheduler`의 프로시저를 사용합니다. 이 패키지에서 몇 가지 파라미터는 프로시저에 공통됩니다. 이 패키지에는 다음과 같은 공통 파라미터가 있습니다.

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
`name`	varchar2	Oracle DB 인스턴스에 대한 공통 스케줄링 작업 수행의 테이블에 나열된 프로시저	—	예	수정할 작업의 이름입니다.
`attribute`	varchar2	'REPEAT_INTERVAL' _NAME'	–	예	수정할 속성입니다. 작업에 대한 반복 간격을 수정하려면 'REPEAT_I

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
					NTERVAL' 을 지정합니다. 작업에 대한 일정 이름을 수정하려면 'SCHEDULE _NAME' 을 지정합니다.
value	varchar2	사용하는 속성에 따라 유효한 일정 간격 또는 일정 이름입니다.	–	예	속성의 새로운 값입니다.

DBMS_SCHEDULER 작업 변경

Oracle Scheduler의 특정 구성 요소를 수정하려면 Oracle 프로시저 dbms_scheduler.set_attribute를 사용합니다. 자세한 내용은 Oracle 문서의 DBMS_SCHEDULER와 SET_ATTRIBUTE Procedure 단원을 참조하세요.

Amazon RDS DB 인스턴스를 작업할 때는, 스키마 이름 SYS를 객체 이름 앞에 붙이십시오. 다음은 월요일 창 객체에 자원 계획 속성을 설정하는 예제입니다.

```
BEGIN
    DBMS_SCHEDULER.SET_ATTRIBUTE(
        name      => 'SYS.MONDAY_WINDOW',
        attribute => 'RESOURCE_PLAN',
        value     => 'resource_plan_1');
END;
/
```

AutoTask 유지 관리 기간 수정

Amazon RDS for Oracle 인스턴스는 유지 관리 기간에 대한 기본 설정을 사용하여 생성됩니다. 이러한 기간 동안 옵티마이저 통계 수집과 같은 자동화된 유지 관리 태스크가 실행됩니다. 기본적으로 유지 관리 기간은 Oracle 데이터베이스 리소스 관리자를 활성화합니다.

이 기간을 수정하려면 DBMS_SCHEDULER 패키지를 사용합니다. 다음과 같은 이유로 유지 관리 기간 설정을 수정해야 할 수 있습니다.

- 유지 관리 작업을 다른 시간에 실행하거나, 다른 설정으로 실행하거나, 전혀 실행하지 않고자 할 수 있습니다. 예를 들어 이 기간의 지속 시간을 수정하거나, 반복 시간 및 간격을 변경할 수 있습니다.
- 유지 관리 중에 리소스 관리자를 사용하도록 활성화할 경우 성능에 미치는 영향을 피하고자 할 수 있습니다. 예를 들어 기본 유지 관리 계획이 지정되어 있고 데이터베이스의 부하가 클 때 유지 관리 기간이 시작되면 `resmgr:cpu quantum`과 같은 대기 이벤트가 발생할 수 있습니다. 이 대기 이벤트는 데이터베이스 리소스 관리자와 관련이 있습니다. 다음과 같은 옵션이 있습니다:
  - DB 인스턴스의 사용량이 적은 시간 동안 유지 관리 기간이 활성화되는지 확인합니다.
  - `resource_plan`속성을 빈 문자열로 설정하여 기본 유지 관리 계획을 비활성화합니다.
  - 파라미터 그룹에서 `resource_manager_plan` 파라미터를 `FORCE:`로 설정합니다. 인스턴스가 Enterprise Edition을 사용하는 경우 이 설정을 사용하면 데이터베이스 리소스 관리자 계획이 활성화되지 않습니다.

유지 관리 기간 설정을 수정하려면

1. Oracle SQL 클라이언트를 사용하여 데이터베이스에 연결합니다.
2. 스케줄러 기간에 대한 현재 구성을 쿼리합니다.

   다음 예에서는 `MONDAY_WINDOW`의 구성을 쿼리합니다.

   ```
   SELECT ENABLED, RESOURCE_PLAN, DURATION, REPEAT_INTERVAL
   FROM   DBA_SCHEDULER_WINDOWS
   WHERE  WINDOW_NAME='MONDAY_WINDOW';
   ```

   다음 출력은 이 기간이 기본값을 사용하고 있음을 보여줍니다.

   ```
   ENABLED         RESOURCE_PLAN                  DURATION         REPEAT_INTERVAL
   --------------- ------------------------------ ---------------
    ------------------------------
   ```

```
TRUE            DEFAULT_MAINTENANCE_PLAN       +000 04:00:00
 freq=daily;byday=MON;byhour=22
                                                                ;byminute=0;
 bysecond=0
```

3. DBMS_SCHEDULER 패키지를 사용하여 이 기간을 수정합니다.

   다음 예에서는 리소스 관리자가 유지 관리 기간 동안 실행되지 않도록 리소스 계획을 null로 설정합니다.

```
BEGIN
  -- disable the window to make changes
  DBMS_SCHEDULER.DISABLE(name=>'"SYS"."MONDAY_WINDOW"',force=>TRUE);

  -- specify the empty string to use no plan
  DBMS_SCHEDULER.SET_ATTRIBUTE(name=>'"SYS"."MONDAY_WINDOW"',
 attribute=>'RESOURCE_PLAN', value=>'');

  -- re-enable the window
  DBMS_SCHEDULER.ENABLE(name=>'"SYS"."MONDAY_WINDOW"');
END;
/
```

   다음 예는 이 기간의 최대 지속 시간을 2시간으로 설정합니다.

```
BEGIN
  DBMS_SCHEDULER.DISABLE(name=>'"SYS"."MONDAY_WINDOW"',force=>TRUE);
  DBMS_SCHEDULER.SET_ATTRIBUTE(name=>'"SYS"."MONDAY_WINDOW"',
 attribute=>'DURATION', value=>'0 2:00:00');
  DBMS_SCHEDULER.ENABLE(name=>'"SYS"."MONDAY_WINDOW"');
END;
/
```

   다음 예는 반복 간격을 매주 월요일 오전 10시로 설정합니다.

```
BEGIN
  DBMS_SCHEDULER.DISABLE(name=>'"SYS"."MONDAY_WINDOW"',force=>TRUE);
  DBMS_SCHEDULER.SET_ATTRIBUTE(name=>'"SYS"."MONDAY_WINDOW"',
 attribute=>'REPEAT_INTERVAL',
 value=>'freq=daily;byday=MON;byhour=10;byminute=0;bysecond=0');
  DBMS_SCHEDULER.ENABLE(name=>'"SYS"."MONDAY_WINDOW"');
END;
```

```
/
```

Oracle Scheduler 작업의 시간대 설정

Oracle Scheduler의 시간대를 수정하려면 Oracle 프로시저 `dbms_scheduler.set_scheduler_attribute`를 사용할 수 있습니다. `dbms_scheduler` 패키지에 대한 자세한 내용은 Oracle 설명서의 DBMS_SCHEDULER 및 SET_SCHEDULER_ATTRIBUTE를 참조하세요.

현재 시간대 설정을 수정하려면

1. SQL Developer와 같은 클라이언트를 사용하여 데이터베이스에 연결합니다. 자세한 내용은 Oracle SQL Developer를 사용하여 DB 인스턴스에 연결 섹션을 참조하세요.
2. *`time_zone_name`*의 시간대를 대체하여 기본 시간대를 다음과 같이 설정합니다.

```
BEGIN
  DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE(
    attribute => 'default_timezone',
    value => 'time_zone_name'
  );
END;
/
```

다음 예제에서는 시간대를 아시아/상하이로 변경합니다.

다음과 같이 현재 시간대를 쿼리하여 시작합니다.

```
SELECT VALUE FROM DBA_SCHEDULER_GLOBAL_ATTRIBUTE WHERE
 ATTRIBUTE_NAME='DEFAULT_TIMEZONE';
```

출력은 현재 시간대가 ETC/UTC임을 보여줍니다.

```
VALUE
-------
Etc/UTC
```

그런 다음 시간대를 아시아/상하이로 설정합니다.

```
BEGIN
```

```
  DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE(
    attribute => 'default_timezone',
    value => 'Asia/Shanghai'
  );
END;
/
```

시스템 시간대 변경에 대한 자세한 내용은 Oracle 시간대 단원을 참조하세요.

SYS가 소유한 Oracle Scheduler 작업 해제

SYS 사용자가 소유한 Oracle Scheduler 작업을 비활성화하려면 `rdsadmin.rdsadmin_dbms_scheduler.disable` 프로시저를 사용해야 합니다.

이 프로시저는 Oracle Scheduler 작업에 `name` 공통 파라미터를 사용합니다. 자세한 내용은 Oracle Scheduler 프로시저용 공통 파라미터 섹션을 참조하세요.

다음 예제는 `SYS.CLEANUP_ONLINE_IND_BUILD` Oracle Scheduler 작업을 비활성화합니다.

```
BEGIN
   rdsadmin.rdsadmin_dbms_scheduler.disable('SYS.CLEANUP_ONLINE_IND_BUILD');
END;
/
```

SYS가 소유한 Oracle Scheduler 작업 켜기

SYS 소유 Oracle Scheduler 작업을 활성화하려면 `rdsadmin.rdsadmin_dbms_scheduler.enable` 프로시저를 사용해야 합니다.

이 프로시저는 Oracle Scheduler 작업에 `name` 공통 파라미터를 사용합니다. 자세한 내용은 Oracle Scheduler 프로시저용 공통 파라미터 섹션을 참조하세요.

다음 예제는 `SYS.CLEANUP_ONLINE_IND_BUILD` Oracle Scheduler 작업을 활성화합니다.

```
BEGIN
   rdsadmin.rdsadmin_dbms_scheduler.enable('SYS.CLEANUP_ONLINE_IND_BUILD');
END;
/
```

CALENDAR 형식 작업에 대한 Oracle Scheduler 반복 간격 수정

CALENDAR 형식의 SYS 소유 Oracle Scheduler 작업을 수정하는 반복 간격을 수정하려면 rdsadmin.rdsadmin_dbms_scheduler.disable 프로시저를 사용합니다.

이 프로시저에서는 Oracle Scheduler 작업을 위해 다음과 같은 공통 파라미터를 사용합니다.

- name
- attribute
- value

자세한 내용은 Oracle Scheduler 프로시저용 공통 파라미터 섹션을 참조하세요.

다음 예제는 SYS.CLEANUP_ONLINE_IND_BUILD Oracle Scheduler 작업의 반복 간격을 수정합니다.

```
BEGIN
     rdsadmin.rdsadmin_dbms_scheduler.set_attribute(
          name      => 'SYS.CLEANUP_ONLINE_IND_BUILD',
          attribute => 'repeat_interval',
          value     => 'freq=daily;byday=FRI,SAT;byhour=20;byminute=0;bysecond=0');
END;
/
```

NAMED 형식 작업에 대한 Oracle Scheduler 반복 간격 수정

일부 Oracle Scheduler 작업은 간격 대신 일정 이름을 사용합니다. 이 유형의 작업에서는 마스터 사용자 스키마에서 이름이 지정된 새 일정을 생성해야 합니다. 이렇게 하려면 표준 Oracle sys.dbms_scheduler.create_schedule 프로시저를 사용합니다. 또한 rdsadmin.rdsadmin_dbms_scheduler.set_attribute procedure를 사용하여 새 명명된 일정을 할당합니다.

이 프로시저에서는 Oracle Scheduler 작업을 위해 다음과 같은 공통 파라미터를 사용합니다.

- name
- attribute
- value

자세한 내용은 Oracle Scheduler 프로시저용 공통 파라미터 섹션을 참조하세요.

다음 예제는 SYS.BSLN_MAINTAIN_STATS_JOB Oracle Scheduler 작업의 반복 간격을 수정합니다.

```
BEGIN
     DBMS_SCHEDULER.CREATE_SCHEDULE (
          schedule_name   => 'rds_master_user.new_schedule',
          start_date      => SYSTIMESTAMP,
          repeat_interval =>
 'freq=daily;byday=MON,TUE,WED,THU,FRI;byhour=0;byminute=0;bysecond=0',
          end_date        => NULL,
          comments        => 'Repeats daily forever');
END;
/

BEGIN
     rdsadmin.rdsadmin_dbms_scheduler.set_attribute (
          name      => 'SYS.BSLN_MAINTAIN_STATS_JOB',
          attribute => 'schedule_name',
          value     => 'rds_master_user.new_schedule');
END;
/
```

Oracle Scheduler 작업 생성을 위한 자동 커밋 해제

DBMS_SCHEDULER.CREATE_JOB에서 Oracle Scheduler 작업을 생성하면, 작업이 즉시 생성되고 변경 사항이 커밋됩니다. 다음을 수행하려면 Oracle Scheduler 작업 생성을 사용자 트랜잭션에 통합해야 할 수 있습니다.

- 사용자 트랜잭션이 롤백될 때 Oracle Schedule 작업을 롤백합니다.
- 기본 사용자 트랜잭션이 커밋될 때 Oracle Scheduler 작업을 생성합니다.

프로시저 rdsadmin.rdsadmin_dbms_scheduler.set_no_commit_flag를 사용하여 이 동작을 설정할 수 있습니다. 이 프로시저에는 파라미터가 없습니다. 이 절차는 다음 RDS for Oracle 릴리스에서 사용할 수 있습니다.

- 21.0.0.0.ru-2022-07.rur-2022-07.r1 이상
- 19.0.0.0.ru-2022-07.rur-2022-07.r1 이상

다음 예에서는 Oracle Scheduler의 자동 커밋을 해제하고 Oracle Scheduler 작업을 생성한 다음 트랜잭션을 롤백합니다. 자동 커밋이 해제된 상태이기 때문에 데이터베이스는 Oracle Scheduler 작업 생성을 롤백합니다.

```
BEGIN
  rdsadmin.rdsadmin_dbms_scheduler.set_no_commit_flag;
  DBMS_SCHEDULER.CREATE_JOB(job_name   => 'EMPTY_JOB',
                            job_type   => 'PLSQL_BLOCK',
                            job_action => 'begin null; end;',
                            auto_drop  => false);
  ROLLBACK;
END;
/

PL/SQL procedure successfully completed.

SELECT * FROM DBA_SCHEDULER_JOBS WHERE JOB_NAME='EMPTY_JOB';

no rows selected
```

RDS for Oracle DB 인스턴스 문제 진단

Oracle 데이터베이스에는 데이터베이스 문제를 조사하는 데 사용할 수 있는 결함 진단 가능성 인프라가 포함되어 있습니다. Oracle 용어에서 문제는 코드 버그 또는 데이터 손상과 같은 중대한 오류입니다. 인시던트는 문제의 발생입니다. 동일한 오류가 세 번 발생하면 인프라에 이 문제의 세 가지 인시던트가 표시됩니다. 자세한 내용은 Oracle 데이터베이스 설명서의 문제 진단 및 해결을 참조하세요.

ADRCI(자동 진단 리포지토리 명령 인터프리터) 유틸리티는 진단 데이터를 관리하는 데 사용하는 Oracle 명령줄 도구입니다. 예를 들어, 이 도구를 사용하여 문제를 조사하고 진단 데이터를 패키징할 수 있습니다. 인시던트 패키지에는 한 인시던트 또는 특정 문제를 참조하는 모든 인시던트에 대한 진단 데이터가 포함됩니다. .zip 파일로 구현된 인시던트 패키지를 Oracle Support에 업로드할 수 있습니다.

관리형 서비스 환경을 제공하기 위해 Amazon RDS는 ADRCI에 대한 shell 액세스를 제공하지 않습니다. RDS for Oracle 인스턴스에 대한 진단 작업을 수행하려면 Amazon RDS 패키지 `rdsadmin.rdsadmin_adrci_util`을 사용합니다.

`rdsadmin_adrci_util`의 함수를 사용하여 문제와 인시던트를 나열하고 패키지징할 수 있으며 추적 파일도 표시할 수 있습니다. 모든 함수는 작업 ID를 반환합니다. 이 ID는 `dbtask-`*`task_id`*`.log`와 같이 ADRCI 출력이 포함된 로그 파일 이름의 일부를 구성합니다. 로그 파일은 BDUMP 디렉터리에 상주합니다. 데이터베이스 로그 파일 다운로드에 설명된 절차에 따라 로그 파일을 다운로드할 수 있습니다.

진단 절차에 대한 공통 파라미터

진단 작업을 수행하려면 Amazon RDS 패키지 `rdsadmin.rdsadmin_adrci_util`의 함수를 사용합니다. 이 패키지에는 다음과 같은 공통 파라미터가 있습니다.

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
`incident_id`	숫자	유효한 인시던트 ID 또는 null	Null	아니요	값이 null이면 이 함수는 모든 인시던트를 표시합니다. 값이 null이 아니고 유효한 인시던트 ID를 나타내면 이 함수는 지정된 인시던트를 표시합니다.
`problem_id`	숫자	유효한 문제 ID 또는 null	Null	아니요	값이 null이면 이 함수에 모든 문제를 표시합니다. 값이 null이 아니고 유효한 문제 ID를 나타내면 이 함수는 지정된 문제를 표시합니다.
`last`	숫자	0보다 큰 유효한 정수 또는 null	Null	아니요	값이 null이면 이 함수는 50개 이하의 항목을 표시합니다. 값이 null이 아니면 이 함수는 지정된 숫자를 표시합니다.

인시던트 나열

Oracle에 대한 진단 인시던트를 나열하려면 Amazon RDS 함수 `rdsadmin.rdsadmin_adrci_util.list_adrci_incidents`를 사용합니다. 기본 또는 세부 모드로 인시던트를 나열할 수 있습니다. 기본적으로 이 함수는 가장 최근 인시던트 50개를 나열합니다.

이 함수는 다음과 같은 공통 파라미터를 사용합니다.

- incident_id
- problem_id
- last

incident_id 및 problem_id를 지정하면 incident_id가 problem_id를 재정의합니다. 자세한 내용은 진단 절차에 대한 공통 파라미터 단원을 참조하십시오.

이 함수는 다음과 같은 추가 파라미터를 사용합니다.

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
detail	부울	TRUE 또는 FALSE	FALSE	아니요	TRUE이면 이 함수는 세부 모드에서 인시던트를 나열합니다. FALSE이면 이 함수는 기본 모드에서 인시던트를 나열합니다.

모든 인시던트를 나열하려면 인수 없이 rdsadmin.rdsadmin_adrci_util.list_adrci_incidents 함수를 쿼리합니다. 쿼리는 작업 ID를 반환합니다.

```
SQL> SELECT rdsadmin.rdsadmin_adrci_util.list_adrci_incidents AS task_id FROM DUAL;

TASK_ID
------------------
1590786706158-3126
```

또는 인수 없이 rdsadmin.rdsadmin_adrci_util.list_adrci_incidents 함수를 호출하고 출력을 SQL 클라이언트 변수에 저장합니다. 다른 문에서 변수를 사용할 수 있습니다.

```
SQL> VAR task_id VARCHAR2(80);
SQL> EXEC :task_id := rdsadmin.rdsadmin_adrci_util.list_adrci_incidents;

PL/SQL procedure successfully completed.
```

로그 파일을 읽으려면 Amazon RDS 프로시저 rdsadmin.rds_file_util.read_text_file을 호출합니다. 작업 ID를 파일 이름의 일부로 제공합니다. 다음 출력은 53523, 53522 및 53521이라는 세 가지 인시던트를 보여줍니다.

```
SQL> SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP',
 'dbtask-'||:task_id||'.log'));

TEXT
--------------------------------------------------------------------------------
2020-05-29 21:11:46.193 UTC [INFO ] Listing ADRCI incidents.
2020-05-29 21:11:46.256 UTC [INFO ]
ADR Home = /rdsdbdata/log/diag/rdbms/orcl_a/ORCL:
*********************************************************************
INCIDENT_ID PROBLEM_KEY                                                 CREATE_TIME
----------- ------------------------------------------------------------
 ----------------------------------------
53523       ORA 700 [EVENT_CREATED_INCIDENT] [942] [SIMULATED_ERROR_003 2020-05-29
 20:15:20.928000 +00:00
53522       ORA 700 [EVENT_CREATED_INCIDENT] [942] [SIMULATED_ERROR_002 2020-05-29
 20:15:15.247000 +00:00
53521       ORA 700 [EVENT_CREATED_INCIDENT] [942] [SIMULATED_ERROR_001 2020-05-29
 20:15:06.047000 +00:00
3 rows fetched


2020-05-29 21:11:46.256 UTC [INFO ] The ADRCI incidents were successfully listed.
2020-05-29 21:11:46.256 UTC [INFO ] The task finished successfully.

14 rows selected.
```

특정 인시던트를 나열하려면 incident_id 파라미터를 사용하여 ID를 지정합니다. 다음 예제에서는 인시던트 53523에 대한 로그 파일만 쿼리합니다.

```
SQL> EXEC :task_id :=
 rdsadmin.rdsadmin_adrci_util.list_adrci_incidents(incident_id=>53523);

PL/SQL procedure successfully completed.

SQL> SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP',
 'dbtask-'||:task_id||'.log'));

TEXT
```

```
--------------------------------------------------------------------------------------------------------------
2020-05-29 21:15:25.358 UTC [INFO ] Listing ADRCI incidents.
2020-05-29 21:15:25.426 UTC [INFO ]
ADR Home = /rdsdbdata/log/diag/rdbms/orcl_a/ORCL:
*************************************************************************
INCIDENT_ID          PROBLEM_KEY
 CREATE_TIME
-------------------- ----------------------------------------------------------
 ---------------------------------
53523                ORA 700 [EVENT_CREATED_INCIDENT] [942] [SIMULATED_ERROR_003
 2020-05-29 20:15:20.928000 +00:00
1 rows fetched


2020-05-29 21:15:25.427 UTC [INFO ] The ADRCI incidents were successfully listed.
2020-05-29 21:15:25.427 UTC [INFO ] The task finished successfully.

12 rows selected.
```

문제 나열

Oracle에 대한 진단 문제를 나열하려면 Amazon RDS 함수 `rdsadmin.rdsadmin_adrci_util.list_adrci_problems`를 사용합니다.

기본적으로 이 함수는 가장 최근 문제 50개를 나열합니다.

이 함수는 공통 파라미터 `problem_id` 및 `last`를 사용합니다. 자세한 내용은 진단 절차에 대한 공통 파라미터 단원을 참조하십시오.

모든 문제에 대한 작업 ID를 가져오려면 인수 없이 `rdsadmin.rdsadmin_adrci_util.list_adrci_problems` 함수를 호출하고 출력을 SQL 클라이언트 변수에 저장합니다.

```
SQL> EXEC :task_id := rdsadmin.rdsadmin_adrci_util.list_adrci_problems;

PL/SQL procedure successfully completed.
```

로그 파일을 읽으려면 `rdsadmin.rds_file_util.read_text_file` 함수를 호출하여 작업 ID를 파일 이름의 일부로 제공합니다. 다음 출력에서 로그 파일에는 1, 2 및 3이라는 세 가지 문제가 표시됩니다.

```
SQL> SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP',
 'dbtask-'||:task_id||'.log'));

TEXT
--------------------------------------------------------------------------------
2020-05-29 21:18:50.764 UTC [INFO ] Listing ADRCI problems.
2020-05-29 21:18:50.829 UTC [INFO ]
ADR Home = /rdsdbdata/log/diag/rdbms/orcl_a/ORCL:
*********************************************************************
PROBLEM_ID   PROBLEM_KEY                                                 LAST_INCIDENT
       LASTINC_TIME
---------- ------------------------------------------------------------ -------------
 -------------------------------
2          ORA 700 [EVENT_CREATED_INCIDENT] [942] [SIMULATED_ERROR_003 53523
 2020-05-29 20:15:20.928000 +00:00
3          ORA 700 [EVENT_CREATED_INCIDENT] [942] [SIMULATED_ERROR_002 53522
 2020-05-29 20:15:15.247000 +00:00
1          ORA 700 [EVENT_CREATED_INCIDENT] [942] [SIMULATED_ERROR_001 53521
 2020-05-29 20:15:06.047000 +00:00
3 rows fetched


2020-05-29 21:18:50.829 UTC [INFO ] The ADRCI problems were successfully listed.
2020-05-29 21:18:50.829 UTC [INFO ] The task finished successfully.

14 rows selected.
```

다음 예제에서는 문제 3만 나열합니다.

```
SQL> EXEC :task_id := rdsadmin.rdsadmin_adrci_util.list_adrci_problems(problem_id=>3);

PL/SQL procedure successfully completed.
```

문제 3에 대한 로그 파일을 읽으려면 `rdsadmin.rds_file_util.read_text_file`을 호출합니다. 작업 ID를 파일 이름의 일부로 제공합니다.

```
SQL> SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP',
 'dbtask-'||:task_id||'.log'));

TEXT
--------------------------------------------------------------------------------
2020-05-29 21:19:42.533 UTC [INFO ] Listing ADRCI problems.
```

```
2020-05-29 21:19:42.599 UTC [INFO ]
ADR Home = /rdsdbdata/log/diag/rdbms/orcl_a/ORCL:
*********************************************************************
PROBLEM_ID PROBLEM_KEY                                                    LAST_INCIDENT
 LASTINC_TIME
---------- ------------------------------------------------------------ -------------
 -------------------------------
3          ORA 700 [EVENT_CREATED_INCIDENT] [942] [SIMULATED_ERROR_002 53522
 2020-05-29 20:15:15.247000 +00:00
1 rows fetched


2020-05-29 21:19:42.599 UTC [INFO ] The ADRCI problems were successfully listed.
2020-05-29 21:19:42.599 UTC [INFO ] The task finished successfully.

12 rows selected.
```

인시던트 패키지 생성

Amazon RDS 함수 `rdsadmin.rdsadmin_adrci_util.create_adrci_package`를 사용하여 인시던트 패키지를 생성할 수 있습니다. 출력은 Oracle Support에 제공할 수 있는.zip 파일입니다.

이 함수는 다음과 같은 공통 파라미터를 사용합니다.

- `problem_id`
- `incident_id`

앞의 파라미터 중 하나를 지정해야 합니다. 두 파라미터를 모두 지정하면 `incident_id`가 `problem_id`를 재정의합니다. 자세한 내용은 진단 절차에 대한 공통 파라미터 섹션을 참조하세요.

특정 인시던트에 대한 패키지를 생성하려면 `rdsadmin.rdsadmin_adrci_util.create_adrci_package` 파라미터를 사용하는 Amazon RDS 함수 `incident_id`를 호출합니다. 다음 예제에서는 인시던트 53523에 대한 패키지를 생성합니다.

```
SQL> EXEC :task_id :=
 rdsadmin.rdsadmin_adrci_util.create_adrci_package(incident_id=>53523);

PL/SQL procedure successfully completed.
```

로그 파일을 읽으려면 rdsadmin.rds_file_util.read_text_file을 호출합니다. 작업 ID를 파일 이름의 일부로 제공할 수 있습니다. 출력은 인시던트 패키지 ORA700EVE_20200529212043_COM_1.zip을 생성했음을 보여줍니다.

```
SQL> SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP',
 'dbtask-'||:task_id||'.log'));

TEXT
--------------------------------------------------------------------------------
2020-05-29 21:20:43.031 UTC [INFO ] The ADRCI package is being created.
2020-05-29 21:20:47.641 UTC [INFO ] Generated package 1 in file /rdsdbdata/log/trace/
ORA700EVE_20200529212043_COM_1.zip, mode complete
2020-05-29 21:20:47.642 UTC [INFO ] The ADRCI package was successfully created.
2020-05-29 21:20:47.642 UTC [INFO ] The task finished successfully.
```

특정 문제에 대한 진단 데이터를 패키징하려면 problem_id 파라미터를 사용하여 ID를 지정합니다. 다음 예제에서는 문제 3에 대한 데이터만 패키징합니다.

```
SQL> EXEC :task_id := rdsadmin.rdsadmin_adrci_util.create_adrci_package(problem_id=>3);

PL/SQL procedure successfully completed.
```

작업 출력을 읽으려면 rdsadmin.rds_file_util.read_text_file을 호출하여 작업 ID를 파일 이름의 일부로 제공합니다. 출력은 인시던트 패키지 ORA700EVE_20200529212111_COM_1.zip을 생성했음을 보여줍니다.

```
SQL> SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP',
 'dbtask-'||:task_id||'.log'));

TEXT
--------------------------------------------------------------------------------
2020-05-29 21:21:11.050 UTC [INFO ] The ADRCI package is being created.
2020-05-29 21:21:15.646 UTC [INFO ] Generated package 2 in file /rdsdbdata/log/trace/
ORA700EVE_20200529212111_COM_1.zip, mode complete
2020-05-29 21:21:15.646 UTC [INFO ] The ADRCI package was successfully created.
2020-05-29 21:21:15.646 UTC [INFO ] The task finished successfully.
```

또한 로그 파일을 다운로드할 수 있습니다. 자세한 내용은 데이터베이스 로그 파일 다운로드 단원을 참조하십시오.

추적 파일 표시

Amazon RDS 함수 `rdsadmin.rdsadmin_adrci_util.show_adrci_tracefile`을 사용하여 추적 디렉터리 아래에 추적 파일을 나열하고 현재 ADR 홈 아래에 모든 인시던트 디렉터리를 나열할 수 있습니다. 추적 파일 및 인시던트 추적 파일의 내용을 표시할 수도 있습니다.

이 함수는 다음 파라미터를 사용합니다.

파라미터 이름	데이터 형식	유효한 값	기본값	필수	설명
`filename`	varchar2	유효한 추적 파일 이름	Null	아니요	값이 null이면 이 함수는 모든 추적 파일을 표시합니다. 값이 null이 아니면 이 함수는 지정된 파일을 표시합니다.

추적 파일을 표시하려면 Amazon RDS 함수 `rdsadmin.rdsadmin_adrci_util.show_adrci_tracefile`을 호출합니다.

```
SQL> EXEC :task_id := rdsadmin.rdsadmin_adrci_util.show_adrci_tracefile;

PL/SQL procedure successfully completed.
```

추적 파일 이름을 나열하려면 Amazon RDS 프로시저 `rdsadmin.rds_file_util.read_text_file`을 호출하여 작업 ID를 파일 이름의 일부로 제공합니다.

```
SQL> SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP',
 'dbtask-'||:task_id||'.log')) WHERE TEXT LIKE '%/alert_%';

TEXT
--------------------------------------------------------------------
     diag/rdbms/orcl_a/ORCL/trace/alert_ORCL.log.2020-05-28
     diag/rdbms/orcl_a/ORCL/trace/alert_ORCL.log.2020-05-27
     diag/rdbms/orcl_a/ORCL/trace/alert_ORCL.log.2020-05-26
     diag/rdbms/orcl_a/ORCL/trace/alert_ORCL.log.2020-05-25
```

```
     diag/rdbms/orcl_a/ORCL/trace/alert_ORCL.log.2020-05-24
     diag/rdbms/orcl_a/ORCL/trace/alert_ORCL.log.2020-05-23
     diag/rdbms/orcl_a/ORCL/trace/alert_ORCL.log.2020-05-22
     diag/rdbms/orcl_a/ORCL/trace/alert_ORCL.log.2020-05-21
     diag/rdbms/orcl_a/ORCL/trace/alert_ORCL.log

9 rows selected.
```

다음 예제에서는 `alert_ORCL.log`에 대한 출력을 생성합니다.

```
SQL> EXEC :task_id := rdsadmin.rdsadmin_adrci_util.show_adrci_tracefile('diag/rdbms/
orcl_a/ORCL/trace/alert_ORCL.log');

PL/SQL procedure successfully completed.
```

로그 파일을 읽으려면 `rdsadmin.rds_file_util.read_text_file`을 호출합니다. 작업 ID를 파일 이름의 일부로 제공합니다. 출력에는 alert_ORCL.log의 처음 10줄이 표시됩니다.

```
SQL> SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP',
 'dbtask-'||:task_id||'.log')) WHERE ROWNUM <= 10;

TEXT
--------------------------------------------------------------------------------
2020-05-29 21:24:02.083 UTC [INFO ] The trace files are being displayed.
2020-05-29 21:24:02.128 UTC [INFO ] Thu May 28 23:59:10 2020
Thread 1 advanced to log sequence 2048 (LGWR switch)
  Current log# 3 seq# 2048 mem# 0: /rdsdbdata/db/ORCL_A/onlinelog/o1_mf_3_hbl2p8xs_.log
Thu May 28 23:59:10 2020
Archived Log entry 2037 added for thread 1 sequence 2047 ID 0x5d62ce43 dest 1:
Fri May 29 00:04:10 2020
Thread 1 advanced to log sequence 2049 (LGWR switch)
  Current log# 4 seq# 2049 mem# 0: /rdsdbdata/db/ORCL_A/onlinelog/o1_mf_4_hbl2qgmh_.log
Fri May 29 00:04:10 2020

10 rows selected.
```

또한 로그 파일을 다운로드할 수 있습니다. 자세한 내용은 데이터베이스 로그 파일 다운로드 단원을 참조하십시오.

Oracle DB 인스턴스에 대한 공통 기타 작업 수행

그 다음에는 Oracle을 실행하는 Amazon RDS DB 인스턴스에서 기타 DBA 작업을 수행하는 방법을 알아봅니다. 관리형 서비스 환경을 제공하기 위해 Amazon RDS는 DB 인스턴스에 대해 shell 액세스를 제공하지 않으며, 고급 권한이 필요한 특정 시스템 절차와 테이블에 대한 액세스를 제한합니다.

주제

- 메인 데이터 스토리지 공간에서 디렉터리 생성 및 삭제
- DB 인스턴스 디렉터리의 파일 목록 표시
- DB 인스턴스 디렉터리의 파일 목록 읽기
- Opatch 파일 액세스
- Advisor 작업 관리
- 테이블스페이스 전송

메인 데이터 스토리지 공간에서 디렉터리 생성 및 삭제

디렉터리를 생성하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.create_directory`를 사용합니다. 디렉터리를 최대 10,000개까지 만들어 메인 데이터 스토리지 공간에 저장할 수 있습니다. 디렉터리를 삭제하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.drop_directory`를 사용합니다.

`create_directory` 및 `drop_directory` 프로시저에는 다음과 같은 필수 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_directory_name	VARCHAR2	—	예	디렉터리의 이름입니다.

다음 예제에서는 `PRODUCT_DESCRIPTIONS`라는 새 디렉터리를 생성합니다.

```
EXEC rdsadmin.rdsadmin_util.create_directory(p_directory_name =>
 'product_descriptions');
```

데이터 딕셔너리는 디렉터리 이름을 대문자로 저장합니다. `DBA_DIRECTORIES`에 쿼리를 실행하면 디렉터리 목록을 표시할 수 있습니다. 실제 호스트 경로 이름은 시스템이 자동으로 선택합니다. 다음은 `PRODUCT_DESCRIPTIONS`라는 디렉터리의 디렉터리 경로를 얻는 예제입니다.

```
SELECT DIRECTORY_PATH
  FROM DBA_DIRECTORIES
 WHERE DIRECTORY_NAME='PRODUCT_DESCRIPTIONS';

DIRECTORY_PATH
----------------------------------------
/rdsdbdata/userdirs/01
```

DB 인스턴스의 마스터 사용자 이름은 새로운 디렉터리에서도 읽기/쓰기 권한이 있으며, 다른 사용자에게 액세스 권한을 부여할 수도 있습니다. EXECUTE 권한은 DB 인스턴스의 디렉터리에 사용할 수 없습니다. 디렉터리는 메인 데이터 스토리지 공간에 생성되어 일정 공간과 I/O 대역폭을 사용합니다.

다음 예제에서는 PRODUCT_DESCRIPTIONS라는 디렉터리를 삭제합니다.

```
EXEC rdsadmin.rdsadmin_util.drop_directory(p_directory_name => 'product_descriptions');
```

> **Note**
>
> Oracle SQL 명령 DROP DIRECTORY를 사용하여 디렉터리를 삭제할 수도 있습니다.

디렉터리를 삭제해도 그 내용은 제거되지 않습니다. `rdsadmin.rdsadmin_util.create_directory` 프로시저가 경로 이름을 재사용하므로 삭제된 디렉터리의 파일이 새로 생성된 디렉터리에 나타날 수도 있습니다. 디렉터리를 삭제하기 전에 `UTL_FILE.FREMOVE`를 사용하여 디렉터리에서 파일을 제거하는 것이 좋습니다. 자세한 내용은 Oracle 설명서의 FREMOVE 프로시저를 참조하세요.

DB 인스턴스 디렉터리의 파일 목록 표시

디렉터리의 파일을 나열하려면 Amazon RDS 프로시저 `rdsadmin.rds_file_util.listdir`을 사용합니다. Oracle 복제본의 경우 이 프로시저가 지원되지 않습니다. `listdir` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_directory	varchar2	—	예	목록을 표시할 디렉터리의 이름입니다.

다음 예에서는 사용자 rdsadmin에게 디렉터리 PRODUCT_DESCRIPTIONS에 대한 읽기/쓰기 권한을 부여한 다음 이 디렉터리에 있는 파일을 나열합니다.

```
GRANT READ,WRITE ON DIRECTORY PRODUCT_DESCRIPTIONS TO rdsadmin;
SELECT * FROM TABLE(rdsadmin.rds_file_util.listdir(p_directory =>
 'PRODUCT_DESCRIPTIONS'));
```

DB 인스턴스 디렉터리의 파일 목록 읽기

텍스트 파일을 읽으려면 Amazon RDS 프로시저 rdsadmin.rds_file_util.read_text_file을 사용합니다. read_text_file 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_directory	varchar2	—	예	파일이 포함된 디렉터리의 이름입니다.
p_filename	varchar2	—	예	읽을 파일의 이름입니다.

다음 예제에서는 rice.txt 디렉터리에 PRODUCT_DESCRIPTIONS 파일을 생성합니다.

```
declare
  fh sys.utl_file.file_type;
begin
  fh := utl_file.fopen(location=>'PRODUCT_DESCRIPTIONS', filename=>'rice.txt',
 open_mode=>'w');
  utl_file.put(file=>fh, buffer=>'AnyCompany brown rice, 15 lbs');
  utl_file.fclose(file=>fh);
end;
/
```

다음 예제에서는 rice.txt 디렉터리에서 PRODUCT_DESCRIPTIONS 파일을 읽습니다.

```
SELECT * FROM TABLE
    (rdsadmin.rds_file_util.read_text_file(
        p_directory => 'PRODUCT_DESCRIPTIONS',
        p_filename  => 'rice.txt'));
```

Opatch 파일 액세스

Opatch는 Oracle 소프트웨어에 패치를 적용하고 롤백할 수 있는 Oracle 유틸리티입니다. 데이터베이스에 적용된 패치를 확인하는 Oracle 메커니즘이 `opatch lsinventory` 명령입니다. BYOL(Bring Your Own Licence) 고객에 대한 서비스 요청을 개시하기 위해 Oracle Support는 해당 `lsinventory` 파일 및 Opatch 에서 생성된 `lsinventory_detail` 파일을 요청합니다.

관리형 서비스 환경을 제공하기 위해 Amazon RDS는 Opatch에 대한 shell 액세스를 제공하지 않습니다. 그 대신, BDUMP 디렉터리의 `lsinventory-`*`dbv`*`.txt`에는 현재 엔진 버전과 관련한 패치 정보가 포함되어 있습니다. 마이너 또는 메이저 업그레이드를 수행하면 Amazon RDS가 패치를 적용한 후 1 시간 이내에 `lsinventory-`*`dbv`*`.txt`를 업데이트합니다. 적용된 패치를 확인하려면 `lsinventory-`*`dbv`*`.txt`를 확인하세요. 이 작업은 `opatch lsinventory` 명령을 실행하는 것과 유사합니다.

> **Note**
>
> 이 단원의 예제에서는 BDUMP 디렉터리의 이름이 `BDUMP`라고 가정합니다. 읽기 전용 복제본에서는 BDUMP 디렉터리 이름이 다릅니다. 읽기 전용 복제본에서 `V$DATABASE.DB_UNIQUE_NAME`을 쿼리하여 BDUMP 이름을 가져오는 방법에 대한 자세한 내용은 파일 나열 단원을 참조하세요.

인벤토리 파일은 Amazon RDS 명명 규칙 `lsinventory-`*`dbv`*`.txt` 및 `lsinventory_detail-`*`dbv`*`.txt`를 사용합니다. 여기서 *`dbv`*는 DB 버전의 전체 이름입니다. `lsinventory-`*`dbv`*`.txt` 파일은 모든 DB 버전에서 사용할 수 있습니다. 해당 `lsinventory_detail-`*`dbv`*`.txt`는 19.0.0.0, ru-2020-01.rur-2020-01.r1 이상 버전에서 사용할 수 있습니다.

예를 들어 DB 버전이 19.0.0.0.ru-2021-07.rur-2021-07.r1인 경우 인벤토리 파일의 이름은 다음과 같습니다.

```
lsinventory-19.0.0.0.ru-2021-07.rur-2021-07.r1.txt
lsinventory_detail-19.0.0.0.ru-2021-07.rur-2021-07.r1.txt
```

DB 엔진의 현재 버전과 일치하는 파일을 다운로드해야 합니다.

콘솔

콘솔을 사용하여 인벤토리 파일을 다운로드하려면

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 보고자 하는 로그 파일을 보유한 DB 인스턴스의 이름을 선택합니다.
4. 로그 및 이벤트 탭을 선택합니다.
5. 아래로 스크롤하여 [Logs] 섹션을 찾습니다.
6. 로그 섹션에서 `lsinventory`를 검색합니다.
7. 액세스할 파일을 선택한 다음 다운로드를 선택합니다.

SQL

SQL 클라이언트에서 `lsinventory-`*dbv*`.txt`를 읽으려면 `SELECT` 문을 사용하면 됩니다. 이 기술에서는 `rdsadmin` 함수인 `rdsadmin.rds_file_util.read_text_file` 또는 `rdsadmin.tracefile_listing`을 사용합니다.

다음 예제 쿼리에서 *dbv*를 Oracle DB 버전으로 바꿉니다. 예를 들어, DB 버전은 19.0.0.0.ru-2020-04.rur-2020-04.r1일 수 있습니다.

```
SELECT text
FROM   TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP', 'lsinventory-dbv.txt'));
```

PL/SQL

SQL 클라이언트에서 `lsinventory-`*dbv*`.txt`를 읽으려면 PL/SQL 프로그램을 작성하면 됩니다. 이 프로그램은 `utl_file`을 사용하여 파일을 읽고 `dbms_output`을 사용하여 인쇄합니다. 이는 Oracle에서 제공하는 패키지입니다.

다음 예제 프로그램에서 *dbv*를 Oracle DB 버전으로 바꿉니다. 예를 들어, DB 버전은 19.0.0.0.ru-2020-04.rur-2020-04.r1일 수 있습니다.

```
SET SERVEROUTPUT ON
DECLARE
  v_file              SYS.UTL_FILE.FILE_TYPE;
  v_line              VARCHAR2(1000);
  v_oracle_home_type  VARCHAR2(1000);
  c_directory         VARCHAR2(30) := 'BDUMP';
```

```
  c_output_file       VARCHAR2(30) := 'lsinventory-dbv.txt';
BEGIN
  v_file := SYS.UTL_FILE.FOPEN(c_directory, c_output_file, 'r');
  LOOP
    BEGIN
      SYS.UTL_FILE.GET_LINE(v_file, v_line,1000);
      DBMS_OUTPUT.PUT_LINE(v_line);
    EXCEPTION
      WHEN no_data_found THEN
        EXIT;
    END;
  END LOOP;
END;
/
```

또는 `rdsadmin.tracefile_listing`을 쿼리하고 출력을 파일로 스풀링합니다. 다음 예제에서는 출력을 `/tmp/tracefile.txt`로 스풀링합니다.

```
SPOOL /tmp/tracefile.txt
SELECT *
FROM   rdsadmin.tracefile_listing
WHERE  FILENAME LIKE 'lsinventory%';
SPOOL OFF;
```

Advisor 작업 관리

Oracle Database에는 다양한 Advisor가 포함되어 있습니다. 각 Advisor는 자동화 및 수동 작업을 지원합니다. `rdsadmin.rdsadmin_util` 패키지의 프로시저를 사용하여 일부 Advisor 작업을 관리할 수 있습니다.

Advisor 작업 프로시저는 다음 엔진 버전에서 사용할 수 있습니다.

- Oracle Database 21c(21.0.0)
- 19.0.0.0.ru-2021-01.rur-2021-01.r1 이상의 Oracle Database 19c 버전

  자세한 내용은 Amazon RDS for Oracle 릴리스 정보의 버전 19.0.0.0.ru-2021-01.rur-2021-01.r1을 참조하세요.

주제

- Advisor 작업에 대한 파라미터 설정

- AUTO_STATS_ADVISOR_TASK 비활성화
- AUTO_STATS_ADVISOR_TASK 다시 활성화

Advisor 작업에 대한 파라미터 설정

일부 Advisor 작업에 대한 파라미터를 설정하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.advisor_task_set_parameter`를 사용합니다. `advisor_task_set_parameter` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_task_name	varchar2	—	예	파라미터를 변경할 Advisor 작업의 이름입니다. 유효한 값은 다음과 같습니다. • AUTO_STATS_ADVISOR_TASK • INDIVIDUAL_STATS_ADVISOR_TASK • SYS_AUTO_SPM_EVOLVE_TASK • SYS_AUTO_SQL_TUNING_TASK

p_parameter	varchar2	—	예	작업 파라미터의 이름입니다. Advisor 작업에 유요한 파라미터를 찾으려면 다음 쿼리를 실행합니다. *p_task_name* 을 p_task_name 의 유효한 값으로 바꿉니다.
				`COL PARAMETER_NAME FORMAT a30`
				`COL PARAMETER_VALUE FORMAT a30`
				`SELECT PARAMETER_NAME, PARAMETER _VALUE`
				`FROM DBA_ADVISOR_PARAMETERS`
				`WHERE TASK_NAME='` *p_task_name* `'`
				`AND PARAMETER_VALUE != 'UNUSED'`
				`ORDER BY PARAMETER_NAME;`

p_value	varchar2	—	예	작업 파라미터의 값입니다. 작업 파라미터의 유효한 값을 찾으려면 다음 쿼리를 실행합니다. *p_task_name* 을 p_task_name 의 유효한 값으로 바꿉니다.

파라미터 이름	데이터 형식	기본값	필수	설명
				COL PARAMETER_NAME FORMAT a30 COL PARAMETER_VALUE FORMAT a30 SELECT PARAMETER_NAME, PARAMETER _VALUE FROM DBA_ADVISOR_PARAMETERS WHERE TASK_NAME=' *p_task_name* ' AND PARAMETER_VALUE != 'UNUSED' ORDER BY PARAMETER_NAME;

다음 PL/SQL 프로그램은 ACCEPT_PLANS에 대해 FALSE를 SYS_AUTO_SPM_EVOLVE_TASK로 설정합니다. SQL Plan Management 자동화 작업은 계획을 확인하고 결과에 대한 보고서를 생성하지만 계획을 자동으로 개선하지는 않습니다. 보고서를 사용하여 새 SQL 계획 기준을 식별하고 수동으로 적용할 수 있습니다.

```
BEGIN
  rdsadmin.rdsadmin_util.advisor_task_set_parameter(
    p_task_name => 'SYS_AUTO_SPM_EVOLVE_TASK',
    p_parameter => 'ACCEPT_PLANS',
    p_value     => 'FALSE');
END;
```

다음 PL/SQL 프로그램은 EXECUTION_DAYS_TO_EXPIRE에 대해 10를 AUTO_STATS_ADVISOR_TASK로 설정합니다. 미리 정의된 작업 AUTO_STATS_ADVISOR_TASK는 유지 관리 기간 동안 하루에 한 번 자동으로 실행됩니다. 이 예에서는 작업 실행의 보존 기간을 10일로 설정합니다.

```
BEGIN
  rdsadmin.rdsadmin_util.advisor_task_set_parameter(
    p_task_name => 'AUTO_STATS_ADVISOR_TASK',
    p_parameter => 'EXECUTION_DAYS_TO_EXPIRE',
    p_value     => '10');
END;
```

AUTO_STATS_ADVISOR_TASK 비활성화

AUTO_STATS_ADVISOR_TASK를 비활성화하려면 Amazon RDS 프로시저 rdsadmin.rdsadmin_util.advisor_task_drop을 사용합니다. advisor_task_drop 프로시저는 다음 파라미터를 받습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_task_name	varchar2	—	예	비활성화할 Advisor 작업의 이름입니다. 유일한 유효 값은 AUTO_STATS_ADVISOR_TASK 입니다.

다음 명령은 AUTO_STATS_ADVISOR_TASK를 삭제합니다.

```
EXEC rdsadmin.rdsadmin_util.advisor_task_drop('AUTO_STATS_ADVISOR_TASK')
```

rdsadmin.rdsadmin_util.dbms_stats_init를 사용하여 AUTO_STATS_ADVISOR_TASK를 다시 활성화할 수 있습니다.

AUTO_STATS_ADVISOR_TASK 다시 활성화

AUTO_STATS_ADVISOR_TASK를 다시 활성화하려면 Amazon RDS 프로시저 rdsadmin.rdsadmin_util.dbms_stats_init를 사용합니다. dbms_stats_init 프로시저에는 파라미터가 없습니다.

다음 명령은 AUTO_STATS_ADVISOR_TASK를 다시 활성화합니다.

```
EXEC rdsadmin.rdsadmin_util.dbms_stats_init()
```

테이블스페이스 전송

Amazon RDS 패키지 rdsadmin.rdsadmin_transport_util을 사용하여 온프레미스 Oracle 데이터베이스의 테이블스페이스 세트를 RDS for Oracle DB 인스턴스로 복사합니다. 물리적 수준에서 전송 가능한 테이블스페이스 기능은 소스 데이터 파일과 메타데이터 파일을 대상 인스턴스에 점진적으로 복사합니다. Amazon EFS 또는 Amazon S3를 사용하여 파일을 전송할 수 있습니다. 자세한 내용은 Oracle 전송 가능한 테이블스페이스를 사용한 마이그레이션 단원을 참조하십시오.

주제

- 전송된 테이블스페이스를 DB 인스턴스로 가져오기
- 전송 가능한 테이블스페이스를 DB 인스턴스로 가져오기
- 테이블스페이스 가져오기 후 분리된 파일 나열
- 테이블스페이스 가져오기 후 분리된 파일 삭제

전송된 테이블스페이스를 DB 인스턴스로 가져오기

`rdsadmin.rdsadmin_transport_util.import_xtts_tablespaces` 프로시저를 사용하여 원본 DB 인스턴스에서 이전에 내보낸 테이블스페이스를 복원합니다. 전송 단계에서는 읽기 전용 테이블스페이스를 백업하고 Data Pump 메타데이터를 내보내고 이러한 파일을 대상 DB 인스턴스로 전송하고 테이블스페이스를 가져옵니다. 자세한 내용은 4단계: 테이블스페이스 전송 단원을 참조하십시오.

구문

```
FUNCTION import_xtts_tablespaces(
    p_tablespace_list IN CLOB,
    p_directory_name  IN VARCHAR2,
    p_platform_id     IN NUMBER DEFAULT 13,
    p_parallel        IN INTEGER DEFAULT 0) RETURN VARCHAR2;
```

파라미터

파라미터 이름	데이터 형식	기본값	필수	설명
p_tablespace_list	CLOB	—	예	가져올 테이블스페이스 목록입니다.
p_directory_name	VARCHAR2	—	예	테이블스페이스 백업이 포함된 디렉터리입니다.
p_platform_id	NUMBER	13	아니요	백업 단계에서 지정한 것과 일치하는 플랫폼 ID를 입력합니다. 플랫폼 목록을 찾으려면 V$TRANSPORTABLE_PLATFORM 을

파라미터 이름	데이터 형식	기본값	필수	설명
				쿼리하세요. 기본 플랫폼은 little endian인 Linux x86 64비트입니다.
p_parallel	INTEGER	0	아니요	병렬 처리의 정도입니다. 기본적으로 병렬 처리는 비활성화됩니다.

예시

다음 예에서는 *DATA_PUMP_DIR* 디렉터리에서 *TBS1*, *TBS2* 및 *TBS3* 테이블스페이스를 가져옵니다. 소스 플랫폼은 플랫폼 ID가 6인 AIX 기반 시스템(64비트)입니다. V$TRANSPORTABLE_PLATFORM 쿼리를 통해 플랫폼 ID를 찾을 수 있습니다.

```
VAR task_id CLOB

BEGIN
  :task_id:=rdsadmin.rdsadmin_transport_util.import_xtts_tablespaces(
        'TBS1,TBS2,TBS3',
        'DATA_PUMP_DIR',
        p_platform_id => 6);
END;
/

PRINT task_id
```

전송 가능한 테이블스페이스를 DB 인스턴스로 가져오기

`rdsadmin.rdsadmin_transport_util.import_xtts_metadata` 프로시저를 사용하여 전송 가능한 테이블스페이스 메타데이터를 RDS for Oracle DB 인스턴스로 가져옵니다. 작업 중에 메타데이터 가져오기 상태가 `rdsadmin.rds_xtts_operation_info` 테이블에 표시됩니다. 자세한 내용은 5단계: 대상 DB 인스턴스에 테이블스페이스 메타데이터 가져오기 단원을 참조하십시오.

구문

```
PROCEDURE import_xtts_metadata(
```

```
    p_datapump_metadata_file IN SYS.DBA_DATA_FILES.FILE_NAME%TYPE,
    p_directory_name         IN VARCHAR2,
    p_exclude_stats          IN BOOLEAN DEFAULT FALSE,
    p_remap_tablespace_list  IN CLOB DEFAULT NULL,
    p_remap_user_list        IN CLOB DEFAULT NULL);
```

파라미터

파라미터 이름	데이터 형식	기본값	필수	설명
p_datapump_metadata_file	SYS.DBA_DATA_FILES.FILE_NAME%TYPE	—	예	전송 가능한 테이블스페이스에 대한 메타데이터가 포함된 Oracle Data Pump 파일의 이름입니다.
p_directory_name	VARCHAR2	—	예	Data Pump 파일이 포함된 디렉터리입니다.
p_exclude_stats	BOOLEAN	FALSE	아니요	통계 제외 여부를 나타내는 플래그입니다.
p_remap_tablespace_list	CLOB	NULL	아니요	메타데이터를 가져오는 동안 재매핑할 테이블스페이스 목록입니다. *from_tbs*:*to_tbs* 형식을 사용합니다. 예를 들어, users:user_data 를 지정합니다.
p_remap_user_list	CLOB	NULL	아니요	메타데이터를 가져오는 동안 재매핑할 사용자 스키마 목록입니다. *from_schema_name* :*to_schema_name* 형식을 사

파라미터 이름	데이터 형식	기본값	필수	설명
				용합니다. 예를 들어, hr:human_resources 를 지정합니다.

예시

이 예시에서는 *DATA_PUMP_DIR*에 위치한 *xttdump.dmp* 파일에서 테이블스페이스 메타데이터를 가져옵니다.

```
BEGIN
  rdsadmin.rdsadmin_transport_util.import_xtts_metadata('xttdump.dmp','DATA_PUMP_DIR');
END;
/
```

테이블스페이스 가져오기 후 분리된 파일 나열

rdsadmin.rdsadmin_transport_util.list_xtts_orphan_files 프로시저를 사용하여 테이블스페이스 가져오기 후 분리된 데이터 파일을 나열합니다. 데이터 파일을 식별한 후 rdsadmin.rdsadmin_transport_util.cleanup_incomplete_xtts_import를 호출하여 삭제할 수 있습니다.

구문

```
FUNCTION list_xtts_orphan_files RETURN xtts_orphan_files_list_t PIPELINED;
```

예시

다음 예에서는 rdsadmin.rdsadmin_transport_util.list_xtts_orphan_files 프로시저를 호출합니다. 출력에는 분리된 두 개의 데이터 파일이 표시됩니다.

```
SQL> SELECT * FROM TABLE(rdsadmin.rdsadmin_transport_util.list_xtts_orphan_files);

FILENAME       FILESIZE
-------------- ---------
datafile_7.dbf 104865792
datafile_8.dbf 104865792
```

테이블스페이스 가져오기 후 분리된 파일 삭제

`rdsadmin.rdsadmin_transport_util.list_xtts_orphan_files` 프로시저를 사용하여 테이블스페이스 가져오기 후 분리된 데이터 파일을 삭제합니다. 이 명령을 실행하면 `rds-xtts-delete_xtts_orphaned_files-YYYY-MM-DD.HH24-MI-SS.FF.log` 이름 형식을 사용하는 로그 파일이 BDUMP 디렉터리에 생성됩니다. `rdsadmin.rdsadmin_transport_util.cleanup_incomplete_xtts_import` 프로시저를 사용하여 분리된 파일을 찾습니다. `rdsadmin.rds_file_util.read_text_file` 프로시저를 호출하여 로그 파일을 읽을 수 있습니다. 자세한 내용은 6단계: 남은 파일 정리 단원을 참조하십시오.

구문

```
PROCEDURE cleanup_incomplete_xtts_import(
    p_directory_name IN VARCHAR2);
```

파라미터

파라미터 이름	데이터 형식	기본값	필수	설명
p_directory_name	VARCHAR2	—	예	분리된 데이터 파일이 포함된 디렉터리입니다.

예시

다음 예시에서는 *DATA_PUMP_DIR*에서 분리된 데이터 파일을 삭제합니다.

```
BEGIN
  rdsadmin.rdsadmin_transport_util.cleanup_incomplete_xtts_import('DATA_PUMP_DIR');
END;
/
```

다음 예시에서는 이전 명령으로 생성된 로그 파일을 읽습니다.

```
SELECT *
FROM TABLE(rdsadmin.rds_file_util.read_text_file(
       p_directory => 'BDUMP',
       p_filename  => 'rds-xtts-
delete_xtts_orphaned_files-2023-06-01.09-33-11.868894000.log'));
```

```
TEXT
--------------------------------------------------------------------------------
orphan transported datafile datafile_7.dbf deleted.
orphan transported datafile datafile_8.dbf deleted.
```

RDS for Oracle 고급 기능 구성

RDS for Oracle은 HugePages, 인스턴스 스토어 및 확장 데이터 유형을 비롯한 다양한 고급 기능을 지원합니다.

주제

- RDS for Oracle 인스턴스 스토어에 임시 데이터 저장
- RDS for Oracle 인스턴스에 HugePages 활성화
- RDS for Oracle에서 확장 데이터 유형 활성화

RDS for Oracle 인스턴스 스토어에 임시 데이터 저장

Oracle DB 클래스에 지원되는 RDS에서 임시 테이블스페이스 및 데이터베이스 스마트 플래시 캐시(플래시 캐시)에 인스턴스 스토어를 사용하세요.

주제

- RDS for Oracle 인스턴스 스토어 개요
- RDS for Oracle 인스턴스 스토어 활성화
- RDS for Oracle 인스턴스 스토어 구성
- Oracle 읽기 전용 복제본에서 인스턴스 스토어로 작업하기
- 인스턴스 스토어와 Amazon EBS에 임시 테이블스페이스 그룹 구성
- RDS for Oracle 인스턴스 스토어 제거

RDS for Oracle 인스턴스 스토어 개요

인스턴스 스토어는 RDS for Oracle DB 인스턴스에 블록 수준의 임시 스토리지를 제공합니다. 자주 변경되는 정보를 임시로 저장하는 데 인스턴스 스토어를 사용할 수 있습니다.

인스턴스 스토어는 호스트 컴퓨터에 물리적으로 연결된 NVMe 디바이스를 기반으로 합니다. 이 스토리지는 짧은 지연 시간, 임의 I/O 성능, 순차 읽기 처리량에 최적화되어 있습니다.

인스턴스 스토어의 크기는 DB 인스턴스 유형에 따라 다릅니다. 인스턴스 스토어 유형에 대한 자세한 내용은 Linux 인스턴스용 Amazon Elastic Compute Cloud 사용 설명서의 Amazon EC2 인스턴스 스토어 섹션을 참조하세요.

주제

- RDS for Oracle 인스턴스 스토어의 데이터 유형
- RDS for Oracle 인스턴스 스토어의 이점
- RDS for Oracle 인스턴스 스토어가 지원되는 인스턴스 클래스
- RDS for Oracle 인스턴스 스토어가 지원되는 엔진 버전
- RDS for Oracle 인스턴스 스토어가 지원되는 AWS 리전
- RDS for Oracle 인스턴스 스토어의 비용

RDS for Oracle 인스턴스 스토어의 데이터 유형

다음 유형의 RDS for Oracle 임시 데이터를 인스턴스 스토어에 배치할 수 있습니다.

임시 테이블스페이스

Oracle Database는 임시 테이블스페이스를 사용하여 메모리에 들어가지 않는 중간 쿼리 결과를 저장합니다. 쿼리가 클수록 일시적으로 캐시해야 하지만 유지할 필요는 없는 많은 양의 중간 데이터가 생성될 수 있습니다. 특히 임시 테이블스페이스는 정렬, 해시 집계 및 조인에 유용합니다. RDS for Oracle DB 인스턴스가 Enterprise Edition 또는 Standard Edition 2를 사용하는 경우 인스턴스 스토어에 임시 테이블스페이스를 배치할 수 있습니다.

플래시 캐시

플래시 캐시는 통상적인 경로에서의 단일 블록 랜덤 읽기 성능을 향상시킵니다. 가장 좋은 방법은 활성 데이터 세트의 대부분을 수용할 수 있도록 캐시의 크기를 조정하는 것입니다. RDS for Oracle DB 인스턴스가 Enterprise Edition을 사용하는 경우 인스턴스 스토어에 플래시 캐시를 배치할 수 있습니다.

기본적으로 인스턴스 스토어는 임시 테이블스페이스용으로 구성되며 플래시 캐시에 대해서는 구성되지 않습니다. Oracle 데이터 파일과 데이터베이스 로그 파일을 인스턴스 스토어에 배치할 수 없습니다.

RDS for Oracle 인스턴스 스토어의 이점

손실되어도 되는 임시 파일 및 캐시를 저장하기 위해 인스턴스 스토어를 사용하는 것을 고려해 볼 수 있습니다. DB 성능을 개선하고 싶거나 워크로드 증가로 인해 Amazon EBS 스토리지의 성능 문제가 발생하는 경우 인스턴스 스토어를 지원하는 인스턴스 클래스로 확장하는 것을 고려해 보세요.

임시 테이블스페이스와 플래시 캐시를 인스턴스 스토어에 배치하면 다음과 같은 이점을 얻을 수 있습니다.

- 읽기 지연 시간이 단축됨

- 처리량이 향상됨
- Amazon EBS 볼륨의 로드가 감소함
- Amazon EBS 로드 감소로 스토리지 및 스냅샷 비용이 절감됨
- 높은 IOPS를 프로비저닝할 필요가 줄어들어 전체 비용이 절감될 수 있음

임시 테이블스페이스를 인스턴스 스토어에 배치하면 임시 공간을 사용하는 쿼리의 성능을 즉시 높일 수 있습니다. 인스턴스 스토어에 플래시 캐시를 배치하면 캐시된 블록 읽기는 일반적으로 Amazon EBS 읽기보다 지연 시간이 훨씬 짧습니다. 플래시 캐시는 성능 이점을 제공하기 전에 '워밍업'되어야 합니다. 데이터베이스는 블록이 데이터베이스 버퍼 캐시에서 노후화됨에 따라 플래시 캐시에 블록을 쓰기 때문에 캐시가 저절로 워밍업됩니다.

> **Note**
>
> 캐시 관리로 인해 플래시 캐시가 성능 오버헤드를 유발하는 경우도 있습니다. 프로덕션 환경에서 플래시 캐시를 켜기 전에 테스트 환경에서 워크로드를 분석하고 캐시를 테스트하는 것이 좋습니다.

RDS for Oracle 인스턴스 스토어가 지원되는 인스턴스 클래스

Amazon RDS는 다음 DB 인스턴스 클래스에 인스턴스 스토어를 지원합니다.

- db.m5d
- db.r5d
- db.x2idn
- db.x2iedn

RDS for Oracle은 BYOL 라이선스 모델에 대한 위의 DB 인스턴스 클래스만 지원합니다. 자세한 내용은 지원되는 RDS for Oracle DB 인스턴스 클래스 및 EE 및 SE2용 기존 보유 라이선스 사용(BYOL) 단원을 참조하세요.

지원되는 DB 인스턴스 유형에 대한 총 인스턴스 스토리지를 확인하려면 AWS CLI에서 다음 명령을 실행합니다.

Example

```
aws ec2 describe-instance-types \
```

```
  --filters "Name=instance-type,Values=*5d.*large*" \
  --query "InstanceTypes[?contains(InstanceType,'m5d')||contains(InstanceType,'r5d')]
[InstanceType, InstanceStorageInfo.TotalSizeInGB]" \
  --output table
```

위 명령은 인스턴스 스토어의 원시 디바이스 크기를 반환합니다. RDS for Oracle은 이 공간의 일부를 구성에 사용합니다. 임시 테이블스페이스 또는 플래시 캐시에 사용할 수 있는 인스턴스 스토어의 공간이 약간 더 작습니다.

RDS for Oracle 인스턴스 스토어가 지원되는 엔진 버전

다음 RDS for Oracle 엔진 버전에 인스턴스 스토어가 지원됩니다.

- 21.0.0.0.ru-2022-01.rur-2022-01.r1 이상의 Oracle Database 21c 버전
- 19.0.0.0.ru-2021-10.rur-2021-10.r1 이상의 Oracle Database 19c 버전

RDS for Oracle 인스턴스 스토어가 지원되는 AWS 리전

인스턴스 스토어는 이러한 인스턴스 유형 중 하나 이상이 지원되는 모든 AWS 리전에서 사용 가능합니다. db.m5d 및 db.r5d 인스턴스 클래스에 대한 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요. Amazon RDS for Oracle에서 지원하는 인스턴스 클래스에 대한 자세한 내용은 RDS for Oracle DB 인스턴스 클래스 섹션을 참조하세요.

RDS for Oracle 인스턴스 스토어의 비용

인스턴스 스토어의 비용은 인스턴스 스토어가 활성화된 인스턴스의 비용에 포함됩니다. RDS for Oracle DB 인스턴스에서 인스턴스 스토어를 활성화해도 추가 비용이 발생하지 않습니다. 인스턴스 스토어가 활성화된 인스턴스에 대한 자세한 내용은 RDS for Oracle 인스턴스 스토어가 지원되는 인스턴스 클래스 섹션을 참조하세요.

RDS for Oracle 인스턴스 스토어 활성화

RDS for Oracle 임시 데이터에 인스턴스 스토어를 활성화하려면 다음 중 하나를 수행합니다.

- 지원되는 인스턴스 클래스를 사용하여 RDS for Oracle DB 인스턴스를 생성합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 단원을 참조하십시오.
- 지원되는 인스턴스 클래스를 사용하도록 RDS for Oracle DB 인스턴스를 수정합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

RDS for Oracle 인스턴스 스토어 구성

기본적으로 인스턴스 스토어 공간의 100%가 임시 테이블스페이스에 할당됩니다. 플래시 캐시 및 임시 테이블스페이스에 공간을 할당하도록 인스턴스 스토어를 구성하려면 인스턴스의 파라미터 그룹에서 다음 파라미터를 설정합니다.

db_flash_cache_size={DBInstanceStore*{0,2,4,6,8,10}/10}

이 파라미터는 플래시 캐시에 할당된 스토리지 공간의 양을 지정합니다. 이 파라미터는 Oracle Database Enterprise Edition에서만 유효합니다. 기본값은 `{DBInstanceStore*0/10}`입니다. `db_flash_cache_size`에 0이 아닌 값을 설정한 경우, 사용자가 인스턴스를 재시작한 후 RDS for Oracle 인스턴스에서 플래시 캐시를 활성화합니다.

rds.instance_store_temp_size={DBInstanceStore*{0,2,4,6,8,10}/10}

이 파라미터는 임시 테이블스페이스에 할당된 스토리지 공간의 양을 지정합니다. 기본값은 `{DBInstanceStore*10/10}`입니다. 이 파라미터는 Oracle Database Enterprise Edition의 경우 수정 가능하며 Standard Edition 2의 경우 읽기 전용입니다. `rds.instance_store_temp_size`에 0이 아닌 값을 설정한 경우 Amazon RDS가 임시 테이블스페이스를 위해 인스턴스 스토어에 공간을 할당합니다.

인스턴스 스토어를 사용하지 않는 DB 인스턴스에 대해 `db_flash_cache_size` 및 `rds.instance_store_temp_size` 파라미터를 설정할 수 있습니다. 이 경우 두 설정 모두 `0`으로 평가되어 기능이 비활성화니다. 이 경우 서로 다른 크기의 인스턴스 및 인스턴스 스토어를 사용하지 않는 인스턴스에 동일한 파라미터 그룹을 사용할 수 있습니다. 이러한 파라미터를 수정할 경우 관련 인스턴스를 재부팅하여 변경 사항을 적용해야 합니다.

> **Important**
>
> 임시 테이블스페이스에 공간을 할당하는 경우 Amazon RDS는 임시 테이블스페이스를 자동으로 생성하지 않습니다. 인스턴스 스토어에 임시 테이블스페이스를 생성하는 방법을 알아보려면 인스턴스 스토어에 임시 테이블스페이스 생성 섹션을 참조하세요.

위 파라미터의 합산 값은 10/10 또는 100%를 초과할 수 없습니다. 다음 테이블에는 올바른 파라미터 설정과 잘못된 파라미터 설정이 나와 있습니다.

db_flash_cache_size 설정	rds.instance_store_temp_size 설정	설명
db_flash_cache_size={DBInstanceStore*0/10}	rds.instance_store_temp_size={DBInstanceStore*10/10}	이는 Oracle Database의 모든 버전에 올바른 구성입니다. Amazon RDS가 인스턴스 스토어 공간의 100%를 임시 테이블스페이스에 할당합니다. 이 값이 기본값입니다.
db_flash_cache_size={DBInstanceStore*10/10}	rds.instance_store_temp_size={DBInstanceStore*0/10}	Oracle Database Enterprise Edition에서만 올바른 구성입니다. Amazon RDS가 인스턴스 스토어 공간의 100%를 플래시 캐시에 할당합니다.
db_flash_cache_size={DBInstanceStore*2/10}	rds.instance_store_temp_size={DBInstanceStore*8/10}	Oracle Database Enterprise Edition에서만 올바른 구성입니다. Amazon RDS가 인스턴스 스토어 공간의 20%를 플래시 캐시에 할당하고, 인스턴스

db_flash_cache_size 설정	rds.instance_store_temp_size 설정	설명
		스토어 공간의 80%를 임시 테이블스페이스에 할당합니다.
db_flash_cache_size={DBInstanceStore*6/10}	rds.instance_store_temp_size={DBInstanceStore*4/10}	Oracle Database Enterprise Edition에서만 올바른 구성입니다. Amazon RDS가 인스턴스 스토어 공간의 60%를 플래시 캐시에 할당하고, 인스턴스 스토어 공간의 40%를 임시 테이블스페이스에 할당합니다.

db_flash_cache_size 설정	rds.instance_store_temp_size 설정	설명
db_flash_cache_size={DBInstanceStore*2/10}	rds.instance_store_temp_size={DBInstanceStore*4/10}	Oracle Database Enterprise Edition에서만 올바른 구성입니다. Amazon RDS가 인스턴스 스토어 공간의 20%를 플래시 캐시에 할당하고, 인스턴스 스토어 공간의 40%를 임시 테이블스페이스에 할당합니다.
db_flash_cache_size={DBInstanceStore*8/10}	rds.instance_store_temp_size={DBInstanceStore*8/10}	인스턴스 스토어 공간의 합산 비율이 100%를 초과하기 때문에 잘못된 구성입니다. 이러한 경우 Amazon RDS가 시도에 실패합니다.

DB 인스턴스 유형 변경 시 고려 사항

DB 인스턴스 유형을 변경하면 인스턴스 스토어의 플래시 캐시 또는 임시 테이블스페이스 구성에 영향을 미칠 수 있습니다. 다음과 같은 수정 사항과 그 효과를 고려하세요.

인스턴스 스토어를 지원하는 DB 인스턴스를 스케일 업하거나 스케일 다운합니다.

다음 값이 인스턴스 스토어의 새로운 크기에 비례하여 증가하거나 감소합니다.

- 플래시 캐시의 새로운 크기

- 인스턴스 스토어에 있는 임시 테이블스페이스에 할당된 공간

  예를 들어 db.m5d.4xlarge 인스턴스의 `db_flash_cache_size={DBInstanceStore*6/10}` 설정은 약 340GB의 플래시 캐시 공간을 제공합니다. 인스턴스 유형을 db.m5d.8xlarge로 스케일 업하면 플래시 캐시 공간이 약 680GB로 늘어납니다.

인스턴스 스토어를 사용하지 않는 DB 인스턴스를 인스턴스 스토어를 사용하는 인스턴스로 수정합니다.

`db_flash_cache_size`를 `0`보다 큰 값으로 설정하면 플래시 캐시가 구성됩니다. `rds.instance_store_temp_size`를 `0`보다 큰 값으로 설정하면 임시 테이블스페이스에서 사용할 수 있도록 인스턴스 스토어 공간이 할당됩니다. RDS for Oracle은 인스턴스 스토어로 임시 파일을 자동으로 이동하지 않습니다. 할당된 공간 사용에 대한 자세한 내용은 인스턴스 스토어에 임시 테이블스페이스 생성 또는 읽기 전용 복제본의 인스턴스 스토어에 임시 파일 추가 섹션을 참조하세요.

인스턴스 스토어를 사용하는 DB 인스턴스를 인스턴스 스토어를 사용하지 않는 인스턴스로 수정합니다.

이 경우 RDS for Oracle은 플래시 캐시를 제거합니다. RDS는 현재 Amazon EBS 볼륨의 인스턴스 스토어에 있는 임시 파일을 다시 생성합니다. 새로운 임시 파일의 최대 크기는 `rds.instance_store_temp_size` 파라미터 이전 크기입니다.

Oracle 읽기 전용 복제본에서 인스턴스 스토어로 작업하기

읽기 전용 복제본은 인스턴스 스토어의 플래시 캐시와 임시 테이블스페이스를 지원합니다. 플래시 캐시는 기본 DB 인스턴스와 동일한 방식으로 작동하지만, 임시 테이블스페이스의 경우 다음과 같은 차이점이 있습니다.

- 읽기 전용 복제본에 임시 테이블스페이스를 생성할 수 없습니다. 기본 인스턴스에서 새 임시 테이블스페이스를 생성하는 경우 RDS for Oracle은 임시 파일 없이 테이블스페이스 정보를 복제합니다. 새 임시 파일을 추가하려면 다음 방법 중 하나를 사용합니다.
  - Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.add_inst_store_tempfile`을 사용합니다. RDS for Oracle은 읽기 전용 복제본의 인스턴스 스토어에 임시 파일을 생성하여 지정된 임시 테이블스페이스에 임시 파일을 추가합니다.
  - `ALTER TABLESPACE … ADD TEMPFILE` 명령을 실행합니다. RDS for Oracle은 임시 파일을 Amazon EBS 스토리지에 저장합니다.

> **Note**
>
> 임시 파일 크기 및 스토리지 유형은 기본 DB 인스턴스와 읽기 전용 복제본에서 다를 수 있습니다.

- 기본 임시 테이블스페이스 설정은 기본 DB 인스턴스에서만 관리할 수 있습니다. RDS for Oracle은 이 설정을 모든 읽기 전용 복제본에 복제합니다.
- 임시 테이블스페이스 그룹은 기본 DB 인스턴스에서만 구성할 수 있습니다. RDS for Oracle은 이 설정을 모든 읽기 전용 복제본에 복제합니다.

인스턴스 스토어와 Amazon EBS에 임시 테이블스페이스 그룹 구성

인스턴스 스토어와 Amazon EBS 모두에 임시 테이블스페이스를 포함하도록 임시 테이블스페이스 그룹을 구성할 수 있습니다. 이 방법은 `rds.instance_store_temp_size`의 최대 설정에서 허용하는 것보다 더 큰 임시 저장소가 필요한 경우에 유용합니다.

인스턴스 스토어와 Amazon EBS 모두에 임시 테이블스페이스 그룹을 구성하면 두 테이블스페이스의 성능 특성이 크게 달라집니다. Oracle Database는 내부 알고리즘을 기반으로 쿼리를 제공할 테이블스페이스를 선택합니다. 따라서 비슷한 쿼리의 성능이 달라질 수 있습니다.

일반적으로 다음과 같이 인스턴스 스토어에 임시 테이블스페이스를 생성합니다.

1. 인스턴스 스토어에 임시 테이블스페이스를 생성합니다.
2. 새 테이블스페이스를 데이터베이스 기본 임시 테이블스페이스로 설정합니다.

인스턴스 스토어의 테이블스페이스 크기가 충분하지 않은 경우 다음과 같이 추가 임시 스토리지를 생성할 수 있습니다.

1. 인스턴스 스토어의 임시 테이블스페이스를 임시 테이블스페이스 그룹에 할당합니다.
2. Amazon EBS에 임시 테이블스페이스가 없는 경우 새 임시 테이블스페이스를 생성합니다.
3. Amazon EBS의 임시 테이블스페이스를 인스턴스 스토어 테이블스페이스가 포함된 동일한 테이블스페이스 그룹에 할당합니다.
4. 새 테이블스페이스를 데이터베이스 기본 임시 테이블스페이스로 설정합니다.

다음 예에서는 인스턴스 스토어의 임시 테이블스페이스 크기가 애플리케이션 요구 사항을 충족하지 않는다고 가정합니다. 이 예에서는 인스턴스 스토어에 임시 테이블스페이스

temp_in_inst_store를 생성하여 테이블스페이스 그룹 temp_group에 할당하고 이 그룹에 이름이 temp_in_ebs인 기존 Amazon EBS 테이블스페이스를 추가하고 이 그룹을 기본 임시 테이블스페이스로 설정합니다.

```
SQL> EXEC rdsadmin.rdsadmin_util.create_inst_store_tmp_tblspace('temp_in_inst_store');

PL/SQL procedure successfully completed.

SQL> ALTER TABLESPACE temp_in_inst_store TABLESPACE GROUP temp_group;

Tablespace altered.

SQL> ALTER TABLESPACE temp_in_ebs TABLESPACE GROUP temp_group;

Tablespace altered.

SQL> EXEC rdsadmin.rdsadmin_util.alter_default_temp_tablespace('temp_group');

PL/SQL procedure successfully completed.

SQL> SELECT * FROM DBA_TABLESPACE_GROUPS;

GROUP_NAME                     TABLESPACE_NAME
------------------------------ ------------------------------
TEMP_GROUP                     TEMP_IN_EBS
TEMP_GROUP                     TEMP_IN_INST_STORE

SQL> SELECT PROPERTY_VALUE FROM DATABASE_PROPERTIES WHERE
 PROPERTY_NAME='DEFAULT_TEMP_TABLESPACE';

PROPERTY_VALUE
--------------
TEMP_GROUP
```

RDS for Oracle 인스턴스 스토어 제거

인스턴스 스토어를 제거하려면 db.m5 또는 db.r5 등 인스턴스 스토어를 지원하지 않는 인스턴스 유형을 사용하도록 RDS for Oracle DB 인스턴스를 수정합니다.

RDS for Oracle 인스턴스에 HugePages 활성화

Amazon RDS for Oracle은 Linux 커널 HugePages를 지원해 데이터베이스 확장성을 높입니다. HugePages를 사용하면 페이지 표가 작아지고 메모리 관리에 사용되는 CPU 시간이 줄어 대용량 데이터베이스 인스턴스의 성능이 높아집니다. 자세한 내용은 Oracle 문서의 Overview of HugePages 단원을 참조하세요.

HugePages는 지원되는 모든 버전의 RDS for Oracle에서 사용할 수 있습니다.

`use_large_pages` 파라미터는 DB 인스턴스에서 HugePages의 활성화 여부를 제어합니다. 이 파라미터는 `ONLY`, `FALSE` 또는 `{DBInstanceClassHugePagesDefault}`로 설정할 수 있습니다. Oracle용 기본 DB 파라미터 그룹에서는 `use_large_pages` 파라미터가 `{DBInstanceClassHugePagesDefault}`로 설정됩니다.

DB 인스턴스에서 HugePages의 자동 활성화 여부를 제어하기 위해 파라미터 그룹에서 `DBInstanceClassHugePagesDefault` 수식 변수를 사용할 수 있습니다. 값은 다음과 같이 결정합니다.

- 다음 표에서 언급한 DB 인스턴스 클래스의 경우 `DBInstanceClassHugePagesDefault`는 항상 `FALSE`가 기본값이며 `use_large_pages`는 `FALSE`로 평가됩니다. 이때는 DB 인스턴스 클래스의 메모리 크기가 14GiB 이상이라면 이러한 DB 인스턴스 클래스의 HugePages를 수동으로 활성화할 수 있습니다.
- 다음 표에서 언급하지 않은 DB 인스턴스 클래스의 경우 DB 인스턴스 클래스의 메모리가 14GiB 미만이면 `DBInstanceClassHugePagesDefault`는 항상 `FALSE`로 평가됩니다. 또한 `use_large_pages`는 `FALSE`로 평가됩니다.
- 다음 표에서 언급하지 않은 DB 인스턴스 클래스의 경우 인스턴스 클래스의 메모리가 14GiB 이상에서 100GiB 미만이면 `DBInstanceClassHugePagesDefault`의 기본값이 `TRUE`로 평가됩니다. 또한 `use_large_pages`는 `ONLY`로 평가됩니다. `use_large_pages`를 `FALSE`로 설정하면 HugePages를 수동으로 비활성화할 수 있습니다.
- 다음 표에서 언급하지 않은 DB 인스턴스 클래스의 경우 인스턴스 클래스의 메모리가 100GiB 이상이면 `DBInstanceClassHugePagesDefault`는 항상 `TRUE`로 평가됩니다. 또한 `use_large_pages`는 `ONLY`로 평가되며 HugePages는 비활성화할 수 없습니다.

다음 DB 인스턴스 클래스의 경우, HugePages가 기본적으로 활성화되지 않습니다.

DB 인스턴스 클래스 패밀리	HugePages가 기본적으로 활성화되지 않는 DB 인스턴스 클래스
db.m5	db.m5.large
db.m4	db.m4.large, db.m4.xlarge, db.m4.2xlarge, db.m4.4xlarge, db.m4.10xlarge
db.t3	db.t3.micro, db.t3.small, db.t3.medium, db.t3.large

DB 인스턴스 클래스에 대한 자세한 내용은 에 대한 DB 인스턴스 클래스의 하드웨어 사양 단원을 참조하세요.

신규 또는 기존 DB 인스턴스에 HugePages를 수동으로 활성화하려면 `use_large_pages` 파라미터를 `ONLY`로 설정합니다. Oracle Automatic Memory Management(AMM)에 HugePages를 사용할 수 없습니다. 파라미터 `use_large_pages`를 `ONLY`로 설정하면 `memory_target` 및 `memory_max_target`도 `0`으로 설정해야 합니다. DB 인스턴스를 위한 DB 파라미터를 설정하는 자세한 방법은 Amazon RDS의 파라미터 그룹 단원을 참조하세요.

`sga_target`, `sga_max_size`, `pga_aggregate_target` 파라미터도 설정할 수 있습니다. 시스템 글로벌 영역(SGA)과 프로그램 글로벌 영역(PGA) 메모리 파라미터를 설정할 때 값을 모두 더합니다. 사용 가능한 인스턴스 메모리(`DBInstanceClassMemory`)에서 이 합산 값을 빼면 HugePages 할당 후 가용 메모리를 알 수 있습니다. 최소 2GiB 또는 총 가용 인스턴스 메모리의 10퍼센트 중에서 적은 용량을 비워두어야 합니다.

파라미터를 구성한 후 DB 인스턴스를 재부팅해야 변경 사항이 적용됩니다. 자세한 내용은 DB 인스턴스 재부팅 섹션을 참조하세요.

> **Note**
>
> 장애 조치 없이 인스턴스를 재부팅할 때까지 Oracle DB 인스턴스는 SGA 관련 초기화 파라미터에 대한 변경 사항을 연기합니다. Amazon RDS 콘솔에서 재부팅을 선택하지만 장애 조치로 재부팅을 선택하지 않습니다. AWS CLI에서 `reboot-db-instance` 파라미터로 `--no-force-failover` 명령을 호출합니다. DB 인스턴스는 장애 조치 중이나 인스턴스를 다시 시작하는 기타 유지 관리 작업 중에는 SGA 관련 파라미터를 처리하지 않습니다.

다음은 HugePages를 수동으로 활성화할 수 있는 샘플 파라미터 구성입니다. 사용자의 필요에 맞게 값을 설정해야 합니다.

```
memory_target            = 0
memory_max_target        = 0
pga_aggregate_target     = {DBInstanceClassMemory*1/8}
sga_target               = {DBInstanceClassMemory*3/4}
sga_max_size             = {DBInstanceClassMemory*3/4}
use_large_pages          = ONLY
```

파라미터 그룹에서 파라미터 값이 다음과 같이 설정되어 있다고 가정합니다.

```
memory_target            = IF({DBInstanceClassHugePagesDefault}, 0,
 {DBInstanceClassMemory*3/4})
memory_max_target        = IF({DBInstanceClassHugePagesDefault}, 0,
 {DBInstanceClassMemory*3/4})
pga_aggregate_target     = IF({DBInstanceClassHugePagesDefault},
 {DBInstanceClassMemory*1/8}, 0)
sga_target               = IF({DBInstanceClassHugePagesDefault},
 {DBInstanceClassMemory*3/4}, 0)
sga_max_size             = IF({DBInstanceClassHugePagesDefault},
 {DBInstanceClassMemory*3/4}, 0)
use_large_pages          = {DBInstanceClassHugePagesDefault}
```

이 파라미터 그룹은 메모리가 100GiB 미만인 db.r4 DB 인스턴스 클래스에서 사용됩니다. 이렇게 파라미터를 설정하고 use_large_pages가 {DBInstanceClassHugePagesDefault}로 설정되면 db.r4 인스턴스에서 HugePages가 활성화됩니다.

또 다른 예제에서는 파라미터 그룹의 파라미터 값이 다음과 같이 설정되어 있다고 가정합니다.

```
memory_target           = IF({DBInstanceClassHugePagesDefault}, 0,
 {DBInstanceClassMemory*3/4})
memory_max_target       = IF({DBInstanceClassHugePagesDefault}, 0,
 {DBInstanceClassMemory*3/4})
pga_aggregate_target    = IF({DBInstanceClassHugePagesDefault},
 {DBInstanceClassMemory*1/8}, 0)
sga_target              = IF({DBInstanceClassHugePagesDefault},
 {DBInstanceClassMemory*3/4}, 0)
sga_max_size            = IF({DBInstanceClassHugePagesDefault},
 {DBInstanceClassMemory*3/4}, 0)
use_large_pages         = FALSE
```

위의 파라미터 그룹은 db.r4 DB 인스턴스 클래스와 db.r5 DB 인스턴스 클래스 모두에서 메모리가 100GiB 미만일 때 사용됩니다. 이 파라미터 설정을 사용하면 db.r4 및 db.r5 인스턴스에서 HugePages가 비활성화됩니다.

> **Note**
>
> 위의 파라미터 그룹이 메모리 크기가 100GiB 이상인 db.r4 DB 인스턴스 클래스 또는 db.r5 DB 인스턴스에서 사용될 경우에는 FALSE의 `use_large_pages` 설정이 ONLY 설정으로 재정의 됩니다. 이때는 재정의에 대한 알림 메시지가 고객에게 전송됩니다.

DB 인스턴스에 HugePages가 활성화된 후 향상된 모니터링을 활성화하여 방대한 페이지 정보를 볼 수 있습니다. 자세한 내용은 Enhanced Monitoring을 사용하여 OS 지표 모니터링 단원을 참조하십시오.

RDS for Oracle에서 확장 데이터 유형 활성화

Amazon RDS for Oracle은 확장 데이터 유형을 지원합니다. 확장 데이터 유형의 경우 VARCHAR2, NVARCHAR2, RAW 데이터 유형의 최대 크기는 32,767바이트입니다. 확장 데이터 유형을 사용하려면 `MAX_STRING_SIZE` 파라미터를 EXTENDED로 설정하세요. 자세한 내용은 Oracle 설명서의 확장 데이터 유형 단원을 참조하세요.

확장 데이터 유형을 사용하지 않을 경우 `MAX_STRING_SIZE` 파라미터를 STANDARD(기본값)로 유지하세요. 이 경우 VARCHAR2 및 NVARCHAR2 데이터 유형의 크기 제한은 4,000바이트이며 RAW 데이터 유형의 크기 제한은 2,000바이트입니다.

신규 또는 기존 DB 인스턴스에 대해 확장 데이터 유형을 활성화할 수 있습니다. 신규 DB 인스턴스의 경우 확장 데이터 형식을 활성화하면 일반적으로 DB 인스턴스 생성 시간이 길어집니다. 기존 DB 인스턴스의 경우 변환 과정 중 DB 인스턴스를 사용할 수 없습니다.

확장 데이터 유형에 대한 고려 사항

DB 인스턴스에 확장 데이터 유형을 활성화할 때는 다음 사항을 고려하세요.

- 신규 또는 기존 DB 인스턴스에 대해 확장 데이터 형식을 활성화하면 인스턴스를 재부팅해야만 변경 사항이 적용됩니다.
- 확장 데이터 유형을 활성화하면 데이터 유형에 표준 크기를 사용하도록 DB 인스턴스를 변경할 수 없습니다. `MAX_STRING_SIZE` 파라미터를 STANDARD로 다시 설정하면 `incompatible-parameters` 상태가 됩니다.

- 확장 데이터 유형을 사용하는 DB 인스턴스를 복원하면 MAX_STRING_SIZE 파라미터를 EXTENDED로 설정한 상태에서 파라미터 그룹을 지정해야 합니다. 복원 과정에서 MAX_STRING_SIZE를 STANDARD로 설정한 상태에서 기본값 파라미터 그룹 또는 기타 파라미터 그룹을 지정한 경우에는 incompatible-parameters 상태가 됩니다.
- incompatible-parameters 설정 때문에 DB 인스턴스 상태가 MAX_STRING_SIZE가 되면 MAX_STRING_SIZE 파라미터를 EXTENDED로 설정하고 DB 인스턴스를 재부팅해야만 DB 인스턴스를 사용 가능합니다.

신규 DB 인스턴스에 확장 데이터 유형 활성화

MAX_STRING_SIZE를 EXTENDED로 설정한 DB 인스턴스를 생성하면 인스턴스가 기본값인 STANDARD로 설정된 MAX_STRING_SIZE을 표시합니다. 인스턴스를 재부팅하여 변경 사항을 활성화합니다.

신규 DB 인스턴스에 확장 데이터 유형을 활성화하는 방법

1. 파라미터 그룹의 MAX_STRING_SIZE 파라미터를 EXTENDED로 설정하세요.

   파라미터를 설정하려면 새 파라미터 그룹을 생성하거나 기존 DB 파라미터 그룹을 수정하면 됩니다.

   자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.
2. RDS for Oracle DB 인스턴스를 새로 생성합니다.

   자세한 내용은 Amazon RDS DB 인스턴스 생성 단원을 참조하십시오.
3. MAX_STRING_SIZE를 EXTENDED로 설정한 파라미터 그룹과 DB 인스턴스를 연결합니다.

   자세한 내용은 Amazon RDS DB 인스턴스 생성 단원을 참조하십시오.
4. 파라미터 변경 사항을 적용하려면 DB 인스턴스를 재부팅하세요.

   자세한 내용은 DB 인스턴스 재부팅 단원을 참조하십시오.

기존 DB 인스턴스에 확장 데이터 유형 활성화

확장 데이터 유형을 활성화하도록 DB 인스턴스를 수정하면 RDS는 확장 크기를 사용하도록 데이터베이스의 데이터를 변환합니다. 변환 및 가동 중지는 파라미터를 변경한 다음에 데이터베이스를 재부팅할 때 발생합니다. 변환 과정 중에는 DB 인스턴스를 사용할 수 없습니다.

데이터 변환에 걸리는 시간은 DB 인스턴스 클래스, 데이터베이스 크기 및 마지막 DB 스냅샷의 시간에 따라 달라집니다. 가동 중지를 줄이려면 재부팅 직전에 스냅샷을 생성하는 것이 좋습니다. 이렇게 하면 변환 워크플로우 중에 발생하는 백업 시간이 단축됩니다.

> **Note**
>
> 확장 데이터 유형을 활성화한 후에는 변환 과정 중 시간인 특정 시점으로 복원을 수행할 수 없습니다. 변환 직전 또는 변환 직후 시간으로는 복원할 수 있습니다.

기존 DB 인스턴스에 확장 데이터 유형을 활성화하는 방법

1. 데이터베이스의 스냅샷을 만드십시오.

   데이터베이스에 잘못된 객체가 있는 경우 Amazon RDS는 이들 객체를 다시 컴파일하려고 합니다. Amazon RDS가 잘못된 객체를 다시 컴파일할 수 없는 경우 확장 데이터 유형으로의 변환은 실패할 수 있습니다. 변환에 문제가 있는 경우 스냅샷을 사용하면 데이터베이스를 복원할 수 있습니다. 변환하기 전에 항상 잘못된 객체가 있는지 점검하고 잘못된 객체를 수정하거나 제거하세요. 프로덕션 데이터베이스의 경우 먼저 DB 인스턴스의 복사본에서 변환 프로세스를 테스트하는 것이 좋습니다.

   자세한 내용은 Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성 섹션을 참조하세요.
2. 파라미터 그룹의 `MAX_STRING_SIZE` 파라미터를 `EXTENDED`로 설정하세요.

   파라미터를 설정하려면 새 파라미터 그룹을 생성하거나 기존 DB 파라미터 그룹을 수정하면 됩니다.

   자세한 내용은 Amazon RDS의 파라미터 그룹 섹션을 참조하세요.
3. `MAX_STRING_SIZE`를 `EXTENDED`로 설정한 파라미터 그룹과 연결하도록 DB 인스턴스를 수정하세요.

   자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.
4. 파라미터 변경 사항을 적용하려면 DB 인스턴스를 재부팅하세요.

   자세한 내용은 DB 인스턴스 재부팅 섹션을 참조하세요.

Amazon RDS의 Oracle로 데이터 가져오기

Amazon RDS DB for Oracle 인스턴스로 데이터를 가져오는 방법은 다음 사항에 따라 다릅니다.

- 보유하고 있는 데이터의 양
- 데이터베이스의 데이터베이스 객체 수
- 데이터베이스의 데이터베이스 객체 다양성

예를 들어, 요구 사항에 따라 다음과 같은 도구를 사용할 수 있습니다.

- Oracle SQL Developer - 20MB 데이터베이스를 가져옵니다.
- Oracle Data Pump - 복합 데이터베이스 또는 수백 메가바이트나 수 테라바이트 크기의 데이터베이스를 가져옵니다. 예를 들어 온프레미스 데이터베이스에서 RDS for Oracle DB 인스턴스로 테이블스페이스를 전송할 수 있습니다. Amazon S3 또는 Amazon EFS를 사용하여 데이터 파일 및 메타데이터를 전송할 수 있습니다. 자세한 내용은 Oracle 전송 가능한 테이블스페이스를 사용한 마이그레이션, Amazon EFS 통합, Amazon S3 통합 단원을 참조하세요.
- AWS Database Migration Service(AWS DMS) - 다운타임 없이 데이터베이스를 마이그레이션합니다. AWS DMS에 대한 자세한 내용은 AWS Database Migration Service란 무엇입니까? 및 블로그 게시물 AWS DMS를 사용하여 거의 0에 가까운 다운타임으로 Oracle 데이터베이스 마이그레이션을 참조하세요.

> **Important**
>
> 이전 마이그레이션 기술을 사용하기 전에 데이터베이스를 백업하는 것이 좋습니다. 데이터를 가져온 후 스냅샷을 생성하여 RDS for Oracle DB 인스턴스를 백업할 수 있습니다. 나중에 스냅샷을 복원할 수 있습니다. 자세한 내용은 데이터 백업, 복원 및 내보내기 섹션을 참조하세요.

많은 데이터베이스 엔진의 경우 대상 데이터베이스로 전환할 준비가 될 때까지 진행 중인 복제가 계속될 수 있습니다. AWS DMS를 사용하여 동일하거나 다른 데이터베이스 엔진에서 RDS for Oracle로 마이그레이션할 수 있습니다. 다른 데이터베이스 엔진에서 마이그레이션하는 경우 AWS Schema Conversion Tool을 사용하여 AWS DMS에서 마이그레이션되지 않는 스키마 객체를 마이그레이션할 수 있습니다.

주제

- Oracle SQL Developer를 사용한 가져오기

- Oracle 전송 가능한 테이블스페이스를 사용한 마이그레이션
- Oracle Data Pump를 사용한 가져오기
- Oracle 내보내기/가져오기를 통해 가져오기
- Oracle SQL*Loader를 사용하여 가져오기
- Oracle 구체화된 보기로 마이그레이션

Oracle SQL Developer를 사용한 가져오기

Oracle SQL Developer는 Oracle에서 무상으로 배포한 그래픽 Java 도구입니다. SQL Developer는 두 Oracle 데이터베이스 간의 데이터 마이그레이션이나 MySQL 등의 다른 데이터베이스에서 Oracle 데이터베이스로 데이터 마이그레이션을 위한 옵션을 제공합니다. 이 도구는 작은 데이터베이스를 마이그레이션하는 데 매우 적합합니다.

데스크톱 컴퓨터(Windows, Linux 또는 Mac) 또는 서버 중 하나에 이 도구를 설치할 수 있습니다. SQL Developer를 설치한 후에는 SQL Developer를 사용하여 원본 및 대상 데이터베이스에 연결할 수 있습니다. 도구 메뉴의 Database Copy 명령을 사용하여 데이터를 RDS for Oracle DB 인스턴스에 복사합니다.

SQL Developer를 다운로드하려면 http://www.oracle.com/technetwork/developer-tools/sql-developer를 참조하십시오.

데이터 마이그레이션을 시작하기 전에 Oracle SQL Developer 제품을 읽는 것이 좋습니다. Oracle에서는 MySQL 및 SQL Server 등 다른 데이터베이스에서 마이그레이션하는 방법을 설명하는 문서도 제공합니다. 자세한 정보는 Oracle 설명서에서 http://www.oracle.com/technetwork/database/migration을 참조하십시오.

Oracle 전송 가능한 테이블스페이스를 사용한 마이그레이션

Oracle 전송 가능한 테이블스페이스 기능을 사용하여 온프레미스 Oracle 데이터베이스의 테이블스페이스 세트를 RDS for Oracle DB 인스턴스로 복사할 수 있습니다. 물리적 수준에서 Amazon EFS 또는 Amazon S3를 사용하여 소스 데이터 파일과 메타데이터 파일을 대상 DB 인스턴스에 전송합니다. 전송 가능한 테이블스페이스 기능은 `rdsadmin.rdsadmin_transport_util` 패키지를 사용합니다. 이 패키지의 구문 및 의미는 테이블스페이스 전송 섹션을 참조하세요.

테이블스페이스를 전송하는 방법을 설명하는 블로그 게시물은 전송 가능한 테이블스페이스를 사용하도록 Oracle 데이터베이스를 AWS로 마이그레이션 및 RMAN을 사용한 Amazon RDS for Oracle 전송 가능 테이블스페이스를 참조하세요.

주제

- Oracle 전송 가능한 테이블스페이스 개요
- 1단계: 소스 호스트 설정
- 2단계: 전체 테이블스페이스 백업 준비
- 3단계: 증분 백업 생성 및 전송
- 4단계: 테이블스페이스 전송
- 5단계: 전송된 테이블스페이스 검증
- 6단계: 남은 파일 정리

Oracle 전송 가능한 테이블스페이스 개요

전송 가능한 테이블스페이스 세트는 전송 중인 테이블스페이스 세트에 대한 데이터 파일과 테이블스페이스 메타데이터를 포함하는 내보내기 덤프 파일로 구성됩니다. 전송 가능한 테이블스페이스와 같은 물리적 마이그레이션 솔루션에서는 데이터 파일, 구성 파일 및 Data Pump 덤프 파일과 같은 물리적 파일을 전송합니다.

주제

- 전송 가능한 테이블스페이스의 장점 및 단점
- 전송 가능한 테이블스페이스의 제한 사항
- 전송 가능한 테이블스페이스의 사전 요구 사항

전송 가능한 테이블스페이스의 장점 및 단점

가동 중지 시간을 최소화하면서 하나 이상의 대규모 테이블스페이스를 RDS로 마이그레이션해야 하는 경우 전송 가능한 테이블스페이스를 사용하는 것이 좋습니다. 전송 가능한 테이블스페이스에는 논리적 마이그레이션에 비해 다음과 같은 이점이 있습니다.

- 대부분의 다른 Oracle 마이그레이션 솔루션보다 가동 중지 시간이 짧습니다.
- 전송 가능한 테이블스페이스 기능은 물리적 파일만 복사하므로 논리적 마이그레이션에서 발생할 수 있는 데이터 무결성 오류와 논리적 손상을 방지합니다.
- 추가 라이선스가 필요하지 않습니다.
- 예를 들어 Oracle Solaris 플랫폼에서 Linux로 마이그레이션하는 등 다양한 플랫폼 및 엔디안(endianness) 유형 간에 테이블스페이스 세트를 마이그레이션할 수 있습니다. 하지만 Windows 서버 간에 테이블스페이스를 주고받는 것은 지원되지 않습니다.

> **Note**
>
> Linux는 완벽하게 테스트되고 지원됩니다. 일부 UNIX 버전은 테스트되지 않았습니다.

전송 가능한 테이블스페이스를 사용하는 경우 Amazon S3 또는 Amazon EFS를 사용하여 데이터를 전송할 수 있습니다.

- EFS를 사용하는 경우 백업은 가져오는 동안 EFS 파일 시스템에 남아 있습니다. 나중에 파일을 제거할 수 있습니다. 이 기법에서는 DB 인스턴스에 EBS 스토리지를 프로비저닝할 필요가 없습니다. 따라서 S3 대신 Amazon EFS를 사용하는 것이 좋습니다. 자세한 내용은 Amazon EFS 통합 섹션을 참조하세요.
- S3를 사용하는 경우 DB 인스턴스에 연결된 EBS 스토리지에 RMAN 백업을 다운로드합니다. 가져오는 동안 파일은 EBS 스토리지에 남아 있습니다. 가져온 후에는 DB 인스턴스에 할당된 상태로 남아 있는 이 공간을 비워 여유 공간을 확보할 수 있습니다.

전송 가능한 테이블스페이스의 주요 단점은 Oracle Database에 대해 비교적 높은 수준의 지식이 필요하다는 것입니다. 자세한 내용은 Oracle Database 관리자 안내서의 데이터베이스 간 테이블스페이스 전송을 참조하세요.

전송 가능한 테이블스페이스의 제한 사항

RDS for Oracle에서 이 기능을 사용하는 경우 전송 가능한 테이블스페이스에 대한 Oracle Database 제한 사항이 적용됩니다. 자세한 내용은 Oracle Database 관리자 안내서의 전송 가능한 테이블스페이스 제한 사항 및 데이터 전송에 대한 일반 제한 사항을 참조하세요. RDS for Oracle의 전송 가능한 테이블스페이스에 대한 다음과 같은 추가 제한 사항에 유의하세요.

- 소스 데이터베이스와 대상 데이터베이스 모두 Standard Edition 2(SE2)를 사용할 수 없습니다. Enterprise Edition만 지원됩니다.
- Oracle Database 11g 데이터베이스를 소스로 사용할 수 없습니다. RMAN 크로스 플랫폼 전송 가능한 테이블스페이스 기능은 Oracle Database 11g에서 지원하지 않는 RMAN 전송 메커니즘을 사용합니다.
- 전송 가능한 테이블스페이스를 사용하여 RDS for Oracle DB 인스턴스에서 데이터를 마이그레이션할 수 없습니다. RDS for Oracle DB 인스턴스로 데이터를 마이그레이션하는 경우에만 전송 가능한 테이블스페이스를 사용할 수 있습니다.
- Windows 운영 체제는 지원되지 않습니다.

- 하위 릴리스 수준에서는 테이블스페이스를 데이터베이스로 전송할 수 없습니다. 대상 데이터베이스는 릴리스 수준이 소스 데이터베이스 이상이어야 합니다. 예를 들어 Oracle Database 21c에서 Oracle Database 19c로 테이블스페이스를 전송할 수 없습니다.
- SYSTEM 및 SYSAUX와 같은 관리 테이블스페이스는 전송할 수 없습니다.
- PL/SQL 패키지, Java 클래스, 뷰, 트리거, 시퀀스, 사용자, 역할 및 임시 테이블과 같은 비데이터 객체는 전송할 수 없습니다. 비데이터 객체를 전송하려면 객체를 수동으로 만들거나 Data Pump 메타데이터 내보내기 및 가져오기를 사용하세요. 자세한 내용은 내 Oracle 지원 노트 1454872.1을 참조하세요.
- 암호화된 테이블스페이스 또는 암호화된 열을 사용하는 테이블스페이스를 전송할 수 없습니다.
- Amazon S3를 사용하여 파일을 전송하는 경우 지원되는 최대 파일 크기는 5TiB입니다.
- 소스 데이터베이스에서 Spatial과 같은 Oracle 옵션을 사용하는 경우 대상 데이터베이스에 동일한 옵션이 구성되어 있지 않으면 테이블스페이스를 전송할 수 없습니다.
- Oracle 복제본 구성에서는 테이블스페이스를 RDS for Oracle DB 인스턴스로 전송할 수 없습니다. 이 문제를 해결하려면 모든 복제본을 삭제하고 테이블스페이스를 전송한 다음 복제본을 다시 생성하면 됩니다.

전송 가능한 테이블스페이스의 사전 요구 사항

시작하기 전에 다음 작업을 완료하세요.

- My Oracle Support의 다음 문서에 설명된 전송 가능한 테이블스페이스의 요구 사항을 검토하세요.
  - 교차 플랫폼 증분 백업을 사용하여 전송 가능한 테이블스페이스 가동 시간 단축(문서 ID 2471245.1)
  - 전송 가능한 테이블스페이스(TTS) 제한 사항 및 한계: 세부 정보, 참조 및 해당하는 경우 버전(문서 ID 1454872.1)
  - 전송 가능한 테이블스페이스(TTS)에 대한 기본 참고 사항 - 일반적인 질문 및 문제(문서 ID 1166564.1)
- 엔디안 변환을 계획하세요. 소스 플랫폼 ID를 지정하는 경우 RDS for Oracle은 엔디안을 자동으로 변환합니다. 플랫폼 ID를 찾는 방법을 알아보려면 동일한 Data Guard 구성의 이기종 기본 및 물리적 대기에 대한 Data Guard 지원(문서 ID 413484.1)을 참조하세요.
- 대상 DB 인스턴스에서 전송 가능한 테이블스페이스 기능이 사용 설정되어 있는지 확인하세요. 이 기능은 다음 쿼리를 실행할 때 ORA-20304 오류가 발생하지 않는 경우에만 사용 설정됩니다.

```
SELECT * FROM TABLE(rdsadmin.rdsadmin_transport_util.list_xtts_orphan_files);
```

전송 가능한 테이블스페이스 기능이 사용 설정되지 않았으면 DB 인스턴스를 재부팅하세요. 자세한 내용은 DB 인스턴스 재부팅 섹션을 참조하세요.

- 소스 및 대상 데이터베이스에서 시간대 파일이 동일한지 확인합니다.
- 소스 및 대상 데이터베이스의 데이터베이스 문자 세트가 다음 요구 사항 중 하나를 충족하는지 확인합니다.
  - 문자 세트가 동일해야 합니다.
  - 문자 세트가 호환되어야 합니다. 호환성 요구 사항 목록은 Oracle Database 설명서의 General Limitations on Transporting Data를 참조하세요.
- Amazon S3를 사용하여 파일을 전송하려는 경우 다음을 수행하세요.
  - 파일 전송에 Amazon S3 버킷을 사용할 수 있고 Amazon S3 버킷이 DB 인스턴스와 동일한 AWS 리전에 있어야 합니다. 지침을 보려면 Amazon Simple Storage Service 시작 안내서에서 버킷 생성을 참조하세요.
  - Amazon S3와 RDS for Oracle 통합을 위한 IAM 권한 구성의 지침에 따라 Amazon RDS 통합을 위해 Amazon S3 버킷을 준비해야 합니다.
- Amazon EFS를 사용하여 파일을 전송하려는 경우 Amazon EFS 통합의 지침에 따라 EFS를 구성해야 합니다.
- 대상 DB 인스턴스에서 자동 백업을 켜는 것이 좋습니다. 메타데이터 가져오기 단계가 실패할 수 있으므로 DB 인스턴스를 가져오기 전 상태로 복원할 수 있어야 합니다. 그러면 테이블스페이스를 다시 백업하고 전송하고 가져올 필요가 없습니다.

1단계: 소스 호스트 설정

이 단계에서는 My Oracle Support에서 제공하는 전송 테이블스페이스 스크립트를 복사하고 필요한 구성 파일을 설정합니다. 다음 단계에서는 소스 호스트가 대상 인스턴스로 전송할 테이블스페이스가 포함된 데이터베이스를 실행하고 있습니다.

소스 호스트를 설정하는 방법

1. 소스 호스트에 Oracle 홈의 소유자로 로그인합니다.
2. `ORACLE_HOME` 및 `ORACLE_SID` 환경 변수가 소스 데이터베이스를 가리키는지 확인합니다.
3. 데이터베이스에 관리자로 로그인하고 시간대 버전, DB 문자 세트 및 국가 문자 세트가 대상 데이터베이스와 동일한지 확인합니다.

```
SELECT * FROM V$TIMEZONE_FILE;
```

```
SELECT * FROM NLS_DATABASE_PARAMETERS
  WHERE PARAMETER IN ('NLS_CHARACTERSET','NLS_NCHAR_CHARACTERSET');
```

4. Oracle Support 노트 2471245.1에 설명된 대로 전송 가능한 테이블스페이스 유틸리티를 설정합니다.

   설정에는 소스 호스트의 `xtt.properties` 파일 편집이 포함됩니다. 다음 샘플 `xtt.properties` 파일은 `/dsk1/backups` 디렉터리에 있는 세 개의 테이블스페이스에 대한 백업을 지정합니다. 이는 대상 DB 인스턴스로 전송하려는 테이블스페이스입니다. 또한 엔디안을 자동으로 변환할 소스 플랫폼 ID를 지정합니다.

   > **Note**
   >
   > 유효한 플랫폼 ID 정보는 동일한 Data Guard 구성의 이기종 기본 및 물리적 대기에 대한 Data Guard 지원(문서 ID 413484.1)을 참조하세요.

```
#linux system
platformid=13
#list of tablespaces to transport
tablespaces=TBS1,TBS2,TBS3
#location where backup will be generated
src_scratch_location=/dsk1/backups
#RMAN command for performing backup
usermantransport=1
```

2단계: 전체 테이블스페이스 백업 준비

이 단계에서는 테이블스페이스를 처음으로 백업하고 대상 호스트로 백업을 전송한 다음, `rdsadmin.rdsadmin_transport_util.import_xtts_tablespaces` 프로시저에 따라 테이블스페이스를 복원합니다. 이 단계가 완료되면 초기 테이블스페이스 백업이 대상 DB 인스턴스에 상주하며 증분 백업으로 업데이트할 수 있습니다.

주제

- 1단계: 소스 호스트의 테이블스페이스 백업
- 2단계: 백업 파일을 대상 DB 인스턴스로 전송
- 3단계: 대상 DB 인스턴스에 테이블스페이스 가져오기

1단계: 소스 호스트의 테이블스페이스 백업

이 단계에서는 xttdriver.pl 스크립트를 사용하여 테이블스페이스의 전체 백업을 만듭니다. xttdriver.pl의 출력은 TMPDIR 환경 변수에 저장됩니다.

테이블스페이스를 백업하는 방법

1. 테이블스페이스가 읽기 전용 모드인 경우 ALTER TABLESPACE 권한이 있는 사용자로 소스 데이터베이스에 로그인하고 테이블스페이스를 읽기/쓰기 모드로 설정합니다. 그렇지 않은 경우 다음 단계로 건너뜁니다.

   다음 예에서는 tbs1, tbs2 및 tbs3를 읽기/쓰기 모드로 설정합니다.

   ```
   ALTER TABLESPACE tbs1 READ WRITE;
   ALTER TABLESPACE tbs2 READ WRITE;
   ALTER TABLESPACE tbs3 READ WRITE;
   ```

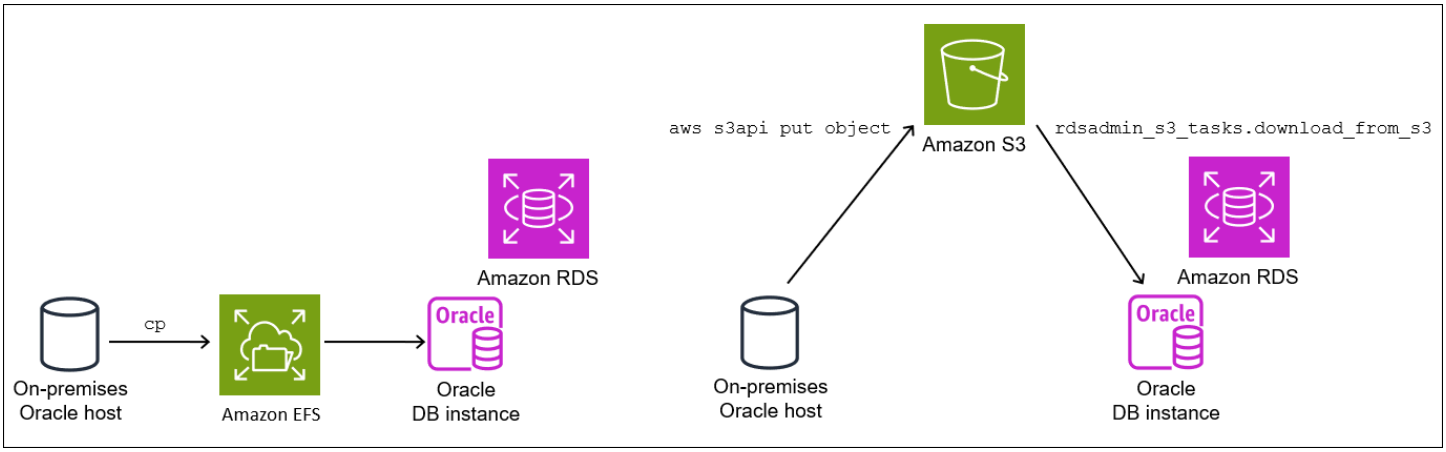
2. xttdriver.pl 스크립트를 사용하여 테이블스페이스를 백업합니다. 선택적으로 스크립트를 디버그 모드에서 실행하도록 --debug를 지정할 수 있습니다.

   ```
   export TMPDIR=location_of_log_files
   cd location_of_xttdriver.pl
   $ORACLE_HOME/perl/bin/perl xttdriver.pl --backup
   ```

2단계: 백업 파일을 대상 DB 인스턴스로 전송

이 단계에서는 스크래치 위치에서 백업 및 구성 파일을 대상 DB 인스턴스로 복사합니다. 다음 옵션 중 하나를 선택하세요.

- 소스 및 대상 호스트가 Amazon EFS 파일 시스템을 공유하는 경우 cp 등의 운영 체제 유틸리티를 사용하여 스크래치 위치에서 백업 파일과 res.txt 파일을 공유 디렉터리로 복사합니다. 3단계: 대상 DB 인스턴스에 테이블스페이스 가져오기 단원을 참조하십시오.
- Amazon S3 버킷으로 백업을 스테이징해야 하는 경우 다음 단계를 완료하세요.



2.2단계: 백업을 Amazon S3 버킷에 업로드

스크래치 디렉터리의 백업과 `res.txt` 파일을 Amazon S3 버킷에 업로드합니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 객체 업로드를 참조하세요.

2.3단계: Amazon S3 버킷에서 대상 DB 인스턴스로 백업 다운로드

이 단계에서는 `rdsadmin.rdsadmin_s3_tasks.download_from_s3` 프로시저를 사용하여 RDS for Oracle DB 인스턴스에 백업을 다운로드합니다.

Amazon S3 버킷에서 백업을 다운로드하는 방법

1. SQL*Plus 또는 Oracle SQL Developer를 시작하고 RDS for Oracle DB 인스턴스에 로그인합니다.
2. Amazon RDS `rdsadmin.rdsadmin_s3_tasks.download_from_s3` 프로시저를 사용하여 Amazon S3 버킷에서 대상 DB 인스턴스로 백업을 다운로드합니다. 다음 예제에서는 *amzn-s3-demo-bucket*이라는 이름의 Amazon S3 버킷에서 모든 파일을 *DATA_PUMP_DIR* 디렉터리로 다운로드합니다.

```
EXEC UTL_FILE.FREMOVE ('DATA_PUMP_DIR', 'res.txt');
SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3(
  p_bucket_name    =>  'amzn-s3-demo-bucket',
  p_directory_name =>  'DATA_PUMP_DIR')
AS TASK_ID FROM DUAL;
```

   SELECT 문은 VARCHAR2 데이터 형식으로 작업 ID를 반환합니다. 자세한 내용은 Amazon S3 버킷의 파일을 Oracle DB 인스턴스로 다운로드 섹션을 참조하세요.

3단계: 대상 DB 인스턴스에 테이블스페이스 가져오기

테이블스페이스를 대상 DB 인스턴스로 복원하려면 `rdsadmin.rdsadmin_transport_util.import_xtts_tablespaces` 프로시저를 사용합니다. 이 프로시저는 데이터 파일을 올바른 엔디안(endian) 형식으로 자동 변환합니다.

Linux 이외의 플랫폼에서 가져오는 경우 `import_xtts_tablespaces` 호출 시 `p_platform_id` 파라미터를 사용하여 소스 플랫폼을 지정합니다. 지정하는 플랫폼 ID가 2단계: 소스 호스트의 테이블스페이스 메타데이터 내보내기의 `xtt.properties` 파일에서 지정하는 ID와 일치해야 합니다.

대상 DB 인스턴스에 테이블스페이스 가져오기

1. Oracle SQL 클라이언트를 시작하고 마스터 사용자로 대상 RDS for Oracle DB 인스턴스에 로그인합니다.
2. 가져올 테이블스페이스와 백업이 포함된 디렉터리를 지정하여 `rdsadmin.rdsadmin_transport_util.import_xtts_tablespaces` 프로시저를 실행합니다.

   다음 예에서는 *DATA_PUMP_DIR* 디렉터리에서 *TBS1*, *TBS2* 및 *TBS3* 테이블스페이스를 가져옵니다. 소스 플랫폼은 플랫폼 ID가 6인 AIX 기반 시스템(64비트)입니다. V$TRANSPORTABLE_PLATFORM 쿼리를 통해 플랫폼 ID를 찾을 수 있습니다.

   ```
   VAR task_id CLOB

   BEGIN
     :task_id:=rdsadmin.rdsadmin_transport_util.import_xtts_tablespaces(
           'TBS1,TBS2,TBS3',
           'DATA_PUMP_DIR',
           p_platform_id => 6);
   END;
   /

   PRINT task_id
   ```

3. (선택 사항) `rdsadmin.rds_xtts_operation_info` 테이블을 쿼리하여 진행 상황을 모니터링합니다. `xtts_operation_state` 열에는 `EXECUTING`, `COMPLETED` 또는 `FAILED` 값이 표시됩니다.

   ```
   SELECT * FROM rdsadmin.rds_xtts_operation_info;
   ```

> **Note**
>
> 장기 실행 작업의 경우 V$SESSION_LONGOPS, V$RMAN_STATUS 및 V$RMAN_OUTPUT을 쿼리할 수도 있습니다.

4. 이전 단계의 작업 ID를 사용하여 완료된 가져오기의 로그를 확인합니다.

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP',
 'dbtask-'||'&task_id'||'.log'));
```

다음 단계로 이동하기 전에 가져오기가 성공적으로 완료되었는지 확인합니다.

3단계: 증분 백업 생성 및 전송

이 단계에서는 소스 데이터베이스가 활성 상태인 동안 정기적으로 증분 백업을 만들어 전송합니다. 이 방법을 사용하면 최종 테이블스페이스 백업의 크기가 줄어듭니다. 증분 백업을 여러 개 수행하는 경우 대상 인스턴스에 적용하려면 먼저 마지막 증분 백업 이후에 `res.txt` 파일을 복사해야 합니다.

가져오기 단계가 선택 사항이라는 점을 제외하고 단계는 2단계: 전체 테이블스페이스 백업 준비와 동일합니다.

4단계: 테이블스페이스 전송

이 단계에서는 읽기 전용 테이블스페이스를 백업하고 Data Pump 메타데이터를 내보내고 이러한 파일을 대상 호스트로 전송하고 테이블스페이스와 메타데이터를 모두 가져옵니다.

주제

- 1단계: 읽기 전용 테이블스페이스 백업
- 2단계: 소스 호스트의 테이블스페이스 메타데이터 내보내기
- 3단계: (Amazon S3만 해당) 백업 및 내보내기 파일을 대상 DB 인스턴스로 전송
- 4단계: 대상 DB 인스턴스에 테이블스페이스 가져오기
- 5단계: 대상 DB 인스턴스에 테이블스페이스 메타데이터 가져오기

1단계: 읽기 전용 테이블스페이스 백업

이 단계는 1단계: 소스 호스트의 테이블스페이스 백업과 동일하지만 한 가지 차이점은 테이블스페이스를 마지막으로 백업하기 전에 테이블스페이스를 읽기 전용 모드로 설정한다는 점입니다.

다음 예에서는 tbs1, tbs2 및 tbs3를 읽기 전용 모드로 설정합니다.

```
ALTER TABLESPACE tbs1 READ ONLY;
ALTER TABLESPACE tbs2 READ ONLY;
ALTER TABLESPACE tbs3 READ ONLY;
```

2단계: 소스 호스트의 테이블스페이스 메타데이터 내보내기

소스 호스트에서 expdb 유틸리티를 실행하여 테이블스페이스 메타데이터를 내보냅니다. 다음 예에서는 *DATA_PUMP_DIR* 디렉터리의 *TBS1*, *TBS2*, *TBS3* 테이블스페이스를 *xttdump.dmp* 덤프 파일로 내보냅니다.

```
expdp username/pwd \
dumpfile=xttdump.dmp \
directory=DATA_PUMP_DIR \
statistics=NONE \
transport_tablespaces=TBS1,TBS2,TBS3 \
transport_full_check=y \
logfile=tts_export.log
```

*DATA_PUMP_DIR*이 Amazon EFS의 공유 디렉터리인 경우 4단계: 대상 DB 인스턴스에 테이블스페이스 가져오기로 건너뛰세요.

3단계: (Amazon S3만 해당) 백업 및 내보내기 파일을 대상 DB 인스턴스로 전송

Amazon S3를 사용하여 테이블스페이스 백업과 Data pump 내보내기 파일을 스테이징하려면 다음 단계를 완료하세요.

3.1단계: 소스 호스트에서 Amazon S3 버킷으로 백업 및 덤프 파일 업로드

소스 호스트에서 Amazon S3 버킷으로 백업 및 덤프 파일을 업로드합니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 객체 업로드를 참조하세요.

3.2단계: Amazon S3 버킷에서 대상 DB 인스턴스로 백업 및 덤프 파일 다운로드

이 단계에서는 rdsadmin.rdsadmin_s3_tasks.download_from_s3 프로시저를 사용하여 RDS for Oracle DB 인스턴스에 백업 및 덤프 파일을 다운로드합니다. 2.3단계: Amazon S3 버킷에서 대상 DB 인스턴스로 백업 다운로드 섹션의 단계를 따르세요.

4단계: 대상 DB 인스턴스에 테이블스페이스 가져오기

`rdsadmin.rdsadmin_transport_util.import_xtts_tablespaces` 프로시저를 사용하여 테이블스페이스를 복원합니다. 이 프로시저의 구문 및 의미는 전송된 테이블스페이스를 DB 인스턴스로 가져오기 섹션을 참조하세요.

> **Important**
>
> 최종 테이블스페이스 가져오기를 완료한 후 다음 단계는 Oracle Data Pump 메타데이터를 가져오는 것입니다. 가져오기에 실패할 경우 DB 인스턴스를 실패 이전 상태로 되돌리는 것이 중요합니다. 따라서 Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성의 지침에 따라 DB 인스턴스의 DB 스냅샷을 만드는 것이 좋습니다. 스냅샷에는 가져온 테이블스페이스가 모두 포함되므로 가져오기가 실패할 경우 백업 및 가져오기 프로세스를 반복하지 않아도 됩니다.
>
> 대상 DB 인스턴스에 자동 백업이 켜져 있는데 메타데이터를 가져오기 전에 Amazon RDS가 유효한 스냅샷이 시작되었음을 감지하지 못하면 RDS는 스냅샷 생성을 시도합니다. 인스턴스 활동에 따라 이 스냅샷은 성공하거나 실패할 수 있습니다. 유효한 스냅샷이 검색되지 않거나 스냅샷을 시작할 수 없는 경우 메타데이터 가져오기가 오류와 함께 종료됩니다.

대상 DB 인스턴스에 테이블스페이스 가져오기

1. Oracle SQL 클라이언트를 시작하고 마스터 사용자로 대상 RDS for Oracle DB 인스턴스에 로그인합니다.
2. 가져올 테이블스페이스와 백업이 포함된 디렉터리를 지정하여 `rdsadmin.rdsadmin_transport_util.import_xtts_tablespaces` 프로시저를 실행합니다.

   다음 예에서는 *DATA_PUMP_DIR* 디렉터리에서 *TBS1*, *TBS2* 및 *TBS3* 테이블스페이스를 가져옵니다.

   ```
   BEGIN

    :task_id:=rdsadmin.rdsadmin_transport_util.import_xtts_tablespaces('TBS1,TBS2,TBS3','DATA_
   END;
   /
   PRINT task_id
   ```

3. (선택 사항) rdsadmin.rds_xtts_operation_info 테이블을 쿼리하여 진행 상황을 모니터링합니다. xtts_operation_state 열에는 EXECUTING, COMPLETED 또는 FAILED 값이 표시됩니다.

```
SELECT * FROM rdsadmin.rds_xtts_operation_info;
```

> **Note**
>
> 장기 실행 작업의 경우 V$SESSION_LONGOPS, V$RMAN_STATUS 및 V$RMAN_OUTPUT을 쿼리할 수도 있습니다.

4. 이전 단계의 작업 ID를 사용하여 완료된 가져오기의 로그를 확인합니다.

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP',
 'dbtask-'||'&task_id'||'.log'));
```

다음 단계로 이동하기 전에 가져오기가 성공적으로 완료되었는지 확인합니다.

5. Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성의 지침에 따라 수동 DB 스냅샷을 생성합니다.

5단계: 대상 DB 인스턴스에 테이블스페이스 메타데이터 가져오기

이 단계에서는 rdsadmin.rdsadmin_transport_util.import_xtts_metadata 프로시저에 따라 전송 가능한 테이블스페이스 메타데이터를 RDS for Oracle DB 인스턴스로 가져옵니다. 이 프로시저의 구문 및 의미는 전송 가능한 테이블스페이스를 DB 인스턴스로 가져오기 섹션을 참조하세요. 작업 중에는 가져오기 상태가 rdsadmin.rds_xtts_operation_info 테이블에 표시됩니다.

> **Important**
>
> 메타데이터를 가져오기 전에 테이블스페이스를 가져온 후 DB 스냅샷이 성공적으로 생성되었는지 확인하는 것이 좋습니다. 가져오기 단계가 실패하면 DB 인스턴스를 복원하고 가져오기 오류를 해결한 다음 가져오기를 다시 시도하세요.

RDS for Oracle DB 인스턴스로 Data Pump 메타데이터 가져오기

1. Oracle SQL 클라이언트를 시작하고 마스터 사용자로 대상 DB 인스턴스에 로그인합니다.

2. 전송된 테이블스페이스에 스키마를 소유하는 사용자가 아직 없다면 생성합니다.

```
CREATE USER tbs_owner IDENTIFIED BY password;
```

3. 덤프 파일의 이름과 디렉터리 위치를 지정하여 메타데이터를 가져옵니다.

```
BEGIN

 rdsadmin.rdsadmin_transport_util.import_xtts_metadata('xttdump.dmp','DATA_PUMP_DIR');
END;
/
```

4. (선택 사항) 전송 가능한 테이블스페이스 기록 테이블을 쿼리하여 메타데이터 가져오기 상태를 확인합니다.

```
SELECT * FROM rdsadmin.rds_xtts_operation_info;
```

   작업이 완료되면 테이블스페이스가 읽기 전용 모드가 됩니다.

5. (선택 사항) 로그 파일을 확인합니다.

   다음 예에서는 BDUMP 디렉터리의 내용을 나열한 다음 가져오기 로그를 쿼리합니다.

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.listdir(p_directory => 'BDUMP'));

SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file(
  p_directory => 'BDUMP',
  p_filename => 'rds-xtts-
import_xtts_metadata-2023-05-22.01-52-35.560858000.log'));
```

5단계: 전송된 테이블스페이스 검증

이 선택적 단계에서는 `rdsadmin.rdsadmin_rman_util.validate_tablespace` 프로시저를 사용하여 전송된 테이블스페이스를 검증한 다음 테이블스페이스를 읽기/쓰기 모드로 설정합니다.

전송된 데이터를 검증하는 방법

1. SQL*Plus 또는 SQL Developer를 시작하고 마스터 사용자로 대상 DB 인스턴스에 로그인합니다.
2. `rdsadmin.rdsadmin_rman_util.validate_tablespace` 프로시저를 사용하여 테이블스페이스를 검증합니다.

```
SET SERVEROUTPUT ON
BEGIN
    rdsadmin.rdsadmin_rman_util.validate_tablespace(
        p_tablespace_name     => 'TBS1',
        p_validation_type     => 'PHYSICAL+LOGICAL',
        p_rman_to_dbms_output => TRUE);
    rdsadmin.rdsadmin_rman_util.validate_tablespace(
        p_tablespace_name     => 'TBS2',
        p_validation_type     => 'PHYSICAL+LOGICAL',
        p_rman_to_dbms_output => TRUE);
    rdsadmin.rdsadmin_rman_util.validate_tablespace(
        p_tablespace_name     => 'TBS3',
        p_validation_type     => 'PHYSICAL+LOGICAL',
        p_rman_to_dbms_output => TRUE);
END;
/
```

3. 테이블스페이스를 읽기/쓰기 모드로 설정합니다.

```
ALTER TABLESPACE TBS1 READ WRITE;
ALTER TABLESPACE TBS2 READ WRITE;
ALTER TABLESPACE TBS3 READ WRITE;
```

6단계: 남은 파일 정리

이 선택적 단계에서는 불필요한 파일을 모두 제거합니다. `rdsadmin.rdsadmin_transport_util.list_xtts_orphan_files` 프로시저를 사용하여 테이블스페이스 가져오기 후 분리된 데이터 파일을 나열한 다음, `rdsadmin.rdsadmin_transport_util.list_xtts_orphan_files` 프로시저를 사용하여 삭제합니다. 이 프로시저의 구문 및 의미는 테이블스페이스 가져오기 후 분리된 파일 나열 및 테이블스페이스 가져오기 후 분리된 파일 삭제 섹션을 참조하세요.

남은 파일을 정리하는 방법

1. 다음과 같이 *DATA_PUMP_DIR*에서 오래된 백업을 제거합니다.

   a. `rdsadmin.rdsadmin_file_util.listdir`을 실행하여 백업 파일을 나열합니다.

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.listdir(p_directory =>
 'DATA_PUMP_DIR'));
```

b. UTL_FILE.FREMOVE를 호출하여 백업을 하나씩 제거합니다.

```
EXEC UTL_FILE.FREMOVE ('DATA_PUMP_DIR', 'backup_filename');
```

2. 테이블스페이스를 가져왔지만 테이블스페이스에 대한 메타데이터를 가져오지 않은 경우 다음과 같이 분리된 데이터 파일을 삭제할 수 있습니다.

a. 삭제해야 하는 분리된 데이터 파일을 나열합니다. 다음 예에서는 rdsadmin.rdsadmin_transport_util.list_xtts_orphan_files 프로시저를 호출합니다.

```
SQL> SELECT * FROM
 TABLE(rdsadmin.rdsadmin_transport_util.list_xtts_orphan_files);

FILENAME       FILESIZE
-------------- ---------
datafile_7.dbf 104865792
datafile_8.dbf 104865792
```

b. rdsadmin.rdsadmin_transport_util.cleanup_incomplete_xtts_import 프로시저를 실행하여 분리된 파일을 삭제합니다.

```
BEGIN

 rdsadmin.rdsadmin_transport_util.cleanup_incomplete_xtts_import('DATA_PUMP_DIR');
END;
/
```

정리 작업을 수행하면 BDUMP 디렉터리에 rds-xtts-delete_xtts_orphaned_files-*YYYY-MM-DD.HH24-MI-SS.FF*.log 이름 형식을 사용하는 로그 파일이 생성됩니다.

c. 이전 단계에서 생성된 로그 파일을 읽습니다. 다음 예에서는 rds-xtts-delete_xtts_orphaned_files-*2023-06-01.09-33-11.868894000*.log 로그를 읽습니다.

```
SELECT *
```

```
FROM TABLE(rdsadmin.rds_file_util.read_text_file(
       p_directory => 'BDUMP',
       p_filename  => 'rds-xtts-
delete_xtts_orphaned_files-2023-06-01.09-33-11.868894000.log'));

TEXT
--------------------------------------------------------------------------------
orphan transported datafile datafile_7.dbf deleted.
orphan transported datafile datafile_8.dbf deleted.
```

3. 테이블스페이스를 가져오고 테이블스페이스에 대한 메타데이터도 가져왔지만 호환성 오류나 기타 Oracle Data Pump 문제가 발생한 경우 부분적으로 전송된 데이터 파일을 다음과 같이 정리하세요.

   a. DBA_TABLESPACES를 쿼리하여 부분적으로 전송된 데이터 파일이 포함된 테이블스페이스를 나열합니다.

   ```
   SQL> SELECT TABLESPACE_NAME FROM DBA_TABLESPACES WHERE PLUGGED_IN='YES';

   TABLESPACE_NAME
   --------------------------------------------------------------------------------
   TBS_3
   ```

   b. 테이블스페이스와 부분적으로 전송된 데이터 파일을 삭제합니다.

   ```
   DROP TABLESPACE TBS_3 INCLUDING CONTENTS AND DATAFILES;
   ```

Oracle Data Pump를 사용한 가져오기

Oracle Data Pump는 Oracle 데이터를 덤프 파일로 내보내고 다른 Oracle 데이터베이스로 가져올 수 있는 유틸리티입니다. Oracle Data Pump는 Oracle 내보내기/가져오기 유틸리티를 장기적으로 대체합니다. Oracle Data Pump는 Oracle 데이터베이스에서 Amazon RDS DB 인스턴스로 대량의 데이터를 이동하는 기본적인 방법입니다.

이 섹션의 예에서는 Oracle 데이터베이스로 데이터를 가져오는 한 방법만 설명하지만, Oracle Data Pump는 다른 여러 가져오기 방법을 지원합니다. 자세한 내용은 Oracle 데이터베이스 설명서를 참조하세요.

이 부분의 예제에서는 DBMS_DATAPUMP 패키지를 사용합니다. Oracle Data Pump 명령줄 유틸리티 impdp 및 expdp를 사용하여 동일한 작업을 수행할 수 있습니다. Oracle 인스턴트 클라이언트를 포함

하여 Oracle 클라이언트 설치의 일부로 원격 호스트에 이러한 유틸리티를 설치할 수 있습니다. 자세한 내용을 알아보려면 Oracle Instant Client를 사용하여 Amazon RDS for Oracle DB 인스턴스에 대해 Data Pump 가져오기 또는 내보내기를 실행하려면 어떻게 해야 합니까?를 참조하세요.

주제

- Oracle Data Pump 개요
- Oracle Data Pump와 Amazon S3 버킷으로 데이터 가져오기
- Oracle Data Pump와 데이터베이스 링크로 데이터 가져오기

Oracle Data Pump 개요

Oracle Data Pump는 다음 구성 요소로 이루어집니다.

- 명령줄 클라이언트 `expdp` 및 `impdp`
- `DBMS_DATAPUMP` PL/SQL 패키지
- `DBMS_METADATA` PL/SQL 패키지

다음 시나리오에 Oracle Data Pump를 사용하면 됩니다.

- Oracle 데이터베이스(온프레미스 또는 Amazon EC2 인스턴스 중 하나)에서 RDS for Oracle DB 인스턴스로 데이터를 가져옵니다.
- RDS for Oracle DB 인스턴스에서 Oracle 데이터베이스(온프레미스 또는 Amazon EC2 인스턴스)로 데이터를 가져옵니다.
- RDS for Oracle DB 인스턴스 간에 데이터를 가져옵니다(예: EC2-Classic에서 VPC로 데이터 마이그레이션).

Oracle Data Pump 유틸리티를 다운로드하려면 Oracle Technology Network 웹사이트의 Oracle 데이터베이스 소프트웨어 다운로드를 참조하세요. Oracle 데이터베이스 버전 간에 마이그레이션할 때 호환성 고려 사항은 Oracle 데이터베이스 설명서를 참조하세요.

Oracle Data Pump 워크플로우

일반적으로 Oracle Data Pump를 사용하는 단계는 다음과 같습니다.

1. 데이터를 소스 데이터베이스의 덤프 파일로 내보냅니다.

2. 덤프 파일을 대상 RDS for Oracle DB 인스턴스로 업로드합니다. Amazon S3 버킷을 사용하거나 두 데이터베이스 간 데이터베이스 링크를 사용하여 전송할 수 있습니다.
3. 덤프 파일의 데이터를 RDS for Oracle DB 인스턴스로 가져옵니다.

Oracle Data Pump 모범 사례

Oracle Data Pump를 사용하여 데이터를 RDS for Oracle 인스턴스로 가져오는 경우 다음과 같은 모범 사례를 적용하는 것이 좋습니다.

- 특정 스키마 및 객체를 가져오려면 `schema` 또는 `table` 모드로 가져오기를 수행하십시오.
- 가져오는 스키마를 애플리케이션에 필요한 스키마로 제한하십시오.
- `full` 모드로 가져오거나 시스템 유지관리 구성 요소의 스키마를 가져오지 않습니다.

  RDS for Oracle은 SYS 또는 SYSDBA 관리 사용자에 대한 액세스를 허용하지 않으므로 이 작업을 수행하면 Oracle 데이터 딕셔너리가 손상되고 데이터베이스 안정성에 영향을 줄 수 있습니다.
- 대량의 데이터를 로드할 경우 다음을 수행합니다.
  1. 덤프 파일을 대상 RDS for Oracle DB 인스턴스로 전송합니다.
  2. DB 인스턴스의 스냅샷을 만듭니다.
  3. 가져오기를 테스트하여 성공적으로 수행되는지 확인합니다.

  데이터베이스 구성 요소가 무효화된 경우 DB 인스턴스를 삭제하고 DB 스냅샷에서 다시 생성할 수 있습니다. 복원된 DB 인스턴스에는 DB 스냅샷을 가져왔을 때 DB 인스턴스에 준비된 모든 덤프 파일이 포함됩니다.
- Oracle Data Pump 내보내기 파라미터 `TRANSPORT_TABLESPACES`, `TRANSPORTABLE` 또는 `TRANSPORT_FULL_CHECK`를 사용하여 생성된 덤프 파일을 가져오지 마세요. RDS for Oracle DB 인스턴스는 이러한 덤프 파일 가져오기를 지원하지 않습니다.
- SYS, SYSTEM, RDSADMIN, RDSSEC 및 RDS_DATAGUARD에 Oracle 스케줄러 객체가 포함되어 있고 다음 범주에 속하는 덤프 파일은 가져오지 마세요.
  - 작업
  - 프로그램
  - Schedules
  - 체인
  - 규칙
  - 평가 컨텍스트

- 규칙 세트

RDS for Oracle DB 인스턴스는 이러한 덤프 파일 가져오기를 지원하지 않습니다.

- 지원되지 않는 Oracle Scheduler 객체를 제외하려면 Data Pump 내보내기 중에 추가 지시문을 사용합니다. DBMS_DATAPUMP를 사용하는 경우 DBMS_METADATA.START_JOB 앞에 METADATA_FILTER를 추가로 넣을 수 있습니다.

```
DBMS_DATAPUMP.METADATA_FILTER(
  v_hdnl,
  'EXCLUDE_NAME_EXPR',
  q'[IN (SELECT NAME FROM SYS.OBJ$
         WHERE TYPE# IN (66,67,74,79,59,62,46)
         AND OWNER# IN
           (SELECT USER# FROM SYS.USER$
            WHERE NAME IN ('RDSADMIN','SYS','SYSTEM','RDS_DATAGUARD','RDSSEC')
            )
        )
  ]',
  'PROCOBJ'
);
```

expdp를 사용하는 경우 다음 예에 표시된 exclude 지시문이 포함된 파라미터 파일을 생성합니다. 그런 다음 PARFILE=*parameter_file* 명령과 함께 expdp을 사용합니다.

```
exclude=procobj:"IN
  (SELECT NAME FROM sys.OBJ$
   WHERE TYPE# IN (66,67,74,79,59,62,46)
   AND OWNER# IN
     (SELECT USER# FROM SYS.USER$
      WHERE NAME IN ('RDSADMIN','SYS','SYSTEM','RDS_DATAGUARD','RDSSEC')
     )
  )"
```

Oracle Data Pump와 Amazon S3 버킷으로 데이터 가져오기

다음 가져오기 프로세스에서는 Oracle Data Pump와 Amazon S3 버킷을 사용합니다. 단계는 다음과 같습니다.

1. Oracle DBMS_DATAPUMP 패키지를 사용한 소스 데이터베이스의 데이터를 내보냅니다.

2. 덤프 파일을 Amazon S3 버킷에 저장합니다.
3. 덤프 파일을 Amazon S3 버킷에서 대상 RDS for Oracle DB 인스턴스의 `DATA_PUMP_DIR` 디렉터리로 다운로드합니다.
4. `DBMS_DATAPUMP` 패키지를 사용하여 복사된 덤프 파일의 데이터를 RDS for Oracle DB 인스턴스로 가져옵니다.

주제

- Oracle Data Pump와 Amazon S3 버킷으로 데이터를 가져오기 위한 요건
- 1단계: RDS for Oracle 대상 DB 인스턴스의 데이터베이스 사용자에게 권한 부여
- 2단계: DBMS_DATAPUMP를 사용하여 데이터를 덤프 파일로 내보내기
- 3단계: 덤프 파일을 Amazon S3 버킷에 업로드
- 4단계: Amazon S3 버킷에서 대상 DB 인스턴스로 덤프 파일 다운로드
- 5단계: DBMS_DATAPUMP를 사용하여 덤프 파일을 대상 DB 인스턴스로 가져오기
- 6단계: 정리

Oracle Data Pump와 Amazon S3 버킷으로 데이터를 가져오기 위한 요건

이 프로세스를 수행하려면 다음 요구 사항이 충족되어야 합니다.

- 파일 전송에 Amazon S3 버킷을 사용할 수 있고 Amazon S3 버킷이 DB 인스턴스와 동일한 AWS 리전에 있어야 합니다. 지침을 보려면 Amazon Simple Storage Service 시작 안내서에서 버킷 생성을 참조하세요.
- Amazon S3 버킷에 업로드하는 객체는 5TB 이하여야 합니다. Amazon S3의 객체 작업에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서를 참조하세요.

> **Note**
>
> 덤프 파일이 5TB를 초과하면 병렬 옵션을 사용하여 Oracle Data Pump 내보내기를 실행할 수 있습니다. 이 작업은 개별 파일에 대해 5TB 제한을 초과하지 않도록 데이터를 여러 덤프 파일로 분산합니다.

- Amazon S3와 RDS for Oracle 통합을 위한 IAM 권한 구성의 지침에 따라 Amazon RDS 통합을 위한 Amazon S3 버킷을 준비해야 합니다.
- 원본 인스턴스 및 대상 DB 인스턴스에 덤프 파일을 저장할 수 있는 충분한 스토리지 공간이 있는지 확인해야 합니다.

> **Note**
>
> 이 프로세스는 덤프 파일을 모든 Oracle DB 인스턴스의 사전 구성된 디렉터리인 `DATA_PUMP_DIR` 디렉터리로 가져옵니다. 이 디렉터리는 데이터 파일과 동일한 스토리지 볼륨에 위치합니다. 덤프 파일을 가져올 때 기존 Oracle 데이터 파일은 더 많은 공간을 사용합니다. 따라서 DB 인스턴스가 공간의 추가 사용을 수용할 수 있는지 확인해야 합니다. 가져온 덤프 파일은 자동으로 삭제되거나 `DATA_PUMP_DIR` 디렉터리에서 제거됩니다. 가져온 덤프 파일을 제거하려면 Oracle 웹사이트에 있는 UTL_FILE.FREMOVE를 사용하십시오.

1단계: RDS for Oracle 대상 DB 인스턴스의 데이터베이스 사용자에게 권한 부여

이 단계에서는 데이터를 가져올 스키마를 생성하고 사용자에게 필요한 권한을 부여합니다.

사용자를 생성하고 RDS for Oracle 대상 인스턴스에 필요한 권한을 부여하는 방법

1. SQL*Plus 또는 Oracle SQL Developer를 사용하여 데이터를 가져올 RDS for Oracle DB 인스턴스에 마스터 사용자로 로그인합니다. DB 인스턴스 연결에 대한 정보는 Oracle DB 인스턴스에 연결 섹션을 참조하세요.
2. 데이터를 가져오려면 먼저 테이블 스페이스를 생성해야 합니다. 자세한 내용은 테이블스페이스 생성과 크기 조정을 참조하세요.
3. 사용자 계정을 생성하고 데이터를 가져올 사용자 계정이 없는 경우 필요한 권한 및 역할을 부여합니다. 데이터를 다수의 사용자 스키마로 가져오려는 경우에는 사용자 계정을 각각 생성한 후에 필요한 권한과 역할을 부여합니다.

   예를 들어 다음 SQL 문은 새로운 사용자를 생성하고 해당 사용자가 소유한 스키마에 데이터를 가져오는 데 필요한 권한과 역할을 부여합니다. 다음 단계에서 *schema_1*을 이 단계의 스키마 이름으로 대체합니다.

   ```
   CREATE USER schema_1 IDENTIFIED BY my_password;
   GRANT CREATE SESSION, RESOURCE TO schema_1;
   ALTER USER schema_1 QUOTA 100M ON users;
   ```

   > **Note**
   >
   > 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

위 문은 새로운 사용자에게 CREATE SESSION 권한과 RESOURCE 역할을 부여합니다. 어떤 데이터베이스 객체를 가져오느냐에 따라 권한과 역할이 추가로 필요할 수 있습니다.

2단계: DBMS_DATAPUMP를 사용하여 데이터를 덤프 파일로 내보내기

덤프 파일을 만들려면 DBMS_DATAPUMP 패키지를 사용합니다.

Oracle 데이터를 덤프 파일로 내보내는 방법

1. SQL Plus 또는 Oracle SQL Developer를 사용하여 관리 사용자 권한으로 소스 RDS for Oracle DB 인스턴스에 연결합니다. 소스 데이터베이스가 RDS for Oracle DB 인스턴스인 경우 Amazon RDS 마스터 사용자 권한으로 연결합니다.
2. DBMS_DATAPUMP 프로시저를 호출하여 데이터를 내보냅니다.

   다음 스크립트는 *SCHEMA_1* 스키마를 DATA_PUMP_DIR 디렉터리에 있는 sample.dmp 덤프 파일로 내보냅니다. *SCHEMA_1*을 내보내려는 스키마 이름으로 바꿉니다.

```
DECLARE
  v_hdnl NUMBER;
BEGIN
  v_hdnl := DBMS_DATAPUMP.OPEN(
    operation => 'EXPORT',
    job_mode  => 'SCHEMA',
    job_name  => null
  );
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl         ,
    filename  => 'sample.dmp'   ,
    directory => 'DATA_PUMP_DIR',
    filetype  => dbms_datapump.ku$_file_type_dump_file
  );
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'sample_exp.log',
    directory => 'DATA_PUMP_DIR' ,
    filetype  => dbms_datapump.ku$_file_type_log_file
  );
  DBMS_DATAPUMP.METADATA_FILTER(v_hdnl,'SCHEMA_EXPR','IN (''SCHEMA_1'')');
  DBMS_DATAPUMP.METADATA_FILTER(
    v_hdnl,
```

```
    'EXCLUDE_NAME_EXPR',
    q'[IN (SELECT NAME FROM SYS.OBJ$
           WHERE TYPE# IN (66,67,74,79,59,62,46)
           AND OWNER# IN
             (SELECT USER# FROM SYS.USER$
              WHERE NAME IN ('RDSADMIN','SYS','SYSTEM','RDS_DATAGUARD','RDSSEC')
             )
          )
    ]',
    'PROCOBJ'
  );
  DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
/
```

> **Note**
>
> Data Pump는 작업을 비동기적으로 시작합니다. Data Pump 작업 모니터링에 대한 자세한 내용은 Oracle 설명서의 Monitoring Job Status를 참조하세요.

3. (선택 사항) `rdsadmin.rds_file_util.read_text_file` 프로시저를 호출하여 내보내기 로그의 내용을 확인할 수 있습니다. 자세한 내용은 DB 인스턴스 디렉터리의 파일 목록 읽기을 참조하세요.

3단계: 덤프 파일을 Amazon S3 버킷에 업로드

Amazon RDS 프로시저 `rdsadmin.rdsadmin_s3_tasks.upload_to_s3`를 사용하여 덤프 파일을 Amazon S3 버킷에 업로드합니다. 다음 예제에서는 `DATA_PUMP_DIR` 디렉터리에서 모든 파일을 *amzn-s3-demo-bucket*이라는 이름의 Amazon S3 버킷에 업로드합니다.

```
SELECT rdsadmin.rdsadmin_s3_tasks.upload_to_s3(
  p_bucket_name    =>  'amzn-s3-demo-bucket',
  p_directory_name =>  'DATA_PUMP_DIR')
AS TASK_ID FROM DUAL;
```

SELECT 문은 VARCHAR2 데이터 형식으로 작업 ID를 반환합니다. 자세한 내용은 RDS for Oracle DB 인스턴스에서 Amazon S3 버킷으로 파일 업로드을 참조하세요.

4단계: Amazon S3 버킷에서 대상 DB 인스턴스로 덤프 파일 다운로드

Amazon RDS `rdsadmin.rdsadmin_s3_tasks.download_from_s3` 프로시저를 사용하여 단계를 수행합니다. 디렉터리에 파일을 다운로드할 때 동일한 이름의 파일이 디렉터리에 이미 있는 경우, 이 `download_from_s3` 프로시저는 다운로드를 건너뜁니다. 다운로드 디렉터리에서 파일을 제거하려면 Oracle 웹사이트에 있는 UTL_FILE.FREMOVE를 사용하세요.

덤프 파일을 다운로드하는 방법

1. SQL*Plus 또는 Oracle SQL Developer를 시작하여 Amazon RDS 대상 Oracle DB 인스턴스에서 마스터로 로그인합니다.
2. Amazon RDS `rdsadmin.rdsadmin_s3_tasks.download_from_s3` 프로시저를 사용하여 덤프 파일을 다운로드합니다.

   다음 예에서는 *amzn-s3-demo-bucket*이라는 이름의 Amazon S3 버킷에서 모든 파일을 `DATA_PUMP_DIR` 디렉터리로 다운로드합니다.

   ```
   SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3(
     p_bucket_name    =>  'amzn-s3-demo-bucket',
     p_directory_name =>  'DATA_PUMP_DIR')
   AS TASK_ID FROM DUAL;
   ```

   `SELECT` 문은 VARCHAR2 데이터 형식으로 작업 ID를 반환합니다. 자세한 내용은 Amazon S3 버킷의 파일을 Oracle DB 인스턴스로 다운로드을 참조하세요.

5단계: DBMS_DATAPUMP를 사용하여 덤프 파일을 대상 DB 인스턴스로 가져오기

`DBMS_DATAPUMP`를 사용하여 스키마를 RDS for Oracle DB 인스턴스로 가져옵니다. `METADATA_REMAP`과 같은 추가 옵션이 필요할 수 있습니다.

대상 DB 인스턴스로 데이터를 가져오는 방법

1. SQL*Plus 또는 SQL Developer를 시작하여 마스터 사용자로 RDS for Oracle DB 인스턴스에 로그인합니다.
2. `DBMS_DATAPUMP` 프로시저를 직접 호출하여 데이터를 가져옵니다.

   다음 예에서는 `sample_copied.dmp`에서 대상 DB 인스턴스로 *SCHEMA_1* 데이터를 가져옵니다.

   ```
   DECLARE
   ```

```
  v_hdnl NUMBER;
BEGIN
  v_hdnl := DBMS_DATAPUMP.OPEN(
    operation => 'IMPORT',
    job_mode  => 'SCHEMA',
    job_name  => null);
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'sample_copied.dmp',
    directory => 'DATA_PUMP_DIR',
    filetype  => dbms_datapump.ku$_file_type_dump_file);
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'sample_imp.log',
    directory => 'DATA_PUMP_DIR',
    filetype  => dbms_datapump.ku$_file_type_log_file);
  DBMS_DATAPUMP.METADATA_FILTER(v_hdnl,'SCHEMA_EXPR','IN (''SCHEMA_1'')');
  DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
/
```

> **Note**
>
> Data Pump 작업은 비동기로 시작됩니다. Data Pump 작업 모니터링에 대한 자세한 정보는 Oracle 설명서의 Monitoring Job Status를 참조하십시오. `rdsadmin.rds_file_util.read_text_file` 절차를 사용하여 가져오기 로그의 내용을 볼 수 있습니다. 자세한 내용은 DB 인스턴스 디렉터리의 파일 목록 읽기을 참조하세요.

3. 대상 DB 인스턴스의 스키마 테이블을 나열하여 데이터 가져오기 작업을 확인합니다.

   예를 들어 다음 쿼리는 *SCHEMA_1*의 테이블 수를 반환합니다.

```
SELECT COUNT(*) FROM DBA_TABLES WHERE OWNER='SCHEMA_1';
```

6단계: 정리

데이터를 가져온 후에는 유지하지 않을 파일을 삭제할 수 있습니다.

불필요한 파일을 제거하는 방법

1. SQL*Plus 또는 SQL Developer를 시작하여 마스터 사용자로 RDS for Oracle DB 인스턴스에 로그인합니다.
2. 다음 명령을 사용하여 DATA_PUMP_DIR의 파일을 나열합니다.

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.listdir('DATA_PUMP_DIR')) ORDER BY
 MTIME;
```

3. 다음 명령을 사용하여 DATA_PUMP_DIR에서 더 이상 필요하지 않은 파일을 삭제합니다.

```
EXEC UTL_FILE.FREMOVE('DATA_PUMP_DIR','filename');
```

예를 들어, 다음 명령은 sample_copied.dmp라는 파일을 삭제합니다.

```
EXEC UTL_FILE.FREMOVE('DATA_PUMP_DIR','sample_copied.dmp');
```

Oracle Data Pump와 데이터베이스 링크로 데이터 가져오기

다음 가져오기 프로세스에서는 Oracle Data Pump 및 Oracle DBMS_FILE_TRANSFER 패키지를 사용합니다. 단계는 다음과 같습니다.

1. 소스 Oracle 데이터베이스(온프레미스 데이터베이스, Amazon EC2 인스턴스 또는 RDS for Oracle DB 인스턴스 등)에 연결합니다.
2. DBMS_DATAPUMP 패키지를 사용하여 데이터를 내보냅니다.
3. DBMS_FILE_TRANSFER.PUT_FILE을 사용하여 데이터베이스 링크를 통해 연결된 대상 RDS for Oracle DB 인스턴스의 DATA_PUMP_DIR 디렉터리에 Oracle 데이터베이스의 덤프 파일을 복사합니다.
4. DBMS_DATAPUMP 패키지를 사용하여 복사된 덤프 파일의 데이터를 RDS for Oracle DB 인스턴스로 가져옵니다.

Oracle Data Pump 및 DBMS_FILE_TRANSFER 패키지를 사용하는 가져오기 프로세스는 다음 단계로 이루어집니다.

주제

- Oracle Data Pump와 데이터베이스 링크로 데이터를 가져오기 위한 요건

- 1단계: RDS for Oracle 대상 DB 인스턴스 사용자에게 권한 부여
- 2단계: 소스 데이터베이스에서 사용자에게 권한 부여
- 3단계: DBMS_DATAPUMP를 사용하여 덤프 파일 생성
- 4단계: 대상 DB 인스턴스의 데이터베이스 링크 생성
- 5단계: DBMS_FILE_TRANSFER를 사용하여 내보낸 덤프 파일을 대상 DB 인스턴스로 복사
- 6단계: DBMS_DATAPUMP를 사용하여 대상 DB 인스턴스로 데이터 파일 가져오기
- 7단계: 정리

Oracle Data Pump와 데이터베이스 링크로 데이터를 가져오기 위한 요건

이 프로세스를 수행하려면 다음 요구 사항이 충족되어야 합니다.

- `DBMS_FILE_TRANSFER` 및 `DBMS_DATAPUMP` 패키지에 대한 실행 권한이 있어야 합니다.
- 원본 DB 인스턴스의 `DATA_PUMP_DIR` 디렉터리에 대한 쓰기 권한이 있어야 합니다.
- 원본 인스턴스 및 대상 DB 인스턴스에 덤프 파일을 저장할 수 있는 충분한 스토리지 공간이 있는지 확인해야 합니다.

> **Note**
>
> 이 프로세스는 덤프 파일을 모든 Oracle DB 인스턴스의 사전 구성된 디렉터리인 `DATA_PUMP_DIR` 디렉터리로 가져옵니다. 이 디렉터리는 데이터 파일과 동일한 스토리지 볼륨에 위치합니다. 덤프 파일을 가져올 때 기존 Oracle 데이터 파일은 더 많은 공간을 사용합니다. 따라서 DB 인스턴스가 공간의 추가 사용을 수용할 수 있는지 확인해야 합니다. 가져온 덤프 파일은 자동으로 삭제되거나 `DATA_PUMP_DIR` 디렉터리에서 제거됩니다. 가져온 덤프 파일을 제거하려면 Oracle 웹사이트에 있는 UTL_FILE.FREMOVE를 사용하십시오.

1단계: RDS for Oracle 대상 DB 인스턴스 사용자에게 권한 부여

RDS for Oracle 대상 DB 인스턴스에서 사용자에게 권한을 부여하려면 다음 단계를 수행합니다.

1. SQL Plus 또는 Oracle SQL Developer를 사용하여 데이터를 가져올 RDS for Oracle DB 인스턴스에 연결합니다. Amazon RDS 마스터 사용자 권한으로 연결합니다. DB 인스턴스 연결에 대한 자세한 정보는 Oracle DB 인스턴스에 연결 단원을 참조하십시오.
2. 데이터를 가져오려면 먼저 테이블 스페이스를 생성해야 합니다. 자세한 내용은 테이블스페이스 생성과 크기 조정 섹션을 참조하세요.

3. 데이터를 가져올 사용자 계정이 존재하지 않으면 사용자 계정을 생성한 후 필요한 권한과 역할을 부여합니다. 데이터를 다수의 사용자 스키마로 가져오려는 경우에는 사용자 계정을 각각 생성한 후에 필요한 권한과 역할을 부여합니다.

   예를 들어 다음 명령은 *schema_1*이라는 신규 사용자를 생성하고, 해당 사용자의 스키마로 데이터를 가져오는 데 필요한 권한과 역할을 부여합니다.

```
CREATE USER schema_1 IDENTIFIED BY my-password;
GRANT CREATE SESSION, RESOURCE TO schema_1;
ALTER USER schema_1 QUOTA 100M ON users;
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

   앞선 예에서는 새로운 사용자에게 CREATE SESSION 권한과 RESOURCE 역할을 부여합니다. 하지만 가져오는 데이터베이스 객체에 따라 권한과 역할이 추가로 필요할 수도 있습니다.

> **Note**
>
> 다음 단계에서 *schema_1*을 이 단계의 스키마 이름으로 대체합니다.

2단계: 소스 데이터베이스에서 사용자에게 권한 부여

SQL*Plus 또는 Oracle SQL Developer를 사용하여 가져올 데이터를 포함하는 RDS for Oracle DB 인스턴스에 연결합니다. 필요할 경우 사용자 계정을 생성하고 필요한 권한을 부여합니다.

> **Note**
>
> 원본 데이터베이스가 Amazon RDS 인스턴스인 경우 이 단계를 건너뛸 수 있습니다. 이 경우 Amazon RDS 마스터 사용자 계정을 사용하여 데이터를 내보냅니다.

다음 명령은 새 사용자를 생성하고 필요한 권한을 부여합니다.

```
CREATE USER export_user IDENTIFIED BY my-password;
GRANT CREATE SESSION, CREATE TABLE, CREATE DATABASE LINK TO export_user;
```

```
ALTER USER export_user QUOTA 100M ON users;
GRANT READ, WRITE ON DIRECTORY data_pump_dir TO export_user;
GRANT SELECT_CATALOG_ROLE TO export_user;
GRANT EXECUTE ON DBMS_DATAPUMP TO export_user;
GRANT EXECUTE ON DBMS_FILE_TRANSFER TO export_user;
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

3단계: DBMS_DATAPUMP를 사용하여 덤프 파일 생성

덤프 파일을 만들려면 다음을 수행합니다.

1. SQL*Plus 또는 Oracle SQL Developer를 통해 관리 사용자 권한으로, 또는 2단계에서 생성한 사용자 권한으로 소스 Oracle 인스턴스에 연결합니다. 소스 데이터베이스가 Amazon RDS for Oracle DB 인스턴스인 경우 Amazon RDS 마스터 사용자 권한으로 연결합니다.
2. Oracle Data Pump 유틸리티를 사용하여 덤프 파일을 생성합니다.

   다음 스크립트는 DATA_PUMP_DIR 디렉터리에 sample.dmp라는 덤프 파일을 생성합니다.

```
DECLARE
  v_hdnl NUMBER;
BEGIN
  v_hdnl := DBMS_DATAPUMP.OPEN(
    operation => 'EXPORT' ,
    job_mode  => 'SCHEMA' ,
    job_name  => null
  );
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'sample.dmp'    ,
    directory => 'DATA_PUMP_DIR' ,
    filetype  => dbms_datapump.ku$_file_type_dump_file
  );
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl           ,
    filename  => 'sample_exp.log' ,
    directory => 'DATA_PUMP_DIR'  ,
    filetype  => dbms_datapump.ku$_file_type_log_file
  );
```

```
  DBMS_DATAPUMP.METADATA_FILTER(
    v_hdnl              ,
    'SCHEMA_EXPR'       ,
    'IN (''SCHEMA_1'')'
  );
  DBMS_DATAPUMP.METADATA_FILTER(
    v_hdnl,
    'EXCLUDE_NAME_EXPR',
    q'[IN (SELECT NAME FROM sys.OBJ$
           WHERE TYPE# IN (66,67,74,79,59,62,46)
           AND OWNER# IN
             (SELECT USER# FROM SYS.USER$
              WHERE NAME IN ('RDSADMIN','SYS','SYSTEM','RDS_DATAGUARD','RDSSEC')
             )
          )
    ]',
    'PROCOBJ'
  );
  DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
/
```

> **Note**
>
> Data Pump 작업은 비동기로 시작됩니다. Data Pump 작업 모니터링에 대한 자세한 정보는 Oracle 설명서의 Monitoring Job Status를 참조하십시오. `rdsadmin.rds_file_util.read_text_file` 절차를 사용하여 내보내기 로그의 내용을 볼 수 있습니다. 자세한 내용은 DB 인스턴스 디렉터리의 파일 목록 읽기 섹션을 참조하세요.

4단계: 대상 DB 인스턴스의 데이터베이스 링크 생성

소스 DB 인스턴스와 대상 DB 인스턴스 간에 데이터베이스 링크를 생성합니다. 데이터베이스 링크를 생성하고 내보내기 덤프 파일을 전송하려면 로컬 Oracle 인스턴스가 DB 인스턴스와 네트워크로 연결되어 있어야 합니다.

이번 단계에서도 이전 단계와 동일한 사용자 계정에 연결합니다.

동일한 VPC 또는 피어링된 VPC 내에서 두 DB 인스턴스 간에 데이터베이스 링크를 생성하려면 두 DB 인스턴스에 서로에게 이르는 유효한 경로가 있어야 합니다. 각 DB 인스턴스의 보안 그룹은 다른 DB 인

스턴스로(부터)의 수신 및 발신을 허용해야 합니다. 보안 그룹 인바운드 또는 아웃바운드 규칙은 동일한 VPC 또는 피어링된 VPC에서 보안 그룹을 참조할 수 있습니다. 자세한 내용은 VPC의 DB 인스턴스에 사용하기 위한 데이터베이스 링크 조정 섹션을 참조하세요.

다음 명령은 대상 DB 인스턴스의 Amazon RDS 마스터 사용자에게 연결하는 to_rds라는 데이터베이스 링크를 생성합니다.

```
CREATE DATABASE LINK to_rds
  CONNECT TO <master_user_account> IDENTIFIED BY <password>
  USING '(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<dns or ip address of remote db>)
         (PORT=<listener port>))(CONNECT_DATA=(SID=<remote SID>)))';
```

5단계: DBMS_FILE_TRANSFER를 사용하여 내보낸 덤프 파일을 대상 DB 인스턴스로 복사

DBMS_FILE_TRANSFER를 사용하여 원본 데이터베이스의 덤프 파일을 대상 DB 인스턴스로 복사합니다. 다음 스크립트는 원본 인스턴스에 있는 sample.dmp라는 덤프 파일을 to_rds(이전 단계에서 생성됨)라는 대상 데이터베이스 링크로 복사합니다.

```
BEGIN
  DBMS_FILE_TRANSFER.PUT_FILE(
    source_directory_object       => 'DATA_PUMP_DIR',
    source_file_name              => 'sample.dmp',
    destination_directory_object  => 'DATA_PUMP_DIR',
    destination_file_name         => 'sample_copied.dmp',
    destination_database          => 'to_rds' );
END;
/
```

6단계: DBMS_DATAPUMP를 사용하여 대상 DB 인스턴스로 데이터 파일 가져오기

DB 인스턴스에서 Oracle Data Pump를 사용하여 스키마를 가져옵니다. METADATA_REMAP 등 추가 옵션이 필요할 수 있습니다.

Amazon RDS 마스터 사용자 계정으로 DB 인스턴스에 연결하여 데이터를 가져옵니다.

```
DECLARE
  v_hdnl NUMBER;
BEGIN
  v_hdnl := DBMS_DATAPUMP.OPEN(
    operation => 'IMPORT',
    job_mode  => 'SCHEMA',
```

```
    job_name  => null);
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'sample_copied.dmp',
    directory => 'DATA_PUMP_DIR',
    filetype  => dbms_datapump.ku$_file_type_dump_file );
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'sample_imp.log',
    directory => 'DATA_PUMP_DIR',
    filetype  => dbms_datapump.ku$_file_type_log_file);
  DBMS_DATAPUMP.METADATA_FILTER(v_hdnl,'SCHEMA_EXPR','IN (''SCHEMA_1'')');
  DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
/
```

> **Note**
>
> Data Pump 작업은 비동기로 시작됩니다. Data Pump 작업 모니터링에 대한 자세한 정보는 Oracle 설명서의 Monitoring Job Status를 참조하십시오. `rdsadmin.rds_file_util.read_text_file` 절차를 사용하여 가져오기 로그의 내용을 볼 수 있습니다. 자세한 내용은 DB 인스턴스 디렉터리의 파일 목록 읽기 섹션을 참조하세요.

DB 인스턴스에서 해당 사용자의 테이블을 보고 데이터 가져오기를 확인할 수 있습니다. 예를 들어 다음 쿼리는 *schema_1*의 테이블 수를 반환합니다.

```
SELECT COUNT(*) FROM DBA_TABLES WHERE OWNER='SCHEMA_1';
```

7단계: 정리

데이터를 가져온 후에는 유지하지 않을 파일을 삭제할 수 있습니다. 다음 명령을 사용하여 DATA_PUMP_DIR의 파일을 나열할 수 있습니다.

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.listdir('DATA_PUMP_DIR')) ORDER BY MTIME;
```

DATA_PUMP_DIR에서 더 이상 필요하지 않은 파일을 삭제하려면 다음 명령을 사용합니다.

```
EXEC UTL_FILE.FREMOVE('DATA_PUMP_DIR','<file name>');
```

예를 들어, 다음 명령은 "sample_copied.dmp"라는 파일을 삭제합니다.

```
EXEC UTL_FILE.FREMOVE('DATA_PUMP_DIR','sample_copied.dmp');
```

Oracle 내보내기/가져오기를 통해 가져오기

다음 조건에서는 Oracle 내보내기/가져오기 유틸리티를 마이그레이션에 사용하는 것이 효과적일 수도 있습니다.

- 데이터 크기가 작습니다.
- 이진 플로트 및 더블과 같은 데이터 유형은 필요하지 않습니다.

가져오기 프로세스는 필요한 스키마 객체를 생성합니다. 따라서 객체를 생성하기 위해 스크립트를 먼저 실행할 필요가 없습니다.

내보내기 및 가져오기 유틸리티로 Oracle Instant Client를 설치하기 위한 가장 쉬운 방법은 Oracle을 설치하는 것입니다. 소프트웨어를 다운로드하려면 https://www.oracle.com/database/technologies/instant-client.html을 방문하세요. 설명서를 보려면 Oracle Database 유틸리티 설명서의 SQL*Loader, 내보내기, 가져오기를 위한 Instant Client를 참조하세요.

테이블을 내보낸 다음 가져오기

1. exp 명령을 사용하여 소스 데이터베이스에서 테이블을 내보냅니다.

   다음 명령은 tab1, tab2 및 tab3이라는 테이블을 내보냅니다. 덤프 파일은 exp_file.dmp입니다.

   ```
   exp cust_dba@ORCL FILE=exp_file.dmp TABLES=(tab1,tab2,tab3) LOG=exp_file.log
   ```

   내보내기 프로세스에서는 지정된 테이블에 대한 스키마 및 데이터를 모두 포함하는 이진 덤프 파일을 생성합니다.
2. 이제 imp 명령을 사용하여 이 스키마와 데이터를 대상 데이터베이스로 가져옵니다.

   다음 명령을 실행하면 덤프 파일 exp_file.dmp에서 tab1, tab2, tab3 테이블을 가져옵니다.

   ```
   imp cust_dba@targetdb FROMUSER=cust_schema TOUSER=cust_schema \
   TABLES=(tab1,tab2,tab3) FILE=exp_file.dmp LOG=imp_file.log
   ```

사용자의 필요에 맞게 여러 버전의 내보내기 및 가져오기가 있습니다. 자세한 내용은 Oracle 설명서를 참조하십시오.

Oracle SQL*Loader를 사용하여 가져오기

객체 수가 제한되어 있는 대규모 데이터베이스의 경우 Oracle SQL*Loader를 사용할 수도 있습니다. 원본 데이터베이스에서 내보내고 대상 데이터베이스로 로드하는 프로세스는 스키마와 매우 밀접한 관계가 있으므로, 다음 예에서는 샘플 스키마 객체를 생성하여 원본에서 내보내고 대상 데이터베이스에 로드합니다.

Oracle SQL*Loader를 설치하는 가장 쉬운 방법은 Oracle Instant Client를 설치하는 것입니다. 소프트웨어를 다운로드하려면 https://www.oracle.com/database/technologies/instant-client.html을 방문하세요. 설명서를 보려면 Oracle Database 유틸리티 설명서의 SQL*Loader, 내보내기, 가져오기를 위한 Instant Client를 참조하세요.

Oracle SQL*Loader를 사용하여 데이터를 가져오려면

1. 다음 SQL 문을 사용해 샘플 원본 테이블을 생성합니다.

```
CREATE TABLE customer_0 TABLESPACE users
   AS (SELECT ROWNUM id, o.*
       FROM   ALL_OBJECTS o, ALL_OBJECTS x
       WHERE  ROWNUM <= 1000000);
```

2. 대상 RDS for Oracle DB 인스턴스에서 데이터를 로드하는 데 사용되는 대상 테이블을 생성합니다. `WHERE 1=2` 절을 사용하면 `ALL_OBJECTS`의 구조를 복사하지만 행은 복사하지 않게 됩니다.

```
CREATE TABLE customer_1 TABLESPACE users
  AS (SELECT 0 AS ID, OWNER, OBJECT_NAME, CREATED
      FROM   ALL_OBJECTS
      WHERE  1=2);
```

3. 원본 데이터베이스의 데이터를 텍스트 파일로 내보냅니다. 다음 예에서는 SQL*Plus를 사용합니다. 실제 데이터의 경우 대개 데이터베이스의 모든 객체에 대한 내보내기를 수행하는 스크립트를 생성해야 할 것입니다.

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY/MM/DD HH24:MI:SS'

SET LINESIZE 800 HEADING OFF FEEDBACK OFF ARRAY 5000 PAGESIZE 0
SPOOL customer_0.out
SET MARKUP HTML PREFORMAT ON
```

```
SET COLSEP ','

SELECT id, owner, object_name, created
FROM   customer_0;

SPOOL OFF
```

4. 데이터를 설명하는 제어 파일을 생성해야 합니다. 이 단계를 수행하려면 스크립트를 작성해야 할 수 있습니다.

```
cat << EOF > sqlldr_1.ctl
load data
infile customer_0.out
into table customer_1
APPEND
fields terminated by "," optionally enclosed by '"'
(
  id           POSITION(01:10)    INTEGER EXTERNAL,
  owner        POSITION(12:41)    CHAR,
  object_name  POSITION(43:72)    CHAR,
  created      POSITION(74:92)    date "YYYY/MM/DD HH24:MI:SS"
)
```

필요할 경우 이전 코드에서 생성된 파일을 Amazon EC2 인스턴스 등의 스테이징 영역으로 복사합니다.

5. 대상 데이터베이스에 대한 적절한 사용자 이름 및 암호와 함께 SQL*Loader를 사용하여 데이터를 가져옵니다.

```
sqlldr cust_dba@targetdb CONTROL=sqlldr_1.ctl BINDSIZE=10485760 READSIZE=10485760
 ROWS=1000
```

Oracle 구체화된 보기로 마이그레이션

Oracle 구체화된 보기 복제를 사용하여 최신 데이터 세트를 효율적으로 마이그레이션할 수 있습니다. 복제를 사용하면 대상 테이블을 원본 테이블과 동기화된 상태로 유지할 수 있습니다. 따라서 필요한 경우 나중에 Amazon RDS로 전환할 수 있습니다.

구체화된 보기로 마이그레이션하려면 먼저 다음 요구 사항을 충족해야 합니다.

- 대상 데이터베이스에서 원본 데이터베이스로의 액세스를 구성합니다. 다음 예에서는 SQL*Net을 통해 RDS for Oracle 대상 데이터베이스가 원본에 연결할 수 있도록 원본 데이터베이스에서 액세스 규칙이 활성화되었습니다.
- RDS for Oracle DB 인스턴스에서 원본 데이터베이스로 연결되는 데이터베이스 링크를 생성합니다.

구체화된 보기를 사용하여 데이터 마이그레이션

1. 원본 및 RDS for Oracle 대상 인스턴스 모두에서 동일한 암호로 인증할 수 있는 사용자 계정을 생성합니다. 다음 예에서는 이름이 dblink_user인 사용자를 생성합니다.

```
CREATE USER dblink_user IDENTIFIED BY my-password
  DEFAULT TABLESPACE users
  TEMPORARY TABLESPACE temp;

GRANT CREATE SESSION TO dblink_user;

GRANT SELECT ANY TABLE TO dblink_user;

GRANT SELECT ANY DICTIONARY TO dblink_user;
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

2. 새로 생성된 사용자를 사용하여 RDS for Oracle 대상 인스턴스에서 원본 인스턴스로의 데이터베이스 링크를 생성합니다.

```
CREATE DATABASE LINK remote_site
  CONNECT TO dblink_user IDENTIFIED BY my-password
  USING '(description=(address=(protocol=tcp) (host=my-host)
    (port=my-listener-port)) (connect_data=(sid=my-source-db-sid)))';
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

3. 링크를 테스트합니다.

```
SELECT * FROM V$INSTANCE@remote_site;
```

4. 원본 인스턴스에서 주 키와 구체화 보기 로그를 사용하여 샘플 테이블을 생성합니다.

```
CREATE TABLE customer_0 TABLESPACE users
  AS (SELECT ROWNUM id, o.*
      FROM   ALL_OBJECTS o, ALL_OBJECTS x
      WHERE  ROWNUM <= 1000000);

ALTER TABLE customer_0 ADD CONSTRAINT pk_customer_0 PRIMARY KEY (id) USING INDEX;

CREATE MATERIALIZED VIEW LOG ON customer_0;
```

5. 대상 RDS for Oracle DB 인스턴스에서 구체화된 보기를 생성합니다.

```
CREATE MATERIALIZED VIEW customer_0
  BUILD IMMEDIATE REFRESH FAST
  AS (SELECT *
      FROM   cust_dba.customer_0@remote_site);
```

6. 대상 RDS for Oracle DB 인스턴스에서 구체화된 보기를 새로 고칩니다.

```
EXEC DBMS_MVIEW.REFRESH('CUSTOMER_0', 'f');
```

7. 구체화된 보기를 삭제하고 PRESERVE TABLE 절을 포함하여 구체화된 보기 컨테이너 테이블과 그 내용을 보관합니다.

```
DROP MATERIALIZED VIEW customer_0 PRESERVE TABLE;
```

보관한 테이블에는 끊긴 구체화된 보기와 같은 이름이 있습니다.

Amazon RDS의 Oracle의 읽기 전용 복제본 작업

Oracle DB 인스턴스 간 복제를 구성하기 위해 복제본 데이터베이스를 생성할 수 있습니다. Amazon RDS 읽기 전용 복제본의 개요는 Amazon RDS 읽기 전용 복제본의 개요 섹션을 참조하세요. Oracle 복제본과 다른 DB 엔진 간의 차이점에 대한 요약은 DB 엔진별 읽기 전용 복제본의 차이점 섹션을 참조하세요.

주제

- RDS for Oracle 복제본 개요
- RDS for Oracle 복제본에 대한 요구 사항 및 고려 사항
- Oracle 복제본 생성 준비
- 탑재된 모드에서 RDS for Oracle 복제본 생성
- RDS for Oracle 복제본 모드 수정
- RDS for Oracle 복제본 백업 작업
- Oracle Data Guard 전환 수행
- RDS for Oracle 복제본 문제 해결

RDS for Oracle 복제본 개요

Oracle 복제본 데이터베이스는 프라이머리 데이터베이스의 실제 복사본입니다. 읽기 전용 모드의 Oracle 복제본을 읽기 전용 복제본이라고 합니다. 탑재된 모드의 Oracle 복제본을 탑재된 복제본이라고 합니다. Oracle 데이터베이스는 복제본에서의 쓰기를 허용하지 않지만, 복제본을 승격하여 쓰기 가능한 상태로 만들 수 있습니다. 승격된 읽기 전용 복제본에는 승격 요청이 이루어진 시점까지 복제된 데이터가 있습니다.

다음 동영상은 RDS for Oracle 재해 복구에 대한 유용한 개요를 제공합니다.

자세한 내용은 블로그 게시물 Amazon RDS for Oracle 교차 리전 자동 백업으로 관리형 재해 복구 - 1부 및 Amazon RDS for Oracle 교차 리전 자동 백업으로 관리형 재해 복구 - 2부를 참조하세요.

주제

- 읽기 전용 복제본 및 탑재된 복제본
- CDB 읽기 전용 복제본
- 아카이브된 다시 실행 로그 보존
- Oracle 복제 중 중단

읽기 전용 복제본 및 탑재된 복제본

Oracle 복제본을 생성하거나 수정할 때 다음 모드 중 하나로 배치할 수 있습니다.

읽기 전용

이 값이 기본값입니다. Active Data Guard는 원본 데이터베이스의 변경 사항을 모든 읽기 전용 복제본 데이터베이스로 전송하고 적용합니다.

원본 DB 인스턴스 하나에서 최대 5개까지 읽기 전용 복제본을 생성할 수 있습니다. 모든 DB 엔진에 적용되는 읽기 전용 복제본에 대한 일반적인 정보는 DB 인스턴스 읽기 전용 복제본 작업 단원을 참조하세요. RDS for Oracle에 대한 자세한 내용은 Oracle 설명서의 Oracle Data Guard concepts and administration(Oracle Data Guard 개념 및 관리)를 참조하세요.

탑재

이 경우 복제는 Oracle Data Guard를 사용하지만 복제본 데이터베이스는 사용자 연결을 허용하지 않습니다. 탑재된 복제본의 주된 용도는 리전 간 재해 복구입니다.

탑재된 복제본은 읽기 전용 워크로드를 처리할 수 없습니다. 탑재된 복제본은 아카이브된 로그 보존 정책에 관계없이 아카이브된 다시 실행 로그 파일을 적용한 후 삭제합니다.

동일한 원본 DB 인스턴스에 대해 탑재된 DB 복제본과 읽기 전용 DB 복제본을 조합하여 생성할 수 있습니다. 읽기 전용 복제본을 탑재된 모드로 변경하거나 탑재된 복제본을 읽기 전용 모드로 변경할 수 있습니다. 두 경우 모두 Oracle 데이터베이스는 아카이브된 로그 보존 설정을 유지합니다.

CDB 읽기 전용 복제본

RDS for Oracle이 단일 테넌트 구성에서만 Oracle Database 19c 및 21c CDB에 대해 Data Guard 읽기 전용 복제본을 지원합니다. 비CDB에서와 마찬가지로 CDB에서도 읽기 전용 복제본을 생성, 관리 및 승격할 수 있습니다. 마운트된 복제본도 지원됩니다. 이점은 다음과 같습니다.

- 관리형 재해 복구, 고가용성, 복제본에 대한 읽기 전용 액세스
- 다른 AWS 리전에서 읽기 전용 복제본 생성 가능
- 기존 RDS 읽기 전용 복제본 API와의 통합: CreateDBInstanceReadReplica, PromoteReadReplica, SwitchoverReadReplica

이 기능을 사용하려면 복제본과 기본 DB 인스턴스 모두에 대해 Active Data Guard 라이선스와 Oracle Database Enterprise Edition 라이선스가 필요합니다. CDB 아키텍처 사용과 관련된 추가 비용은 없습니다. DB 인스턴스에 대한 비용만 지불하면 됩니다.

CDB 아키텍처의 단일 테넌트 및 다중 테넌트 구성에 대한 자세한 내용은 RDS for Oracle CDB 개요 섹션을 참조하세요.

아카이브된 다시 실행 로그 보존

기본 DB 인스턴스에 리전 간 읽기 전용 복제본이 없는 경우 Amazon RDS for Oracle이 원본 DB 인스턴스에 대한 최소 2시간의 아카이브된 다시 실행 로그를 유지합니다. 이는 `rdsadmin.rdsadmin_util.set_configuration`에서 `archivelog retention hours`에 대한 설정과 무관하게 true입니다.

RDS는 2시간 후 또는 아카이브 로그 보존 시간 설정이 지난 후 중 더 긴 시간이 경과한 후에 소스 DB 인스턴스에서 로그를 제거합니다. RDS는 아카이브 로그 보존 시간이 데이터베이스에 성공적으로 적용된 경우에만 해당 설정이 경과된 후 읽기 전용 복제본에서 로그를 제거합니다.

경우에 따라 기본 DB 인스턴스에 하나 이상의 리전 간 읽기 전용 복제본이 있을 수 있습니다. 이 경우 Amazon RDS for Oracle은 원본 DB 인스턴스에 대한 트랜잭션 로그가 전송되어 모든 리전 간 읽기 전용 복제본에 적용될 때까지 이 로그를 유지합니다. `rdsadmin.rdsadmin_util.set_configuration`에 대한 내용은 보관된 다시 실행 로그 보존을 참조하세요.

Oracle 복제 중 중단

읽기 전용 복제본을 생성하면 Amazon RDS가 원본 DB 인스턴스의 DB 스냅샷을 캡처하고 복제를 시작합니다. DB 스냅샷 작업이 시작될 때 원본 DB 인스턴스에서 매우 짧은 I/O 중단이 발생합니다. 이러한 I/O 중단은 일반적으로 1초 정도 지속됩니다. 원본 DB 인스턴스가 다중 AZ 배포인 경우에는 I/O 중단을 방지할 수 있습니다. 이 경우에는 보조 DB 인스턴스에서 스냅샷을 생성하기 때문입니다.

DB 스냅샷은 Oracle 복제본이 됩니다. Amazon RDS는 서비스 중단 없이 원본 데이터베이스 및 복제본에 필요한 파라미터와 권한을 설정합니다. 마찬가지로 복제본을 삭제해도 중단이 발생하지 않습니다.

RDS for Oracle 복제본에 대한 요구 사항 및 고려 사항

Oracle 복제본을 생성하기 전에 다음 요구 사항 및 고려 사항을 숙지하세요.

주제

- Oracle 복제본에 대한 버전 및 라이선스 요구 사항
- RDS for Oracle 복제본에 대한 옵션 그룹 제한 사항
- RDS for Oracle 복제본에 대한 Backup 및 복구 고려 사항

- RDS for Oracle 복제본에 대한 Oracle Data Guard 요구 사항 및 제한 사항
- RDS for Oracle 복제본에 대한 기타 고려 사항

Oracle 복제본에 대한 버전 및 라이선스 요구 사항

RDS for Oracle 복제본을 생성하기 전에 다음 사항을 고려하세요.

- 복제본이 읽기 전용 모드인 경우 Active Data Guard 라이선스가 있는지 확인하세요. 복제본을 탑재된 모드로 배치하는 경우 Active Data Guard 라이선스가 필요하지 않습니다. Oracle DB 엔진만 탑재된 복제본을 지원합니다.
- Oracle 복제본은 Oracle Enterprise Edition(EE)에 대해서만 지원됩니다.
- 비CDB의 Oracle 복제본은 Oracle Database 19c를 실행하는 비CDB 인스턴스를 사용하여 생성된 DB 인스턴스에 대해서만 지원됩니다.
- Oracle 복제본은 두 개 이상의 vCPU가 있는 DB 인스턴스 클래스에서 실행 중인 DB 인스턴스에만 사용할 수 있습니다. 소스 DB 인스턴스는 db.t3.small 인스턴스 클래스를 사용할 수 없습니다.
- 소스 DB 인스턴스의 Oracle DB 엔진 버전과 모든 복제본은 동일해야 합니다. Amazon RDS는 복제본의 유지 관리 기간과 상관없이 소스 DB 인스턴스를 업그레이드한 직후 복제본을 업그레이드합니다. 리전 간 복제본의 메이저 버전 업그레이드의 경우 Amazon RDS에서 자동으로 다음을 수행합니다.
  - 대상 버전에 대한 옵션 그룹을 생성합니다.
  - 모든 옵션과 옵션 설정을 원래 옵션 그룹에서 새 옵션 그룹으로 복사합니다.
  - 업그레이드된 리전 간 복제본을 새 옵션 그룹과 연결합니다.

  DB 엔진 버전 업그레이드에 대한 자세한 내용은 RDS for Oracle DB 엔진 업그레이드 단원을 참조하세요.

RDS for Oracle 복제본에 대한 옵션 그룹 제한 사항

RDS for Oracle 복제본의 옵션 그룹으로 작업할 때는 다음 사항을 고려하세요.

- 소스와 복제본이 동일한 AWS 리전에 있는 경우 소스 DB 인스턴스 옵션 그룹과 다른 복제본 옵션 그룹을 사용할 수 없습니다.

  원본 옵션 그룹 또는 원본 옵션 그룹 멤버십에 대한 수정 사항은 Oracle 복제본으로 전파됩니다. 이러한 변경 사항은 복제본의 유지 관리 기간과 상관없이 원본 DB 인스턴스에 적용된 직후 복제본에 적용됩니다. 옵션 그룹에 대한 자세한 내용은 옵션 그룹 작업 단원을 참조하십시오.

- 복제본에 대해 자동으로 생성되는 전용 옵션 그룹에서 RDS for Oracle 교차 리전 복제본을 제거할 수 없습니다.
- RDS for Oracle 교차 리전 복제본에 대한 전용 옵션 그룹을 다른 DB 인스턴스에 추가할 수 없습니다.
- 다음 옵션을 제외하고 RDS for Oracle 리전 간 복제본의 전용 옵션 그룹에서 비복제 옵션을 추가하거나 제거할 수 없습니다.
  - `NATIVE_NETWORK_ENCRYPTION`
  - `OEM`
  - `OEM_AGENT`
  - `SSL`

  RDS for Oracle 리전 간 복제본에 다른 옵션을 추가하려면 원본 DB 인스턴스의 옵션 그룹에 추가하세요. 이 옵션은 모든 원본 DB 인스턴스의 복제본에도 설치됩니다. 라이선스가 있는 옵션의 경우 복제본에 대한 라이선스가 충분해야 합니다.

  RDS for Oracle 리전 간 복제본을 승격하면 승격된 복제본은 옵션 관리를 포함하여 다른 Oracle DB 인스턴스와 동일하게 동작합니다. 원본 DB 인스턴스를 삭제하여 복제본을 명시적 또는 암시적으로 승격할 수 있습니다.

  옵션 그룹에 대한 자세한 내용은 옵션 그룹 작업 단원을 참조하세요.
- RDS for Oracle 리전 간 복제본에는 `EFS_INTEGRATION` 옵션을 추가할 수 없습니다.

RDS for Oracle 복제본에 대한 Backup 및 복구 고려 사항

RDS for Oracle 복제본을 생성하기 전에 다음 사항을 고려하세요.

- RDS for Oracle 복제본의 스냅샷을 생성하거나 자동 백업을 설정하려면 백업 보존 기간을 수동으로 설정해야 합니다. 자동 백업은 기본적으로 켜져 있지 않습니다.
- 복제본 백업을 복원할 때는 백업이 완료된 시간이 아니라 데이터베이스 시간으로 복원됩니다. 이 데이터베이스 시간이란 백업에 있는 데이터의 가장 최근에 적용된 트랜잭션 시간입니다. 복제본이 기본 복제본보다 몇 분 또는 몇 시간 정도 지연될 수 있기 때문에 이러한 차이는 상당합니다.

  차이점을 찾으려면 `describe-db-snapshots` 명령을 사용하세요. 복제본 백업의 데이터베이스 시간인 `snapshotDatabaseTime`와 기본 데이터베이스에서 가장 최근에 적용된 트랜잭션인 `OriginalSnapshotCreateTime` 필드를 비교합니다.

RDS for Oracle 복제본에 대한 Oracle Data Guard 요구 사항 및 제한 사항

RDS Custom for Oracle 복제를 생성하기 전에 다음과 같은 요구 사항과 제한 사항에 유의하세요.

- 기본 DB 인스턴스가 멀티테넌트 아키텍처의 단일 테넌트 구성을 사용하는 경우 다음을 고려하세요.
  - Enterprise Edition으로 Oracle Database 19c 이상을 사용해야 합니다.
  - 기본 CDB 인스턴스는 `ACTIVE` 수명 주기에 속해야 합니다.
  - 비CDB 인스턴스를 CDB 인스턴스로 변환하고 해당 복제본을 동일한 작업에서 변환할 수 없습니다. 대신 비CDB 복제본을 삭제하고 기본 DB 인스턴스를 CDB로 변환한 다음, 새 복제본을 생성하세요.
- 기본 인스턴스의 로그인 트리거는 `RDS_DATAGUARD` 사용자와 `AUTHENTICATED_IDENTITY` 값이 `RDS_DATAGUARD` 또는 `rdsdb`인 모든 사용자에 대한 액세스를 허용해야 합니다. 또한 트리거는 `RDS_DATAGUARD` 사용자의 현재 스키마를 설정하지 않아야 합니다.
- Data Guard 브로커 프로세스에서 연결을 차단하지 않으려면 제한된 세션을 활성화하지 마십시오. 제한된 세션에 대한 자세한 내용은 제한 세션 활성화 및 비활성화 단원을 참조하십시오.

RDS for Oracle 복제본에 대한 기타 고려 사항

RDS for Oracle 복제본을 생성하기 전에 다음 사항을 고려하세요.

- DB 인스턴스가 하나 이상의 리전 간 복제본의 원본인 경우 원본 DB는 모든 리전 간 복제본에 적용될 때까지 아카이브된 재실행 로그 파일을 유지합니다. 아카이브된 다시 실행 로그 때문에 스토리지 소비가 증가할 수 있습니다.
- RDS 자동화를 방해하지 않으려면 특정 사용자가 기본 및 복제본 데이터베이스에 로그온할 수 있도록 시스템 트리거가 허용해야 합니다. 시스템 트리거에는 DDL, 로그온 및 데이터베이스 역할 트리거가 포함되어 있습니다. 다음 예제 코드에 나열된 사용자를 제외하도록 트리거에 코드를 추가하는 것이 좋습니다.

```
-- Determine who the user is
SELECT SYS_CONTEXT('USERENV','AUTHENTICATED_IDENTITY') INTO CURRENT_USER FROM DUAL;
-- The following users should always be able to login to either the Primary or
 Replica
IF CURRENT_USER IN ('master_user', 'SYS', 'SYSTEM', 'RDS_DATAGUARD', 'rdsdb') THEN
RETURN;
END IF;
```

- 블록 변경 내용 추적은 읽기 전용 복제본에 대해 지원되지만 마운트된 복제본에는 지원되지 않습니다. 마운트된 복제본을 읽기 전용 복제본으로 변경한 다음 블록 변경 내용 추적을 활성화할 수 있습니다. 자세한 내용은 블록 변경 추적 활성화 및 비활성화 섹션을 참조하세요.
- 소스 데이터베이스가 Secrets Manager를 사용하여 마스터 사용자 자격 증명을 관리하는 경우 Oracle 읽기 전용 복제본을 만들 수 없습니다.

Oracle 복제본 생성 준비

복제본 사용을 시작하려면 먼저 다음 작업을 수행해야 합니다.

주제

- 자동 백업 활성화
- 강제 로깅 모드 활성화
- 로깅 구성 변경
- MAX_STRING_SIZE 파라미터 설정
- 컴퓨팅 및 스토리지 리소스 계획

자동 백업 활성화

임의의 DB 인스턴스를 원본 DB 인스턴스로 사용하려면 원본 DB 인스턴스의 자동 백업을 활성화해야 합니다. 이 절차를 수행하는 방법에 대한 자세한 내용은 자동 백업 활성화 단원을 참조하세요.

강제 로깅 모드 활성화

강제 로깅 모드를 활성화하는 것이 좋습니다. 강제 로깅 모드에서 Oracle 데이터베이스는 `NOLOGGING`이 데이터 정의 언어(DDL) 문과 함께 사용되더라도 다시 실행 레코드를 기록합니다.

강제 로깅 모드를 활성화하려면

1. SQL Developer와 같은 클라이언트 도구를 사용하여 Oracle 데이터베이스에 로그인합니다.
2. 다음 절차를 실행하여 강제 로깅 모드를 활성화합니다.

```
exec rdsadmin.rdsadmin_util.force_logging(p_enable => true);
```

이 프로시저에 대한 자세한 내용은 강제 로깅 설정 단원을 참조하십시오.

로깅 구성 변경

크기가 m인 n개의 온라인 다시 실행 로그의 경우 RDS는 기본 DB 인스턴스와 모든 복제본에 크기가 m인 n+1 대기 로그를 자동으로 생성합니다. 기본 복제본의 로깅 구성을 변경할 때마다 변경 내용이 복제본에 자동으로 전파됩니다.

로깅 구성을 변경하는 경우 다음 지침을 고려하세요.

- DB 인스턴스를 복제본의 소스로 만들기 전에 변경을 완료하는 것이 좋습니다. RDS for Oracle은 인스턴스가 소스가 된 후의 인스턴스 업데이트도 지원합니다.
- 기본 DB 인스턴스의 로깅 구성을 변경하기 전에 각 복제본에 새 구성을 수용할 수 있는 충분한 스토리지가 있는지 확인하세요.

Amazon RDS 프로시저 `rdsadmin.rdsadmin_util.add_logfile` 및 `rdsadmin.rdsadmin_util.drop_logfile`을 사용하여 DB 인스턴스의 로깅 구성을 수정할 수 있습니다. 자세한 내용은 온라인 다시 실행 로그 추가 및 온라인 다시 실행 로그 드롭 단원을 참조하세요.

MAX_STRING_SIZE 파라미터 설정

Oracle 복제본을 생성하기 전에 원본 DB 인스턴스와 복제본의 `MAX_STRING_SIZE` 파라미터 설정이 동일한지 확인하세요. 동일한 파라미터 그룹과 연결하여 이 작업을 수행할 수 있습니다. 원본 및 복제본에 대해 다른 파라미터 그룹이 있는 경우 `MAX_STRING_SIZE`를 동일한 값으로 설정할 수 있습니다. 이 파라미터에 대한 자세한 내용은 신규 DB 인스턴스에 확장 데이터 유형 활성화 단원을 참조하세요.

컴퓨팅 및 스토리지 리소스 계획

원본 DB 인스턴스와 해당 복제본이 운영 로드에 맞게 컴퓨팅 및 스토리지 면에서 제대로 크기가 조정되었는지 확인합니다. 복제본이 컴퓨팅, 네트워크 또는 스토리지 리소스 용량에 도달하면 복제본은 해당 소스에서 변경 사항을 수신하거나 적용하는 것을 중지합니다. Amazon RDS for Oracle은 개입을 통해 소스 DB 인스턴스와 해당 복제본 간의 긴 복제본 지연 시간을 줄이지 않습니다. 복제본의 스토리지 및 CPU 리소스를 해당 원본 및 다른 복제본과 별도로 수정할 수 있습니다.

탑재된 모드에서 RDS for Oracle 복제본 생성

기본적으로 Oracle 복제본은 읽기 전용입니다. 탑재된 모드에서 복제본을 생성하려면 콘솔, AWS CLI 또는 RDS API를 사용합니다.

콘솔

소스 Oracle DB 인스턴스에서 탑재된 복제본을 생성하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 탑재된 복제본의 소스로 사용할 Oracle DB 인스턴스를 선택합니다.
4. 작업에서 읽기 전용 복제본 만들기를 선택합니다.
5. 복제본 모드에서 탑재를 선택합니다.
6. 사용하려는 설정을 선택합니다. DB 인스턴스 식별자에 읽기 전용 복제본의 이름을 입력합니다. 필요에 따라 다른 설정을 조정합니다.
7. 리전의 경우 탑재된 복제본을 시작할 리전을 선택합니다.
8. 인스턴스 크기 및 스토리지 유형을 선택합니다. 읽기 전용 복제본의 원본 DB 인스턴스와 동일한 DB 인스턴스 클래스와 스토리지 유형을 사용하는 것이 좋습니다.
9. 탑재된 복제본에 대한 장애 조치 지원을 위해 다른 가용 영역에 대기 복제본을 생성하려면 다중 AZ 배포에서 대시 인스턴스 생성을 선택합니다. 탑재된 복제본을 다중 AZ DB 인스턴스로 생성하는 작업은 원본 데이터베이스가 다중 AZ DB 인스턴스인지 여부와는 무관합니다.
10. 사용하고자 하는 기타 설정을 선택합니다.
11. 복제본 생성을 선택합니다.

데이터베이스 페이지에서 탑재된 복제본에는 복제본 역할이 있습니다.

AWS CLI

탑재된 모드에서 Oracle 복제본을 생성하려면 `--replica-mode` 명령 create-db-instance-read-replica에서 `mounted`를 AWS CLI로 설정합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance-read-replica \
    --db-instance-identifier myreadreplica \
    --source-db-instance-identifier mydbinstance \
    --replica-mode mounted
```

Windows의 경우:

```
aws rds create-db-instance-read-replica ^
    --db-instance-identifier myreadreplica ^
    --source-db-instance-identifier mydbinstance ^
    --replica-mode mounted
```

읽기 전용 복제본을 탑재된 상태로 변경하려면 `--replica-mode` 명령 modify-db-instance에서 `mounted`를 AWS CLI로 설정합니다. 탑재된 복제본을 읽기 전용 모드로 배치하려면 `--replica-mode`를 `open-read-only`로 설정합니다.

RDS API

탑재된 모드에서 Oracle 복제본을 생성하려면 RDS API 작업 CreateDBInstanceReadReplica에서 `ReplicaMode=mounted`를 지정합니다.

RDS for Oracle 복제본 모드 수정

기존 복제본의 복제본 모드를 변경하려면 콘솔, AWS CLI 또는 RDS API를 사용합니다. 탑재된 모드로 변경하면 복제본이 모든 활성 연결을 해제합니다. 읽기 전용 모드로 변경하면 Amazon RDS는 Active Data Guard를 초기화합니다.

변경 작업은 몇 분 정도 걸릴 수 있습니다. 작업 중에는 DB 인스턴스 상태가 수정 중으로 변경됩니다. 상태 변경에 대한 자세한 내용은 Amazon RDS DB 인스턴스 상태 보기 단원을 참조하세요.

콘솔

Oracle 복제본의 복제본 모드를 탑재에서 읽기 전용으로 변경하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 탑재된 복제본 데이터베이스를 선택합니다.
4. 수정을 선택합니다.
5. 복제본 모드의 경우 읽기 전용을 선택합니다.
6. 사용하려는 기타 설정을 선택합니다.
7. [Continue]를 선택합니다.

8. 수정 사항 예약에 대해 즉시 적용을 선택합니다.
9. DB 인스턴스 수정을 선택합니다.

AWS CLI

읽기 전용 복제본을 탑재된 모드로 변경하려면 `--replica-mode` 명령 modify-db-instance에서 `mounted`를 AWS CLI로 설정합니다. 탑재된 복제본을 읽기 전용 모드로 변경하려면 `--replica-mode`를 `open-read-only`로 설정합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier myreadreplica \
    --replica-mode mode
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier myreadreplica ^
    --replica-mode mode
```

RDS API

읽기 전용 복제본을 탑재된 모드로 변경하려면 ModifyDBInstance에서 `ReplicaMode=mounted`를 설정합니다. 탑재된 복제본을 읽기 전용 모드로 변경하려면 `ReplicaMode=read-only`를 설정합니다.

RDS for Oracle 복제본 백업 작업

RDS for Oracle 복제본의 백업을 생성하고 복원할 수 있습니다. 자동 백업과 수동 스냅샷이 모두 지원됩니다. 자세한 내용은 데이터 백업, 복원 및 내보내기 섹션을 참조하세요. 다음 섹션에서는 기본 복제본과 RDS for Oracle 복제본의 백업 관리 간의 주요 차이점을 설명합니다.

RDS for Oracle 복제본 백업 켜기

Oracle 복제본에서는 기본적으로 자동 백업이 켜져 있지 않습니다. 백업 보존 기간을 0이 아닌 양수 값으로 설정하여 자동 백업을 켭니다.

콘솔

자동 백업을 즉시 활성화하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 다음 수정하려는 DB 인스턴스 또는 다중 AZ DB 클러스터를 선택합니다.
3. 수정을 선택합니다.
4. 백업 보존 기간으로 0이 아닌 양수 값(예: 3일)을 선택합니다.
5. [Continue]를 선택합니다.
6. 즉시 적용을 선택합니다.
7. DB 인스턴스 수정 또는 클러스터 수정을 선택하여 변경 내용을 저장하고 자동 백업을 활성화합니다.

AWS CLI

자동 백업을 활성화하려면 AWS CLI modify-db-instance 또는 modify-db-cluster 명령을 사용합니다.

다음 파라미터를 포함합니다.

- `--db-instance-identifier`(또는 다중 AZ DB 클러스터의 경우 `--db-cluster-identifier`)
- `--backup-retention-period`
- `--apply-immediately` 또는 `--no-apply-immediately`

다음 예에서는 백업 보존 기간을 3일로 설정하여 자동 백업을 활성화합니다. 변경이 바로 적용됩니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier my_db_instance  \
    --backup-retention-period 3 \
    --apply-immediately
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier my_db_instance  ^
    --backup-retention-period 3 ^
    --apply-immediately
```

RDS API

자동 백업을 활성화하려면 RDS API ModifyDBInstance 또는 ModifyDBCluster 작업을 다음 필수 파라미터와 함께 사용합니다.

- `DBInstanceIdentifier` 또는 `DBClusterIdentifier`
- `BackupRetentionPeriod`

RDS for Oracle 복제본 백업 복구

기본 인스턴스의 백업을 복원할 수 있는 것처럼 Oracle 복제본 백업을 복원할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- DB 인스턴스 복원
- Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원

복제본 백업을 복원할 때 주요 고려 사항은 복원할 시점을 결정하는 것입니다. 이 데이터베이스 시간이란 백업에 있는 데이터의 가장 최근에 적용된 트랜잭션 시간입니다. 복제본 백업을 복원할 때는 백업이 완료된 시간이 아니라 데이터베이스 시간으로 복원됩니다. RDS for Oracle 복제본이 기본 복제본보다 몇 분 또는 몇 시간 정도 지연될 수 있기 때문에 이러한 차이는 상당합니다. 따라서 복제본 백업의 데이터베이스 시간, 즉 복원하는 시점이 백업 생성 시간보다 훨씬 빠를 수 있습니다.

데이터베이스 시간과 생성 시간의 차이를 구하려면 `describe-db-snapshots` 명령을 사용합니다. 복제본 백업의 데이터베이스 시간인 `SnapshotDatabaseTime`와 기본 데이터베이스에서 가장 최근에 적용된 트랜잭션인 `OriginalSnapshotCreateTime` 필드를 비교합니다. 다음 예에서는 두 날짜 간의 차이 일수를 반환합니다.

```
aws rds describe-db-snapshots \
    --db-instance-identifier my-oracle-replica
    --db-snapshot-identifier my-replica-snapshot
```

```
{
    "DBSnapshots": [
        {
            "DBSnapshotIdentifier": "my-replica-snapshot",
            "DBInstanceIdentifier": "my-oracle-replica",
            "SnapshotDatabaseTime": "2022-07-26T17:49:44Z",
            ...
            "OriginalSnapshotCreateTime": "2021-07-26T19:49:44Z"
        }
    ]
}
```

Oracle Data Guard 전환 수행

전환은 기본 데이터베이스와 대기 데이터베이스 간의 역할 전환입니다. 전환하는 동안 원래 기본 데이터베이스는 대기 역할로 전환되고 원래 대기 데이터베이스는 기본 역할로 전환됩니다.

Oracle Data Guard 환경에서 기본 데이터베이스는 하나 이상의 대기 데이터베이스를 지원합니다. 기본 데이터베이스에서 대기 데이터베이스로 관리형 전환 기반 역할 전환을 수행할 수 있습니다. 전환은 기본 데이터베이스와 대기 데이터베이스 간의 역할 전환입니다. 전환하는 동안 원래 기본 데이터베이스는 대기 역할로 전환되고 원래 대기 데이터베이스는 기본 역할로 전환됩니다.

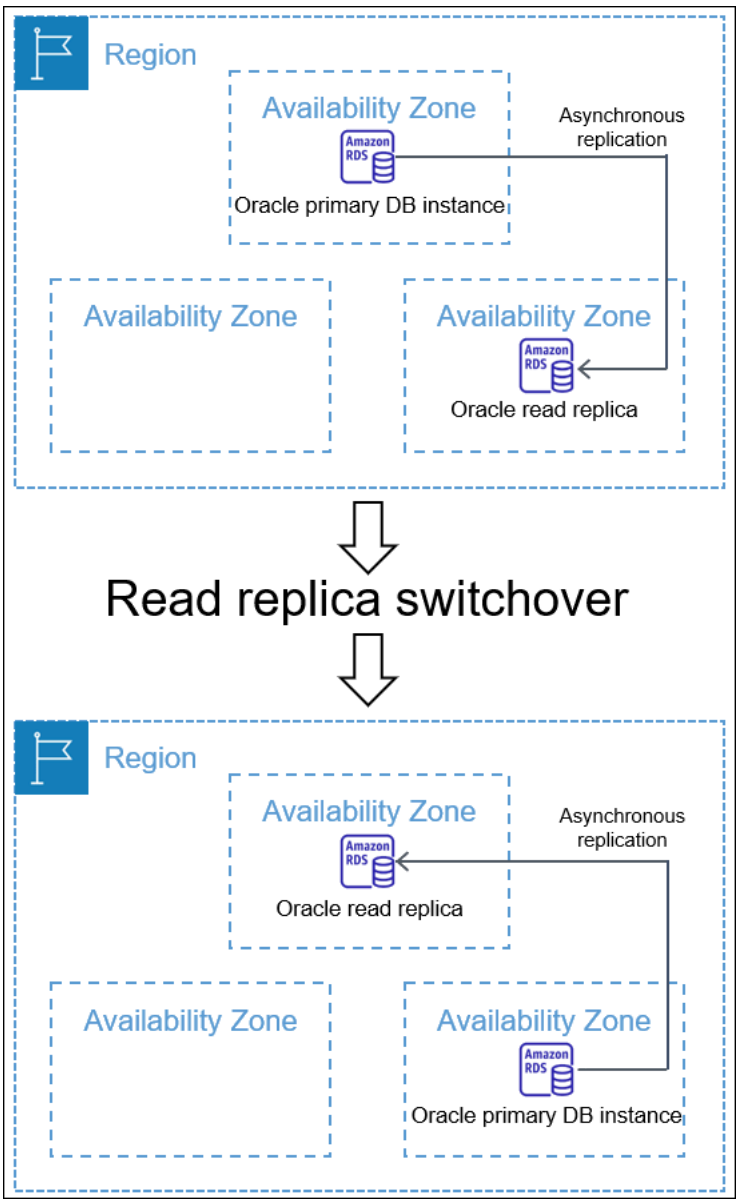
주제

- Oracle Data Guard 전환 개요
- Oracle Guard 전환 요구 사항
- Oracle Data Guard 전환 단계
- Oracle Data Guard 전환 모니터링

Oracle Data Guard 전환 개요

mazon RDS는 Oracle Database 복제본에 대한 완전관리형 전환 기반 역할 전환을 지원합니다. 마운트되거나 읽기 전용으로 열려 있는 대기 데이터베이스로만 전환을 시작할 수 있습니다.

복제본은 별도의 AWS 리전에 있거나 단일 리전의 다른 가용 영역(AZ)에 있을 수 있습니다. 모든 AWS 리전가 지원됩니다.



전환은 읽기 전용 복제본 승격과는 다릅니다. 전환 시 소스 및 복제본 DB 인스턴스의 역할이 변경됩니다. 승격 시 읽기 전용 복제본은 소스 DB 인스턴스가 되지만, 소스 DB 인스턴스는 복제본이 되지 않습니다. 자세한 내용은 읽기 전용 복제본을 독립 DB 인스턴스로 승격 단원을 참조하십시오.

주제

- Oracle Data Guard 전환 이점
- 지원되는 Oracle Database 버전
- Oracle Data Guard 전환 비용
- Oracle Data Guard 전환 작동 방식

Oracle Data Guard 전환 이점

RDS for Oracle 읽기 전용 복제본과 마찬가지로 관리형 전환은 Oracle Data Guard에 의존합니다. 이 작업은 데이터 손실이 전혀 없도록 설계되었습니다. Amazon RDS는 전환의 다음 측면을 자동화합니다.

- 기본 데이터베이스와 지정된 대기 데이터베이스의 역할을 반대로 하여 새 대기 데이터베이스를 원래 대기 데이터베이스와 동일한 상태(마운트 또는 읽기 전용)로 만듭니다.
- 데이터 일관성을 보장합니다.
- 전환 후에도 복제 구성을 유지합니다.
- 반복 반전을 지원하여 새 대기 데이터베이스를 원래의 기본 역할로 되돌릴 수 있습니다.

지원되는 Oracle Database 버전

Oracle Data Guard 전환은 Oracle Database 19c 이상 버전 릴리스에서 지원됩니다.

Oracle Data Guard 전환 비용

Oracle Data Guard 전환 기능에는 추가 비용이 발생하지 않습니다. Oracle Database Enterprise Edition에는 탑재된 모드에서 대기 데이터베이스에 대한 지원이 포함되어 있습니다. 대기 데이터베이스를 읽기 전용 모드에서 열려면 Oracle Active Data Guard 옵션이 필요합니다.

Oracle Data Guard 전환 작동 방식

Oracle Data Guard 전환은 완전관리형 작업입니다. CLI 명령 `switchover-read-replica`을 발행하여 대기 데이터베이스의 전환을 시작합니다. 그런 다음 Amazon RDS가 복제 구성의 기본 및 대기 역할을 수정합니다.

이 원래 대기 및 원본 기본은 전환 이전에 존재하는 역할입니다. 이 신규 대기 및 신규 기본은 전환 이후에 존재하는 역할입니다. 방관자 복제본은 Oracle Data Guard 환경에서 대기 데이터베이스 역할을 하지만 역할을 전환하지 않는 복제 데이터베이스입니다.

주제

- Oracle Data Guard 전환 단계
- Oracle Data Guard 전환 단계

Oracle Data Guard 전환 단계

전환을 수행하려면 Amazon RDS는 다음 단계를 따라야 합니다.

1. 원래 기본 데이터베이스에서 새 트랜잭션을 차단합니다. 전환 중에 Amazon RDS는 Oracle Data Guard 구성의 모든 데이터베이스에 대한 복제를 중단합니다. 전환 중에는 원래 기본 데이터베이스가 쓰기 요청을 처리할 수 없습니다.
2. 적용되지 않은 트랜잭션을 원래 대기 데이터베이스로 출하하고 적용합니다.
3. 새 대기 데이터베이스를 읽기 전용 또는 마운트된 모드로 재시작합니다. 모드는 전환 전 원래 대기 데이터베이스의 열린 상태에 따라 달라집니다.
4. 새 기본 데이터베이스를 읽기/쓰기 모드에서 엽니다.

Oracle Data Guard 전환 단계

Amazon RDS는 기본 및 대기 데이터베이스의 역할을 전환합니다. 애플리케이션을 다시 연결하고 원하는 다른 구성을 수행하는 것은 사용자의 책임입니다.

주제

- 성공 기준
- 새 기본 데이터베이스에 연결
- 새 기본 데이터베이스 구성

성공 기준

Oracle Data Guard 전환은 원래 대기 데이터베이스가 다음을 수행할 때 성공합니다.

- 새 기본 데이터베이스로서의 역할로 전환
- 재구성 완료

가동 중지 시간을 제한하기 위해 새 기본 데이터베이스는 가능한 한 빨리 활성화됩니다. Amazon RDS는 방관자 복제본을 비동기적으로 구성하기 때문에 이러한 복제본은 원래 기본 데이터베이스 이후에 활성화될 수 있습니다.

새 기본 데이터베이스에 연결

Amazon RDS는 전환 후 현재 데이터베이스 연결을 새 기본 데이터베이스로 전파하지 않습니다. Oracle Data Guard 전환이 완료되면 애플리케이션을 새 기본 데이터베이스에 다시 연결합니다.

새 기본 데이터베이스 구성

새 기본 데이터베이스로 전환하기 위해 Amazon RDS는 원래 대기 데이터베이스의 모드를 열림 상태로 변경합니다. 역할 변경은 이 데이터베이스의 유일한 변경 사항입니다. Amazon RDS는 다중 AZ 복제본과 같은 기능을 설정하지 않습니다.

다른 옵션을 사용하여 교차 리전 복제본으로 전환하는 경우 새 기본 데이터베이스는 자체 옵션을 유지합니다. Amazon RDS는 원래 기본 데이터베이스의 옵션을 마이그레이션하지 않습니다. 원래의 기본 데이터베이스에 SSL, NNE, OEM 및 OEM_AGENT와 같은 옵션이 있는 경우 Amazon RDS는 이를 새로운 기본 데이터베이스로 전파하지 않습니다.

Oracle Guard 전환 요구 사항

Oracle Data Guard 전환을 시작하기 전에 복제 환경이 다음 요구 사항을 충족하는지 확인하세요.

- 원래 대기 데이터베이스가 마운트되거나 읽기 전용으로 열려 있습니다.
- 자동 백업은 원래 대기 데이터베이스에서 활성화됩니다.
- 원래 기본 데이터베이스와 원래 대기 데이터베이스가 `available` 상태입니다.
- 원래 기본 데이터베이스와 원래 대기 데이터베이스에 `required`, `next window` 또는 `in progress` 상태의 보류 중인 유지 관리 작업이 없습니다. 이러한 상태의 작업은 전환을 차단합니다. 보류 중인 유지 관리 업데이트의 상태를 확인하는 방법은 보류 중인 유지 관리 업데이트 보기 섹션을 참조하세요.

  `available` 상태의 보류 중인 유지 관리 작업은 전환을 차단하지 않습니다. RDS for Oracle은 `available` 상태의 운영 체제(OS) 업데이트를 자주 릴리스합니다. 이러한 보류 중인 OS 업데이트는 다음 유지 관리 기간으로 예약하여 `next window` 상태가 되지 않는 한 전환을 차단하지 않습니다.

  > **Note**
  >
  > 전환을 실행할 수 있도록 예약된 유지 관리 작업을 연기하려면 RDS 콘솔에서 작업을 선택한 다음 업그레이드 보류를 선택합니다. 보류 중인 유지 관리 작업을 적용하거나 유지 관리 기간을 전환 전 간격으로 이동하여 전환이 차단되는 것을 방지할 수도 있습니다. 자세한 내용은 re:Post 문서 How to remove RDS pending maintenance items를 참조하세요.

- 원래 대기 데이터베이스가 복제 중 상태입니다.

- 기본 데이터베이스 또는 대기 데이터베이스가 현재 전환 수명 주기 내에 있을 때는 전환을 시작하려고 하지 않습니다. 전환 후 복제본 데이터베이스를 재구성하는 경우 Amazon RDS에서는 다른 전환을 시작할 수 없습니다.

  > **Note**
  >
  > 방관자 복제본은 전환 대상이 아닌 Oracle Data Guard 구성의 복제본입니다. 방관자 복제본은 전환 중에 어떤 상태로든 존재할 수 있습니다.

- 원래 대기 데이터베이스에는 원래 기본 데이터베이스에 원하는 만큼 가까운 구성이 있습니다. 원래의 기본 데이터베이스와 원래 대기 데이터베이스의 옵션이 서로 다른 시나리오를 가정해 보겠습니다. 전환이 완료된 후 Amazon RDS는 원래 기본 데이터베이스와 동일한 옵션을 갖도록 새 기본 데이터베이스를 자동으로 재구성하지 않습니다.
- 전환을 시작하기 전에 원하는 다중 AZ 배포를 구성합니다. Amazon RDS는 전환의 일환으로 다중 AZ를 관리하지 않습니다. 다중 AZ 배포는 그대로 유지됩니다.

  db_maz가 다중 AZ 배포의 기본 데이터베이스이고 db_sz가 단일 AZ 복제본이라고 가정해 보겠습니다. db_maz에서 db_sz로의 전환을 시작합니다. 다음으로 db_maz가 다중 AZ 복제본 데이터베이스이고 db_sz가 단일 AZ 기본 데이터베이스입니다. 새 기본 데이터베이스는 이제 다중 AZ 배포로 보호되지 않습니다.
- 기본 데이터베이스는 리전 간 전환에 대비하여 복제 구성 외부의 DB 인스턴스와 동일한 옵션 그룹을 사용하지 않습니다. 리전 간 전환이 성공하려면 현재 기본 데이터베이스와 해당 읽기 전용 복제본이 현재 기본 데이터베이스의 옵션 그룹을 사용하는 유일한 DB 인스턴스여야 합니다. 그렇지 않으면 Amazon RDS에서 전환을 차단합니다.

Oracle Data Guard 전환 단계

RDS for Oracle용 읽기 전용 복제본을 기본 역할로 전환하고 이전 기본 DB 인스턴스를 복제본 역할로 전환할 수 있습니다.

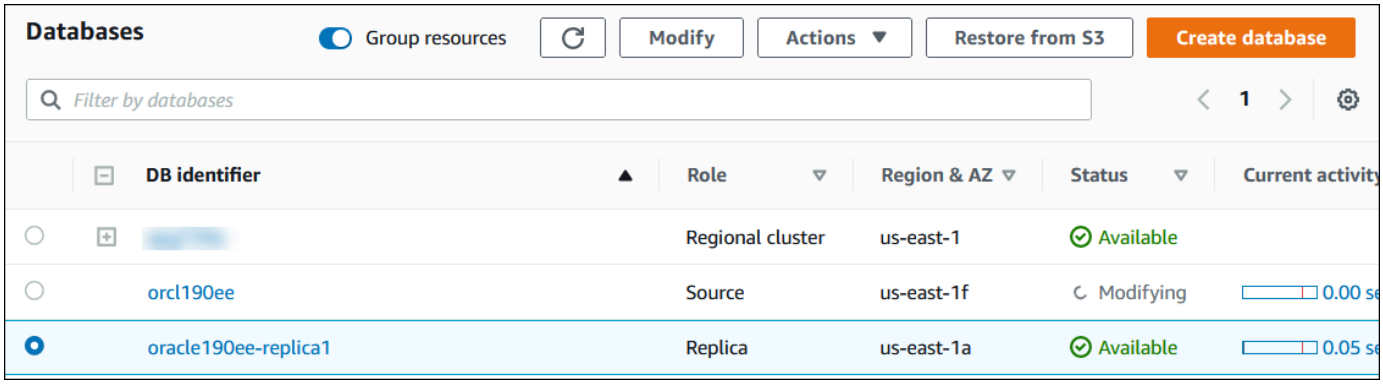
콘솔

Oracle 읽기 전용 복제본을 기본 DB 역할로 전환하려면

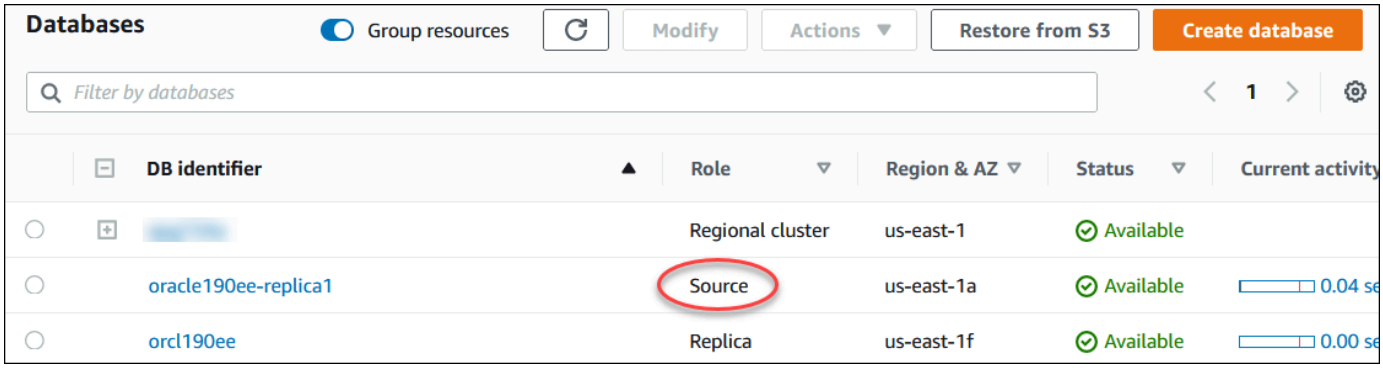
1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 콘솔에서 데이터베이스를 선택합니다.

데이터베이스 창이 표시됩니다. 각 읽기 전용 복제본은 역할 열에 복제본이라고 표시됩니다.

3. 기본 역할로 전환하려는 읽기 전용 복제본을 선택합니다.
4. 작업에서는 복제본 전환을 선택합니다.
5. 동의를 선택합니다. 그런 다음 복제본 전환을 선택합니다.
6. 데이터베이스 페이지에서 전환 진행 상황을 모니터링합니다.



전환이 완료되면 전환 대상의 역할이 복제본에서 소스로 바뀝니다.



AWS CLI

Oracle 복제본을 기본 DB 역할로 전환하려면 AWS CLI `switchover-read-replica` 명령을 사용합니다. 다음 예에서는 *`replica-to-be-made-primary`*라는 Oracle 복제본을 새 기본 데이터베이스로 만듭니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds switchover-read-replica \
    --db-instance-identifier replica-to-be-made-primary
```

Windows의 경우:

```
aws rds switchover-read-replica ^
    --db-instance-identifier replica-to-be-made-primary
```

RDS API

Oracle 복제본을 기본 DB 역할로 전환하려면 필수 파라미터 `DBInstanceIdentifier`를 사용하여 Amazon RDS API `SwitchoverReadReplica` 작업을 호출합니다. 이 파라미터는 기본 DB 역할을 맡으려는 Oracle 복제본의 이름을 지정합니다.

Oracle Data Guard 전환 모니터링

인스턴스 상태를 확인하려면 AWS CLI 명령 `describe-db-instances`를 사용합니다. 다음 명령은 DB 인스턴스 *orcl2*의 상태를 확인합니다. 이 데이터베이스는 전환 전에는 대기 데이터베이스였지만 전환 후에는 새로운 기본 데이터베이스입니다.

```
aws rds describe-db-instances \
    --db-instance-identifier orcl2
```

전환이 성공적으로 완료되었는지 확인하려면 `V$DATABASE.OPEN_MODE`를 쿼리합니다. 새 기본 데이터베이스의 값이 `READ WRITE`인지 확인합니다.

```
SELECT OPEN_MODE FROM V$DATABASE;
```

전환 관련 이벤트를 찾으려면 AWS CLI 명령 `describe-events`를 사용합니다. 다음 예제에서는 *orcl2*에서 이벤트를 찾습니다.

```
aws rds describe-events \
    --source-identifier orcl2 \
    --source-type db-instance
```

RDS for Oracle 복제본 문제 해결

이 섹션에서는 발생할 수 있는 복제 문제 및 해결 방법에 대해 설명합니다.

주제

- Oracle 복제 지연 모니터링
- 트리거 추가 또는 수정 후 Oracle 복제 실패 문제 해결

Oracle 복제 지연 모니터링

Amazon CloudWatch에서 복제 지연 시간을 모니터링하려면 Amazon RDS `ReplicaLag` 지표를 확인합니다. 복제본 지연 시간에 대한 자세한 내용은 읽기 전용 복제본 모니터링 및 Amazon RDS에 대한 Amazon CloudWatch 지표 단원을 참조하세요.

읽기 전용 복제본 지연 시간이 너무 긴 경우 다음 보기를 쿼리합니다.

- `V$ARCHIVED_LOG` – 읽기 전용 복제본에 적용된 커밋을 표시합니다.
- `V$DATAGUARD_STATS` – `ReplicaLag` 지표를 구성하는 구성 요소의 세부 분류를 표시합니다.
- `V$DATAGUARD_STATUS` – Oracle의 내부 복제 프로세스의 로그 출력을 표시합니다.

탑재된 복제본의 경우, 지연 시간이 너무 긴 경우 `V$` 보기를 쿼리할 수 없습니다. 대신 다음을 수행합니다.

- CloudWatch에서 `ReplicaLag` 지표를 확인합니다.
- 콘솔에서 복제본에 대한 알림 로그 파일을 확인합니다. 복구 메시지에서 오류를 찾습니다. 메시지에는 기본 시퀀스 번호와 비교할 수 있는 로그 시퀀스 번호가 포함됩니다. 자세한 정보는 Amazon RDS for Oracle 데이터베이스 로그 파일의 내용을 참조하세요.

트리거 추가 또는 수정 후 Oracle 복제 실패 문제 해결

트리거를 추가하거나 수정하고 나서 복제가 실패하면 트리거가 문제가 될 수 있습니다. RDS에서 복제를 위해 필요한 다음 사용자 계정을 트리거가 제외하는지 확인합니다.

- 관리자 권한을 가진 사용자 계정
- `SYS`
- `SYSTEM`
- `RDS_DATAGUARD`
- `rdsdb`

자세한 정보는 RDS for Oracle 복제본에 대한 기타 고려 사항의 내용을 참조하세요.

Oracle DB 인스턴스에 옵션 추가

Amazon RDS에서 옵션은 추가 기능입니다. 다음에는 Oracle DB 엔진을 실행하는 Amazon RDS 인스턴스에 추가할 수 있는 옵션에 대한 설명이 나와 있습니다.

주제

- Oracle DB 옵션 개요
- Amazon S3 통합
- Oracle Application Express(APEX)
- Amazon EFS 통합
- Oracle Java 가상 머신
- Oracle Enterprise Manager
- Oracle 레이블 보안
- Oracle Locator
- Oracle 기본 네트워크 암호화
- Oracle OLAP
- Oracle 보안 소켓 Layer
- Oracle Spatial
- Oracle SQLT
- Oracle Statspack
- Oracle 시간대
- Oracle 시간대 파일 자동 업그레이드
- Oracle Transparent Data Encryption
- Oracle UTL_MAIL
- Oracle XML DB

Oracle DB 옵션 개요

Oracle 데이터베이스에 대한 옵션을 활성화하려면 먼저 옵션 그룹에 추가한 다음 옵션 그룹을 DB 인스턴스에 연결해야 합니다. 자세한 내용은 옵션 그룹 작업 단원을 참조하십시오.

주제

- Oracle Database 옵션 요약
- 다른 버전에 대해 지원되는 옵션
- 특정 옵션의 메모리 요구 사항

Oracle Database 옵션 요약

Oracle DB 인스턴스에 대해 다음 옵션을 추가할 수 있습니다.

옵션	옵션 ID
Amazon S3 통합	S3_INTEGRATION
Oracle Application Express(APEX)	APEX APEX-DEV
Oracle Enterprise Manager	OEM OEM_AGENT
Oracle Java 가상 머신	JVM
Oracle 레이블 보안	OLS
Oracle Locator	LOCATOR
Oracle 기본 네트워크 암호화	NATIVE_NETWORK_ENCRYPTION
Oracle OLAP	OLAP
Oracle 보안 소켓 Layer	SSL
Oracle Spatial	SPATIAL
Oracle SQLT	SQLT
Oracle Statspack	STATSPACK
Oracle 시간대	Timezone

옵션	옵션 ID
Oracle 시간대 파일 자동 업그레이드	TIMEZONE_FILE_AUTOUPGRADE
Oracle Transparent Data Encryption	TDE
Oracle UTL_MAIL	UTL_MAIL
Oracle XML DB	XMLDB

다른 버전에 대해 지원되는 옵션

RDS for Oracle에서는 지원되지 않는 버전에 옵션을 추가할 수 없습니다. 다른 Oracle Database 버전에서 지원되는 RDS 옵션을 확인하려면 `aws rds describe-option-group-options` 명령을 사용하세요. 다음 예에서는 Oracle Database 19c Enterprise Edition에 대해 지원되는 옵션을 나열합니다.

```
aws rds describe-option-group-options \
    --engine-name oracle-ee \
    --major-engine-version 19
```

자세한 내용은 AWS CLI 명령 참조에서 describe-option-group-options를 참조하세요.

특정 옵션의 메모리 요구 사항

일부 옵션은 DB 인스턴스에서 실행하려면 추가 메모리가 필요합니다. 예를 들어 Oracle Enterprise Manager Database Control은 약 300MB의 RAM을 사용합니다. 작은 DB 인스턴스에 대해 이 옵션을 활성화할 경우 메모리 제약으로 인해 성능 문제가 발생할 수 있습니다. 데이터베이스에서 RAM에 대한 필요성이 줄어들도록 Oracle 파라미터를 조정할 수 있습니다. 또는 더 큰 DB 인스턴스로 확장할 수 있습니다.

Amazon S3 통합

RDS for Oracle DB 인스턴스와 Amazon S3 버킷 사이에서 파일을 전송할 수 있습니다. Oracle Data Pump와 같은 Oracle Database 기능과의 Amazon S3 통합을 사용할 수 있습니다. 예를 들어 Amazon S3의 Data Pump 파일을 DB 인스턴스로 다운로드할 수 있습니다. 자세한 내용은 Amazon RDS의 Oracle로 데이터 가져오기을 참조하세요.

> **Note**
>
> DB 인스턴스와 Amazon S3 버킷은 같은 AWS 리전에 있어야 합니다.

주제

- Amazon S3와 RDS for Oracle 통합을 위한 IAM 권한 구성
- Amazon S3 통합 옵션 추가
- Amazon RDS for Oracle와 Amazon S3 버킷 사이의 파일 전송
- Amazon S3 통합 문제 해결
- Amazon S3 통합 옵션 제거

Amazon S3와 RDS for Oracle 통합을 위한 IAM 권한 구성

RDS for Oracle이 Amazon S3와 통합되게 하려면 DB 인스턴스에 Amazon S3 버킷에 대한 액세스 권한이 있어야 합니다. DB 인스턴스가 사용하는 Amazon VPC가 Amazon S3 엔드포인트에 대한 액세스 권한을 제공할 필요가 없습니다.

RDS for Oracle은 한 계정의 DB 인스턴스와 다른 계정의 Amazon S3 버킷 간에 파일 전송을 지원합니다. 추가 단계가 필요한 경우 다음 섹션에 설명되어 있습니다.

주제

- 1단계: Amazon RDS 역할에 대한 IAM 정책 생성
- 2단계: (선택 사항) Amazon S3 버킷에 대한 IAM 정책 생성
- 3단계: DB 인스턴스에 대한 IAM 역할 생성 및 정책 연결
- 4단계: IAM 역할을 RDS for Oracle DB 인스턴스와 연결하는 방법

1단계: Amazon RDS 역할에 대한 IAM 정책 생성

이 단계에서는 Amazon S3 버킷과 RDS DB 인스턴스 간에 파일을 전송하는 데 필요한 권한이 있는 AWS Identity and Access Management(IAM) 정책을 만듭니다. 이 단계에서는 이미 S3 버킷을 생성한 것으로 가정합니다.

정책을 생성하기 전에 다음 정보를 기록해 둡니다.

- 버킷의 Amazon 리소스 이름(ARN)
- 버킷에서 SSE-KMS 또는 SSE-S3 암호화를 사용하는 경우 AWS KMS 키에 대한 ARN

> **Note**
>
> RDS for Oracle DB 인스턴스는 SSE-C로 암호화된 Amazon S3 버킷에 액세스할 수 없습니다.

자세한 내용은 Amazon Simple Storage Service 사용 설명서의 서버 측 암호화를 사용하여 데이터 보호를 참조하세요.

콘솔

Amazon S3 버킷에 Amazon RDS 액세스를 허용하는 IAM 정책을 생성하려면

1. IAM 관리 콘솔을 엽니다.
2. 액세스 관리에서 정책을 선택합니다.
3. 정책 생성(Create Policy)을 선택합니다.
4. [Visual editor] 탭에서 [Choose a service]를 선택한 다음 [S3]을 선택합니다.
5. 작업에서 모두 확장을 선택한 후 Amazon S3 버킷에서 Amazon RDS(으)로 파일을 전송하는 데 필요한 버킷 권한 및 객체 권한을 선택합니다. 예를 들어, 다음을 수행합니다.
   - 목록을 확장한 후 ListBucket을 선택합니다.
   - 읽기를 확장한 후 GetObject를 선택합니다.
   - 쓰기(Write)를 확장한 다음 PutObject 및 DeleteObject를 선택합니다.
   - 권한 관리(Permissions management)를 확장한 다음 PutObjectAcl을 선택합니다. 이 권한은 다른 계정에서 소유한 버킷에 파일을 업로드할 계획이고 이 계정이 버킷 콘텐츠를 완전히 제어해야 하는 경우에 필요합니다.

객체 권한(Object permissions)은 Amazon S3 객체 작업에 대한 권한입니다. 이 권한은 버킷 자체가 아닌 버킷의 객체에 부여해야 합니다. 자세한 내용은 객체 작업에 대한 권한을 참조하세요.

6. 리소스를 선택하고 다음을 수행합니다.

   a. 특정 항목을 선택합니다.

   b. bucket의 경우 ARN 추가를 선택합니다. 버킷 ARN을 입력합니다. 버킷 이름은 자동으로 입력됩니다. 그런 다음 추가를 선택합니다.

   c. object 리소스가 표시되면 ARN 추가를 선택하여 리소스를 수동으로 추가하거나 임의를 선택합니다.

   > **Note**
   >
   > Amazon Resource Name(ARN)(Amazon 리소스 이름(ARN))을 더 구체적인 ARN 값으로 설정하여 Amazon RDS에서 Amazon S3 버킷의 특정 파일 또는 폴더에만 액세스하도록 허용할 수 있습니다. Amazon S3에 대한 액세스 정책을 정의하는 방법에 대한 자세한 내용은 Amazon S3 리소스에 대한 액세스 권한 관리 단원을 참조하십시오.

7. (선택 사항) 추가 권한 추가를 선택하여 정책에 리소스를 추가합니다. 예를 들어, 다음을 수행합니다.

   a. 버킷이 사용자 지정 KMS 키로 암호화되면 서비스에 대해 KMS를 선택합니다.

   b. 수동 작업의 경우 다음을 선택합니다.

      - 암호화
      - ReEncrypt 시작 지점 및 ReEncrypt 종료 지점
      - Decrypt
      - DescribeKey
      - GenerateDataKey

   c. 리소스에서 특정 항목을 선택합니다.

   d. key의 경우 ARN 추가를 선택합니다. 사용자 지정 키의 ARN을 리소스로 입력한 다음, 추가를 선택합니다.

      자세한 내용은 Amazon Simple Storage Service 사용 설명서의 AWS Key Management Service(SSE-KMS)에 저장된 KMS 키로 서버 측 암호화를 사용하여 데이터 보호를 참조하세요.

e. Amazon RDS가 다른 버킷에 액세스하게 하려면 이러한 버킷에 대한 ARN을 추가합니다. 또는 Amazon S3의 모든 버킷 및 객체에 대한 액세스 권한을 부여할 수도 있습니다.

8. Next: Tags(다음: 태그)를 선택한 후 Next: Review(다음: 검토)를 선택합니다.
9. 이름에서 IAM 정책의 이름을 입력합니다(예: `rds-s3-integration-policy`). IAM 역할을 만들어 DB 인스턴스와 연결할 때 이 이름을 사용합니다. Description 값(선택 사항)을 추가할 수도 있습니다.
10. 정책 생성을 선택합니다.

AWS CLI

Amazon RDS에 Amazon S3 버킷에 대한 액세스 권한을 부여하는 AWS Identity and Access Management(IAM) 정책을 생성합니다. 정책을 생성한 후 정책의 ARN을 기록해 둡니다. 이후 단계에 이 ARN이 필요합니다.

필요한 액세스 유형에 따라 정책에 적절한 조치를 포함합니다.

- `GetObject` – Amazon S3 버킷에서 Amazon RDS로 파일을 전송하는 데 필요합니다.
- `ListBucket` – Amazon S3 버킷에서 Amazon RDS로 파일을 전송하는 데 필요합니다.
- `PutObject` – Amazon RDS에서 Amazon S3 버킷으로 파일을 전송하는 데 필요합니다.

다음 AWS CLI 명령은 이 옵션으로 *`rds-s3-integration-policy`*라는 IAM 정책을 만듭니다. *`amzn-s3-demo-bucket`*이라는 버킷에 대한 액세스 권한을 부여합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws iam create-policy \
   --policy-name rds-s3-integration-policy \
   --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3integration",
         "Action": [
           "s3:GetObject",
           "s3:ListBucket",
           "s3:PutObject"
```

```
        ],
        "Effect": "Allow",
        "Resource": [
          "arn:aws:s3:::amzn-s3-demo-bucket",
          "arn:aws:s3:::amzn-s3-demo-bucket/*"
        ]
      }
    ]
  }'
```

다음 예제에는 사용자 지정 KMS 키에 대한 사용 권한이 포함되어 있습니다.

```
aws iam create-policy \
   --policy-name rds-s3-integration-policy \
   --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3integration",
         "Action": [
           "s3:GetObject",
           "s3:ListBucket",
           "s3:PutObject",
           "kms:Decrypt",
           "kms:Encrypt",
           "kms:ReEncrypt*",
           "kms:GenerateDataKey",
           "kms:DescribeKey",
         ],
         "Effect": "Allow",
         "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket",
           "arn:aws:s3:::amzn-s3-demo-bucket/*",
           "arn:aws:kms:::your-kms-arn"
         ]
       }
     ]
   }'
```

Windows의 경우:

```
aws iam create-policy ^
   --policy-name rds-s3-integration-policy ^
```

```
   --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3integration",
         "Action": [
           "s3:GetObject",
           "s3:ListBucket",
           "s3:PutObject"
         ],
         "Effect": "Allow",
         "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket",
           "arn:aws:s3:::amzn-s3-demo-bucket/*"
         ]
       }
     ]
   }'
```

다음 예제에는 사용자 지정 KMS 키에 대한 사용 권한이 포함되어 있습니다.

```
aws iam create-policy ^
   --policy-name rds-s3-integration-policy ^
   --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3integration",
         "Action": [
           "s3:GetObject",
           "s3:ListBucket",
           "s3:PutObject",
           "kms:Decrypt",
           "kms:Encrypt",
           "kms:ReEncrypt",
           "kms:GenerateDataKey",
           "kms:DescribeKey",
         ],
         "Effect": "Allow",
         "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket",
           "arn:aws:s3:::amzn-s3-demo-bucket/*",
           "arn:aws:kms:::your-kms-arn"
```

```
        ]
      }
    ]
}'
```

2단계: (선택 사항) Amazon S3 버킷에 대한 IAM 정책 생성

이 단계는 다음 조건에서만 필요합니다.

- 한 계정(계정 A)에서 Amazon S3 버킷에 파일을 업로드하고 다른 계정(계정 B)에서 액세스할 계획입니다.
- 계정 B가 버킷을 소유하고 있습니다.
- 계정 B는 버킷에 로드된 객체를 완전히 제어해야 합니다.

앞서 나온 조건에 해당하지 않는 경우 3단계: DB 인스턴스에 대한 IAM 역할 생성 및 정책 연결으로 건너뜁니다.

버킷 정책을 생성하려면 다음이 있어야 합니다.

- 계정 A의 계정 ID
- 계정 A의 사용자 이름
- 계정 B의 Amazon S3 버킷에 대한 ARN 값

콘솔

버킷 정책 생성 또는 편집

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷 목록에서 버킷 정책을 만들 버킷 이름 또는 버킷 정책을 편집할 버킷 이름을 선택합니다.
3. Permissions를 선택합니다.
4. 버킷 정책에서 편집을 선택합니다. 버킷 정책 편집(Edit bucket policy) 페이지가 열립니다.
5. 버킷 정책 편집 페이지에서 Amazon S3 사용 설명서의 정책 예제를 탐색하거나 정책 생성기를 선택하여 정책을 자동으로 생성하거나 정책 섹션에서 JSON을 편집합니다.

   정책 생성기(Policy generator)를 선택하면 새 창에서 AWS 정책 생성기가 열립니다.

   a. AWS 정책 생성기 페이지의 정책 유형 선택에서 S3 버킷 정책을 선택합니다.

b. 제공된 필드에 정보를 입력하여 명령문을 추가한 다음 명령문 추가(Add Statement)를 선택합니다. 명령문을 추가하려는 만큼 반복합니다. 이러한 필드에 대한 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

> **Note**
>
> 편의상 버킷 정책 편집(Edit bucket policy) 페이지는 정책(Policy) 텍스트 필드 위에 현재 버킷의 버킷 ARN(Bucket ARN)(Amazon 리소스 이름)을 표시합니다. AWS 정책 생성기 페이지의 명령문에 사용하기 위해 이 ARN을 복사할 수 있습니다.

c. 명령문 추가를 마친 후 정책 생성(Generate Policy)을 선택합니다.

d. 생성된 정책 텍스트를 복사하고 닫기(Close)를 선택하고 Amazon S3 콘솔의 버킷 정책 편집(Edit bucket policy) 페이지로 돌아갑니다.

6. 정책(Policy) 상자에서 기존 정책을 편집하거나 정책 생성기에서 버킷 정책을 붙여 넣습니다. 정책을 저장하기 전에 보안 경고, 오류, 일반 경고 및 제안 사항을 해결해야 합니다.

   JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Example permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::account-A-ID:account-A-user"
      },
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-destination-bucket",
        "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
      ]
    }
  ]
}
```

7. 변경 사항 저장을 선택하면 버킷 권한 페이지로 돌아갑니다.

3단계: DB 인스턴스에 대한 IAM 역할 생성 및 정책 연결

이 단계에서는 1단계: Amazon RDS 역할에 대한 IAM 정책 생성에서 IAM 정책을 생성했다고 가정합니다. 그리고 RDS for Oracle DB 인스턴스에 대한 역할을 생성한 다음 해당 역할에 정책을 연결합니다.

콘솔

Amazon S3 버킷에 Amazon RDS 액세스를 허용하는 IAM 역할을 생성하려면

1. IAM 관리 콘솔을 엽니다.
2. 탐색 창에서 Roles를 선택합니다.
3. 역할 생성(Create role)을 선택합니다.
4. AWS 서비스를 선택합니다.
5. 기타 AWS 서비스 사용 사례에서 RDS를 선택한 다음 RDS - 데이터베이스에 역할 추가를 선택합니다. 이후 다음을 선택합니다.
6. 권한 정책의 검색에 1단계: Amazon RDS 역할에 대한 IAM 정책 생성에서 생성한 IAM 정책의 이름을 입력하고, 목록에 표시된 정책을 선택합니다. 그런 다음 다음을 선택합니다.
7. 역할 이름에 IAM 역할의 이름을 입력합니다(예: `rds-s3-integration-role`). 설명 값(선택 사항)을 추가할 수도 있습니다.
8. 역할 생성을 선택합니다.

AWS CLI

역할을 생성하고 역할에 정책을 연결하는 방법

1. Amazon RDS가 Amazon S3 버킷에 액세스하기 위해 사용자 대신 가정할 수 있는 IAM 역할을 만듭니다.

   서비스 권한을 특정 리소스로 제한하는 리소스 기반 신뢰 관계의 `aws:SourceArn` 및 `aws:SourceAccount` 전역 조건 컨텍스트 키를 사용하는 것이 좋습니다. 이는 혼동된 대리자 문제를 방지하는 가장 효과적인 방법입니다.

   전역 조건 컨텍스트 키를 모두 사용하고 `aws:SourceArn` 값에 계정 ID가 포함되도록 할 수 있습니다. 이 경우 `aws:SourceAccount` 값과 `aws:SourceArn` 값의 계정이 동일한 문에서 사용될 때 동일한 계정 ID를 사용해야 합니다.

   - 단일 리소스에 대한 교차 서비스 액세스를 원하는 경우 `aws:SourceArn`을 사용하세요.

- 해당 계정의 모든 리소스가 교차 서비스 사용과 연결되도록 허용하려는 경우 `aws:SourceAccount`를 사용하세요.

신뢰 정책에서는 역할에 액세스하는 리소스의 전체 Amazon 리소스 이름(ARN)이 포함된 `aws:SourceArn` 전역 조건 컨텍스트 키를 사용해야 합니다.

다음 AWS CLI 명령은 이 목적으로 *rds-s3-integration-role*이라는 역할을 생성합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws iam create-role \
   --role-name rds-s3-integration-role \
   --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                 "aws:SourceAccount": "my_account_ID",
                 "aws:SourceArn": "arn:aws:rds:Region:my_account_ID:db:dbname"
             }
         }
       }
     ]
   }'
```

Windows의 경우:

```
aws iam create-role ^
   --role-name rds-s3-integration-role ^
   --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
```

```
        "Effect": "Allow",
        "Principal": {
           "Service": "rds.amazonaws.com"
         },
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "my_account_ID",
                "aws:SourceArn": "arn:aws:rds:Region:my_account_ID:db:dbname"
            }
        }
      }
    ]
  }'
```

자세한 내용은 IAM 사용 설명서의 IAM 사용자에게 권한을 위임하기 위한 역할 생성을 참조하십시오.

2. 역할이 생성되면 역할의 ARN을 기록하십시오. 이후 단계에 이 ARN이 필요합니다.
3. 생성한 정책을 생성한 역할에 연결하십시오.

   다음 AWS CLI 명령은 정책을 *rds-s3-integration-role*이라는 역할에 연결합니다.

   Example

   대상 LinuxmacOS, 또는Unix:

   ```
   aws iam attach-role-policy \
      --policy-arn your-policy-arn \
      --role-name rds-s3-integration-role
   ```

   Windows의 경우:

   ```
   aws iam attach-role-policy ^
      --policy-arn your-policy-arn ^
      --role-name rds-s3-integration-role
   ```

   *your-policy-arn*을 이전 단계에서 기록한 정책 ARN으로 바꾸세요.

4단계: IAM 역할을 RDS for Oracle DB 인스턴스와 연결하는 방법

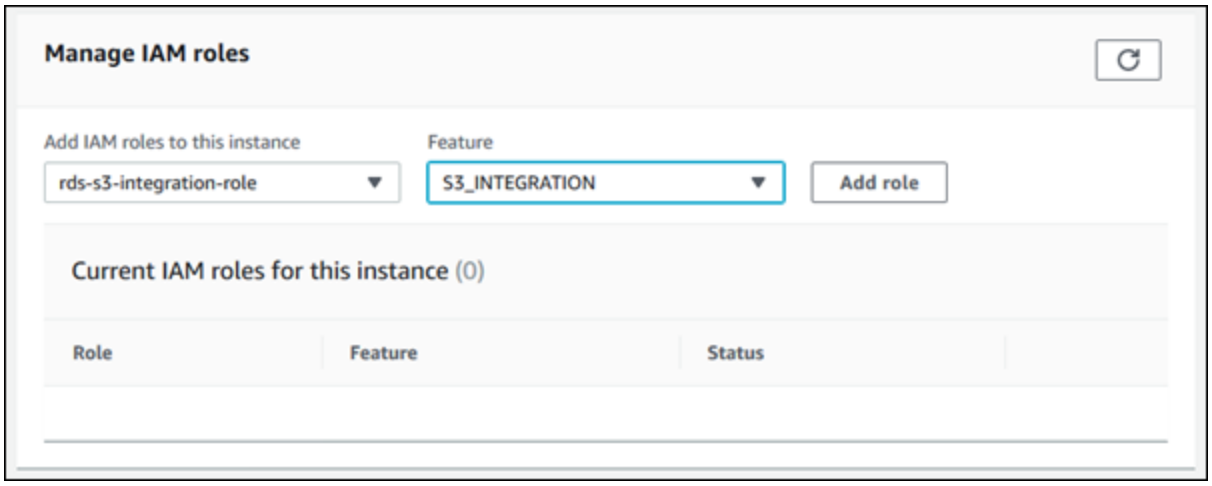
Amazon S3 통합 권한을 구성하는 마지막 단계는 IAM 역할을 DB 인스턴스와 연결하는 것입니다. 다음과 같은 요구 사항을 확인합니다.

- 필수 Amazon S3 권한 정책이 연결된 IAM 역할에 대한 액세스 권한이 있어야 합니다.
- 한 번에 하나의 IAM 역할만 RDS for Oracle DB 인스턴스에 연결할 수 있습니다.
- DB 인스턴스는 사용 가능 상태여야 합니다.

콘솔

IAM 역할을 RDS for Oracle DB 인스턴스와 연결하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 세부 정보를 표시하고자 하는 RDS for Oracle DB 인스턴스 이름을 선택합니다.
4. 연결성 및 보안(Connectivity & security) 탭에서 페이지 하단의 IAM 역할 관리(Manage IAM roles) 섹션이 나올 때까지 아래로 스크롤합니다.
5. IAM 역할을 이 인스턴스에 추가에는 3단계: DB 인스턴스에 대한 IAM 역할 생성 및 정책 연결에서 생성한 역할을 선택합니다.
6. 기능에서 S3_INTEGRATION을 선택하십시오.



7. [Add role]을 선택합니다.

AWS CLI

다음 AWS CLI 명령은 *mydbinstance*라는 Oracle DB 인스턴스에 역할을 추가합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds add-role-to-db-instance \
   --db-instance-identifier mydbinstance \
   --feature-name S3_INTEGRATION \
   --role-arn your-role-arn
```

Windows의 경우:

```
aws rds add-role-to-db-instance ^
   --db-instance-identifier mydbinstance ^
   --feature-name S3_INTEGRATION ^
   --role-arn your-role-arn
```

*your-role-arn*을 이전 단계에서 기록한 역할 ARN으로 바꿉니다. `S3_INTEGRATION` 옵션에 대해 `--feature-name`을 지정해야 합니다.

Amazon S3 통합 옵션 추가

Amazon RDS for Oracle을 Amazon S3와 통합하려면 DB 인스턴스가 `S3_INTEGRATION` 옵션을 포함하는 옵션 그룹과 연결되어 있어야 합니다.

콘솔

Amazon S3 통합을 위한 옵션 그룹을 구성하려면

1. 새 옵션 그룹을 만들거나 `S3_INTEGRATION` 옵션을 추가할 수 있는 기존 옵션 그룹을 식별하십시오.

   옵션 그룹의 생성에 대한 자세한 내용은 옵션 그룹 생성 단원을 참조하십시오.
2. [S3_INTEGRATION] 옵션을 옵션 그룹에 추가합니다.

   옵션 그룹에 옵션을 추가하는 방법에 대한 자세한 내용은 옵션 그룹에 옵션 추가 단원을 참조하십시오.
3. 새 RDS for Oracle DB 인스턴스를 생성하고 옵션 그룹을 연결하거나, RDS for Oracle DB 인스턴스를 수정하여 옵션 그룹을 연결합니다.

DB 인스턴스 생성에 대한 자세한 내용은 Amazon RDS DB 인스턴스 생성 단원을 참조하십시오.

DB 인스턴스 수정에 대한 자세한 내용은 [g70][/g70][g69][/g69] 섹션을 참조하세요.

AWS CLI

Amazon S3 통합을 위한 옵션 그룹을 구성하려면

1. 새 옵션 그룹을 만들거나 S3_INTEGRATION 옵션을 추가할 수 있는 기존 옵션 그룹을 식별하십시오.

   옵션 그룹의 생성에 대한 자세한 내용은 옵션 그룹 생성 단원을 참조하십시오.
2. [S3_INTEGRATION] 옵션을 옵션 그룹에 추가합니다.

   예를 들어 다음 AWS CLI 명령은 S3_INTEGRATION 옵션을 **myoptiongroup**이라는 옵션 그룹에 추가합니다.

   Example

   대상 LinuxmacOS, 또는Unix:

   ```
   aws rds add-option-to-option-group \
      --option-group-name myoptiongroup \
      --options OptionName=S3_INTEGRATION,OptionVersion=1.0
   ```

   Windows의 경우:

   ```
   aws rds add-option-to-option-group ^
      --option-group-name myoptiongroup ^
      --options OptionName=S3_INTEGRATION,OptionVersion=1.0
   ```
3. 새 RDS for Oracle DB 인스턴스를 생성하고 옵션 그룹을 연결하거나, RDS for Oracle DB 인스턴스를 수정하여 옵션 그룹을 연결합니다.

   DB 인스턴스 생성에 대한 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.

   RDS for Oracle DB 인스턴스 수정에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

Amazon RDS for Oracle와 Amazon S3 버킷 사이의 파일 전송

RDS for Oracle DB 인스턴스와 Amazon S3 버킷 간에 파일을 전송하려면 Amazon RDS 패키지 `rdsadmin_s3_tasks`를 사용하면 됩니다. 파일을 업로드할 때 GZIP으로 파일을 압축하고 다운로드 시 압축을 풀 수 있습니다.

주제

- 파일 전송 요구 사항 및 제한 사항
- RDS for Oracle DB 인스턴스에서 Amazon S3 버킷으로 파일 업로드
- Amazon S3 버킷의 파일을 Oracle DB 인스턴스로 다운로드
- 파일 전송 상태 모니터링

파일 전송 요구 사항 및 제한 사항

DB 인스턴스와 Amazon S3 버킷 사이에서 파일을 전송하기 전에 다음에 유의하세요.

- `rdsadmin_s3_tasks` 패키지는 단일 디렉터리에서 파일을 전송합니다. 전송에 하위 디렉터리를 포함시킬 수 없습니다.
- Amazon S3 버킷의 최대 객체 크기는 5TB입니다.
- `rdsadmin_s3_tasks`에서 생성한 작업은 비동기적으로 실행됩니다.
- Data Pump 디렉터리(예: `DATA_PUMP_DIR` 또는 사용자 생성 디렉터리)에서 파일을 업로드할 수 있습니다. Oracle 백그라운드 프로세스에서 사용하는 디렉터리(예: adump, bdump, 또는 `trace` 디렉터리)에서는 파일을 업로드할 수 없습니다.
- 다운로드 제한은 `download_from_s3`의 프로시저 직접 호출당 파일 2,000개입니다. Amazon S3에서 2,000개 이상의 파일을 다운로드해야 하는 경우 프로시저 호출당 2,000개 이하의 파일로 다운로드를 개별 작업으로 분할하세요.
- 다운로드 폴더에 파일이 있고 같은 이름의 파일을 다운로드하려고 하면 `download_from_s3`를 통해 다운로드를 건너뜁니다. 다운로드 디렉터리에서 파일을 제거하려면 PL/SQL 프로시저 UTL_FILE.FREMOVE를 사용하세요.

RDS for Oracle DB 인스턴스에서 Amazon S3 버킷으로 파일 업로드

DB 인스턴스에서 Amazon S3 버킷으로 파일을 업로드하려면 절차 `rdsadmin.rdsadmin_s3_tasks.upload_to_s3`를 참조하면 됩니다. 예를 들어 Oracle Recovery Manager(RMAN) 백업 파일 또는 Oracle Data Pump 파일을 업로드할 수 있습니다. 객체 작업에 대한

자세한 내용은 Amazon Simple Storage Service 사용 설명서를 참조하세요. RMAN 백업 수행에 대한 자세한 내용은 Oracle DB 인스턴스에 대한 공통 RMAN 작업 수행 단원을 참조하세요.

`rdsadmin.rdsadmin_s3_tasks.upload_to_s3` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
`p_bucket_name`	VARCHAR2	–	필수	파일을 업로드할 Amazon S3 버킷의 이름입니다.
`p_directory_name`	VARCHAR2	–	필수	파일을 업로드할 Oracle 디렉터리 객체의 이름입니다. 이 디렉터리는 사용자가 생성한 디렉터리 객체 또는 Data Pump 디렉터리(예: `DATA_PUMP_DIR`)일 수 있습니다. 백그라운드 프로세스에서 사용하는 디렉터리(예: `adump`, `bdump`, 또는 `trace` 디렉터리)에서는 파일을 업로드할 수 없습니다. **Note** 지정된 디렉터리의 파일만 업로드할 수 있습니다. 지정된 디렉터리의 하위 디렉터리에서는 파일을 업로드할 수 없습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_s3_prefix	VARCHAR2	–	필수	파일이 업로드되는 Amazon S3 파일 이름 접두사입니다. 빈 접두사는 모든 파일을 지정된 Amazon S3 버킷의 최상위 레벨에 업로드하며, 접두사를 파일 이름에 추가하지 않습니다. 예를 들어 접두사가 folder_1/oradb 이면 파일이 folder_1에 업로드됩니다. 이 경우 oradb 접두사가 각 파일에 추가됩니다.
p_prefix	VARCHAR2	–	필수	업로드되기 위해 파일 이름과 일치해야 하는 파일 이름 접두사입니다. 빈 접두사는 지정된 디렉터리의 모든 파일을 업로드합니다.
p_compression_level	NUMBER	0	선택 사항	GZIP 압축 수준입니다. 사용할 수 있는 값은 0~9입니다. • 0 - 압축 없음 • 1 - 가장 빠른 압축 • 9 - 최고 수준의 압축

파라미터 이름	데이터 형식	기본값	필수	설명
p_bucket_owner_full_control	VARCHAR2	–	선택 사항	버킷에 대한 액세스 제어 설정입니다. 유일하게 유효한 값은 null과 FULL_CONTROL 입니다. 이 설정은 한 계정(계정 A)의 파일을 다른 계정(계정 B)에서 소유한 버킷으로 업로드하고 계정 B가 해당 파일을 완전히 제어해야 하는 경우에만 필요합니다.

rdsadmin.rdsadmin_s3_tasks.upload_to_s3 프로시저의 반환 값이 작업 ID입니다.

다음 예제에서는 *DATA_PUMP_DIR* 디렉터리에 있는 모든 파일을 *amzn-s3-demo-bucket*이라는 이름의 Amazon S3 버킷에 업로드합니다. 파일이 압축되지 않습니다.

```
SELECT rdsadmin.rdsadmin_s3_tasks.upload_to_s3(
      p_bucket_name    =>  'amzn-s3-demo-bucket',
      p_prefix         =>  '',
      p_s3_prefix      =>  '',
      p_directory_name =>  'DATA_PUMP_DIR')
   AS TASK_ID FROM DUAL;
```

다음 예제에서는 *db* 디렉터리에 있는 *DATA_PUMP_DIR* 접두사의 모든 파일을 *amzn-s3-demo-bucket*이라는 이름의 Amazon S3 버킷에 업로드합니다. Amazon RDS는 파일에 최고 수준의 GZIP 압축을 적용합니다.

```
SELECT rdsadmin.rdsadmin_s3_tasks.upload_to_s3(
      p_bucket_name       =>  'amzn-s3-demo-bucket',
      p_prefix            =>  'db',
      p_s3_prefix         =>  '',
      p_directory_name    =>  'DATA_PUMP_DIR',
      p_compression_level =>  9)
   AS TASK_ID FROM DUAL;
```

다음 예제에서는 *DATA_PUMP_DIR* 디렉터리에 있는 모든 파일을 *amzn-s3-demo-bucket*이라는 이름의 Amazon S3 버킷에 업로드합니다. 파일은 dbfiles 폴더에 업로드됩니다. 이 예제에서 GZIP 압축 수준은 가장 빠른 압축 수준인 *1*입니다.

```
SELECT rdsadmin.rdsadmin_s3_tasks.upload_to_s3(
      p_bucket_name       =>  'amzn-s3-demo-bucket',
      p_prefix            =>  '',
      p_s3_prefix         =>  'dbfiles/',
      p_directory_name    =>  'DATA_PUMP_DIR',
      p_compression_level =>  1)
   AS TASK_ID FROM DUAL;
```

다음 예제에서는 *DATA_PUMP_DIR* 디렉터리에 있는 모든 파일을 *amzn-s3-demo-bucket*이라는 이름의 Amazon S3 버킷에 업로드합니다. 파일은 dbfiles 폴더에 업로드되고 ora는 각 파일 이름의 시작 부분에 추가됩니다. 압축이 적용되지 않습니다.

```
SELECT rdsadmin.rdsadmin_s3_tasks.upload_to_s3(
      p_bucket_name    =>  'amzn-s3-demo-bucket',
      p_prefix         =>  '',
      p_s3_prefix      =>  'dbfiles/ora',
      p_directory_name =>  'DATA_PUMP_DIR')
   AS TASK_ID FROM DUAL;
```

다음 예제에서는 명령이 계정 A에서 실행되지만, 계정 B에서 버킷 콘텐츠를 완전히 제어해야 한다고 가정합니다. 명령 rdsadmin_s3_tasks.upload_to_s3는 *DATA_PUMP_DIR* 디렉터리의 모든 파일을 *s3bucketOwnedByAccountB*라는 이름의 버킷으로 전송합니다. 액세스 제어가 FULL_CONTROL로 설정되어 계정 B가 버킷의 파일에 액세스할 수 있습니다. GZIP 압축 수준은 속도 및 파일 크기가 균형을 이루는 *6*입니다.

```
SELECT rdsadmin.rdsadmin_s3_tasks.upload_to_s3(
      p_bucket_name               =>  's3bucketOwnedByAccountB',
      p_prefix                    =>  '',
      p_s3_prefix                 =>  '',
      p_directory_name            =>  'DATA_PUMP_DIR',
      p_bucket_owner_full_control =>  'FULL_CONTROL',
      p_compression_level         =>  6)
   AS TASK_ID FROM DUAL;
```

각 예에서, SELECT 문은 VARCHAR2 데이터 형식으로 작업 ID를 반환합니다.

작업의 출력 파일을 표시하여 결과를 볼 수 있습니다.

```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-task-id.log'));
```

*task-id*를 절차에서 반환된 작업 ID로 대체합니다.

> **Note**
>
> 작업은 비동기식으로 실행됩니다.

Amazon S3 버킷의 파일을 Oracle DB 인스턴스로 다운로드

RDS for Oracle 인스턴스로 Amazon S3 버킷의 파일을 다운로드하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_s3_tasks.download_from_s3`를 사용하십시오.

`download_from_s3` 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
`p_bucket_name`	VARCHAI	–	필수	파일을 다운로드할 수 있는 Amazon S3 버킷의 이름입니다.
`p_directory_name`	VARCHAI	–	필수	파일을 다운로드할 Oracle 디렉터리 객체의 이름입니다. 이 디렉터리는 사용자가 생성한 디렉터리 객체 또는 Data Pump 디렉터리(예: `DATA_PUMP_DIR`)일 수 있습니다.
`p_error_on_zero_downloads`	VARCHAI	FALSE	선택 사항	Amazon S3 버킷에 접두사와 일치하는 객체가 없을 때 작업에서 오류가 발생하는지 여부를 결정하는 플래그입니다. 이 파라미터가 설정되지 않았거나 FALSE(기본값)로 설정된 경우 작업은 객체를 찾을 수 없다는 메시지를 인쇄하지만 예외를 발생시키거나 실패하지는 않습니다. 이 파라미터가 TRUE이면 작업에서 예외가 발생하고 실패합니다.

파라미터 이름	데이터 형식	기본값	필수	설명
				일치 테스트에 실패할 수 있는 접두사 사양의 예로는 ' import/test9.log' 와 같은 접두사의 공백과 test9.log 및 test9.LOG 와 같은 대/소문자 불일치가 있습니다.
p_s3_prefix	VARCHAI	–	필수	다운로드되기 위해 파일 이름과 일치해야 하는 파일 이름 접두사입니다. 빈 접두사는 지정된 Amazon S3 버킷에 모든 상위 수준 파일을 다운로드하고, 버킷의 폴더에는 파일을 다운로드하지 않습니다. 이 프로시저는 접두사와 일치하는 첫 레벨 폴더에서만 Amazon S3개 객체를 다운로드합니다. 지정된 접두사와 일치하는 중첩된 디렉터리 구조는 다운로드되지 않습니다. 예를 들어 Amazon S3 버킷에 folder_1/folder_2/folder_3 폴더 구조가 있다고 가정합니다. 'folder_1/folder_2/' 접두사를 지정합니다. 이 경우 folder_2의 파일만 다운로드되고 folder_1 또는 folder_3의 파일은 다운로드되지 않습니다. 대신에 'folder_1/folder_2' 접두사를 지정하는 경우에는 folder_1 접두사와 일치하는 'folder_2' 의 모든 파일들이 다운로드되고, folder_2의 파일들은 다운로드되지 않습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
p_decompression_format	VARCHAR	–	선택 사항	압축 형식입니다. 유효 값은 압축 해제 미적용 시 NONE, 압축 해제 적용 시 GZIP입니다.

rdsadmin.rdsadmin_s3_tasks.download_from_s3 프로시저의 반환 값이 작업 ID입니다.

다음 예에서는 *amzn-s3-demo-bucket*라는 Amazon S3 버킷의 모든 파일을 *DATA_PUMP_DIR* 디렉터리로 다운로드합니다. 파일은 압축되지 않으므로 압축 해제가 적용되지 않습니다.

```
SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3(
      p_bucket_name    =>  'amzn-s3-demo-bucket',
      p_directory_name =>  'DATA_PUMP_DIR')
   AS TASK_ID FROM DUAL;
```

다음 예제에서는 *db*이라는 Amazon S3 버킷에서 접두사가 *amzn-s3-demo-bucket*인 모든 파일을 *DATA_PUMP_DIR* 디렉터리로 다운로드합니다. 파일이 GZIP으로 압축되므로 압축 해제가 적용됩니다. 파라미터 p_error_on_zero_downloads는 접두사 오류 검사를 켜므로 접두사가 버킷의 파일과 일치하지 않으면 작업이 예외를 발생시키고 실패합니다.

```
SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3(
      p_bucket_name               =>  'amzn-s3-demo-bucket',
      p_s3_prefix                 =>  'db',
      p_directory_name            =>  'DATA_PUMP_DIR',
      p_decompression_format      =>  'GZIP',
      p_error_on_zero_downloads   =>  'TRUE')
   AS TASK_ID FROM DUAL;
```

다음 예제에서는 *myfolder/*이라는 Amazon S3 버킷의 *amzn-s3-demo-bucket* 폴더에 있는 모든 파일을 *DATA_PUMP_DIR* 디렉터리로 다운로드합니다. p_s3_prefix 파라미터를 사용하여 Amazon S3 폴더를 지정합니다. 업로드된 파일은 GZIP으로 압축되지만, 다운로드 중에는 압축이 풀리지 않습니다.

```
SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3(
      p_bucket_name          =>  'amzn-s3-demo-bucket',
      p_s3_prefix            =>  'myfolder/',
      p_directory_name       =>  'DATA_PUMP_DIR',
```

```
      p_decompression_format =>  'NONE')
   AS TASK_ID FROM DUAL;
```

다음 예에서는 *amzn-s3-demo-bucket*이라는 Amazon S3 버킷의 *mydumpfile.dmp* 파일을 *DATA_PUMP_DIR* 디렉터리로 다운로드합니다. 압축 해제가 적용되지 않습니다.

```
SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3(
      p_bucket_name    =>  'amzn-s3-demo-bucket',
      p_s3_prefix      =>  'mydumpfile.dmp',
      p_directory_name =>  'DATA_PUMP_DIR')
   AS TASK_ID FROM DUAL;
```

각 예에서, SELECT 문은 VARCHAR2 데이터 형식으로 작업 ID를 반환합니다.

작업의 출력 파일을 표시하여 결과를 볼 수 있습니다.

```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-task-id.log'));
```

*task-id*를 절차에서 반환된 작업 ID로 대체합니다.

> **Note**
>
> 작업은 비동기식으로 실행됩니다.
> UTL_FILE.FREMOVE Oracle 프로시저를 사용하여 디렉터리에서 파일을 제거할 수 있습니다. 자세한 내용은 Oracle 설명서의 FREMOVE 프로시저를 참조하십시오.

파일 전송 상태 모니터링

파일 전송 작업은 Amazon RDS 이벤트가 시작되고 완료될 때 이벤트를 게시합니다. 이벤트 메시지에 파일 전송을 위한 작업 ID가 포함됩니다. 이벤트 보기에 대한 자세한 내용은 Amazon RDS 이벤트 보기 단원을 참조하십시오.

진행 중인 작업의 상태를 bdump 파일에서 볼 수 있습니다. 이 bdump 파일은 /rdsdbdata/log/trace 디렉터리에 위치합니다. 각 bdump 파일 이름은 다음 형식으로 되어 있습니다.

```
dbtask-task-id.log
```

*task-id*를 모니터링하고자 하는 작업의 ID로 바꾸십시오.

> **Note**
>
> 작업은 비동기식으로 실행됩니다.

`rdsadmin.rds_file_util.read_text_file` 저장 프로시저를 사용하여 bdump 파일의 내용을 볼 수 있습니다. 예를 들어, 다음 쿼리는 *dbtask-1234567890123-1234.log* bdump 파일의 내용을 반환합니다.

```
SELECT text FROM
 table(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-1234567890123-1234.log'));
```

다음 샘플에서는 전송 실패에 대한 로그 파일을 보여줍니다.

```
TASK_ID




---------------------------------------------------------------------------------------------
1234567890123-1234


TEXT









---------------------------------------------------------------------------------------------
2023-04-17 18:21:33.993 UTC [INFO ] File #1: Uploading the file /rdsdbdata/datapump/
A123B4CDEF567890G1234567890H1234/sample.dmp to Amazon S3 with bucket name amzn-s3-demo-
bucket and key sample.dmp.
```

```
2023-04-17 18:21:34.188 UTC [ERROR] RDS doesn't have permission to write to Amazon S3
 bucket name amzn-s3-demo-bucket and key sample.dmp.
2023-04-17 18:21:34.189 UTC [INFO ] The task failed.
```

Amazon S3 통합 문제 해결

문제 해결 팁을 보려면 AWS re:Post 문서인 Amazon RDS for Oracle을 Amazon S3와 통합할 때 발생하는 문제를 해결하려면 어떻게 해야 합니까?를 참조하세요.

Amazon S3 통합 옵션 제거

DB 인스턴스에서 Amazon S3 통합 옵션을 제거할 수 있습니다.

DB 인스턴스에서 Amazon S3 통합 옵션을 제거하려면 다음 중 하나를 수행합니다.

- 여러 DB 인스턴스에서 Amazon S3 통합 옵션을 제거하려면 DB 인스턴스가 속한 옵션 그룹에서 해당 `S3_INTEGRATION` 옵션을 제거합니다. 이 변경은 해당 옵션 그룹을 사용하는 모든 DB 인스턴스에 영향을 미칩니다. 자세한 내용은 옵션 그룹에서 옵션 제거을 참조하세요.
- 단일 DB 인스턴스에서 Amazon S3 통합 옵션을 제거하려면 인스턴스를 수정하고 `S3_INTEGRATION` 옵션이 포함되지 않은 다른 옵션 그룹을 지정합니다. 기본(비어 있음) 옵션 그룹을 지정하거나 다른 사용자 지정 옵션 그룹을 지정할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정을 참조하세요.

Oracle Application Express(APEX)

Amazon RDS는 APEX 및 APEX-DEV 옵션 사용을 통해 Oracle Application Express(APEX)를 지원합니다. Oracle APEX를 웹 기반 애플리케이션을 위한 런타임 환경 또는 전체 개발 환경으로 배포할 수 있습니다. 사용자는 Oracle APEX를 사용하여 웹 브라우저 내에서 전체 애플리케이션을 빌드할 수 있습니다. 자세한 내용은 Oracle 문서의 Oracle Application Express를 참조하세요.

주제

- Oracle APEX 구성 요소
- 요구 사항 및 제한 사항
- Oracle APEX 및 Oracle Rest Data Services(ORDS) 설정
- Oracle Rest Data Services(ORDS) 구성
- Oracle APEX 업그레이드 및 제거

Oracle APEX 구성 요소

Oracle APEX는 다음과 같은 주요 구성 요소로 구성됩니다.

- 리포지토리 - Oracle APEX 애플리케이션 및 구성 요소에 대한 메타데이터를 저장합니다. 리포지토리는 Amazon RDS DB 인스턴스에 설치되는 테이블, 인덱스 및 기타 객체로 구성됩니다.
- listener - Oracle APEX 클라이언트와의 HTTP 통신을 관리합니다. 리스너는 Amazon EC2 인스턴스, 회사의 온프레미스 서버 또는 사용자의 데스크톱 컴퓨터와 같은 별도의 호스트에 상주합니다. 리스너는 웹 브라우저에서 수신되는 유입 연결을 허용하고 처리를 위해 해당 연결을 Amazon RDS DB 인스턴스에 전달한 다음, 리포지토리의 결과를 브라우저로 다시 보냅니다.

  RDS for Oracle은 다음과 같은 유형의 리스너를 지원합니다.

  - Oracle APEX 버전 5.0 이상은 Oracle REST Data Services(ORDS) 버전 19.1 이상을 사용합니다. 지원되는 최신 버전의 Oracle APEX 및 ORDS를 사용하는 것이 좋습니다. 이 설명서에서는 이전 버전과의 호환성에 대해서만 이전 버전을 설명합니다.
  - Oracle APEX 버전 4.1.1의 경우 Oracle APEX Listener 버전 1.1.4를 사용할 수 있습니다.
  - Oracle HTTP 서버 및 mod_plsql 리스너를 사용할 수 있습니다.

> **Note**
>
> Amazon RDS는 Oracle APEX의 리스너로 PL/SQL 게이트웨이가 내장된 Oracle XML DB HTTP 서버를 지원하지 않습니다. 일반적으로 Oracle은 인터넷 기반 애플리케이션용으로 포함된 PL/SQL 게이트웨이 사용을 권장합니다.

이러한 리스너 유형에 대한 자세한 내용은 Oracle 설명서에서 About Choosing a Web Listener를 참조하세요.

APEX 및 APEX-DEV 옵션을 RDS for Oracle DB 인스턴스에 추가하면 Amazon RDS에서 Oracle APEX 리포지토리만 설치합니다. 리스너를 별도의 호스트에 설치합니다.

요구 사항 및 제한 사항

다음 주제에는 Oracle APEX 및 ORDS의 요구 사항과 제한 사항이 나와 있습니다.

Oracle APEX 버전 요구 사항

APEX 옵션은 DB 인스턴스에 대한 DB 인스턴스 클래스의 스토리지를 사용합니다. 다음은 Oracle APEX에 대한 지원 버전과 대략적인 스토리지 요구 사항입니다.

Oracle APEX 버전	스토리지 요구 사항	지원되는 Oracle Database 버전	Notes
Oracle APEX 버전 24.2.v1	114MiB	모두	이 버전은 패치 37885097: APEX 24.2용 PSE 번들(24.2.0 기반 PSES), PATCH_VERSION 4를 포함합니다.
Oracle APEX 버전 24.1.v1	112MiB	모두	이 버전은 패치 36695709: APEX 24.1용 PSE 번들(24.1.0 기반 PSES), PATCH_VERSION 3을 포함합니다. EC2 인스턴스에 설치할 APEX 이미지 버전이 정확히 동일한 경우 패치 37544819: 24.1.3 PSE BUNDLE FOR APEX 24.1(PSES ON TOP OF 24.1.0)을 다운로드합니다.

Oracle APEX 버전	스토리지 요구 사항	지원되는 Oracle Database 버전	Notes
Oracle APEX 버전 23.2.v1	110MiB	모두	이 버전에는 패치 35895964: APEX 23.2용 PSE 번들(23.2.0 기반 PSES), PATCH_VERSION 6이 포함됩니다. EC2 인스턴스에 설치할 APEX 이미지 버전이 정확히 동일한 경우 패치 37593125: 23.2.6 PSE BUNDLE FOR APEX 23.2(PSES ON TOP OF 23.2.0)을 다운로드합니다.
Oracle APEX 버전 23.1.v1	106MiB	모두	이 버전은 패치 35283657: APEX 23.1용 PSE 번들(23.1.0 기반 PSES), PATCH_VERSION 2를 포함합니다.
Oracle APEX 버전 22.2.v1	106MiB	모두	이 버전은 패치 34628174: APEX 22.2용 PSE 번들(22.2.0 기반 PSES), PATCH_VERSION 4를 포함합니다.
Oracle APEX 버전 22.1.v1	124MiB	모두	이 버전은 패치 34020981: APEX 22.1용 PSE 번들(22.1.0 기반 PSES), PATCH_VERSION 6을 포함합니다.
Oracle APEX 버전 21.1.v1	125MiB	모두	이 버전은 패치 33420059: APEX 21.2용 PSE 번들(21.2.0 기반 PSES), PATCH_VERSION 8을 포함합니다.
Oracle APEX 버전 21.1.v1	125MiB	모두	이 버전은 패치 32598392: APEX 21.1용 PSE 번들, PATCH_VERSION 3를 포함합니다.
Oracle APEX 버전 20.2.v1	148MiB	Oracle Database 21c를 제외한 모든 제품	이 버전은 패치 32006852: APEX 20.2용 PSE 번들, PATCH_VERSION 2020.11.12를 포함합니다. 다음 쿼리를 실행하여 패치 번호와 날짜를 확인할 수 있습니다. `SELECT PATCH_VERSION, PATCH_NUMBER FROM APEX_PATCHES;`

Oracle APEX 버전	스토리지 요구 사항	지원되는 Oracle Database 버전	Notes
Oracle APEX 버전 20.1.v1	173MiB	Oracle Database 21c를 제외한 모든 제품	이 버전은 패치 30990551: APEX 20.1용 PSE 번들, PATCH_VERSION 2020.07.15를 포함합니다.
Oracle APEX 버전 19.2.v1	149 MiB	Oracle Database 21c를 제외한 모든 제품	
Oracle APEX 버전 19.1.v1	148MiB	Oracle Database 21c를 제외한 모든 제품	

다운로드 가능한 Oracle APEX .zip 파일은 Oracle 웹 사이트의 Oracle APEX 이전 릴리스 아카이브를 참조하세요.

Oracle APEX 및 ORDS 사전 조건

Oracle APEX 및 ORDS 사용의 사전 조건은 다음과 같습니다.

- 시스템이 Java 런타임 환경(JRE)을 사용해야 합니다.
- Oracle 클라이언트 설치에는 다음이 포함되어야 합니다.
  - 관리 작업을 위한 SQL*Plus 또는 SQL Developer
  - RDS for Oracle DB 인스턴스에 대한 연결을 구성하기 위한 Oracle Net Services

Oracle APEX 제한 사항

Amazon RDS에서 관리하는 APEX_*version* 사용자 계정은 수정할 수 없습니다. 따라서 이 사용자에게 데이터베이스 프로필을 적용하거나 암호 규칙을 적용할 수 없습니다. APEX_*version*의 프로필 및

암호 설정은 Oracle 및 AWS에서 사전 정의하며 Amazon RDS의 보안 요구 사항을 충족하도록 설계되었습니다.

Oracle APEX 및 Oracle Rest Data Services(ORDS) 설정

다음 주제에서는 Oracle APEX 및 ORDS를 설정하는 데 필요한 단계를 나열합니다.

주제

- DB 인스턴스에 APEX 및 APEX-DEV 옵션 추가
- DB 인스턴스의 퍼블릭 사용자 계정 잠금 해제
- Oracle APEX용 RESTful 서비스 구성
- 별도의 호스트에 ORDS 설치 준비
- Oracle APEX Listener 설정

DB 인스턴스에 APEX 및 APEX-DEV 옵션 추가

RDS for Oracle DB 인스턴스에 APEX 및 APEX-DEV 옵션을 추가하려면 다음을 수행합니다.

1. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 복사 또는 수정합니다.
2. 옵션 그룹에 APEX 및 APEX-DEV 옵션을 추가합니다.
3. 옵션 그룹과 DB 인스턴스를 연결합니다.

APEX 및 APEX-DEV 옵션을 추가하는 경우 DB 인스턴스를 자동으로 다시 시작하는 동안 잠시 작동이 중단됩니다.

> **Note**
>
> APEX 옵션이 설치되면 APEX_MAIL을 사용할 수 있습니다. APEX_MAIL 패키지에 대한 실행 권한은 PUBLIC에 부여되므로 APEX 관리자 계정이 없어도 사용할 수 있습니다.

DB 인스턴스에 APEX 및 APEX-DEV 옵션을 추가하는 방법

1. 사용할 옵션 그룹을 결정합니다. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 사용합니다. 기존 옵션 그룹을 사용하려면 다음 단계로 건너뛰십시오. 그렇지 않으면 다음 설정을 사용하여 사용자 지정 DB 옵션을 생성합니다.

a. [Engine]에서 사용할 Oracle 버전을 선택합니다. 모든 버전에서 APEX 및 APEX-DEV 옵션이 지원됩니다.

b. 메이저 엔진 버전에서 DB 인스턴스의 버전을 선택합니다.

자세한 내용은 옵션 그룹 생성 섹션을 참조하세요.

2. 해당 옵션을 옵션 그룹에 추가합니다. Oracle APEX 런타임 환경만 배포하려면 APEX 옵션만 추가합니다. 전체 개발 환경을 배포하려면 APEX 및 APEX-DEV 옵션을 모두 추가합니다.

   버전에서 사용하고자 하는 Oracle APEX 버전을 선택합니다.

   > **Important**
   >
   > 하나 이상의 DB 인스턴스에 이미 연결되어 있는 기존 옵션 그룹에 APEX 또는 APEX-DEV 옵션을 추가하면 인스턴스가 잠시 중단됩니다. 이때 모든 DB 인스턴스가 자동으로 다시 시작됩니다.

   옵션 추가에 대한 자세한 내용은 옵션 그룹에 옵션 추가 섹션을 참조하세요.

3. 옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스에 적용합니다:

   - 새 DB 인스턴스의 경우, 인스턴스를 시작할 때 옵션 그룹을 적용합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
   - 기존 DB 인스턴스의 경우, 해당 인스턴스를 수정하고 새 옵션 그룹을 연결하여 옵션 그룹을 적용합니다. 기존 DB 인스턴스에 APEX 또는 APEX-DEV 옵션을 추가하는 경우 DB 인스턴스를 자동으로 다시 시작하는 동안 인스턴스가 잠시 중단됩니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

DB 인스턴스의 퍼블릭 사용자 계정 잠금 해제

APEX 또는 APEX-DEV 옵션을 DB 인스턴스에 설치한 후 다음 작업을 수행해야 합니다.

1. APEX_PUBLIC_USER 계정의 암호를 변경합니다.
2. 계정의 잠금을 해제합니다.

이 작업은 Oracle SQL*Plus 명령줄 유틸리티를 사용하여 수행할 수 있습니다. DB 인스턴스에 마스터 사용자로 연결하고 다음 명령을 실행합니다. new_password를 원하는 암호로 바꿉니다.

```
ALTER USER APEX_PUBLIC_USER IDENTIFIED BY new_password;
ALTER USER APEX_PUBLIC_USER ACCOUNT UNLOCK;
```

Oracle APEX용 RESTful 서비스 구성

Oracle APEX에 RESTful 서비스를 구성하려면(Oracle APEX 4.1.1.V1에는 필요하지 않음) SQL*Plus를 사용하여 마스터 사용자로서 DB 인스턴스에 연결합니다. 그런 다음 `rdsadmin.rdsadmin_run_apex_rest_config` 저장 프로시저를 실행합니다. 저장 프로시저를 실행할 때 다음 사용자를 위한 암호를 제공합니다.

- `APEX_LISTENER`
- `APEX_REST_PUBLIC_USER`

저장 프로시저는 이들 사용자를 위해 새 데이터베이스 계정을 만드는 `apex_rest_config.sql` 스크립트를 실행합니다.

> **Note**
>
> Oracle APEX 버전 4.1.1.v1을 위한 구성은 필요하지 않습니다. 이 Oracle APEX 버전에 한해 저장 프로시저를 실행할 필요가 없습니다.

다음 명령으로 저장 프로시저를 실행합니다.

```
EXEC rdsadmin.rdsadmin_run_apex_rest_config('apex_listener_password',
 'apex_rest_public_user_password');
```

별도의 호스트에 ORDS 설치 준비

Amazon EC2 인스턴스, 회사의 온프레미스 서버 또는 사용자의 데스크톱 컴퓨터와 같은 별도의 호스트에 ORDS를 설치합니다. 이 섹션의 예제에서는 호스트가 Linux를 실행하고 `myapexhost.example.com` 이름이 지정되었다고 가정합니다.

ORDS를 설치하려면 먼저 권한 없는 OS 사용자를 생성한 다음, Oracle APEX 설치 파일을 다운로드하여 압축을 풀어야 합니다.

ORDS 설치를 준비하려면

1. `myapexhost.example.com`에 `root`로 로그인합니다.

2. 리스너 설치를 소유할 권한 없는 OS 사용자를 생성합니다. 다음 명령은 apexuser라는 새 사용자를 생성합니다.

```
useradd -d /home/apexuser apexuser
```

다음 명령은 새로운 사용자에게 암호를 할당합니다.

```
passwd apexuser;
```

3. myapexhost.example.com에 apexuser로 로그인하고, Oracle의 Oracle APEX 설치 파일을 /home/apexuser 디렉터리에 다운로드합니다.
   - http://www.oracle.com/technetwork/developer-tools/apex/downloads/index.html
   - Oracle Application Express Prior Release Archives
4. /home/apexuser 디렉터리에서 파일의 압축을 풉니다.

```
unzip apex_version.zip
```

파일의 압축을 풀면 apex 디렉터리에 /home/apexuser 디렉터리가 있습니다.

5. myapexhost.example.com에 apexuser로 로그인한 상태에서 Oracle REST Data Services 파일을 Oracle에서 /home/apexuser 디렉터리로 다운로드합니다. http://www.oracle.com/technetwork/developer-tools/apex-listener/downloads/index.html

Oracle APEX Listener 설정

> **Note**
>
> Oracle APEX Listener는 더 이상 사용되지 않습니다.

Amazon RDS for Oracle은 Oracle APEX 버전 4.1.1 및 Oracle APEX Listener 버전 1.1.4를 계속 지원합니다. 지원되는 최신 버전의 Oracle APEX 및 ORDS를 사용하는 것이 좋습니다.

Amazon EC2 인스턴스, 회사의 온프레미스 서버 또는 사용자의 데스크톱 컴퓨터와 같은 별도의 호스트에 Oracle APEX Listener를 설치합니다. 여기에서는 호스트의 이름이 myapexhost.example.com이고 이 호스트에서 Linux를 실행 중이라고 가정합니다.

Oracle APEX Listener 설치 준비

Oracle APEX Listener를 설치하려면 먼저 권한 없는 OS 사용자를 생성한 다음, Oracle APEX 설치 파일을 다운로드하여 압축을 풀어야 합니다.

Oracle APEX Listener 설치를 준비하려면

1. myapexhost.example.com에 root로 로그인합니다.
2. 리스너 설치를 소유할 권한 없는 OS 사용자를 생성합니다. 다음 명령은 apexuser라는 새 사용자를 생성합니다.

   ```
   useradd -d /home/apexuser apexuser
   ```

   다음 명령은 새로운 사용자에게 암호를 할당합니다.

   ```
   passwd apexuser;
   ```

3. myapexhost.example.com에 apexuser로 로그인하고, Oracle의 Oracle APEX 설치 파일을 /home/apexuser 디렉터리에 다운로드합니다.

   - http://www.oracle.com/technetwork/developer-tools/apex/downloads/index.html
   - Oracle Application Express Prior Release Archives

4. /home/apexuser 디렉터리에서 파일의 압축을 풉니다.

   ```
   unzip apex_<version>.zip
   ```

   파일의 압축을 풀면 apex 디렉터리에 /home/apexuser 디렉터리가 있습니다.

5. myapexhost.example.com에 apexuser로 여전히 로그인되어 있는 경우, Oracle의 Oracle APEX 리스너 파일을 /home/apexuser 디렉터리에 다운로드합니다.

Oracle APEX Listener 설치 및 구성

Oracle APEX를 사용하기 전에 apex.war 파일을 다운로드하고 Java를 사용하여 Oracle APEX Listener를 설치한 다음 리스너를 시작해야 합니다.

Oracle APEX Listener를 설치하고 구성하려면

1. Oracle APEX Listener를 기반으로 새 디렉터리를 생성하고 리스너 파일을 엽니다.

다음 코드를 실행합니다.

```
mkdir /home/apexuser/apexlistener
cd /home/apexuser/apexlistener
unzip ../apex_listener.version.zip
```

2. 다음 코드를 실행합니다.

```
java -Dapex.home=./apex -Dapex.images=/home/apexuser/apex/images -Dapex.erase -
jar ./apex.war
```

3. 다음 프로그램 프롬프트에 정보를 입력합니다.

   - APEX Listener 관리자 이름. 기본값은 adminlistener입니다.
   - APEX Listener 관리자의 암호
   - APEX Listener Manger의 사용자 이름입니다. 기본값은 managerlistener입니다.
   - APEX Listener 관리자의 암호

   다음과 같이 구성을 완료하는 데 필요한 URL이 인쇄됩니다.

```
INFO: Please complete configuration at: http://localhost:8080/apex/
listenerConfigure
Database is not yet configured
```

4. Oracle Application Express를 사용할 수 있도록 Oracle APEX Listener를 계속 실행합니다. 이 구성 절차를 마치면 Listener를 백그라운드에서 실행할 수 있습니다.
5. 웹 브라우저에서 Oracle APEX Listener 프로그램에 제공된 URL로 이동합니다. Oracle Application Express Listener 관리 창이 나타납니다. 다음 정보를 입력합니다.

   - Username(사용자 이름)APEX_PUBLIC_USER –
   - Password(암호) – APEX_PUBLIC_USER에 대한 암호입니다. 앞에서 Oracle APEX 리포지토리를 구성할 때 지정한 암호입니다. 자세한 내용은 DB 인스턴스의 퍼블릭 사용자 계정 잠금 해제 섹션을 참조하세요.
   - Connection Type(연결 유형) – 기본
   - Hostname(호스트 이름) – Amazon RDS DB 인스턴스의 엔드포인트입니다(예: mydb.f9rbfa893tft.us-east-1.rds.amazonaws.com).
   - Port(포트) – 1521

- SID – Amazon RDS DB 인스턴스에 있는 데이터베이스의 이름입니다(예: mydb).

6. Apply(적용)를 선택합니다. Oracle APEX 관리 창이 나타납니다.
7. Oracle APEX admin 사용자용 암호를 설정합니다. 이를 위해 SQL*Plus를 사용하여 DB 인스턴스에 마스터 사용자로 연결하고 다음 명령을 실행합니다.

```
EXEC rdsadmin.rdsadmin_util.grant_apex_admin_role;
grant APEX_ADMINISTRATOR_ROLE to master;
@/home/apexuser/apex/apxchpwd.sql
```

   *master*를 마스터 사용자 이름으로 바꿉니다. apxchpwd.sql 스크립트에서 메시지가 표시되면 새 admin 암호를 입력합니다.
8. 브라우저에서 Oracle APEX 관리 창으로 돌아가서 관리를 선택합니다. 그런 다음 [Application Express Internal Administration]을 선택합니다. 자격 증명을 요구하는 메시지가 표시되면 다음 정보를 입력합니다.

   - User name(사용자 이름)admin –
   - Password(암호) – apxchpwd.sql 스크립트를 사용하여 설정한 암호

   [Login]을 선택한 다음 admin 사용자용 새 암호를 설정합니다.

이제 리스너를 사용할 준비가 끝났습니다.

Oracle Rest Data Services(ORDS) 구성

다음 주제에서는 ORDS 21 및 22의 구성 옵션을 나열합니다.

주제

- ORDS 21 이하 설치 및 구성
- ORDS 22 이상 설치 및 구성

ORDS 21 이하 설치 및 구성

이제 Oracle APEX에서 사용하기 위해 Oracle Rest Data Services(ORDS)를 설치하고 구성할 수 있습니다. Oracle APEX 버전 5.0 이상은 ORDS 버전 19.1~21을 사용합니다. ORDS 22 이상을 설치하는 방법에 대해 알아보려면 ORDS 22 이상 설치 및 구성 섹션을 참조하세요.

Amazon EC2 인스턴스, 회사의 온프레미스 서버 또는 사용자의 데스크톱 컴퓨터와 같은 별도의 호스트에 리스너를 설치합니다. 이 단원의 예에서는 호스트의 이름이 `myapexhost.example.com`이고 이 호스트에서 Linux를 실행 중이라고 가정합니다.

Oracle APEX에서 사용할 ORDS 21 이하를 설치 및 구성하려면 다음과 같이 하세요.

1. Oracle REST data services로 이동하여 Readme를 검토합니다. 필요한 버전의 Java를 설치했는지 확인합니다.
2. ORDS 설치를 위한 새 디렉터리를 만듭니다.

```
mkdir /home/apexuser/ORDS
cd /home/apexuser/ORDS
```

3. Oracle REST 데이터 서비스에서 `ords.version.number.zip` 파일을 다운로드합니다.
4. `/home/apexuser/ORDS` 디렉터리에 파일의 압축을 풉니다.
5. 다중 테넌트 데이터베이스에 ORDS를 설치하는 경우 `/home/apexuser/ORDS/params/ords_params.properties` 파일에 다음 줄을 추가합니다.

```
pdb.disable.lockdown=false
```

6. 마스터 사용자에게 ORDS를 설치하는 데 필요한 권한을 부여합니다.

   Oracle APEX 옵션이 설치된 후 마스터 사용자에게 ORDS 스키마를 설치하는 데 필요한 권한을 부여합니다. 이렇게 하려면 데이터베이스에 연결하고 다음 명령을 실행합니다. *MASTER_USER*를 마스터 사용자의 대문자 이름으로 바꿉니다.

   > **Important**
   >
   > 사용자 이름을 입력할 때 대소문자를 구분하는 식별자로 사용자를 생성하지 않는 한 대문자를 사용합니다. 예를 들어 `CREATE USER myuser` 또는 `CREATE USER MYUSER`를 실행하는 경우 데이터 딕셔너리에 `MYUSER`가 저장됩니다. 그러나 `CREATE USER "MyUser"`에서 큰 따옴표를 사용하는 경우 데이터 딕셔너리를 `MyUser`에 저장합니다. 자세한 내용은 SYS 객체에 대한 SELECT 또는 EXECUTE 권한 부여 섹션을 참조하세요.

```
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_OBJECTS', 'MASTER_USER',
 'SELECT', true);
```

```
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_ROLE_PRIVS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_TAB_COLUMNS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_CONS_COLUMNS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_CONSTRAINTS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_OBJECTS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_PROCEDURES', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_TAB_COLUMNS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_TABLES', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_VIEWS', 'MASTER_USER', 'SELECT',
 true);
exec rdsadmin.rdsadmin_util.grant_sys_object('WPIUTL', 'MASTER_USER', 'EXECUTE',
 true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_SESSION', 'MASTER_USER',
 'EXECUTE', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_UTILITY', 'MASTER_USER',
 'EXECUTE', true);
```

> **Note**
>
> 이들 명령은 ORDS 버전 19.1 이상에 적용됩니다.

7. 다운로드한 ords.war 파일을 사용하여 ORDS 스키마를 설치합니다.

```
java -jar ords.war install advanced
```

프로그램에서 다음 정보를 묻는 메시지를 표시합니다. 기본값은 괄호 안에 표시되어 있습니다. 자세한 내용은 Oracle 설명서의 Introduction to Oracle REST Data Services를 참조하세요.

- 구성 데이터를 저장할 위치를 입력합니다.

  */home/apexuser/ORDS*를 입력합니다. ORDS 구성 파일의 위치입니다.

- 사용할 데이터베이스 연결 유형을 지정합니다. [1] 기본 [2] TNS [3] 사용자 지정 URL [1]에 대한 번호를 입력합니다.

원하는 연결 유형을 선택합니다.

- 데이터베이스 서버 [localhost]의 이름 *DB_instance_endpoint*를 입력합니다.

  기본값을 선택하거나 알맞은 값을 입력합니다.

- 데이터베이스 리스너 포트 [1521]: *DB_instance_port*를 입력합니다.

  기본값을 선택하거나 알맞은 값을 입력합니다.

- 데이터베이스 서비스 이름을 지정하려면 1을, 데이터베이스 SID를 지정하려면 2를 입력합니다 [1]:

  데이터베이스 SID를 지정하려면 2를 선택합니다.

- 데이터베이스 SID[xe]

  기본값을 선택하거나 알맞은 값을 입력합니다.

- Oracle REST Data Services 스키마를 확인/설치하려면 1을 입력하고 이 [1]단계를 건너뛰려면 2를 입력합니다.

  를 선택합니다1 이 단계에서는 ORDS_PUBLIC_USER라는 Oracle REST Data Services 프록시 사용자를 생성합니다.

- ORDS_PUBLIC_USER의 데이터베이스 암호를 입력합니다.

  암호를 입력한 다음 확인합니다.

- 관리자 권한으로 로그인하여 Oracle REST Data Services 스키마를 확인해야 합니다.

  관리자 사용자 이름을 입력합니다. *master_user*

  *master_user*의 데이베이스 암호를 입력합니다. *master_user_password*

  암호를 확인합니다. *master_user_password*

  > **Note**
  >
  > 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

- ORDS_METADATA [SYSAUX]에 대한 기본 테이블 스페이스를 입력합니다.

  ORDS_METADATA [TEMP]에 대한 기본 테이블 스페이스를 입력합니다.

ORDS_PUBLIC_USER [USERS]에 대한 기본 테이블 스페이스를 입력합니다.

ORDS_PUBLIC_USER [TEMP]에 대한 기본 테이블 스페이스를 입력합니다.

- PL/SQL 게이트웨이를 사용하려면 1을 입력하고 이 단계를 건너뛰려면 2을 입력합니다 Oracle Application Express를 사용 중이거나 mod_plsql에서 마이그레이션하는 경우 1을 입력해야 합니다[1].

  기본값을 선택합니다.
- PL/SQL 게이트웨이 데이터베이스 사용자 이름 [APEX_PUBLIC_USER]를 입력합니다.

  기본값을 선택합니다.
- APEX_PUBLIC_USER의 데이터베이스 암호를 입력합니다.

  암호를 입력한 다음 확인합니다.
- Application Express RESTful Services 데이터베이스 사용자(APEX_LISTENER, APEX_REST_PUBLIC_USER)를 위한 암호를 지정하려면 1을 입력하고 이 [1]단계를 건너뛰려면 2를 입력합니다.

  APEX 4.1.1.V1의 경우 2를 선택하고 다른 모든 APEX 버전의 경우 1을 선택합니다.
- [APEX 4.1.1.v1에는 필요하지 않음] APEX_LISTENER의 데이터베이스 암호

  암호를 입력한 다음(필요한 경우) 확인합니다.
- [APEX 4.1.1.v1에는 필요하지 않음] APEX_REST_PUBLIC_USER의 데이터베이스 암호

  암호를 입력한 다음(필요한 경우) 확인합니다.
- 숫자를 입력하여 활성화할 기능을 선택합니다.

  SQL 개발자 웹, REST Enabled SQL 및 데이터베이스 API 등 모든 기능을 활성화하려면 1을 입력합니다.
- 독립 실행형 모드로 시작하려면 1을 입력하고 [1]을 종료하려면 2를 입력합니다.

  1를 입력합니다.
- APEX 정적 리소스 위치를 입력합니다.

  `/home/apexuser`에 APEX 설치 파일의 압축을 푼 경우 `/home/apexuser/apex/images`를 입력합니다. 그렇지 않으면 *`unzip_path`*`/apex/images`를 입력합니다. 여기서 *`unzip_path`*는 파일의 압축을 푼 디렉터리입니다.

- HTTP를 사용하는 경우 1을 입력하고 HTTPS [1]를 사용하는 경우 2를 입력합니다.

  1을 입력하는 경우 HTTP 포트를 지정합니다. 2를 입력하는 경우 HTTPS 포트와 SSL 호스트 이름을 지정합니다. HTTPS 옵션은 인증서 제공 방법을 지정하라는 메시지를 표시합니다.

  - 자체 서명된 인증서를 사용하려면 1을 입력합니다.
  - 자신의 인증서를 제공하려면 2를 입력합니다. 2를 입력하는 경우 SSL 인증서의 경로와 SSL 인증서 프라이빗 키의 경로를 지정합니다.

8. APEX admin 사용자용 암호를 설정합니다. 이를 위해 SQL*Plus를 사용하여 DB 인스턴스에 마스터 사용자로 연결하고 다음 명령을 실행합니다.

```
EXEC rdsadmin.rdsadmin_util.grant_apex_admin_role;
grant APEX_ADMINISTRATOR_ROLE to master;
@/home/apexuser/apex/apxchpwd.sql
```

   *master*를 마스터 사용자 이름으로 바꿉니다. apxchpwd.sql 스크립트에서 메시지가 표시되면 새 admin 암호를 입력합니다.

9. ORDS 리스너를 시작합니다. 다음 코드를 실행합니다.

```
java -jar ords.war
```

   ORDS를 처음으로 시작할 때 APEX 고정 리소스의 위치를 제공하라는 메시지가 나타납니다. 이 이미지 폴더는 APEX 설치 디렉터리의 /apex/images 디렉터리에 위치합니다.

10. 브라우저에서 Oracle APEX 관리 창으로 돌아가서 관리를 선택합니다. 그런 다음 [Application Express Internal Administration]을 선택합니다. 자격 증명을 요구하는 메시지가 표시되면 다음 정보를 입력합니다.

    - User name(사용자 이름)admin –
    - Password(암호) – apxchpwd.sql 스크립트를 사용하여 설정한 암호

    [Login]을 선택한 다음 admin 사용자용 새 암호를 설정합니다.

이제 리스너를 사용할 준비가 끝났습니다.

ORDS 22 이상 설치 및 구성

이제 Oracle APEX에서 사용하기 위해 Oracle Rest Data Services(ORDS)를 설치하고 구성할 수 있습니다. 이 섹션의 예제에서는 별도 호스트의 이름이 `myapexhost.example.com`이고 이 호스트에서 Linux를 실행 중이라고 가정합니다. ORDS 22의 지침은 이전 릴리스의 지침과는 다릅니다.

Oracle APEX에서 사용할 ORDS 22 이상을 설치 및 구성하려면 다음과 같이 하세요.

1. Oracle REST data services로 이동하여 다운로드하려는 ORDS 버전에 대한 Readme를 검토합니다. 필요한 버전의 Java를 설치했는지 확인합니다.
2. ORDS 설치를 위한 새 디렉터리를 만듭니다.

   ```
   mkdir /home/apexuser/ORDS
   cd /home/apexuser/ORDS
   ```

3. Oracle REST data services에서 `ords.version.number.zip` 또는 `ords-latest.zip` 파일을 다운로드합니다.
4. `/home/apexuser/ORDS` 디렉터리에 파일의 압축을 풉니다.
5. 마스터 사용자에게 ORDS를 설치하는 데 필요한 권한을 부여합니다.

   APEX 옵션이 설치된 후 마스터 사용자에게 ORDS 스키마를 설치하는 데 필요한 권한을 부여합니다. 이렇게 하려면 데이터베이스에 로깅하여 다음 명령을 실행합니다. *MASTER_USER*를 마스터 사용자의 대문자 이름으로 바꿉니다.

   > **Important**
   >
   > 사용자 이름을 입력할 때 대소문자를 구분하는 식별자로 사용자를 생성하지 않는 한 대문자를 사용합니다. 예를 들어 `CREATE USER myuser` 또는 `CREATE USER MYUSER`를 실행하는 경우 데이터 딕셔너리에 `MYUSER`가 저장됩니다. 그러나 `CREATE USER "MyUser"`에서 큰 따옴표를 사용하는 경우 데이터 딕셔너리를 `MyUser`에 저장합니다. 자세한 내용은 SYS 객체에 대한 SELECT 또는 EXECUTE 권한 부여 섹션을 참조하세요.

   ```
   exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_OBJECTS', 'MASTER_USER',
    'SELECT', true);
   exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_ROLE_PRIVS', 'MASTER_USER',
    'SELECT', true);
   exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_TAB_COLUMNS', 'MASTER_USER',
    'SELECT', true);
   ```

```
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_CONS_COLUMNS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_CONSTRAINTS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_OBJECTS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_PROCEDURES', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_TAB_COLUMNS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_TABLES', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_VIEWS', 'MASTER_USER', 'SELECT',
 true);
exec rdsadmin.rdsadmin_util.grant_sys_object('WPIUTL', 'MASTER_USER', 'EXECUTE',
 true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_SESSION', 'MASTER_USER',
 'EXECUTE', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_UTILITY', 'MASTER_USER',
 'EXECUTE', true);

exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_LOB', 'MASTER_USER', 'EXECUTE',
 true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_ASSERT', 'MASTER_USER',
 'EXECUTE', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_OUTPUT', 'MASTER_USER',
 'EXECUTE', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_SCHEDULER', 'MASTER_USER',
 'EXECUTE', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('HTP', 'MASTER_USER', 'EXECUTE',
 true);
exec rdsadmin.rdsadmin_util.grant_sys_object('OWA', 'MASTER_USER', 'EXECUTE',
 true);
exec rdsadmin.rdsadmin_util.grant_sys_object('WPG_DOCLOAD', 'MASTER_USER',
 'EXECUTE', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_CRYPTO', 'MASTER_USER',
 'EXECUTE', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_METADATA', 'MASTER_USER',
 'EXECUTE', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_SQL', 'MASTER_USER', 'EXECUTE',
 true);
exec rdsadmin.rdsadmin_util.grant_sys_object('UTL_SMTP', 'MASTER_USER', 'EXECUTE',
 true);
```

```
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_NETWORK_ACL_ADMIN',
 'MASTER_USER', 'EXECUTE', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('SESSION_PRIVS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_USERS', 'MASTER_USER', 'SELECT',
 true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_NETWORK_ACL_PRIVILEGES',
 'MASTER_USER', 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_NETWORK_ACLS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_REGISTRY', 'MASTER_USER',
 'SELECT', true);
```

> **Note**
>
> 위의 명령은 ORDS 22 이상에 적용됩니다.

6. 다운로드한 `ords` 스크립트를 사용하여 ORDS 스키마를 설치합니다. 구성 파일 및 로그 파일을 포함할 디렉터리를 지정합니다. Oracle Corporation은 이러한 디렉터리를 ORDS 제품 소프트웨어가 포함되어 있는 디렉터리 내에 배치하지 말 것을 권장합니다.

```
mkdir -p /home/apexuser/ords_config /home/apexuser/ords_logs

/home/apexuser/ORDS/bin/ords \
  --config /home/apexuser/ords_config \
  install --interactive --log-folder /home/apexuser/ords_logs
```

컨테이너 데이터베이스(CDB) 아키텍처를 실행하는 DB 인스턴스의 경우 ORDS 23.2 이상을 사용하고 ORDS를 설치할 때 `--pdb-skip-disable-lockdown` 인수를 전달합니다.

```
/home/apexuser/ORDS/bin/ords \
  --config /home/apexuser/ords_config \
  install --interactive --log-folder /home/apexuser/ords_logs --pdb-skip-disable-
lockdown
```

프로그램에서 다음 정보를 묻는 메시지를 표시합니다. 기본값은 괄호 안에 표시되어 있습니다. 자세한 내용은 Oracle 설명서의 Introduction to Oracle REST Data Services를 참조하세요.

- `Choose the type of installation:`

데이터베이스에 ORDS 스키마를 설치하고 로컬 ORDS 구성 파일에 데이터베이스 연결 풀을 생성하도록 **2**를 선택합니다.

- `Specify the database connection type to use. Enter number for [1] Basic [2] TNS [3] Custom URL:`

  원하는 연결 유형을 선택합니다. 이 예제에서는 사용자가 **1**을 선택한 것으로 가정합니다.

- `Enter the name of the database server [localhost]:` ***`DB_instance_endpoint`***

  기본값 을 선택하거나 적절한 값을 입력합니다.

- `Enter the database listener port [1521]:` ***`DB_instance_port`***

  기본값 **1521**을 선택하거나 적절한 값을 입력합니다.

- `Enter the database service name [orcl]:`

  RDS for Oracle DB 인스턴스에서 사용하는 데이터베이스 이름을 입력합니다.

- `Provide database user name with administrator privileges`

  Oracle DB 인스턴스의 RDS에 마스터 사용자 이름을 입력합니다.

- `Enter the database password for [username]:`

  Oracle DB 인스턴스의 RDS에 마스터 사용자 비밀번호를 입력합니다.

- `Enter the default tablespace for ORDS_METADATA and ORDS_PUBLIC_USER [SYSAUX]:`
- `Enter the temporary tablespace for ORDS_METADATA [TEMP]. Enter the default tablespace for ORDS_PUBLIC_USER [USERS]. Enter the temporary tablespace for ORDS_PUBLIC_USER [TEMP].`
- `Enter a number to select additional feature(s) to enable [1]:`
- `Enter a number to configure and start ORDS in standalone mode [1]:`

  독립형 모드에서 ORDS를 바로 시작하지 않도록 **2**를 선택합니다.

- `Enter a number to select the protocol [1] HTTP`
- `Enter the HTTP port [8080]:`
- `Enter the APEX static resources location:`

  Oracle APEX 설치 파일(`/home/apexuser/apex/images`)에 대한 경로를 입력합니다.

7. Oracle APEX admin 사용자용 암호를 설정합니다. 이를 위해 SQL*Plus를 사용하여 DB 인스턴스에 마스터 사용자로 연결하고 다음 명령을 실행합니다.

```
EXEC rdsadmin.rdsadmin_util.grant_apex_admin_role;
grant APEX_ADMINISTRATOR_ROLE to master;
@/home/apexuser/apex/apxchpwd.sql
```

   *master*를 마스터 사용자 이름으로 바꿉니다. apxchpwd.sql 스크립트에서 메시지가 표시되면 새 admin 암호를 입력합니다.

8. serve 명령과 함께 ords 스크립트를 사용하여 독립형 모드에서 ORDS를 실행합니다. 프로덕션 배포의 경우 Apache Tomcat 또는 Oracle WebLogic Server 등의 지원되는 Java EE 애플리케이션 서버를 사용하는 것이 좋습니다. 자세한 내용은 Oracle 데이터베이스 설명서의 Oracle REST Data Services 배포 및 모니터링을 참조하세요.

```
/home/apexuser/ORDS/bin/ords \
  --config /home/apexuser/ords_config serve \
  --port 8193 \
  --apex-images /home/apexuser/apex/images
```

   ORDS가 실행 중이지만 Oracle APEX 설치에 액세스할 수 없는 경우, 특히 비CDB 인스턴스에서 다음 오류가 표시될 수 있습니다.

```
The procedure named apex_admin could not be accessed, it may not be declared,
 or the user executing this request may not have been granted execute privilege
 on the procedure, or a function specified by security.requestValidationFunction
 configuration property has prevented access.
```

   이 오류를 해결하려면 config 명령으로 ords 스크립트를 실행하여 ORDS에서 사용하는 요청 검증 함수를 변경합니다. 기본적으로 ORDS는 CDB 인스턴스에서만 지원되는 ords_util.authorize_plsql_gateway 프로시저를 사용합니다. 비CDB 인스턴스의 경우 이 프로시저를 wwv_flow_epg_include_modules.authorize 패키지로 변경할 수 있습니다. 사용 사례에 적합한 요청 검증 기능을 구성하는 모범 사례는 Oracle 데이터베이스 설명서 및 Oracle 지원을 참조하세요.

9. 브라우저에서 Oracle APEX 관리 창으로 돌아가서 관리를 선택합니다. 그런 다음 [Application Express Internal Administration]을 선택합니다. 자격 증명을 요구하는 메시지가 표시되면 다음 정보를 입력합니다.

   - User name(사용자 이름)admin –

- Password(암호) – `apxchpwd.sql` 스크립트를 사용하여 설정한 암호

[Login]을 선택한 다음 `admin` 사용자용 새 암호를 설정합니다.

이제 리스너를 사용할 준비가 끝났습니다.

Oracle APEX 업그레이드 및 제거

Oracle APEX를 업그레이드하거나 제거하려면 이 주제의 지침을 따르세요.

주제

- Oracle APEX 버전 업그레이드
- APEX 및 APEX-DEV 옵션 제거

Oracle APEX 버전 업그레이드

> **Important**
>
> Oracle APEX를 업그레이드하기 전에 DB 인스턴스를 백업하세요. 자세한 내용은 Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성 및 Oracle DB 업그레이드 테스트(을)를 참조하세요.

DB 인스턴스와 함께 Oracle APEX를 업그레이드하려면 다음을 실행합니다.

- 업그레이드된 DB 인스턴스 버전의 새 옵션 그룹을 생성합니다.
- `APEX` 및 `APEX-DEV` 옵션의 업그레이드된 버전을 새 옵션 그룹에 추가합니다. DB 인스턴스가 사용하는 다른 옵션도 포함시켜야 합니다. 자세한 내용은 옵션 그룹 고려 사항 섹션을 참조하세요.
- DB 인스턴스를 업그레이드할 때 업그레이드된 DB 인스턴스의 새 옵션 그룹을 지정합니다.

Oracle APEX 버전을 업그레이드한 후에도 이전 버전의 Oracle APEX 스키마가 데이터베이스에 남아 있을 수 있습니다. 더 이상 필요 없는 경우에는 업그레이드 후에 데이터베이스에서 기존 Oracle APEX 스키마를 제거할 수 있습니다.

Oracle APEX 버전을 업그레이드하고 이전 Oracle APEX 버전에서 RESTful 서비스가 구성되지 않은 경우 RESTful 서비스를 구성하는 것이 좋습니다. 자세한 내용은 Oracle APEX용 RESTful 서비스 구성 섹션을 참조하세요.

DB 인스턴스의 메이저 버전 업그레이드를 계획할 때 대상 데이터베이스 버전과 호환되지 않는 Oracle APEX 버전을 사용하는 경우가 있습니다. 이러한 경우 DB 인스턴스를 업그레이드하려면 먼저 Oracle APEX 버전을 업그레이드해야 합니다. Oracle APEX를 먼저 업그레이드하면 DB 인스턴스를 업그레이드하는 시간을 줄일 수 있습니다.

> **Note**
>
> Oracle APEX 업그레이드 이후 업그레이드된 버전과 사용할 리스너를 설치 및 구성합니다. 지침은 Oracle APEX Listener 설정 섹션을 참조하세요.

APEX 및 APEX-DEV 옵션 제거

DB 인스턴스에서 APEX 및 APEX-DEV 옵션을 제거할 수 있습니다. DB 인스턴스에서 이러한 옵션을 제거하려면 다음 중 하나를 수행합니다.

- 여러 DB 인스턴스에서 APEX 및 APEX-DEV 옵션을 제거하려면 DB 인스턴스가 속한 옵션 그룹에서 해당 옵션을 제거합니다. 이 변경은 해당 옵션 그룹을 사용하는 모든 DB 인스턴스에 영향을 미칩니다. 여러 DB 인스턴스에 연결된 옵션 그룹에서 옵션을 제거하면 DB 인스턴스가 다시 시작되는 동안 인스턴스가 잠시 중단됩니다.

  자세한 내용은 옵션 그룹에서 옵션 제거 섹션을 참조하세요.
- 단일 DB 인스턴스에서 APEX 및 APEX-DEV 옵션을 제거하려면 DB 인스턴스를 수정하고 이러한 옵션이 포함되지 않은 다른 옵션 그룹을 지정합니다. 기본(빈) 옵션 그룹을 지정하거나 다른 사용자 지정 옵션 그룹을 지정할 수 있습니다. 이러한 옵션을 제거하는 경우 DB 인스턴스를 자동으로 다시 시작하는 동안 인스턴스가 잠시 중단됩니다.

  자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

DB 인스턴스에서 APEX 및 APEX-DEV 옵션을 제거하면 데이터베이스에서 APEX 스키마가 제거됩니다.

Amazon EFS 통합

Amazon Elastic File System(Amazon EFS)은 완전히 탄력적인 서버리스 파일 스토리지를 제공하므로 스토리지 용량과 성능을 프로비저닝하거나 관리하지 않고도 파일 데이터를 공유할 수 있습니다. Amazon EFS를 사용하면 파일 시스템을 생성한 다음 NFS 버전 4.0 및 4.1(NFSv4) 프로토콜을 통해 VPC에 탑재할 수 있습니다. 그러면 다른 POSIX 호환 파일 시스템처럼 EFS 파일 시스템을 사용할 수 있습니다. 일반적인 정보는 Amazon What is Amazon Elastic File System이란 무엇인가요?와 Amazon RDS for Oracle을 Amazon EFS와 통합 AWS 블로그를 참조하세요.

주제

- Amazon EFS 통합 개요
- Amazon EFS와 RDS for Oracle 통합을 위한 네트워크 권한 구성
- Amazon EFS와 RDS for Oracle 통합을 위한 IAM 권한 구성
- EFS_INTEGRATION 옵션 추가
- Amazon EFS 파일 시스템 권한 구성
- RDS for Oracle DB 인스턴스와 Amazon EFS 파일 시스템 간 파일 전송
- EFS_INTEGRATION 옵션 제거
- Amazon EFS 통합 문제 해결

Amazon EFS 통합 개요

Amazon EFS를 사용하면 RDS for Oracle DB 인스턴스와 EFS 파일 시스템 사이에서 파일을 전송할 수 있습니다. 예를 들어 EFS를 사용하여 다음 사용 사례를 지원할 수 있습니다.

- 애플리케이션과 여러 데이터베이스 서버 간에 파일 시스템을 공유합니다.
- 전송 가능한 테이블스페이스 데이터 파일을 포함하여 마이그레이션 관련 파일을 위한 공유 디렉터리를 생성합니다. 자세한 내용은 Oracle 전송 가능한 테이블스페이스를 사용한 마이그레이션 단원을 참조하십시오.
- 서버에 추가 스토리지 공간을 할당하지 않고 아카이브된 재실행 로그 파일을 저장하고 공유합니다.
- 파일 읽기 및 쓰기를 위해 `UTL_FILE`과 같은 Oracle Database 유틸리티를 사용합니다.

Amazon EFS 통합의 이점

다른 데이터 전송 솔루션 대신 EFS 파일 시스템을 선택하면 다음과 같은 이점을 얻을 수 있습니다.

- Amazon EFS와 RDS for Oracle DB 인스턴스 사이에서 Oracle Data Pump 파일을 전송할 수 있습니다. Data Pump는 EFS 파일 시스템에서 직접 가져오기 때문에 이러한 파일을 로컬로 복사할 필요가 없습니다. 자세한 내용은 Amazon RDS의 Oracle로 데이터 가져오기 단원을 참조하십시오.
- 데이터베이스 링크를 사용하는 것보다 데이터 마이그레이션이 빠릅니다.
- 파일을 보관할 스토리지 공간을 RDS for Oracle DB 인스턴스에 할당하지 않아도 됩니다.
- EFS 파일 시스템은 프로비저닝할 필요 없이 스토리지의 규모를 자동으로 조정할 수 있습니다.
- Amazon EFS 통합에는 최소 수수료나 설정 비용이 없습니다. 사용한 만큼만 지불합니다.
- Amazon EFS 통합은 전송 중인 데이터의 암호화와 미사용 시 암호화라는 두 가지 형태의 암호화를 지원합니다. 전송 중인 데이터의 암호화는 기본적으로 TLS 버전 1.2를 사용하여 활성화됩니다. Amazon EFS 파일 시스템을 생성할 때 저장 데이터 암호화를 활성화할 수 있습니다. 자세한 내용을 알아보려면 Amazon Elastic File System User Guide(Amazon Elastic File System 사용 설명서)의 Encrypting data at rest(저장 데이터 암호화)를 참조하세요.

Amazon EFS 통합을 위한 요구 사항

다음 요구 사항을 충족하는지 확인합니다.

- 데이터베이스가 데이터베이스 버전 19.0.0.0.ru-2022-07.rur-2022-07.r1 이상을 실행해야 합니다.
- DB 인스턴스와 EFS 파일 시스템은 동일한 AWS 리전, VPC 및 AWS 계정에 있어야 합니다. RDS for Oracle은 EFS에 대한 교차 계정 및 교차 리전 액세스를 지원하지 않습니다.
- VPC에 DNS 확인 및 DNS 호스트 이름이 모두 있어야 합니다. 자세한 내용은 Amazon Virtual Private Cloud 사용 설명서의 VPC의 DNS 속성을 참조하세요.
- EFS 파일 시스템은 Standard 또는 Standard-IA 스토리지 클래스를 사용해야 합니다.
- `mount` 명령에서 DNS 이름을 사용하는 경우 Amazon에서 제공하는 DNS 서버를 사용하도록 VPC가 구성되어 있어야 합니다. 사용자 지정 DNS 서버는 지원되지 않습니다.
- RDS가 아닌 솔루션을 사용하여 EFS 파일 시스템을 백업해야 합니다. RDS for Oracle은 EFS 파일 시스템의 자동 백업 또는 수동 DB 스냅샷을 지원하지 않습니다. 자세한 내용은 Amazon EFS 파일 시스템 백업을 참조하세요.

Amazon EFS와 RDS for Oracle 통합을 위한 네트워크 권한 구성

RDS for Oracle을 Amazon EFS와 통합하려면 DB 인스턴스에 EFS 파일 시스템에 대한 네트워크 액세스 권한이 있어야 합니다. 자세한 내용은 Amazon Elastic File System 사용 설명서의 NFS 클라이언트의 Amazon EFS 파일 시스템에 대한 네트워크 액세스 제어를 참조하세요.

주제

- 보안 그룹을 통한 네트워크 액세스 제어
- 파일 시스템 정책을 통한 네트워크 액세스 제어

보안 그룹을 통한 네트워크 액세스 제어

VPC 보안 그룹과 같은 네트워크 계층 보안 메커니즘을 사용하여 EFS 파일 시스템에 대한 DB 인스턴스 액세스를 제어할 수 있습니다. DB 인스턴스의 EFS 파일 시스템에 대한 액세스를 허용하려면 EFS 파일 시스템이 다음 요구 사항을 충족하는지 확인하세요.

- EFS 탑재 대상은 RDS for Oracle DB 인스턴스에서 사용하는 모든 가용 영역에 있습니다.

  EFS 탑재 대상이 EFS 파일 시스템을 탑재할 수 있는 NFSv4 엔드포인트의 IP 주소를 제공해야 합니다. DB 인스턴스의 가용 영역에서 사용하는 EFS 탑재 대상의 IP 주소로 변환되는 DNS 이름을 사용하여 파일 시스템을 탑재해야 합니다.

  여러 AZ에 있는 DB 인스턴스에서 동일한 EFS 파일 시스템을 사용하도록 구성할 수 있습니다. 다중 AZ의 경우 배포의 각 AZ에 대한 탑재 지점이 필요합니다. DB 인스턴스를 다른 AZ로 이동해야 할 수도 있습니다. 이러한 이유를 고려하여, VPC 내의 각 AZ에 EFS 탑재 지점을 생성하는 것이 좋습니다. 기본적으로 콘솔을 사용하여 새 EFS 파일 시스템을 생성하면 RDS는 모든 AZ에 대한 탑재 대상을 생성합니다.
- 보안 그룹이 탑재 대상에 연결되어 있습니다.
- 보안 그룹에는 TCP/2049(유형 NFS)에서 RDS for Oracle DB 인스턴스의 네트워크 서브넷 또는 보안 그룹을 허용하는 인바운드 규칙이 있습니다.

자세한 내용은 Amazon Elastic File System 사용 설명서의 Amazon EFS 파일 시스템 생성 및 EFS 탑재 대상 및 보안 그룹 생성 및 관리를 참조하세요.

파일 시스템 정책을 통한 네트워크 액세스 제어

RDS for Oracle과의 Amazon EFS 통합은 기본(빈) EFS 파일 시스템 정책을 사용하여 작동합니다. 기본 정책은 인증에 IAM을 사용하지 않습니다. 대신 탑재 대상을 사용하여 파일 시스템에 연결할 수 있는 익명 클라이언트에 대한 전체 액세스 권한을 부여합니다. 기본 정책은 파일 시스템 생성 시를 포함하여 사용자가 구성한 파일 시스템 정책이 존재하지 않을 때마다 적용됩니다. 자세한 내용은 Amazon Elastic File System 사용 설명서에서 기본 EFS 파일 시스템 정책을 참조하세요.

RDS for Oracle을 비롯한 모든 클라이언트의 EFS 파일 시스템에 대한 액세스를 강화하기 위해 IAM 권한을 구성할 수 있습니다. 이 방법에서는 파일 시스템 정책을 생성합니다. 자세한 내용은 Amazon Elastic File System 사용 설명서에서 파일 시스템 정책 생성을 참조하세요.

Amazon EFS와 RDS for Oracle 통합을 위한 IAM 권한 구성

기본적으로 Amazon EFS 통합 기능은 IAM 역할을 사용하지 않습니다. `USE_IAM_ROLE` 옵션 설정은 `FALSE`입니다. RDS for Oracle을 Amazon EFS 및 IAM 역할과 통합하려면 DB 인스턴스에는 Amazon EFS 파일 시스템에 액세스할 수 있는 IAM 권한이 있어야 합니다.

주제

- 1단계: DB 인스턴스의 IAM 역할 생성 및 정책 연결
- 2단계: Amazon EFS 파일 시스템의 파일 시스템 정책 생성
- 3단계: IAM 역할을 RDS for Oracle DB 인스턴스와 연결

1단계: DB 인스턴스의 IAM 역할 생성 및 정책 연결

이 단계에서는 Amazon RDS가 EFS 파일 시스템에 액세스할 수 있도록 RDS for Oracle DB 인스턴스의 역할을 생성합니다.

콘솔

Amazon RDS가 EFS 파일 시스템에 액세스하도록 허용하는 IAM 역할을 생성하는 방법

1. IAM 관리 콘솔을 엽니다.
2. 탐색 창에서 Roles를 선택합니다.
3. Create role(역할 생성)을 선택합니다.
4. AWS 서비스(AWS service)에서 RDS를 선택합니다.
5. 사용 사례 선택을 선택하려면 RDS– 데이터베이스에 역할 추가를 선택하십시오.
6. 다음을 선택합니다.
7. 권한 정책은 추가하지 마세요. 다음을 선택합니다.
8. 역할 이름을 IAM 역할의 이름으로 설정합니다(예: `rds-efs-integration-role`). 설명 값(선택 사항)을 추가할 수도 있습니다.
9. 역할 생성을 선택합니다.

AWS CLI

서비스 권한을 특정 리소스로 제한하려면 리소스 기반 신뢰 관계에 aws:SourceArn 및 aws:SourceAccount 전역 조건 컨텍스트 키를 사용하는 것이 좋습니다. 이는 혼동된 대리자 문제를 방지하는 가장 효과적인 방법입니다.

전역 조건 컨텍스트 키를 모두 사용하고 aws:SourceArn 값에 계정 ID가 포함되도록 할 수 있습니다. 이 경우 aws:SourceAccount 값과 aws:SourceArn 값의 계정이 동일한 문에서 사용될 때 동일한 계정 ID를 사용해야 합니다.

- 단일 리소스에 대한 교차 서비스 액세스를 원하는 경우 aws:SourceArn을 사용하세요.
- 해당 계정의 모든 리소스가 교차 서비스 사용과 연결되도록 허용하려는 경우 aws:SourceAccount를 사용하세요.

신뢰 정책에서는 역할에 액세스하는 리소스의 전체 Amazon 리소스 이름(ARN)이 포함된 aws:SourceArn 전역 조건 컨텍스트 키를 사용해야 합니다.

다음 AWS CLI 명령은 이 목적으로 *rds-efs-integration-role*이라는 역할을 생성합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws iam create-role \
   --role-name rds-efs-integration-role \
   --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                 "aws:SourceAccount": my_account_ID,
                 "aws:SourceArn": "arn:aws:rds:Region:my_account_ID:db:dbname"
             }
         }
       }
     ]
```

```
    }'
```

Windows의 경우:

```
aws iam create-role ^
   --role-name rds-efs-integration-role ^
   --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                 "aws:SourceAccount": my_account_ID,
                 "aws:SourceArn": "arn:aws:rds:Region:my_account_ID:db:dbname"
             }
         }
       }
     ]
   }'
```

자세한 내용은 IAM 사용 설명서의 IAM 사용자에게 권한을 위임하기 위한 역할 생성을 참조하십시오.

2단계: Amazon EFS 파일 시스템의 파일 시스템 정책 생성

이 단계에서는 Amazon EFS 파일 시스템의 파일 시스템 정책을 생성합니다.

EFS 파일 시스템 정책을 생성하거나 편집하는 방법

1. EFS 관리 콘솔을 엽니다.
2. 파일 시스템을 선택합니다.
3. 파일 시스템 페이지에서 파일 시스템 정책을 편집하거나 생성할 파일 시스템을 선택합니다. 해당 파일 시스템의 세부 정보 페이지가 표시됩니다.
4. File system policy(파일 시스템 정책) 탭을 선택합니다.

   정책이 비어 있으면 기본 EFS 파일 시스템 정책이 사용 중인 것입니다. 자세한 내용은 Amazon Elastic File System 사용 설명서에서 기본 EFS 파일 시스템 정책을 참조하세요.

5. 편집을 선택합니다. 파일 시스템 정책 페이지가 나타납니다.
6. 정책 편집기에서 다음과 같은 정책을 입력한 다음 Save(저장)를 선택합니다.

JSON

```
{
    "Version": "2012-10-17",
    "Id": "ExamplePolicy01",
    "Statement": [
        {
            "Sid": "ExampleStatement01",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/rds-efs-integration-
role"
            },
            "Action": [
                "elasticfilesystem:ClientMount",
                "elasticfilesystem:ClientWrite",
                "elasticfilesystem:ClientRootAccess"
            ],
            "Resource": "arn:aws:elasticfilesystem:us-
east-1:123456789012:file-system/fs-1234567890abcdef0"
        }
    ]
}
```

3단계: IAM 역할을 RDS for Oracle DB 인스턴스와 연결

이 단계에서는 IAM 역할을 DB 인스턴스와 연결합니다. 다음 요구 사항에 유의하세요.

- 필수 Amazon EFS 권한 정책이 연결된 IAM 역할에 대한 액세스 권한이 있어야 합니다.
- 한 번에 하나의 IAM 역할만 RDS for Oracle DB 인스턴스에 연결할 수 있습니다.
- 인스턴스의 상태는 사용 가능이어야 합니다.

자세한 내용은 Amazon Elastic File System 사용 설명서의 Amazon EFS ID 및 액세스 관리를 참조하세요.

콘솔

IAM 역할을 RDS for Oracle DB 인스턴스와 연결하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 데이터베이스를 선택합니다.
3. 데이터베이스 인스턴스를 사용할 수 없는 경우 작업을 선택한 수 시작을 선택합니다. 인스턴스 상태가 시작됨으로 표시되면 다음 단계로 이동합니다.
4. 세부 정보를 표시하고자 하는 Oracle DB 인스턴스 이름을 선택합니다.
5. 연결성 및 보안(Connectivity & security) 탭에서 페이지 하단의 IAM 역할 관리(Manage IAM roles) 섹션이 나올 때까지 아래로 스크롤합니다.
6. Add IAM roles to this instance(이 인스턴스에 IAM 역할 추가) 섹션에서 추가할 역할을 선택합니다.
7. Feature(기능)에서 EFS_INTEGRATION을 선택합니다.
8. [Add role]을 선택합니다.

AWS CLI

다음 AWS CLI 명령은 *mydbinstance*라는 Oracle DB 인스턴스에 역할을 추가합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds add-role-to-db-instance \
   --db-instance-identifier mydbinstance \
   --feature-name EFS_INTEGRATION \
   --role-arn your-role-arn
```

Windows의 경우:

```
aws rds add-role-to-db-instance ^
   --db-instance-identifier mydbinstance ^
   --feature-name EFS_INTEGRATION ^
   --role-arn your-role-arn
```

*your-role-arn*을 이전 단계에서 기록한 역할 ARN으로 바꿉니다. EFS_INTEGRATION 옵션에 대해 --feature-name을 지정해야 합니다.

EFS_INTEGRATION 옵션 추가

Amazon RDS for Oracle을 Amazon EFS와 통합하려면 DB 인스턴스가 EFS_INTEGRATION 옵션을 포함하는 옵션 그룹과 연결되어 있어야 합니다.

동일한 옵션 그룹에 속한 여러 Oracle DB 인스턴스는 동일한 EFS 파일 시스템을 공유합니다. 서로 다른 DB 인스턴스가 동일한 데이터에 액세스할 수 있지만 서로 다른 Oracle 디렉터리를 사용하여 액세스를 나눌 수 있습니다. 자세한 정보는 RDS for Oracle DB 인스턴스와 Amazon EFS 파일 시스템 간 파일 전송 섹션을 참조하세요.

콘솔

Amazon EFS 통합을 위한 옵션 그룹을 구성하는 방법

1. 새 옵션 그룹을 만들거나 EFS_INTEGRATION 옵션을 추가할 수 있는 기존 옵션 그룹을 식별하십시오.

   옵션 그룹의 생성에 대한 자세한 내용은 옵션 그룹 생성 단원을 참조하십시오.
2. [EFS_INTEGRATION] 옵션을 옵션 그룹에 추가합니다. EFS_ID 파일 시스템 ID를 지정하고 USE_IAM_ROLE 플래그를 설정해야 합니다.

   자세한 내용은 옵션 그룹에 옵션 추가 단원을 참조하십시오.
3. 다음 방법 중 하나를 사용하여 옵션 그룹을 DB 인스턴스에 연결합니다.

   - 새 Oracle DB 인스턴스를 생성하고 옵션 그룹을 연결합니다. DB 인스턴스 생성에 대한 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
   - 옵션 그룹에 연결하도록 DB 인스턴스를 수정합니다. Oracle DB 인스턴스 수정에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하세요.

AWS CLI

EFS 통합을 위한 옵션 그룹을 구성하는 방법

1. 새 옵션 그룹을 만들거나 EFS_INTEGRATION 옵션을 추가할 수 있는 기존 옵션 그룹을 식별하십시오.

   옵션 그룹의 생성에 대한 자세한 내용은 옵션 그룹 생성 단원을 참조하십시오.

2. [EFS_INTEGRATION] 옵션을 옵션 그룹에 추가합니다.

   예를 들어 다음 AWS CLI 명령은 EFS_INTEGRATION 옵션을 **myoptiongroup**이라는 옵션 그룹에 추가합니다.

   Example

   대상 LinuxmacOS, 또는Unix:

   ```
   aws rds add-option-to-option-group \
      --option-group-name myoptiongroup \
      --options "OptionName=EFS_INTEGRATION,OptionSettings=\
      [{Name=EFS_ID,Value=fs-1234567890abcdef0},{Name=USE_IAM_ROLE,Value=TRUE}]"
   ```

   Windows의 경우:

   ```
   aws rds add-option-to-option-group ^
      --option-group-name myoptiongroup ^
      --options "OptionName=EFS_INTEGRATION,OptionSettings=^
      [{Name=EFS_ID,Value=fs-1234567890abcdef0},{Name=USE_IAM_ROLE,Value=TRUE}]"
   ```

3. 다음 방법 중 하나를 사용하여 옵션 그룹을 DB 인스턴스에 연결합니다.

   - 새 Oracle DB 인스턴스를 생성하고 옵션 그룹을 연결합니다. DB 인스턴스 생성에 대한 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
   - 옵션 그룹에 연결하도록 DB 인스턴스를 수정합니다. Oracle DB 인스턴스 수정에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하세요.

Amazon EFS 파일 시스템 권한 구성

새로 생성된 EFS 파일 시스템에서는 기본적으로 루트 사용자(UID 0)에게만 읽기, 쓰기, 실행 권한이 있습니다. 다른 사용자가 파일 시스템을 수정할 수 있게 하려면 루트 사용자가 다른 사용자에게 액세스 권한을 명시적으로 부여해야 합니다. RDS for Oracle DB 인스턴스 사용자는 others 범주에 속합니다. 자세한 내용은 Amazon Elastic File System 사용 설명서의 NFS(네트워크 파일 시스템) 수준에서 사용자, 그룹, 권한 작업을 참조하세요.

RDS for Oracle DB 인스턴스가 EFS 파일 시스템에 있는 파일을 읽고 쓸 수 있도록 하려면 다음을 수행합니다.

- EFS 파일 시스템을 Amazon EC2 온프레미스 인스턴스에 로컬로 탑재합니다.

- 세분화된 권한을 구성합니다.

예를 들어 `other` 사용자에게 EFS 파일 시스템 루트에 쓸 수 있는 권한을 부여하려면 이 디렉터리에서 `chmod 777`을 실행하세요. 자세한 내용은 Amazon Elastic File System 사용 설명서의 예시 Amazon EFS 파일 시스템 사용 사례 및 권한을 참조하세요.

RDS for Oracle DB 인스턴스와 Amazon EFS 파일 시스템 간 파일 전송

RDS for Oracle 인스턴스와 Amazon EFS 파일 시스템 간에 파일을 전송하려면 하나 이상의 Oracle 디렉터리를 생성하고 DB 인스턴스 액세스를 제어하도록 EFS 파일 시스템 권한을 구성해야 합니다.

주제

- Oracle 디렉터리 생성
- EFS 파일 시스템으로/EFS 파일 시스템에서 데이터 전송: 예제

Oracle 디렉터리 생성

Oracle 디렉터리를 생성하려면 `rdsadmin.rdsadmin_util.create_directory_efs` 프로시저를 사용합니다. 프로시저에는 다음과 같은 파라미터가 있습니다.

파라미터 이름	데이터 형식	기본값	필수	설명
`p_directory_name`	VARCHAR	–	예	Oracle 디렉터리의 이름입니다.
`p_path_on_efs`	VARCHAR	–	예	EFS 파일 시스템의 경로입니다. 경로 이름의 접두사는 `/rdsefs-`*`fsid`*`/` 패턴을 사용합니다. 여기서 *`fsid`*는 EFS 파일 시스템 ID의 자리 표시자입니다. 예를 들어 EFS 파일 시스템의 이름이 `fs-1234567890abcdef0` 이고 이 파일 시스템에 이름이 `mydir`인 하위 디렉터리를 생성하는 경우 다음 값을 지정할 수 있습니다. `/rdsefs-fs-1234567890abcdef0/mydir`

EFS 파일 시스템 fs-1234567890abcdef0에 이름이 /datapump1인 하위 디렉터리를 생성한다고 가정해 보겠습니다. 다음 예제에서는 EFS 파일 시스템의 /datapump1 디렉터리를 가리키는 Oracle 디렉터리 DATA_PUMP_DIR_EFS를 생성합니다. p_path_on_efs 파라미터의 파일 시스템 경로 값에는 문자열 접두사 /rdsefs-가 붙습니다.

```
BEGIN
  rdsadmin.rdsadmin_util.create_directory_efs(
    p_directory_name => 'DATA_PUMP_DIR_EFS',
    p_path_on_efs    => '/rdsefs-fs-1234567890abcdef0/datapump1');
END;
/
```

EFS 파일 시스템으로/EFS 파일 시스템에서 데이터 전송: 예제

다음 예제에서는 Oracle Data Pump를 사용하여 이름이 MY_TABLE인 테이블을 datapump.dmp 파일로 내보냅니다. 이 파일은 EFS 파일 시스템에 있습니다.

```
DECLARE
  v_hdnl NUMBER;
BEGIN
  v_hdnl := DBMS_DATAPUMP.OPEN(operation => 'EXPORT', job_mode => 'TABLE',
 job_name=>null);
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'datapump.dmp',
    directory => 'DATA_PUMP_DIR_EFS',
    filetype  => dbms_datapump.ku$_file_type_dump_file);
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'datapump-exp.log',
    directory => 'DATA_PUMP_DIR_EFS',
    filetype  => dbms_datapump.ku$_file_type_log_file);
  DBMS_DATAPUMP.METADATA_FILTER(v_hdnl,'NAME_EXPR','IN (''MY_TABLE'')');
  DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
/
```

다음 예제에서는 Oracle Data Pump를 사용하여 이름이 MY_TABLE인 테이블을 datapump.dmp 파일로 가져옵니다. 이 파일은 EFS 파일 시스템에 있습니다.

```
DECLARE
```

```
  v_hdnl NUMBER;
BEGIN
  v_hdnl := DBMS_DATAPUMP.OPEN(
    operation => 'IMPORT',
    job_mode  => 'TABLE',
    job_name  => null);
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'datapump.dmp',
    directory => 'DATA_PUMP_DIR_EFS',
    filetype  => dbms_datapump.ku$_file_type_dump_file );
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'datapump-imp.log',
    directory => 'DATA_PUMP_DIR_EFS',
    filetype  => dbms_datapump.ku$_file_type_log_file);
  DBMS_DATAPUMP.METADATA_FILTER(v_hdnl,'NAME_EXPR','IN (''MY_TABLE'')');
  DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
/
```

자세한 내용은 Amazon RDS의 Oracle로 데이터 가져오기 단원을 참조하십시오.

EFS_INTEGRATION 옵션 제거

옵션을 제거하는 단계는 여러 DB 인스턴스에서 EFS_INTEGRATION 옵션을 제거할지 단일 인스턴스에서 제거할지에 따라 달라집니다.

DB 인스턴스 개수	작업	관련 정보
다양함	DB 인스턴스가 속한 옵션 그룹에서 EFS_INTEGRATION 옵션을 제거합니다. 이 변경은 해당 옵션 그룹을 사용하는 모든 인스턴스에 영향을 미칩니다.	옵션 그룹에서 옵션 제거
단일	DB 인스턴스를 수정하고 EFS_INTEGRATION 옵션이 포함되지 않은 다른 옵션 그룹을 지정합니다. 기본(비어 있음) 옵션 그룹을 지정하거나 다른 사용자 지정 옵션 그룹을 지정할 수 있습니다.	Amazon RDS DB 인스턴스 수정

EFS_INTEGRATION 옵션을 제거한 후에는 선택적으로 DB 인스턴스에 연결되어 있던 EFS 파일 시스템을 삭제할 수 있습니다.

Amazon EFS 통합 문제 해결

RDS for Oracle DB 인스턴스는 Amazon EFS 파일 시스템에 대한 연결을 모니터링합니다. 모니터링에서 문제가 감지되면 RDS for Oracle DB 인스턴스는 문제를 수정하고 RDS 콘솔에 이벤트를 게시하려 시도할 수 있습니다. 자세한 내용은 Amazon RDS 이벤트 보기를 참조하세요.

이 섹션의 정보는 Amazon EFS 통합 작업 시 일반적 문제를 진단하고 수정하는 데 도움이 됩니다.

Notification	설명	작업
`The EFS for RDS Oracle instance instance_name isn't available on the primary host. NFS port 2049 of your EFS isn't reachable.`	DB 인스턴스가 EFS 파일 시스템과 통신할 수 없습니다.	다음을 확인하세요. • EFS 파일 시스템이 존재합니다. • EFS 탑재 대상에 연결된 보안 그룹에 TCP/2049(유형 NFS)에서 RDS for Oracle DB 인스턴스의 보안 그룹 또는 네트워크 서브넷을 허용하는 인바운드 규칙이 있습니다.
`The EFS isn't reachable.`	EFS_INTEGRATION 옵션을 설치하는 동안 오류가 발생했습니다.	다음을 확인하세요. • EFS 파일 시스템이 존재합니다. • EFS 탑재 대상에 연결된 보안 그룹에 TCP/2049(유형 NFS)에서 RDS for Oracle DB 인스턴스의 보안 그룹 또는 네트워크 서브넷을 허용하는 인바운드 규칙이 있습니다.

Notification	설명	작업
		• VPC의 enableDns Support 속성이 켜져 있습니다. • VPC에서 Amazon이 제공하는 DNS 서버를 사용하고 있습니다. Amazon EFS 통합은 사용자 지정 DHCP DNS와 함께 작동하지 않습니다.
The associated role with your DB instance wasn't found.	EFS_INTEGRATION 옵션을 설치하는 동안 오류가 발생했습니다.	IAM 역할을 RDS for Oracle DB 인스턴스에 연결했는지 확인합니다.
The associated role with your DB instance wasn't found.	EFS_INTEGRATION 옵션을 설치하는 동안 오류가 발생했습니다. RDS for Oracle은 USE_IAM_ROLE 옵션 설정이 TRUE인 DB 스냅샷에서 복원되었습니다.	IAM 역할을 RDS for Oracle DB 인스턴스에 연결했는지 확인합니다.
The associated role with your DB instance wasn't found.	EFS_INTEGRATION 옵션을 설치하는 동안 오류가 발생했습니다. RDS for Oracle은 USE_IAM_ROLE 옵션 설정이 TRUE인 올인원 CloudFormation 템플릿에서 생성되었습니다.	차선책으로 다음 단계를 따라 완료하세요. 1. IAM 역할 및 기본 옵션 그룹과 함께 DB 인스턴스를 생성하세요. 2. 후속 스택 업데이트 시 EFS_INTEGRATION 옵션과 함께 사용자 지정 옵션 그룹을 추가합니다.
PLS-00302: component 'CREATE_DIRECTORY_EFS' must be declared	이 오류는 Amazon EFS를 지원하지 않는 RDS for Oracle 버전을 사용할 때 발생할 수 있습니다.	RDS for Oracle DB 인스턴스 버전 19.0.0.0.ru-2022-07.rur-2022-07.r1 이상을 사용하고 있는지 확인하세요.

Notification	설명	작업
`Read access of your EFS is denied. Check your file system policy.`	DB 인스턴스가 EFS 파일 시스템을 읽을 수 없습니다.	EFS 파일 시스템이 IAM 역할을 통해 또는 EFS 파일 시스템 수준에서 읽기 액세스를 허용하는지 확인하세요.
N/A	DB 인스턴스가 EFS 파일 시스템에 쓸 수 없습니다.	다음 단계를 따릅니다. 1. EFS 파일 시스템이 Amazon EC2 인스턴스에 탑재되어 있는지 확인합니다. 2. RDS 사용자에게 `others` 그룹 쓰기 액세스 권한을 부여합니다. 가장 간단한 방법은 EFS 파일 시스템의 최상위 디렉터리에서 `chmod 777` 명령을 실행하는 것입니다.

Oracle Java 가상 머신

Amazon RDS는 JVM 옵션을 사용함으로써 Oracle Jave 가상 머신(JVM)을 지원합니다. Oracle Java는 Oracle 데이터베이스에서 Oracle Java 기능을 지원하는 SQL 스키마 및 함수를 제공합니다. 자세한 내용은 Oracle 설명서의 Oracle 데이터베이스에 Jave 도입 단원을 참조하십시오. 모든 버전의 Oracle Database 21c(21.0.0) 및 Oracle Database 19c(19.0.0)에서 Oracle JVM을 사용할 수 있습니다.

Oracle JVM 고려 사항

Amazon RDS의 Java 구현에는 권한 집합이 제한되어 있습니다. 마스터 사용자에게는 RDS_JAVA_ADMIN 역할이 부여되며, JAVA_ADMIN 역할에 의해 부여된 권한의 하위 집합이 부여됩니다. RDS_JAVA_ADMIN 역할에 부여된 권한을 나열하려면 DB 인스턴스에서 다음 쿼리를 실행하십시오.

```
SELECT * FROM dba_java_policy
   WHERE grantee IN ('RDS_JAVA_ADMIN', 'PUBLIC')
   AND enabled = 'ENABLED'
   ORDER BY type_name, name, grantee;
```

Oracle JVM 사전 조건

다음은 Oracle Java 사용 시 사전 조건입니다.

- DB 인스턴스 클래스는 충분히 커야 합니다. Oracle Java는 db.t3.small DB 인스턴스 클래스에 대해 지원되지 않습니다. 자세한 내용은 DB 인스턴스 클래스 단원을 참조하십시오.
- DB 인스턴스는 마이너 버전 자동 업그레이드가 활성화되어 있어야 합니다. 이 옵션을 사용하면 DB 인스턴스가 마이너 DB 엔진 버전 업그레이드를 사용할 수 있을 때 자동으로 수신할 수 있습니다. Amazon RDS는 이 옵션을 사용하여 DB 인스턴스를 최신 Oracle Patch Set Update(PSU) 또는 Release Update(RU)로 업데이트합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

Oracle JVM의 모범 사례

다음은 Oracle Java 사용에 관한 모범 사례입니다.

- 보안을 극대화하기 위해 Secure Sockets Layer(SSL)와 함께 JVM 옵션을 사용합니다. 자세한 내용은 Oracle 보안 소켓 Layer 섹션을 참조하세요.
- DB 인스턴스를 구성하여 네트워크 액세스를 제한하십시오. 자세한 내용은 VPC에서 DB 인스턴스에 액세스하는 시나리오 및 VPC에서 DB 인스턴스를 사용한 작업 단원을 참조하십시오.

- 다음 조건을 충족하는 경우 TLSv1.2를 지원하도록 HTTPS 엔드포인트의 구성을 업데이트합니다.
  - Oracle Java Virtual Machine(JVM)을 사용하여 TLSv1 또는 TLSv1.1 프로토콜을 통해 HTTPS 엔드포인트를 연결합니다.
  - 이 엔드포인트는 TLSv1.2 프로토콜을 지원하지 않습니다.
  - 2021년 4월 릴리스 업데이트가 Oracle DB에 적용되지 않았습니다.

  엔드포인트 구성을 업데이트하면 JVM과 HTTPS 엔드포인트의 연결이 계속해서 작동합니다. Oracle JRE 및 JDK의 TLS 변경 사항에 대한 자세한 내용은 Oracle JRE 및 JDK 암호화 로드맵을 참조하세요.

Oracle JVM 옵션 추가

`JVM` 옵션을 DB 인스턴스에 추가하는 일반적인 프로세스는 다음과 같습니다.

1. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 복사 또는 수정합니다.
2. [] 옵션을 옵션 그룹에 추가합니다.
3. 옵션 그룹을 DB 인스턴스에 연동시킵니다.

`JVM` 옵션을 추가하는 동안 잠시 작동이 중단됩니다. 옵션을 추가한 후 DB 인스턴스를 재시작할 필요가 없습니다. 옵션 그룹이 활성화되는 즉시 Oracle java를 사용 가능합니다.

> **Note**
>
> 이 중단 기간 동안에는 암호 확인 기능이 잠시 비활성화됩니다. 중단 기간 중에 암호 확인 기능과 관련된 이벤트를 볼 수도 있습니다. Oracle DB 인스턴스를 사용하기 전에 암호 확인 기능이 다시 활성화됩니다.

DB 인스턴스에 JVM 옵션을 추가하려면

1. 사용할 옵션 그룹을 결정합니다. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 사용합니다. 기존 옵션 그룹을 사용하려면 다음 단계로 건너뛰십시오. 그렇지 않으면 다음 설정을 사용하여 사용자 지정 DB 옵션을 생성합니다.
   - 엔진에는 DB 인스턴스에서 사용하는 DB 엔진을 선택하십시오(oracle-ee, oracle-se, oracle-se1 또는 oracle-se2).
   - 메이저 엔진 버전에서 DB 인스턴스의 버전을 선택합니다.

자세한 내용은 옵션 그룹 생성 섹션을 참조하세요.

2. [JVM] 옵션을 옵션 그룹에 추가합니다. 옵션 추가에 대한 자세한 내용은 옵션 그룹에 옵션 추가 섹션을 참조하세요.
3. 옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스에 적용합니다:
   - 새 DB 인스턴스의 경우, 인스턴스를 시작할 때 옵션 그룹을 적용하십시오. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
   - 기존 DB 인스턴스의 경우, 해당 인스턴스를 수정하고 새 옵션 그룹을 연결하여 옵션 그룹을 적용하십시오. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.
4. 필수 권한을 사용자에게 부여하십시오.

   Amazon RDS 마스터 사용자는 기본값으로 JVM 옵션을 사용하는 권한을 가져야 합니다. 다른 사용자가 이러한 사용 권한을 필요로 하는 경우 SQL 클라이언트에서 DB 인스턴스에 마스터 사용자로 연결하고 사용자에게 사용 권한을 부여하십시오.

   다음 예제에서는 JVM 사용자에게 test_proc 옵션 사용 권한을 부여합니다.

```
create user test_proc identified by password;
CALL dbms_java.grant_permission('TEST_PROC',
 'oracle.aurora.security.JServerPermission', 'LoadClassInPackage.*', '');
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

   사용자에게 권한이 부여되면 다음 쿼리는 출력을 반환해야 합니다.

```
select * from dba_java_policy where grantee='TEST_PROC';
```

> **Note**
>
> Oracle 사용자 이름은 대소문자를 구분하며 일반적으로 모두 대문자입니다.

Oracle JVM 옵션 제거

DB 인스턴스에서 JVM 옵션을 제거할 수 있습니다. 옵션을 제거하는 동안 잠시 작동이 중단됩니다. JVM 옵션을 제거한 후 DB 인스턴스를 재시작할 필요가 없습니다.

> **Warning**
>
> JVM 옵션을 삭제했을 때 DB 인스턴스가 옵션의 일부로 활성화되어 있는 데이터 형식을 사용하고 있다면 데이터가 손실될 수 있습니다. 따라서 처리 전에 데이터를 백업해야 합니다. 자세한 내용은 데이터 백업, 복원 및 내보내기 섹션을 참조하세요.

DB 인스턴스에서 JVM 옵션을 제거하려면 다음 중 하나를 수행합니다.

- 소속 옵션 그룹에서 JVM 옵션을 제거합니다. 이 변경은 해당 옵션 그룹을 사용하는 모든 DB 인스턴스에 영향을 미칩니다. 자세한 내용은 옵션 그룹에서 옵션 제거 섹션을 참조하세요.
- DB 인스턴스를 수정하고 JVM 옵션이 포함되지 않은 다른 옵션 그룹을 지정합니다. 이 변경은 단일 DB 인스턴스에 영향을 미칩니다. 기본(빈) 옵션 그룹을 지정하거나 다른 사용자 지정 옵션 그룹을 지정할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

Oracle Enterprise Manager

Amazon RDS는 Oracle Enterprise Manager(OEM)를 지원합니다. OEM은 엔터프라이즈 정보 기술의 통합 관리를 위한 Oracle 제품 라인입니다.

Amazon RDS는 Oracle Database 19c 비CDB 또는 CDB에서만 OEM을 지원합니다. 다음 표에서는 지원되는 OEM 옵션에 대해 설명합니다.

옵션	옵션 ID	지원되는 OEM 릴리스
OEM Database Express	`OEM`	OEM Database Express 19c
OEM Management Agent	`OEM_AGENT`	OEM Cloud Control for 13c

> **Note**
>
> OEM 데이터베이스 또는 OEM Management Agent를 사용할 수 있지만 둘 다 사용할 수는 없습니다.

Oracle Enterprise Manager Database Express

Amazon RDS는 OEM 옵션 사용을 통해 Oracle Enterprise Manager Database Express(EM Express)를 지원합니다. Amazon RDS는 CDB 또는 비CDB 아키텍처를 사용하여 Oracle Database 19c용 EM Express를 지원합니다.

EM Express는 데이터베이스에 포함된 웹 기반 데이터베이스 관리 도구이며 개방되었을 때만 사용할 수 있습니다. 주요 성능 관리 및 기본 데이터베이스 관리 기능을 지원합니다. 자세한 내용은 Oracle 설명서의 Introduction to Oracle Enterprise Manager Database Express를 참조하세요.

> **Note**
>
> EM Express는 db.t3.small DB 인스턴스 클래스에서 지원되지 않습니다. DB 인스턴스 클래스에 대한 자세한 내용은 RDS for Oracle DB 인스턴스 클래스 섹션을 참조하세요.

OEM 옵션 설정

Amazon RDS는 OEM 옵션에 대해 다음 설정을 지원합니다.

옵션 설정	유효한 값	설명
포트	정수 값	EM Express를 수신하는 RDS for Oracle DB 인스턴스의 포트입니다. 기본값은 5500입니다.
보안 그룹	—	Port(포트)에 액세스할 수 있는 보안 그룹입니다.

1단계: OEM 옵션 추가

OEM 옵션을 DB 인스턴스에 추가하는 일반적인 프로세스는 다음과 같습니다.

1. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 복사 또는 수정합니다.
2. 옵션을 옵션 그룹에 추가합니다.
3. 옵션 그룹과 DB 인스턴스를 연결합니다.

OEM 옵션을 추가하는 경우 DB 인스턴스를 자동으로 다시 시작하는 동안 잠시 작동이 중단됩니다.

DB 인스턴스에 OEM 옵션을 추가하려면

1. 사용할 옵션 그룹을 결정합니다. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 사용합니다. 기존 옵션 그룹을 사용하려면 다음 단계로 건너뛰십시오. 그렇지 않으면 다음 설정을 사용하여 사용자 지정 DB 옵션을 생성합니다.

   a. [Engine]에서 DB 인스턴스의 Oracle 에디션을 선택합니다.

   b. 메이저 엔진 버전에서 DB 인스턴스의 버전을 선택합니다.

   자세한 내용은 옵션 그룹 생성 섹션을 참조하세요.

2. OEM 옵션을 옵션 그룹에 추가하고 옵션 설정을 구성합니다. 옵션 추가에 대한 자세한 내용은 옵션 그룹에 옵션 추가 섹션을 참조하세요. 각 설정에 대한 자세한 내용은 OEM 옵션 설정 단원을 참조하십시오.

   > **Note**
   >
   > 하나 이상의 DB 인스턴스에 이미 연결되어 있는 기존 옵션 그룹에 OEM 옵션을 추가하면 모든 DB 인스턴스가 자동으로 다시 시작되는 동안 인스턴스가 잠시 중단됩니다.

3. 옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스에 적용합니다:

   - 새 DB 인스턴스의 경우, 인스턴스를 시작할 때 옵션 그룹을 적용하십시오. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
   - 기존 DB 인스턴스의 경우, 해당 인스턴스를 수정하고 새 옵션 그룹을 연결하여 옵션 그룹을 적용하십시오. OEM 옵션을 추가하는 경우 DB 인스턴스를 자동으로 다시 시작하는 동안 잠시 작동이 중단됩니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

> **Note**
>
> AWS CLI를 사용하여 OEM 옵션을 추가할 수도 있습니다. 예제는 옵션 그룹에 옵션 추가을 참조하세요.

2단계: (CDB만 해당) DBSNMP 사용자 계정 잠금 해제

DB 인스턴스는 CDB 아키텍처를 사용하므로, EM Express에 DBSNMP로 로그인해야 합니다. DBSNMP는 CDB에서 일반 사용자입니다. 기본적으로 이 계정은 잠겨 있습니다. DB 인스턴스가 CDB 아키텍처를 사용하지 않는 경우 이 단계를 건너뛰세요.

CDB 인스턴스에서 DBSNMP 사용자 계정을 잠금 해제하려면

1. SQL*Plus 또는 다른 Oracle SQL 애플리케이션에서 마스터 사용자로 DB 인스턴스에 로그인합니다.
2. 다음 저장 프로시저를 실행하여 DBSNMP 계정을 잠금 해제합니다.

```
EXEC rdsadmin.rdsadmin_util.reset_oem_agent_password('new_password');
```

   프로시저가 존재하지 않는다는 오류 메시지가 표시되면 CDB 인스턴스를 재부팅하여 자동으로 설치합니다. 자세한 내용은 DB 인스턴스 재부팅 단원을 참조하십시오.

3단계: 브라우저를 통해 EM Express에 액세스

웹 브라우저에서 EM Express에 액세스하면 사용자 이름과 암호를 묻는 로그인 창이 나타납니다.

브라우저를 통해 EM Express에 액세스하려면

1. 사용자의 Amazon RDS DB 인스턴스에 대한 엔드포인트와 EM Express 포트를 식별합니다. 사용자의 Amazon RDS DB 인스턴스에 대한 엔드포인트를 찾는 방법은 RDS for Oracle DB 인스턴스의 엔드포인트 찾기 단원을 참조하십시오.
2. 브라우저 로케이터 표시줄에 다음 형식을 사용하여 URL을 입력합니다.

```
https://endpoint.rds.amazonaws.com:port/em
```

   예를 들어, Amazon RDS DB 인스턴스의 엔드포인트가 mydb.a1bcde234fgh.us-east-1.rds.amazonaws.com이고 EM Express 포트가 1158인 경우 다음 URL을 사용하여 EM Express에 액세스합니다.

```
https://mydb.f9rbfa893tft.us-east-1.rds.amazonaws.com:1158/em
```

3. 로그인 세부 정보를 입력하라는 메시지가 표시되면 데이터베이스 아키텍처에 따라 다음 작업 중 하나를 수행합니다.

데이터베이스가 CDB가 아닙니다.

DB 인스턴스에 대한 마스터 사용자 이름과 마스터 암호를 입력합니다.

데이터베이스가 CDB입니다.

사용자 및 DBSNMP 암호에 DBSNMP를 입력합니다. Container 필드는 비워둡니다.

OEM Database 설정 수정

OEM Database를 활성화한 후 옵션의 보안 그룹 설정을 수정할 수 있습니다.

옵션 그룹을 DB 인스턴스와 연동한 이후에는 OEM 포트 번호를 수정할 수 없습니다. DB 인스턴스의 OEM 포트 번호를 변경하려면 다음과 같이 합니다.

1. 새 옵션 그룹을 생성합니다.
2. 새 포트 번호가 포함된 OEM 옵션에 새 옵션 그룹을 추가합니다.
3. DB 인스턴스에서 기존 옵션 그룹을 제거합니다.
4. 새 옵션 그룹을 DB 인스턴스에 추가합니다.

옵션 설정을 변경하는 방법에 대한 자세한 내용은 옵션 설정 수정을(를) 참조하십시오. 각 설정에 대한 자세한 내용은 OEM 옵션 설정 단원을 참조하십시오.

OEM Database Express 작업 실행

Amazon RDS 절차를 통해 특정 OEM Database Express 작업을 실행할 수 있습니다. 이 프로시저를 실행하면 다음과 같은 작업을 수행할 수 있습니다.

> **Note**
>
> OEM Database Express 작업은 비동기식으로 실행됩니다.

작업

- OEM Database Express의 웹 사이트 프런트 엔드를 Adobe Flash로 전환
- OEM Database Express의 웹 사이트 프런트 엔드를 Oracle JET로 전환

OEM Database Express의 웹 사이트 프런트 엔드를 Adobe Flash로 전환

> **Note**
>
> 이 작업은 Oracle Database 19c 비 CDB의 경우에만 가능합니다.

Oracle Database 19c부터 Oracle은 Adobe Flash 기반의 이전 OEM Database Express 사용자 인터페이스를 더 이상 사용하지 않습니다. 대신, 이제 OEM Database Express에서 Oracle JET로 구축된 인터페이스를 사용합니다. 새 인터페이스 사용에 문제가 있는 경우 더 이상 사용하지 않는 Flash 기반 인터페이스로 다시 전환할 수 있습니다. 새 인터페이스 사용 시 발생할 수 있는 문제에는 OEM Database Express에 로그인한 후 `Loading` 화면이 멈추는 것이 포함됩니다. Flash 기반의 OEM Database Express 버전에 있는 특정 기능을 사용하지 못할 수도 있습니다.

OEM Database Express 웹 사이트 프런트 엔드를 Adobe Flash로 전환하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_oem_tasks.em_express_frontend_to_flash`를 실행하십시오. 이 프로시저는 `execemx  emx` SQL 명령과 동일합니다.

보안 모범 사례에서는 Adobe Flash의 사용을 권장하지 않습니다. Flash 기반의 OEM Database Express로 되돌릴 수 있지만 가능하면 JET 기반의 OEM Database Express 웹 사이트를 사용하는 것이 좋습니다. Adobe Flash를 사용하도록 되돌리고 Oracle JET를 사용하도록 다시 전환하려면 `rdsadmin.rdsadmin_oem_tasks.em_express_frontend_to_jet` 프로시저를 사용하십시오. Oracle Database 업그레이드 후 최신 버전의 Oracle JET에서 OEM Database Express의 JET 관련 문제를 해결할 수 있습니다. Oracle JET로의 전환에 대한 자세한 내용은 OEM Database Express의 웹 사이트 프런트 엔드를 Oracle JET로 전환 단원을 참조하십시오.

> **Note**
>
> 읽기 전용 복제본의 원본 DB 인스턴스에서 이 작업을 실행하면 읽기 전용 복제본이 해당 OEM Database Express 웹 사이트 프런트 엔드를 Adobe Flash로 전환하게 됩니다.

다음 프로시저 호출은 OEM Database Express 웹 사이트에서 Adobe Flash로 전환하는 작업을 생성하고 작업 ID를 반환합니다.

```
SELECT rdsadmin.rdsadmin_oem_tasks.em_express_frontend_to_flash() as TASK_ID from DUAL;
```

작업의 출력 파일을 표시하여 결과를 볼 수 있습니다.

```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-task-id.log'));
```

*task-id*를 절차에서 반환된 작업 ID로 대체합니다. Amazon RDS 프로시저 `rdsadmin.rds_file_util.read_text_file`에 대한 자세한 내용은 DB 인스턴스 디렉터리의 파일 목록 읽기 단원을 참조하십시오.

AWS Management Console에 대한 Logs & events(로그 및 이벤트) 섹션에서 로그 항목을 검색하여 `task-id`에서 작업의 출력 파일 내용을 볼 수도 있습니다.

OEM Database Express의 웹 사이트 프런트 엔드를 Oracle JET로 전환

> **Note**
>
> 이 작업은 Oracle Database 19c 비 CDB의 경우에만 가능합니다.

OEM Database Express 웹 사이트 프런트 엔드를 Oracle JET로 전환하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_oem_tasks.em_express_frontend_to_jet`를 실행합니다. 이 프로시저는 `execemx omx` SQL 명령과 동일합니다.

기본적으로 19c 이상을 실행하는 Oracle DB 인스턴스에 대한 OEM Database Express 웹 사이트에서는 Oracle JET를 사용합니다. `rdsadmin.rdsadmin_oem_tasks.em_express_frontend_to_flash` 프로시저를 사용하여 OEM Database Express 웹 사이트 프런트 엔드를 Adobe Flash로 전환한 경우 Oracle JET로 다시 전환할 수 있습니다. 이렇게 하려면 `rdsadmin.rdsadmin_oem_tasks.em_express_frontend_to_jet` 프로시저를 사용하십시오. Adobe Flash로 전환하는 방법에 대한 자세한 내용은 OEM Database Express의 웹 사이트 프런트 엔드를 Adobe Flash로 전환 단원을 참조하십시오.

> **Note**
>
> 읽기 전용 복제본의 원본 DB 인스턴스에서 이 작업을 실행하면 읽기 전용 복제본이 해당 OEM Database Express 웹 사이트 프런트 엔드를 Oracle JET로 전환하게 됩니다.

다음 프로시저 호출은 OEM Database Express 웹 사이트를 Oracle JET로 전환하는 작업을 생성하고 작업 ID를 반환합니다.

```
SELECT rdsadmin.rdsadmin_oem_tasks.em_express_frontend_to_jet() as TASK_ID from DUAL;
```

작업의 출력 파일을 표시하여 결과를 볼 수 있습니다.

```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-task-id.log'));
```

*task-id*를 절차에서 반환된 작업 ID로 대체합니다. Amazon RDS 프로시저 `rdsadmin.rds_file_util.read_text_file`에 대한 자세한 내용은 DB 인스턴스 디렉터리의 파일 목록 읽기 단원을 참조하십시오.

AWS Management Console에 대한 Logs & events(로그 및 이벤트) 섹션에서 로그 항목을 검색하여 `task-id`에서 작업의 출력 파일 내용을 볼 수도 있습니다.

OEM Database 옵션 제거

DB 인스턴스에서 OEM 옵션을 제거할 수 있습니다. OEM 옵션을 제거하는 경우 인스턴스를 자동으로 다시 시작하는 동안 인스턴스가 잠시 중단됩니다. 따라서 OEM 옵션을 제거한 후 DB 인스턴스를 다시 시작할 필요가 없습니다.

DB 인스턴스에서 OEM 옵션을 제거하려면 다음 중 하나를 수행합니다.

- OEM Agent가 속한 옵션 그룹에서 OEM 옵션을 제거합니다. 이 변경은 해당 옵션 그룹을 사용하는 모든 DB 인스턴스에 영향을 미칩니다. 자세한 내용은 옵션 그룹에서 옵션 제거 섹션을 참조하세요.
- DB 인스턴스를 수정하고, OEM 옵션이 포함되지 않은 다른 옵션 그룹을 지정합니다. 이 변경은 단일 DB 인스턴스에 영향을 미칩니다. 기본(빈) 옵션 그룹을 지정하거나 다른 사용자 지정 옵션 그룹을 지정할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

Oracle Management Agent for Enterprise Manager Cloud Control

Oracle Enterprise Manager(OEM) Management Agent는 호스트에서 실행되는 대상을 모니터링하고 해당 정보를 미들티어 Oracle Management Service(OMS)에 전달하는 소프트웨어 구성 요소입니다. Amazon RDS는 `OEM_AGENT` 옵션 사용을 통해 Management Agent를 지원합니다.

자세한 내용은 Oracle 설명서에서 Overview of Oracle Enterprise Manager Cloud Control 12c 및 Overview of Oracle Enterprise Manager Cloud Control 13c 단원을 참조하십시오.

주제

- Management Agent 요구 사항
- OMS 호스트 통신 사전 요구 사항
- Management Agent의 제한 사항
- Management Agent 옵션 설정
- Management Agent 옵션을 DB 인스턴스에 활성화
- Management Agent 옵션 제거
- Management Agent를 사용하여 데이터베이스 작업 수행

Management Agent 요구 사항

다음은 Management Agent 사용을 위한 일반적인 요구 사항입니다.

- DB 인스턴스는 Oracle Database 19c(19.0.0.0)를 실행해야 합니다. CDB 아키텍처 또는 비CDB 아키텍처를 사용할 수 있습니다.
- DB 인스턴스에 연결하도록 구성된 Oracle Management Service(OMS)를 사용해야 합니다. 다음과 같은 OMS 요구 사항을 확인합니다.
  - Management Agent 버전 13.5.0.0.v2 및 13.5.0.0.v3에는 OMS 버전 13.5.0.23 또는 24.1이 필요합니다.
  - Management Agent 버전 13.5.0.0.v1에는 OMS 버전 13.5.0.0 또는 24.1이 필요합니다.
  - Management Agent 버전 13.4.0.9.v1 및 13.4.0.9.v2에는 OMS 버전 13.4.0.9 이상과 32198287 패치가 필요합니다.
- 대부분의 경우 OMS에서 DB 인스턴스로 연결을 허용하도록 VPC를 구성해야 합니다. Amazon Virtual Private Cloud(Amazon VPC)에 익숙하지 않은 경우 계속하기 전에 자습서: DB 인스턴스에 사용할 Amazon VPC 생성(IPv4 전용)의 단계를 완료하는 것이 좋습니다.

- 12c 및 13c용 Oracle Enterprise Manager Cloud Control과 함께 Management Agent를 사용할 수 있습니다. OEM 릴리스를 위한 충분한 스토리지 공간이 있는지 확인합니다.
  - OEM 13c 릴리스 5의 경우 최소 8.5GiB이 필요합니다.
  - OEM 13c 릴리스 4의 경우 최소 8.5GiB
  - OEM 13c 릴리스 3의 경우 최소 8.5GiB
  - OEM 13c 릴리스 2의 경우 최소 5.5GiB
  - OEM 13c 릴리스 1의 경우 최소 4.5GiB가 필요합니다.
  - OEM 12c의 경우 최소 2.5GiB
- Management Agent 버전 `OEM_AGENT 13.2.0.0.v3` 및 `13.3.0.0.v2`를 사용 중이고 TCPS 연결을 사용하려면 Oracle 설명서의 대상 데이터베이스와의 통신을 위한 서드 파티 CA 인증서 구성에 설명된 지침을 따르세요. 또한 Oracle Doc ID가 2241358.1인 Oracle 문서의 지침에 따라 OMS에서 JDK를 업데이트하십시오. 그러면 데이터베이스가 지원하는 모든 암호 그룹을 OMS에서 지원하게 됩니다.

> **Note**
>
> Management Agent와 DB 인스턴스 간 TCPS 연결은 Management Agent `OEM_AGENT 13.2.0.0.v3`, `13.3.0.0.v2`, `13.4.0.9.v1` 이상 버전에서 지원됩니다.

OMS 호스트 통신 사전 요구 사항

OMS 호스트와 Amazon RDS DB 인스턴스가 통신해야 합니다. 다음을 수행합니다.

- OMS 호스트가 방화벽 뒤에 있는 경우 Management Agent에서 OMS 호스트로 연결하려면 DB 인스턴스의 IP 주소를 방화벽에 추가합니다. OMS용 방화벽이 다음 네트워크 트래픽을 허용하도록 합니다.

  OMS 호스트에서 DB 인스턴스로의 기준

  OMS 호스트에서 데이터베이스 리스너 포트(기본값 1521) 및 OEM 에이전트 포트(기본값 3872)로의 트래픽을 허용하는 단방향 방화벽 규칙을 구성합니다.

  DB 인스턴스에서 OMS 호스트로의 기준

  DB 인스턴스에서 OMS HTTP 포트(기본값 4903)로의 트래픽을 허용하는 단방향 방화벽 규칙을 구성합니다.

- OMS에 공개적으로 확인할 수 있는 호스트 이름이 있는 경우 OMS에서 Management Agent로 연결하려면 OMS 주소를 보안 그룹에 추가합니다. 보안 그룹에는 DB 리스너 포트 및 Management Agent 포트에 대한 액세스를 허용하는 인바운드 규칙이 있어야 합니다. 보안을 생성하고 인바운드 규칙을 추가하는 방법의 예는 자습서: DB 인스턴스에 사용할 Amazon VPC 생성(IPv4 전용) 단원을 참조하십시오.
- OMS에 공개적으로 확인할 수 있는 호스트 이름이 없는 경우 OMS에서 Management Agent로 연결하려면 다음 중 하나를 사용합니다.
  - OMS가 프라이빗 VPC의 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스에 호스팅된 경우 VPC 피어링을 설정하여 OMS에서 Management Agent로 연결할 수 있습니다. 자세한 내용은 VPC에 있는 DB 인스턴스에 다른 VPC에 있는 EC2 인스턴스가 액세스 섹션을 참조하세요.
  - OMS가 온프레미스에 호스팅된 경우 VPN 연결을 설정하여 OMS에서 Management Agent로 액세스를 허용할 수 있습니다. 자세한 내용은 클라이언트 애플리케이션이 인터넷을 통해 VPC에 있는 DB 인스턴스에 액세스 또는 VPN 연결을 참조하십시오.
- OEM Management Agent 버전 13.5.0.0(v1~v3)을 24.1 OMS 호스트에 연결하려면 구성 옵션에서 TLS 1.2 프로토콜 `TLSv1.2`를 사용하도록 `MINIMUM_TLS_VERSION`을 설정합니다.

Management Agent의 제한 사항

다음은 Management Agent를 사용할 때 적용되는 몇 가지 제한 사항입니다.

- 사용자 지정 Oracle 관리 에이전트 이미지는 제공할 수 없습니다.
- 작업 실행 및 데이터베이스 패치 적용과 같이 호스트 자격 증명이 필요한 관리 작업은 지원되지 않습니다.
- 호스트 측정치 및 프로세스 목록에는 실제 시스템 상태가 반영되지 않을 수 있습니다. 따라서 OEM을 사용하여 루트 파일 시스템이나 마운트 지점 파일 시스템을 모니터링하면 안 됩니다. 운영 체제 모니터링에 대한 자세한 내용은 Enhanced Monitoring을 사용하여 OS 지표 모니터링 단원을 참조하십시오.
- 자동 검색은 지원되지 않습니다. 데이터베이스 대상을 수동으로 추가해야 합니다.
- OMS 모듈 가용성은 데이터베이스 에디션에 따라 다릅니다. 예를 들어, 데이터베이스 성능 진단 및 튜닝 모듈은 Oracle Database Enterprise Edition에만 사용할 수 있습니다.
- Management Agent는 추가 메모리 및 컴퓨팅 리소스를 사용합니다. `OEM_AGENT` 옵션을 활성화한 후 성능 문제가 발생할 경우 더 큰 DB 인스턴스 클래스로 조정하는 것이 좋습니다. 자세한 내용은 DB 인스턴스 클래스 및 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

- Amazon RDS 호스트에서 OEM_AGENT를 실행 중인 사용자에게는 경고 로그에 대한 운영 체제 액세스 권한이 없습니다. 따라서 OEM에서 DB Alert Log 및 DB Alert Log Error Status에 대한 지표를 수집할 수 없습니다.

Management Agent 옵션 설정

Amazon RDS는 Management Agent 옵션에 대해 다음 설정을 지원합니다.

옵션 설정	필수	유효한 값	설명
버전 (AGENT_VERSION)	예	13.5.0.0.v3 13.5.0.0.v2 13.5.0.0.v1 13.4.0.9.v2 13.4.0.9.v1 13.3.0.0.v2 13.3.0.0.v1 13.2.0.0.v3 13.2.0.0.v2 13.2.0.0.v1	Management Agent 소프트웨어의 버전입니다. 현재 지원되는 최소 버전은 13.1.0.0.v1 입니다. AWS CLI 옵션 이름은 OptionVersion 입니다. **Note** AWS GovCloud (US) 리전에서는 13.1 버전을 사용할 수 없습니다.

옵션 설정	필수	유효한 값	설명
		13.1.0.0.v1	
포트 (AGENT_PORT)	예	정수 값	OMS 호스트에 대해 수신 대기하는 DB 인스턴스의 포트입니다. 기본값은 3872입니다. OMS 호스트는 이 포트에 액세스할 수 있는 보안 그룹에 속해야 합니다. AWS CLI 옵션 이름은 Port입니다.
보안 그룹	예	기존 보안 그룹	Port(포트)에 액세스할 수 있는 보안 그룹입니다. OMS 호스트는 이 보안 그룹에 속해야 합니다. AWS CLI 옵션 이름은 VpcSecurityGroupMemberships 또는 DBSecurityGroupMemberships 입니다.
OMS_HOST	예	문자열 값, 예: *my.example.oms*	OMS의 공개적으로 액세스할 수 있는 호스트 이름 또는 IP 주소입니다. AWS CLI 옵션 이름은 OMS_HOST입니다.
OMS_PORT	예	정수 값	Management Agent에 대해 수신 대기하는 OMS 호스트의 HTTPS 업로드 포트입니다. HTTPS 업로드 포트를 결정하려면 OMS 호스트를 연결하고 다음 명령을 실행합니다(SYSMAN 암호 필요). `emctl status oms -details` AWS CLI 옵션 이름은 OMS_PORT입니다.

옵션 설정	필수	유효한 값	설명
AGENT_REGISTRATION_PASSWORD	예	문자열 값	Management Agent가 OMS에 자신을 인증하기 위해 사용하는 암호입니다. OEM_AGENT 옵션을 활성화하기 전에 OMS에서 지속적인 암호를 생성하는 것이 좋습니다. 지속적인 암호가 있으면 단일 Management Agent 옵션 그룹을 여러 Amazon RDS 데이터베이스 간에 공유할 수 있습니다. AWS CLI 옵션 이름은 AGENT_REGISTRATION_PASSWORD 입니다.
ALLOW_TLS_ONLY	아니요	true, false(기본값)	에이전트가 서버로서 수신하는 동안 OEM 에이전트가 TLSv1 프로토콜만 지원하도록 구성하는 값입니다. 해당 설정은 더 이상 지원되지 않습니다. Management Agent 버전 13.1.0.0.v1 이상은 기본적으로 TLS(전송 계층 보안)를 지원합니다.
MINIMUM_TLS_VERSION	아니요	TLSv1 (default) TLSv1.2	에이전트가 서버로서 수신하는 동안 OEM 에이전트에서 지원하는 최소 TLS 버전을 지정하는 값입니다. 지원되지 않는 에이전트 버전에서는 TLSv1 설정만 지원합니다. 13.5.0.0(v1~v3)을 24.1 OMS 호스트에 연결하려면 이 값을 TLSv1.2로 설정합니다.
TLS_CIPHER_SUITE	아니요	Management Agent 옵션 설정을(를) 참조하세요.	에이전트가 서버로서 수신하는 동안 OEM 에이전트가 사용하는 TLS 암호 그룹을 지정하는 값입니다.

다음 표에는 관리 에이전트 옵션에서 지원하는 TLS 암호 그룹이 나와 있습니다.

암호 그룹	지원되는 에이전트 버전	FedRAMP 규정 준수
TLS_RSA_WITH_AES_128_CBC_SHA	모두	아니요
TLS_RSA_WITH_AES_128_CBC_SHA256	13.1.0.0.v1 이상	아니요
TLS_RSA_WITH_AES_256_CBC_SHA	13.2.0.0.v3 이상	아니요
TLS_RSA_WITH_AES_256_CBC_SHA256	13.2.0.0.v3 이상	아니요
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	13.2.0.0.v3 이상	예
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	13.2.0.0.v3 이상	예
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	13.2.0.0.v3 이상	예
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	13.2.0.0.v3 이상	예

Management Agent 옵션을 DB 인스턴스에 활성화

Management Agent 옵션을 활성화하려면 다음 단계를 사용합니다.

주제

- 1단계: Management Agent 옵션을 DB 인스턴스에 추가
- 2단계: DBSNMP 사용자 계정 잠금 해제
- 3단계: Management Agent 콘솔에 대상 추가

1단계: Management Agent 옵션을 DB 인스턴스에 추가

DB 인스턴스에 Management Agent 옵션을 추가하려면 다음을 수행하세요.

1. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 복사 또는 수정합니다.
2. [] 옵션을 옵션 그룹에 추가합니다.
3. 옵션 그룹을 DB 인스턴스에 연동시킵니다.

오류가 발생할 경우 My Oracle Support 문서에서 특정 문제를 해결하는 자세한 내용을 확인합니다.

Management Agent 옵션을 추가한 후 DB 인스턴스를 재시작할 필요가 없습니다. 옵션 그룹이 활성화되자마자 OEM Agent가 활성화됩니다.

OMS 호스트가 신뢰할 수 없는 타사 인증서를 사용하는 경우 Amazon RDS가 다음 오류를 반환합니다.

```
You successfully installed the OEM_AGENT option. Your OMS host is using an untrusted
 third party certificate.
Configure your OMS host with the trusted certificates from your third party.
```

이 오류가 반환되면 문제가 해결될 때까지 관리 에이전트 옵션을 사용할 수 없습니다. 문제 해결에 대한 정보는 My Oracle Support 설명서 2202569.1을 참조하십시오.

콘솔

Management Agent 옵션을 DB 인스턴스에 추가하려면

1. 사용할 옵션 그룹을 결정합니다. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 사용합니다. 기존 옵션 그룹을 사용하려면 다음 단계로 건너뛰십시오. 그렇지 않으면 다음 설정을 사용하여 사용자 지정 DB 옵션을 생성합니다.

   a. [Engine]에서 DB 인스턴스의 Oracle 에디션을 선택합니다.

   b. 메이저 엔진 버전에서 DB 인스턴스의 버전을 선택합니다.

   자세한 내용은 옵션 그룹 생성 섹션을 참조하세요.
2. [OEM_AGENT] 옵션을 옵션 그룹에 추가하고 옵션 설정을 구성합니다. 옵션 추가에 대한 자세한 내용은 옵션 그룹에 옵션 추가 섹션을 참조하세요. 각 설정에 대한 자세한 내용은 Management Agent 옵션 설정 단원을 참조하십시오.
3. 옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스에 적용합니다:

   - 새 DB 인스턴스의 경우, 인스턴스를 시작할 때 옵션 그룹을 적용합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.

- 기존 DB 인스턴스의 경우, 해당 인스턴스를 수정하고 새 옵션 그룹을 연결하여 옵션 그룹을 적용합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

AWS CLI

다음 예에서는 AWS CLI add-option-to-option-group 명령을 사용하여 OEM_AGENT 옵션을 myoptiongroup이라는 옵션 그룹에 추가합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds add-option-to-option-group \
    --option-group-name "myoptiongroup" \
    --options
 OptionName=OEM_AGENT,OptionVersion=13.1.0.0.v1,Port=3872,VpcSecurityGroupMemberships=sg-123456
{Name=OMS_PORT,Value=4903},{Name=AGENT_REGISTRATION_PASSWORD,Value=password}] \
    --apply-immediately
```

Windows의 경우:

```
aws rds add-option-to-option-group ^
    --option-group-name "myoptiongroup" ^
    --options
 OptionName=OEM_AGENT,OptionVersion=13.1.0.0.v1,Port=3872,VpcSecurityGroupMemberships=sg-123456
{Name=OMS_PORT,Value=4903},{Name=AGENT_REGISTRATION_PASSWORD,Value=password}] ^
    --apply-immediately
```

2단계: DBSNMP 사용자 계정 잠금 해제

Management Agent는 DBSNMP 사용자 계정을 사용하여 데이터베이스에 연결하고 Oracle Enterprise Manager에 문제를 보고합니다. CDB에서 DBSNMP는 일반 사용자입니다. 이 사용자 계정은 Management Agent와 OEM Database Express 모두에 필요합니다. 기본적으로 이 계정은 잠겨 있습니다. 이 계정의 잠금을 해제하는 절차는 데이터베이스가 비CDB 아키텍처를 사용하는지 아니면 CDB 아키텍처를 사용하는지에 따라 달라집니다.

DBSNMP 사용자 계정을 잠금 해제하려면

1. SQL*Plus 또는 다른 Oracle SQL 애플리케이션에서 마스터 사용자로 DB 인스턴스에 로그인합니다.
2. 데이터베이스 아키텍처에 따라 다음 작업 중 하나를 수행합니다.

데이터베이스가 CDB가 아닙니다.

다음 SQL 문을 실행합니다.

```
ALTER USER dbsnmp IDENTIFIED BY new_password ACCOUNT UNLOCK;
```

데이터베이스가 CDB입니다.

다음 저장 프로시저를 실행하여 DBSNMP 계정을 잠금 해제합니다.

```
EXEC rdsadmin.rdsadmin_util.reset_oem_agent_password('new_password');
```

프로시저가 존재하지 않는다는 오류 메시지가 표시되면 CDB 인스턴스를 재부팅하여 자동으로 설치합니다. 자세한 내용은 DB 인스턴스 재부팅 섹션을 참조하세요.

3단계: Management Agent 콘솔에 대상 추가

DB 인스턴스를 대상으로 추가하려면 엔드포인트와 포트를 알고 있어야 합니다. 사용자의 Amazon RDS DB 인스턴스에 대한 엔드포인트를 찾는 방법은 RDS for Oracle DB 인스턴스의 엔드포인트 찾기 단원을 참조하십시오. 데이터베이스에서 CDB 아키텍처를 사용하는 경우 CDB$ROOT 컨테이너를 별도로 대상으로 추가하세요.

Management Agent 콘솔에 대상을 추가하려면

1. OMS 콘솔에서 [Setup], [Add Target], [Add Targets Manually]를 선택합니다.
2. [Add Targets Declaratively by Specifying Target Monitoring Properties]를 선택합니다.
3. [Target Type]에서 [Database Instance]를 선택합니다.
4. Monitoring Agent(모니터링 에이전트)에서 RDS DB 인스턴스 식별자와 동일한 식별자가 있는 에이전트를 선택합니다.
5. [Add Manually]를 선택합니다.
6. Amazon RDS DB 인스턴스의 엔드포인트를 입력하거나 호스트 이름 목록에서 이를 선택합니다. 지정된 호스트 이름이 Amazon RDS DB 인스턴스의 엔드포인트와 일치하는지 확인합니다.
7. 다음 데이터베이스 속성을 지정합니다.
   - 대상 이름에 이름을 입력합니다.
   - Database system name(데이터베이스 시스템 이름)에 이름을 입력합니다.

- Monitor username(모니터 사용자 이름)에 **dbsnmp**를 입력합니다.
- 모니터 암호에 2단계: DBSNMP 사용자 계정 잠금 해제의 암호를 입력합니다.
- 역할에 normal을 입력합니다.
- Oracle home path(Oracle 홈 경로)에 **/oracle**을 입력합니다.
- [Listener Machine name]에는 에이전트 식별자가 이미 나타나 있습니다.
- 포트에 데이터베이스 포트를 입력합니다. RDS 기본 포트는 1521입니다.
- 데이터베이스 이름에 데이터베이스 이름을 입력합니다. 데이터베이스가 CDB인 경우 이름은 RDSCDB입니다.

8. [Test Connection]을 선택합니다.
9. 다음을 선택합니다. 모니터링되는 리소스 목록에 대상 데이터베이스가 나타납니다.

Management Agent 옵션 제거

DB 인스턴스에서 OEM Agent를 제거할 수 있습니다. OEM Agent를 제거한 후 DB 인스턴스를 재시작할 필요가 없습니다.

DB 인스턴스에서 OEM Agent를 제거하려면 다음 중 하나를 수행합니다.

- OEM Agent가 속한 옵션 그룹에서 OEM Agent 옵션을 제거합니다. 이 변경은 해당 옵션 그룹을 사용하는 모든 DB 인스턴스에 영향을 미칩니다. 자세한 내용은 옵션 그룹에서 옵션 제거 섹션을 참조하세요.
- DB 인스턴스를 수정하고, OEM Agent 옵션이 포함되지 않은 다른 옵션 그룹을 지정합니다. 이 변경은 단일 DB 인스턴스에 영향을 미칩니다. 기본(빈) 옵션 그룹을 지정하거나 다른 사용자 지정 옵션 그룹을 지정할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

Management Agent를 사용하여 데이터베이스 작업 수행

Amazon RDS 프로시저를 사용하여 Management Agent에서 특정 EMCTL 명령을 실행할 수 있습니다. 이 프로시저를 실행하면 다음과 같은 작업을 수행할 수 있습니다.

> **Note**
>
> 작업은 비동기식으로 실행됩니다.

업무

- Management Agent 보호
- Management Agent의 상태 가져오기
- Management Agent 다시 시작
- Management Agent가 모니터링하는 대상 나열
- Management Agent가 모니터링하는 수집 스레드 나열
- Management Agent 상태 지우기
- Management Agent에서 OMS 업로드
- OMS에 대해 ping 실행
- 진행 중인 작업 상태 보기

Management Agent 보호

Management Agent를 보호하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_oem_agent_tasks.secure_oem_agent`를 실행합니다. 이 프로시저는 `emctl secure agent` 명령을 실행하는 것과 동일합니다.

다음 프로시저에서는 Management Agent를 보호하는 작업을 생성하고 작업 ID를 반환합니다.

```
SELECT rdsadmin.rdsadmin_oem_agent_tasks.secure_oem_agent as TASK_ID from DUAL;
```

작업의 출력 파일을 표시하여 결과를 보려면 진행 중인 작업 상태 보기 섹션을 참조하세요.

Management Agent의 상태 가져오기

Management Agent의 상태를 가져오려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_oem_agent_tasks.get_status_oem_agent`를 실행합니다. 이 프로시저는 `emctl status agent` 명령과 동일합니다.

다음 절차에서는 Management Agent의 상태를 가져오는 작업을 생성하고 작업 ID를 반환합니다.

```
SELECT rdsadmin.rdsadmin_oem_agent_tasks.get_status_oem_agent() as TASK_ID from DUAL;
```

작업의 출력 파일을 표시하여 결과를 보려면 진행 중인 작업 상태 보기 섹션을 참조하세요.

Management Agent 다시 시작

Management Agent를 다시 시작하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_oem_agent_tasks.restart_oem_agent`를 실행합니다. 이 프로시저는 `emctl stop agent` 및 `emctl start agent` 명령을 실행하는 것과 동일합니다.

다음 절차에서는 Management Agent를 다시 시작하는 작업을 생성하고 작업 ID를 반환합니다.

```
SELECT rdsadmin.rdsadmin_oem_agent_tasks.restart_oem_agent as TASK_ID from DUAL;
```

작업의 출력 파일을 표시하여 결과를 보려면 진행 중인 작업 상태 보기 섹션을 참조하세요.

Management Agent가 모니터링하는 대상 나열

Management Agent가 모니터링하는 대상을 나열하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_oem_agent_tasks.list_targets_oem_agent`를 실행합니다. 이 프로시저는 `emctl config agent listtargets` 명령을 실행하는 것과 동일합니다.

다음 절차에서는 Management Agent가 모니터링하는 대상을 나열하는 작업을 생성하고 작업 ID를 반환합니다.

```
SELECT rdsadmin.rdsadmin_oem_agent_tasks.list_targets_oem_agent as TASK_ID from DUAL;
```

작업의 출력 파일을 표시하여 결과를 보려면 진행 중인 작업 상태 보기 섹션을 참조하세요.

Management Agent가 모니터링하는 수집 스레드 나열

Management Agent가 모니터링하는 실행 중, 준비된 및 예약된 수집 스레드를 모두 나열하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_oem_agent_tasks.list_clxn_threads_oem_agent`를 실행합니다. 이 프로시저는 `emctl status agent scheduler` 명령과 동일합니다.

다음 절차에서는 컬렉션 스레드를 나열하는 작업을 생성하고 작업 ID를 반환합니다.

```
SELECT rdsadmin.rdsadmin_oem_agent_tasks.list_clxn_threads_oem_agent() as TASK_ID from
 DUAL;
```

작업의 출력 파일을 표시하여 결과를 보려면 진행 중인 작업 상태 보기 섹션을 참조하세요.

Management Agent 상태 지우기

Management Agent 상태를 지우려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_oem_agent_tasks.clearstate_oem_agent`를 실행합니다. 이 프로시저는 `emctl clearstate agent` 명령을 실행하는 것과 동일합니다.

다음 절차에서는 Management Agent의 상태를 지우는 작업을 생성하고 작업 ID를 반환합니다.

```
SELECT rdsadmin.rdsadmin_oem_agent_tasks.clearstate_oem_agent() as TASK_ID from DUAL;
```

작업의 출력 파일을 표시하여 결과를 보려면 진행 중인 작업 상태 보기 섹션을 참조하세요.

Management Agent에서 OMS 업로드

Management Agent가 관련 OMS(Oracle Management Server)를 업로드하도록 하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_oem_agent_tasks.upload_oem_agent`를 실행합니다. 이 프로시저는 `emclt upload agent` 명령을 실행하는 것과 동일합니다.

다음 절차에서는 Management Agent가 연관된 OMS를 업로드하는 작업을 생성하고 작업 ID를 반환합니다.

```
SELECT rdsadmin.rdsadmin_oem_agent_tasks.upload_oem_agent() as TASK_ID from DUAL;
```

작업의 출력 파일을 표시하여 결과를 보려면 진행 중인 작업 상태 보기 섹션을 참조하세요.

OMS에 대해 ping 실행

Management Agent의 OMS에 대해 ping을 실행하려면 Amazon RDS 프로시저 `rdsadmin.rdsadmin_oem_agent_tasks.ping_oms_oem_agent`를 실행합니다. 이 프로시저는 `emctl pingOMS` 명령을 실행하는 것과 동일합니다.

다음 절차에서는 Management Agent의 OMS에 대해 ping을 수행하는 작업을 생성하고 작업 ID를 반환합니다.

```
SELECT rdsadmin.rdsadmin_oem_agent_tasks.ping_oms_oem_agent() as TASK_ID from DUAL;
```

작업의 출력 파일을 표시하여 결과를 보려면 진행 중인 작업 상태 보기 섹션을 참조하세요.

진행 중인 작업 상태 보기

진행 중인 작업의 상태를 bdump 파일에서 볼 수 있습니다. 이 bdump 파일은 `/rdsdbdata/log/trace` 디렉터리에 위치합니다. 각 bdump 파일 이름은 다음 형식으로 되어 있습니다.

```
dbtask-task-id.log
```

작업을 모니터링하려면 *task-id*를 모니터링하려는 작업의 ID로 바꾸십시오.

bdump 파일의 콘텐츠를 보려면 Amazon RDS 프로시저 `rdsadmin.rds_file_util.read_text_file`을 실행합니다. 다음 쿼리는 `dbtask-1546988886389-2444.log` bdump 파일의 콘텐츠를 반환합니다.

```
SELECT text FROM
 table(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-1546988886389-2444.log'));
```

Amazon RDS 프로시저 `rdsadmin.rds_file_util.read_text_file`에 대한 자세한 내용은 DB 인스턴스 디렉터리의 파일 목록 읽기 단원을 참조하십시오.

Oracle 레이블 보안

Amazon RDS에서는 OLS 옵션을 사용하여 Oracle Database의 Enterprise Edition에 대한 Oracle 레이블 보안을 지원합니다.

대부분의 데이터베이스 보안에서는 객체 수준에서 액세스를 제어합니다. Oracle 레이블 보안에서는 개별 테이블 행에 대한 액세스를 세부적으로 제어합니다. 예를 들어 레이블 보안을 사용하면 정책 기반 관리 모델을 통해 규제 준수를 이행할 수 있습니다. 레이블 보안 정책을 사용하여 중요 데이터에 대한 액세스를 제어하고 적절한 권한을 가진 사용자로 액세스를 제한할 수 있습니다. 자세한 내용은 Oracle 설명서의 Introduction to Oracle Label Security를 참조하십시오.

주제

- Oracle 레이블 보안 요구 사항
- Oracle 레이블 보안 사용 시 고려 사항
- Oracle 레이블 보안 옵션 추가
- 문제 해결

Oracle 레이블 보안 요구 사항

Oracle 레이블 보안에 대한 다음 요구 사항을 숙지하세요.

- DB 인스턴스에서 기본 보유 라이선스 사용 모델을 사용해야 합니다. 자세한 내용은 RDS for Oracle 라이선스 옵션 섹션을 참조하세요.
- Oracle Enterprise Edition에 유효한 라이선스(소프트웨어 업데이트 라이선스 및 지원 포함)가 있어야 합니다.
- Oracle 라이선스에 레이블 보안 옵션이 포함되어 있어야 합니다.

Oracle 레이블 보안 사용 시 고려 사항

Oracle 레이블 보안을 사용하려면 테이블의 특정 행에 대한 액세스를 제어하는 정책을 생성합니다. 자세한 내용은 Oracle 설명서의 Creating an Oracle Label Security Policy를 참조하십시오.

다음을 고려하세요.

- Oracle 레이블 보안은 영구적이고 지속적인 옵션입니다. 이 옵션은 영구적이므로 옵션 그룹에서 제거할 수 없습니다. Oracle 레이블 보안을 옵션 그룹에 추가하고 이를 DB 인스턴스와 연결하면 나중

에 다른 옵션 그룹을 DB 인스턴스와 연결할 수 있지만 이 그룹에는 Oracle 레이블 보안 옵션도 포함되어야 합니다.

- 레이블 보안을 사용할 경우 모든 작업을 LBAC_DBA 역할로 수행합니다. LBAC_DBA 역할은 DB 인스턴스에 대한 마스터 사용자에게 부여됩니다. 다른 사용자에게 레이블 보안 정책을 관리할 수 있는 LBAC_DBA 역할을 부여할 수 있습니다.
- Oracle 레이블 보안에 액세스해야 하는 새로운 사용자에게 OLS_ENFORCEMENT 패키지에 대한 액세스 권한을 부여해야 합니다. OLS_ENFORCEMENT 패키지에 대한 액세스 권한을 부여하려면 마스터 사용자로 DB 인스턴스에 연결하고 다음 SQL 문을 실행합니다.

```
GRANT ALL ON LBACSYS.OLS_ENFORCEMENT TO username;
```

- Oracle Enterprise Manager(OEM) Cloud Control을 통해 레이블 보안을 구성할 수 있습니다. Amazon RDS는 Management Agent 옵션을 통해 OEM Cloud Control을 지원합니다. 자세한 내용은 Oracle Management Agent for Enterprise Manager Cloud Control 단원을 참조하십시오.

Oracle 레이블 보안 옵션 추가

Oracle 레이블 보안 옵션을 DB 인스턴스에 추가하는 일반적인 프로세스는 다음과 같습니다.

1. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 복사 또는 수정합니다.
2. 옵션을 옵션 그룹에 추가합니다.

> **Important**
>
> Oracle 레이블 보안은 영구적이고 지속적인 옵션입니다.

3. 옵션 그룹을 DB 인스턴스에 연결합니다.

레이블 보안 옵션을 추가하면 옵션 그룹이 활성화되고 레이블 보안이 활성화됩니다.

DB 인스턴스에 레이블 보안 옵션을 추가하려면

1. 사용할 옵션 그룹을 결정합니다. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 사용합니다. 기존 옵션 그룹을 사용하려면 다음 단계로 건너뛰십시오. 그렇지 않으면 다음 설정을 사용하여 사용자 지정 DB 옵션을 생성합니다.

   a. [Engine]에서 [oracle-ee]를 선택합니다.

   b. 메이저 엔진 버전에서 DB 인스턴스의 버전을 선택합니다.

자세한 내용은 옵션 그룹 생성 섹션을 참조하세요.

2. [OLS] 옵션을 옵션 그룹에 추가합니다. 옵션 추가에 대한 자세한 내용은 옵션 그룹에 옵션 추가 섹션을 참조하세요.

> **Important**
>
> 하나 이상의 DB 인스턴스에 이미 연결되어 있는 기존 옵션 그룹에 레이블 보안을 추가하면 모든 DB 인스턴스가 다시 시작됩니다.

3. 옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스에 적용합니다:

   - 새 DB 인스턴스의 경우, 인스턴스를 시작할 때 옵션 그룹을 적용합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
   - 기존 DB 인스턴스의 경우, 해당 인스턴스를 수정하고 새 옵션 그룹을 연결하여 옵션 그룹을 적용합니다. 기존 DB 인스턴스에 레이블 보안 옵션을 추가하는 경우 DB 인스턴스를 자동으로 다시 시작하는 동안 인스턴스가 잠시 중단됩니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

문제 해결

다음은 Oracle 레이블 보안을 사용할 때 생길 수 있는 문제입니다.

문제	문제 해결 제안
정책을 생성하려고 하면 다음과 비슷한 오류 메시지가 표시됩니다. `insufficient authorization for the SYSDBA package.`	사용자의 이름이 16자 또는 24자인 경우 레이블 보안 명령을 실행할 수 없는 Oracle 레이블 보안 기능의 알려진 문제입니다. 새 사용자의 이름 문자 수를 변경하고 새 사용자에게 LBAC_DBA를 부여한 다음 새 사용자로 로그인하여 OLS 명령을 실행할 수 있습니다. 자세한 내용은 Oracle Support에 문의하세요.

Oracle Locator

Amazon RDS는 LOCATOR 옵션 사용을 통해 Oracle Locator를 지원합니다. Oracle Locator에는 인터넷 및 무선 서비스 기반 애플리케이션과 파트너 기반 GIS 솔루션을 지원할 때 필요한 기능이 있습니다. Oracle Locator는 Oracle Spatial의 제한된 서브셋입니다. 자세한 내용은 Oracle 문서의 Oracle Locator를 참조하십시오.

> **Important**
>
> Oracle Locator를 사용하면 CVSS(공통 취약성 평가 시스템) 점수가 9 이상인 보안 취약성 또는 기타 발표된 보안 취약성이 있는 경우 Amazon RDS에서 DB 인스턴스를 최신 Oracle PSU로 자동 업데이트합니다.

Oracle Locator에서 지원되는 데이터베이스 릴리스

RDS for Oracle은 Oracle Database 19c용 Oracle Locator를 지원합니다. Oracle Locator는 Oracle Database 21c에서 지원되지 않지만 관련 기능을 Oracle Spatial 옵션에서 사용할 수 있습니다. 이전에는 Spatial 옵션을 사용하려면 추가 라이선스가 필요했습니다. Oracle Locator는 Oracle Spatial 기능의 하위 집합으로서 추가 라이선스가 필요하지 않았습니다. 2019년에 Oracle은 모든 Oracle Spatial 기능이 추가 비용 없이 Enterprise Edition 및 Standard Edition 2 라이선스에 포함되었다고 발표했습니다. 따라서 Oracle Spatial 옵션에 더 이상 추가 라이선스가 필요하지 않습니다. 자세한 내용은 Oracle Database Insider 블로그의 Machine Learning, Spatial and Graph - No License Required!(기계 학습, Spatial 및 그래프 - 라이선스 불필요!)를 참조하세요.

Oracle Locator의 사전 요구 사항

Oracle Locator 사용을 위한 사전 요구 사항은 다음과 같습니다.

- DB 인스턴스 클래스는 충분해야 합니다. Oracle Locator는 db.t3.small DB 인스턴스 클래스에 대해 지원되지 않습니다. 자세한 내용은 RDS for Oracle DB 인스턴스 클래스 단원을 참조하십시오.
- DB 인스턴스는 마이너 버전 자동 업그레이드가 활성화되어 있어야 합니다. 이 옵션을 사용하면 DB 인스턴스를 활성화하여 사용 가능할 때 마이너 DB 엔진 버전 업그레이드를 자동으로 받을 수 있으며, Oracle Java Virtual Machine(JVM)을 설치하는 옵션에 필요합니다. Amazon RDS는 이 옵션을 사용하여 DB 인스턴스를 최신 Oracle Patch Set Update(PSU) 또는 Release Update(RU)로 업데이트합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

Oracle Locator의 모범 사례

다음은 Oracle Locator 사용에 관한 모범 사례입니다.

- 보안을 극대화하기 위해 Secure Sockets Layer(SSL)와 함께 LOCATOR 옵션을 사용합니다. 자세한 내용은 Oracle 보안 소켓 Layer 섹션을 참조하세요.
- DB 인스턴스에 대한 액세스를 제한하도록 DB 인스턴스를 구성합니다. 자세한 내용은 VPC에서 DB 인스턴스에 액세스하는 시나리오 및 VPC에서 DB 인스턴스를 사용한 작업 단원을 참조하십시오.

Oracle Locator 옵션 추가

LOCATOR 옵션을 DB 인스턴스에 추가하는 일반적인 프로세스는 다음과 같습니다.

1. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 복사 또는 수정합니다.
2. [] 옵션을 옵션 그룹에 추가합니다.
3. 옵션 그룹을 DB 인스턴스에 연동시킵니다.

Oracle Java Virtual Machine(JVM)이 DB 인스턴스에 설치되어 있지 않은 경우 LOCATOR 옵션이 추가되는 동안 잠시 중단이 발생합니다. Oracle Java Virtual Machine(JVM)이 DB 인스턴스에 이미 설치되어 있으면 중단이 발생하지 않습니다. 옵션을 추가한 후 DB 인스턴스를 재시작할 필요가 없습니다. 옵션 그룹이 활성화되는 즉시 Oracle Locator를 사용할 수 있습니다.

> **Note**
>
> 이 중단 기간 동안에는 암호 확인 기능이 잠시 비활성화됩니다. 중단 기간 중에 암호 확인 기능과 관련된 이벤트를 볼 수도 있습니다. Oracle DB 인스턴스를 사용하기 전에 암호 확인 기능이 다시 활성화됩니다.

LOCATOR 옵션을 DB 인스턴스에 추가하려면

1. 사용할 옵션 그룹을 결정합니다. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 사용합니다. 기존 옵션 그룹을 사용하려면 다음 단계로 건너뛰십시오. 그렇지 않으면 다음 설정을 사용하여 사용자 지정 DB 옵션을 생성합니다.

   a. [Engine]에서 DB 인스턴스의 Oracle 에디션을 선택합니다.

   b. 메이저 엔진 버전에서 DB 인스턴스의 버전을 선택합니다.

자세한 내용은 옵션 그룹 생성 섹션을 참조하세요.

2. [LOCATOR] 옵션을 옵션 그룹에 추가합니다. 옵션 추가에 대한 자세한 내용은 옵션 그룹에 옵션 추가 섹션을 참조하세요.
3. 옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스에 적용합니다:

   - 새 DB 인스턴스의 경우, 인스턴스를 시작할 때 옵션 그룹을 적용합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
   - 기존 DB 인스턴스의 경우, 해당 인스턴스를 수정하고 새 옵션 그룹을 연결하여 옵션 그룹을 적용합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

Oracle Locator 사용

Oracle Locator 옵션을 활성화한 후에 사용할 수 있습니다. Oracle Locator 기능만 사용해야 합니다. Oracle Spatial 기능을 사용하려면 Oracle Spatial 라이선스가 있어야 합니다.

Oracle Locator에 지원되는 기능 목록은 Oracle 설명서에서 Locator에 포함되는 기능을 참조하십시오.

Oracle Locator에 지원되지 않는 기능 목록은 Oracle 설명서에서 Locator에 포함되지 않는 기능을 참조하십시오.

Oracle Locator 옵션 제거

LOCATOR 옵션에서 제공하는 데이터 유형을 사용하는 모든 객체를 삭제한 후 DB 인스턴스에서 옵션을 삭제할 수 있습니다. Oracle Java Virtual Machine(JVM)이 DB 인스턴스에 설치되어 있지 않은 경우 LOCATOR 옵션이 제거되는 동안 잠시 중단이 발생합니다. Oracle Java Virtual Machine(JVM)이 DB 인스턴스에 이미 설치되어 있으면 중단이 발생하지 않습니다. LOCATOR 옵션을 제거한 후 DB 인스턴스를 재시작할 필요가 없습니다.

LOCATOR 옵션을 삭제하려면

1. 데이터를 백업합니다.

   > **Warning**
   >
   > 인스턴스에서 옵션의 일부로 활성화된 데이터 유형을 사용하고 LOCATOR 옵션을 제거하면 데이터가 손실될 수 있습니다. 자세한 내용은 데이터 백업, 복원 및 내보내기 섹션을 참조하세요.

2. 기존 객체가 LOCATOR 옵션의 데이터 유형 또는 기능을 참조하는지 확인합니다.

   LOCATOR 옵션이 있는 경우 LOCATOR 옵션이 없는 새 옵션 그룹을 적용할 때 인스턴스가 중단될 수 있습니다. 다음 쿼리를 사용하여 객체를 식별할 수 있습니다.

```
SELECT OWNER, SEGMENT_NAME, TABLESPACE_NAME, BYTES/1024/1024 mbytes
FROM   DBA_SEGMENTS
WHERE  SEGMENT_TYPE LIKE '%TABLE%'
AND    (OWNER, SEGMENT_NAME) IN
       (SELECT DISTINCT OWNER, TABLE_NAME
        FROM   DBA_TAB_COLUMNS
        WHERE  DATA_TYPE='SDO_GEOMETRY'
        AND    OWNER <> 'MDSYS')
ORDER BY 1,2,3,4;

SELECT OWNER, TABLE_NAME, COLUMN_NAME
FROM   DBA_TAB_COLUMNS
WHERE  DATA_TYPE = 'SDO_GEOMETRY'
AND    OWNER <> 'MDSYS'
ORDER BY 1,2,3;
```

3. LOCATOR 옵션의 데이터 유형 또는 기능을 참조하는 모든 객체를 삭제합니다.
4. 다음 중 하나를 수행하십시오.

   - 소속 옵션 그룹에서 LOCATOR 옵션을 제거합니다. 이 변경은 해당 옵션 그룹을 사용하는 모든 DB 인스턴스에 영향을 미칩니다. 자세한 내용은 옵션 그룹에서 옵션 제거 섹션을 참조하세요.
   - DB 인스턴스를 수정하고 LOCATOR 옵션이 포함되지 않은 다른 옵션 그룹을 지정합니다. 이 변경은 단일 DB 인스턴스에 영향을 미칩니다. 기본(빈) 옵션 그룹을 지정하거나 다른 사용자 지정 옵션 그룹을 지정할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

Oracle 기본 네트워크 암호화

Amazon RDS는 Oracle 기본 네트워크 암호화(NNE)를 지원합니다. NATIVE_NETWORK_ENCRYPTION 옵션을 사용하면 DB 인스턴스 간 이동 시 데이터를 암호화할 수 있습니다. Amazon RDS는 Oracle 데이터베이스의 모든 버전에 대해 NNE를 지원합니다.

Oracle 기본 네트워크 암호화에 대한 자세한 설명은 본 문서의 범위에서 벗어나지만 배포할 솔루션을 결정하려면 각 알고리즘과 키의 장단점은 잘 알고 있어야 합니다. Oracle 기본 네트워크 암호화를 통해 제공되는 알고리즘 및 키에 대한 자세한 내용은 Oracle 설명서의 Configuring Network Data Encryption을 참조하십시오. AWS 보안에 대한 자세한 내용은 AWS 보안 센터를 참조하세요.

> **Note**
>
> Native Network Encryption 또는 Secure Sockets Layer를 사용할 수 있지만 둘 다 사용할 수는 없습니다. 자세한 내용은 Oracle 보안 소켓 Layer 단원을 참조하십시오.

주제

- NATIVE_NETWORK_ENCRYPTION 옵션 설정
- NATIVE_NETWORK_ENCRYPTION 옵션 추가
- sqlnet.ora에서 NNE 값 설정
- NATIVE_NETWORK_ENCRYPTION 옵션 설정 수정
- NATIVE_NETWORK_ENCRYPTION 옵션 제거

NATIVE_NETWORK_ENCRYPTION 옵션 설정

서버와 클라이언트 모두에서 암호화 요구 사항을 지정할 수 있습니다. DB 인스턴스는 예를 들어 데이터베이스 링크를 사용하여 다른 데이터베이스에 연결할 때 클라이언트 역할을 수행할 수 있습니다. 서버 측에서 암호화를 강제하지 않도록 할 수 있습니다. 예를 들어 서버에 필요하기 때문에 모든 클라이언트 통신에 암호화를 사용하도록 강제하지 않을 수 있습니다. 이 경우 SQLNET.*CLIENT 옵션을 사용하여 클라이언트 측에서 암호화를 강제할 수 있습니다.

Amazon RDS는 NATIVE_NETWORK_ENCRYPTION 옵션에 대해 다음 설정을 지원합니다.

> **Note**
>
> 쉼표를 사용하여 옵션 설정의 값을 구분하는 경우 쉼표 뒤에 공백을 넣지 마십시오.

옵션 설정	유효값	기본값	설명
SQLNET.ALLOW_WEAK_CRYPTO_CLIENTS	TRUE, FALSE	TRUE	비보안 암호를 사용하는 클라이언트가 데이터베이스에 연결을 시도할 때 서버의 동작입니다. TRUE인 경우, 클라이언트는 2021년 7월 PSU에 패치되지 않은 경우에도 연결할 수 있습니다. 설정이 FALSE인 경우 클라이언트는 2021년 7월 PSU로 패치된 경우에만 데이터베이스에 연결할 수 있습니다. SQLNET.ALLOW_WEAK_CRYPTO_CLIENTS 를 FALSE로 설정하기 전에 다음 조건을 충족하는지 확인하세요. • SQLNET.ENCRYPTION_TYPES_SERVER 및 SQLNET.ENCRYPTION_TYPES_CLIENT 는 DES, 3DES 또는 RC4가 아닌 일치하는 하나의 암호화 방법이 있습니다(모든 키 길이). • SQLNET.CHECKSUM_TYPES_SERVER 와 SQLNET.CHECKSUM_TYPES_CLIENT 는 MD5가 아닌 일치하는 보안 체크섬 방법이 하나 있습니다. • 클라이언트는 2021년 7월 PSU로 패치됩니다. 클라이언트가 패치되지 않은 경우 클라이언트는 연결을 끊고ORA-12269 오류를 받습니다.
SQLNET.ALLOW_WEAK_CRYPTO	TRUE, FALSE	TRUE	비보안 암호를 사용하는 클라이언트가 데이터베이스에 연결을 시도할 때 서버의 동작입니다. 다음 암호는 안전하지 않은 것으로 간주됩니다. • DES 암호화 방법(모든 키 길이)

옵션 설정	유효값	기본값	설명
			• 3DES 암호화 방법(모든 키 길이) • RC4 암호화 방법(모든 키 길이) • MD5 체크섬 방법 설정이 TRUE인 경우 클라이언트는 앞의 비보안 암호를 사용할 때 연결할 수 있습니다. 설정이 FALSE인 경우 데이터베이스는 클라이언트가 앞의 비보안 암호를 사용할 때 연결하는 것을 막습니다. SQLNET.ALLOW_WEAK_CRYPTO 를 FALSE로 설정하기 전에 다음 조건을 충족하는지 확인하세요. • SQLNET.ENCRYPTION_TYPES_SERVER 및 SQLNET.ENCRYPTION_TYPES_CLIENT 는 DES, 3DES 또는 RC4가 아닌 일치하는 하나의 암호화 방법이 있습니다(모든 키 길이). • SQLNET.CHECKSUM_TYPES_SERVER 와 SQLNET.CHECKSUM_TYPES_CLIENT 는 MD5가 아닌 일치하는 보안 체크섬 방법이 하나 있습니다. • 클라이언트는 2021년 7월 PSU로 패치됩니다. 클라이언트가 패치되지 않은 경우 클라이언트는 연결을 끊고ORA-12269 오류를 받습니다.

옵션 설정	유효값	기본값	설명
SQLNET.CRYPTO_CHECKSUM_CLIENT	Accepted, Rejected, Requested, Required	Requested	DB 인스턴스가 클라이언트 또는 클라이언트 역할을 하는 서버에 연결된 경우의 데이터 무결성 동작입니다. DB 인스턴스가 데이터베이스 링크를 사용하는 경우 해당 DB 인스턴스는 클라이언트 역할을 합니다. Requested 은(는) 클라이언트에서 DB 인스턴스가 체크섬을 수행할 필요가 없음을 나타냅니다.
SQLNET.CRYPTO_CHECKSUM_SERVER	Accepted, Rejected, Requested, Required	Requested	클라이언트 또는 클라이언트 역할을 하는 서버가 DB 인스턴스에 연결된 경우의 데이터 무결성 동작입니다. DB 인스턴스가 데이터베이스 링크를 사용하는 경우 해당 DB 인스턴스는 클라이언트 역할을 합니다. Requested 은(는) DB 인스턴스에서 클라이언트가 체크섬을 수행할 필요가 없음을 나타냅니다.
SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT	SHA256, SHA384, SHA512, SHA1, MD5	SHA256, SHA384, SHA512	체크섬 알고리즘 목록입니다. 단일 값 또는 쉼표로 구분된 값 목록을 지정할 수 있습니다. 쉼표를 사용하는 경우 쉼표 뒤에 공백을 삽입하지 마십시오. 그렇지 않으면 InvalidParameterValue 오류가 발생합니다. 이 파라미터 및 SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER 은(는) 공통 암호를 가져야 합니다.

옵션 설정	유효값	기본값	설명
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER	SHA256, SHA384, SHA512, SHA1, MD5	SHA256, SHA384, SHA512, SHA1, MD5	체크섬 알고리즘 목록입니다. 단일 값 또는 쉼표로 구분된 값 목록을 지정할 수 있습니다. 쉼표를 사용하는 경우 쉼표 뒤에 공백을 삽입하지 마십시오. 그렇지 않으면 InvalidParameterValue 오류가 발생합니다. 이 파라미터 및 SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT 은(는) 공통 암호를 가져야 합니다.
SQLNET.ENCRYPTION_CLIENT	Accepted, Rejected, Requested, Required	Requested	클라이언트 또는 클라이언트 역할을 하는 서버가 DB 인스턴스에 연결된 경우의 클라이언트의 암호화 동작입니다. DB 인스턴스가 데이터베이스 링크를 사용하는 경우 해당 DB 인스턴스는 클라이언트 역할을 합니다. Requested 은(는) 클라이언트에서 서버의 트래픽을 암호화할 필요가 없음을 나타냅니다.
SQLNET.ENCRYPTION_SERVER	Accepted, Rejected, Requested, Required	Requested	클라이언트 또는 클라이언트 역할을 하는 서버가 DB 인스턴스에 연결된 경우의 서버의 암호화 동작입니다. DB 인스턴스가 데이터베이스 링크를 사용하는 경우 해당 DB 인스턴스는 클라이언트 역할을 합니다. Requested 는 DB 인스턴스에서 클라이언트의 송신 트래픽을 암호화할 필요가 없음을 나타냅니다.

옵션 설정	유효값	기본값	설명
SQLNET.ENCRYPTION_TYPES_CLIENT	RC4_256, AES256, AES192, 3DES168, RC4_128, AES128, 3DES112, RC4_56, DES, RC4_40, DES40	RC4_256, AES256, AES192, 3DES168, RC4_128, AES128, 3DES112, RC4_56, DES, RC4_40, DES40	클라이언트에서 사용하는 암호화 알고리즘 목록입니다. 클라이언트에서는 알고리즘이 성공하거나 목록의 끝에 도달할 때까지 각 알고리즘을 순서대로 진행하여 서버 입력 복호화를 시도합니다. Amazon RDS는 다음과 같은 Oracle의 기본 목록을 사용합니다. RDS는 RC4_256으로 시작하고 목록을 순서대로 진행합니다. 순서를 변경하거나 DB 인스턴스가 수락할 알고리즘을 제한할 수 있습니다. 1. RC4_256: RSA RC4(256비트 키 크기) 2. AES256: AES(256비트 키 크기) 3. AES192: AES(192비트 키 크기) 4. 3DES168112: 3키 Triple-DES(비트 유효 키 크기) 5. RC4_128: RSA RC4(128비트 키 크기) 6. AES128: AES(128비트 키 크기) 7. 3DES1122키 Triple-DES(80비트 유효 키 크기) 8. RC4_56: RSA RC4(56비트 키 크기) 9. DES: Standard DES(56비트 키 크기) 10RC4_40: RSA RC4(40비트 키 크기) 11DES40: DES40(40비트 키 크기) 단일 값 또는 쉼표로 구분된 값 목록을 지정할 수 있습니다. 쉼표인 경우 쉼표 뒤에 공백을 삽입하지 마십시오. 그렇지

옵션 설정	유효값	기본값	설명
			않으면 InvalidParameterValue 오류가 발생합니다. 이 파라미터 및 SQLNET.SQLNET.ENCRYPTION_TYPES_SERVER 은(는) 공통 암호를 가져야 합니다.

옵션 설정	유효값	기본값	설명
SQLNET.ENCRYPTION_TYPES_SERVER	RC4_256, AES256, AES192, 3DES168, RC4_128, AES128, 3DES112, RC4_56, DES, RC4_40, DES40	RC4_256, AES256, AES192, 3DES168, RC4_128, AES128, 3DES112, RC4_56, DES, RC4_40, DES40	DB 인스턴스에서 사용하는 암호화 알고리즘 목록입니다. DB 인스턴스에서는 알고리즘이 성공하거나 목록의 끝에 도달할 때까지 각 알고리즘을 순서대로 사용하여 클라이언트 입력을 해독합니다. Amazon RDS는 다음과 같은 Oracle의 기본 목록을 사용합니다. 순서를 변경하거나 클라이언트가 수락할 알고리즘을 제한할 수 있습니다. 1. RC4_256: RSA RC4(256비트 키 크기) 2. AES256: AES(256비트 키 크기) 3. AES192: AES(192비트 키 크기) 4. 3DES168112: 3키 Triple-DES(비트 유효 키 크기) 5. RC4_128: RSA RC4(128비트 키 크기) 6. AES128: AES(128비트 키 크기) 7. 3DES1122키 Triple-DES(80비트 유효 키 크기) 8. RC4_56: RSA RC4(56비트 키 크기) 9. DES: Standard DES(56비트 키 크기) 10RC4_40: RSA RC4(40비트 키 크기) 11DES40: DES40(40비트 키 크기) 단일 값 또는 쉼표로 구분된 값 목록을 지정할 수 있습니다. 쉼표인 경우 쉼표 뒤에 공백을 삽입하지 마십시오. 그렇지 않으면 InvalidParameterValue 오류가 발생합니다.

옵션 설정	유효값	기본값	설명
			이 파라미터 및 SQLNET.SQLNET.ENCRYPTION_TYPES_SERVER 은(는) 공통 암호를 가져야 합니다.

NATIVE_NETWORK_ENCRYPTION 옵션 추가

NATIVE_NETWORK_ENCRYPTION 옵션을 DB 인스턴스에 추가하는 일반적인 프로세스는 다음과 같습니다.

1. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 복사 또는 수정합니다.
2. [] 옵션을 옵션 그룹에 추가합니다.
3. 옵션 그룹을 DB 인스턴스에 연결합니다.

옵션 그룹이 활성화되면 NNE가 활성화됩니다.

AWS Management Console을 사용하여 DB 인스턴스에 NATIVE_NETWORK_ENCRYPTION 옵션을 추가하는 방법

1. [Engine]에서 사용할 Oracle 버전을 선택합니다. NNE는 모든 에디션에서 지원됩니다.
2. 메이저 엔진 버전에서 DB 인스턴스의 버전을 선택합니다.

   자세한 내용은 옵션 그룹 생성 단원을 참조하십시오.
3. 옵션 그룹에 NATIVE_NETWORK_ENCRYPTION 옵션을 추가합니다. 옵션 추가에 대한 자세한 내용은 옵션 그룹에 옵션 추가 섹션을 참조하세요.

   > **Note**
   >
   > NATIVE_NETWORK_ENCRYPTION 옵션을 추가한 후 DB 인스턴스를 재시작할 필요가 없습니다. 옵션 그룹이 활성화되자마자 NNE가 활성화됩니다.
4. 옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스에 적용합니다:
   - 새 DB 인스턴스의 경우, 인스턴스를 시작할 때 옵션 그룹을 적용합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.

- 기존 DB 인스턴스의 경우, 해당 인스턴스를 수정하고 새 옵션 그룹을 연결하여 옵션 그룹을 적용합니다. NATIVE_NETWORK_ENCRYPTION 옵션을 추가한 후 DB 인스턴스를 재시작할 필요가 없습니다. 옵션 그룹이 활성화되자마자 NNE가 활성화됩니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

sqlnet.ora에서 NNE 값 설정

Oracle 기본 네트워크 암호화를 사용하여 서버 측 및 클라이언트 측에서 네트워크 암호화를 설정할 수 있습니다. 해당 클라이언트는 DB 인스턴스에 연결하는 데 사용되는 컴퓨터입니다. sqlnet.ora에서 다음과 같은 클라이언트 설정을 지정할 수 있습니다.

- `SQLNET.ALLOW_WEAK_CRYPTO`
- `SQLNET.ALLOW_WEAK_CRYPTO_CLIENTS`
- `SQLNET.CRYPTO_CHECKSUM_CLIENT`
- `SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT`
- `SQLNET.ENCRYPTION_CLIENT`
- `SQLNET.ENCRYPTION_TYPES_CLIENT`

자세한 내용은 Oracle 설명서의 Configuring Network Data Encryption and Integrity for Oracle Servers and Clients를 참조하십시오.

DB 인스턴스가 애플리케이션의 연결 요청을 거부하는 경우가 있습니다. 예를 들어 클라이언트와 서버의 암호화 알고리즘이 일치하지 않는 경우 거부가 발생할 수 있습니다. Oracle 기본 네트워크 암호화를 테스트하려면 클라이언트의 sqlnet.ora 파일에 다음 줄을 추가합니다.

```
DIAG_ADR_ENABLED=off
TRACE_DIRECTORY_CLIENT=/tmp
TRACE_FILE_CLIENT=nettrace
TRACE_LEVEL_CLIENT=16
```

연결이 시도될 때 선행 줄이 클라이언트에서 `/tmp/nettrace*`(이)라는 추적 파일을 생성합니다. 추적 파일에는 연결 관련 정보가 포함되어 있습니다. Oracle 기본 네트워크 암호화를 사용할 때 연결 관련 문제에 대한 자세한 내용은 Oracle 데이터베이스 설명서의 About Negotiating Encryption and Integrity를 참조하세요.

NATIVE_NETWORK_ENCRYPTION 옵션 설정 수정

NATIVE_NETWORK_ENCRYPTION 옵션을 활성화한 후 해당 설정을 수정할 수 있습니다. 현재는 AWS CLI 또는 RDS API로만 NATIVE_NETWORK_ENCRYPTION 옵션 설정을 수정할 수 있습니다. 콘솔은 사용할 수 없습니다. 다음 예제에서는 옵션의 두 가지 설정을 수정합니다.

```
aws rds add-option-to-option-group \
    --option-group-name my-option-group \
    --options
 "OptionName=NATIVE_NETWORK_ENCRYPTION,OptionSettings=[{Name=SQLNET.CRYPTO_CHECKSUM_TYPES_SERVE
{Name=SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER,Value=SHA256}]" \
    --apply-immediately
```

CLI를 사용하여 옵션 설정을 수정하는 방법은 AWS CLI 섹션을 참조하세요. 각 설정에 대한 자세한 내용은 NATIVE_NETWORK_ENCRYPTION 옵션 설정 단원을 참조하십시오.

주제

- CRYPTO_CHECKSUM_* 값 수정
- ALLOW_WEAK_CRYPTO* 설정 수정

CRYPTO_CHECKSUM_* 값 수정

NATIVE_NETWORK_ENCRYPTION 옵션 설정을 수정하는 경우 다음 옵션 설정에 공통 암호가 하나 이상 있어야 합니다.

- SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER
- SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT

다음 예제에서는 SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER를 수정한 시나리오를 보여줍니다. 구성은 CRYPTO_CHECKSUM_TYPES_CLIENT 및 CRYPTO_CHECKSUM_TYPES_SERVER가 모두 SHA256을 사용해서 유효합니다.

옵션 설정	수정 전 값	수정 후 값
SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT	**SHA256**, SHA384, SHA512	변경 사항이 없습니다

옵션 설정	수정 전 값	수정 후 값
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER	**SHA256**, SHA384, SHA512, SHA1, MD5	SHA1,MD5, **SHA256**

또 다른 예를 들어, SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER을(를) 기본 설정에서 SHA1,MD5(으)로 수정하려고 한다고 가정합니다. 이 경우 SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT을(를) SHA1 또는 MD5(으)로 설정해야 합니다. 이러한 알고리즘은 SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT에 대한 기본값에 포함되지 않습니다.

ALLOW_WEAK_CRYPTO* 설정 수정

기본값으로 SQLNET.ALLOW_WEAK_CRYPTO* 옵션을 FALSE로 설정하려면 다음 조건을 충족하는지 확인하세요.

- SQLNET.ENCRYPTION_TYPES_SERVER 및 SQLNET.ENCRYPTION_TYPES_CLIENT는 일치하는 보안 암호화 방법이 하나 있습니다. DES, 3DES 또는 RC4가 아닌 경우 방법이 안전한 것으로 간주됩니다(모든 키 길이).
- SQLNET.CHECKSUM_TYPES_SERVER 및 SQLNET.CHECKSUM_TYPES_CLIENT는 일치하는 보안 체크섬 방법이 하나 있습니다. MD5가 아닌 경우 방법은 안전한 것으로 간주됩니다.
- 클라이언트는 2021년 7월 PSU로 패치됩니다. 클라이언트가 패치되지 않은 경우 클라이언트는 연결을 끊고ORA-12269 오류를 받습니다.

다음 예는 NNE 설정 샘플을 보여줍니다. SQLNET.ENCRYPTION_TYPES_SERVER 및 SQLNET.ENCRYPTION_TYPES_CLIENT를 FALSE로 설정하려는 경우 비보안 연결을 차단합니다. 체크섬 옵션 설정은 두 가지 모두 SHA256을 가지고 있어서 필수 구성 요소를 충족합니다. 그러나 SQLNET.ENCRYPTION_TYPES_CLIENT 및 SQLNET.ENCRYPTION_TYPES_SERVER는 DES, 3DES 및 RC4 암호화 방법을 사용하여 안전하지 않습니다. 따라서 SQLNET.ALLOW_WEAK_CRYPTO* 옵션이 FALSE로 설정되면 먼저 SQLNET.ENCRYPTION_TYPES_SERVER 및 SQLNET.ENCRYPTION_TYPES_CLIENT를 AES256과 같은 보안 암호화 방법으로 설정합니다.

옵션 설정	값
SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT	SHA256, SHA384, SHA512

옵션 설정	값
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER	SHA1,MD5,SHA256
SQLNET.ENCRYPTION_TYPES_CLIENT	RC4_256, 3DES168, DES40
SQLNET.ENCRYPTION_TYPES_SERVER	RC4_256, 3DES168, DES40

NATIVE_NETWORK_ENCRYPTION 옵션 제거

DB 인스턴스에서 NNE를 제거할 수 있습니다.

DB 인스턴스에서 NATIVE_NETWORK_ENCRYPTION 옵션을 제거하려면 다음 중 하나를 수행합니다.

- 여러 DB 인스턴스에서 옵션을 제거하려면 DB 인스턴스가 속한 옵션 그룹에서 해당 NATIVE_NETWORK_ENCRYPTION 옵션을 제거합니다. 이 변경은 해당 옵션 그룹을 사용하는 모든 DB 인스턴스에 영향을 미칩니다. NATIVE_NETWORK_ENCRYPTION 옵션을 제거한 후 DB 인스턴스를 재시작할 필요가 없습니다. 자세한 내용은 옵션 그룹에서 옵션 제거 단원을 참조하십시오.
- 단일 DB 인스턴스에서 해당 옵션을 제거하려면 DB 인스턴스를 수정하고 NATIVE_NETWORK_ENCRYPTION 옵션이 포함되지 않은 다른 옵션 그룹을 지정합니다. 기본(빈) 옵션 그룹을 지정하거나 다른 사용자 지정 옵션 그룹을 지정할 수 있습니다. NATIVE_NETWORK_ENCRYPTION 옵션을 제거한 후 DB 인스턴스를 재시작할 필요가 없습니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

Oracle OLAP

Amazon RDS는 OLAP 옵션 사용을 통해 Oracle OLAP를 지원합니다. 이 옵션은 Oracle DB 인스턴스에 대한 OLAP(On-line Analytical Processing)를 제공합니다. Oracle OLAP를 사용하여 OLAP 표준에 따라 차원 객체 및 큐브를 생성함으로써 많은 양의 데이터를 분석할 수 있습니다. 자세한 내용은 Oracle 설명서를 참조하십시오.

> **Important**
>
> Oracle OLAP를 사용하면 CVSS(공통 취약성 평가 시스템) 점수가 9 이상인 보안 취약성 또는 기타 발표된 보안 취약성이 있는 경우 Amazon RDS에서 DB 인스턴스를 최신 Oracle PSU로 자동 업데이트합니다.

Amazon RDS는 Oracle Database 19c 이상의 Enterprise Edition용 Oracle OLAP을 지원합니다.

Oracle OLAP의 사전 조건

다음은 Oracle OLAP 사용 시 사전 조건입니다.

- Oracle에서 제공하는 Oracle OLAP 라이선스를 보유해야 합니다. 자세한 내용은 Oracle 설명서의 라이선스 정보를 참조하십시오.
- DB 인스턴스는 충분한 인스턴스 클래스여야 합니다. Oracle OLAP는 db.t3.small DB 인스턴스 클래스에 대해 지원되지 않습니다. 자세한 내용은 RDS for Oracle DB 인스턴스 클래스 단원을 참조하십시오.
- DB 인스턴스는 마이너 버전 자동 업그레이드가 활성화되어 있어야 합니다. 이 옵션을 사용하면 DB 인스턴스를 활성화하여 사용 가능할 때 마이너 DB 엔진 버전 업그레이드를 자동으로 받을 수 있으며, Oracle Java Virtual Machine(JVM)을 설치하는 옵션에 필요합니다. Amazon RDS는 이 옵션을 사용하여 DB 인스턴스를 최신 Oracle Patch Set Update(PSU) 또는 Release Update(RU)로 업데이트합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.
- DB 인스턴스에는 이름이 OLAPSYS인 사용자가 없어야 합니다. 이 경우 OLAP 옵션 설치가 실패합니다.

Oracle OLAP의 모범 사례

다음은 Oracle OLAP 사용에 관한 모범 사례입니다.

- 보안을 극대화하기 위해 Secure Sockets Layer(SSL)와 함께 OLAP 옵션을 사용합니다. 자세한 내용은 Oracle 보안 소켓 Layer 섹션을 참조하세요.
- DB 인스턴스에 대한 액세스를 제한하도록 DB 인스턴스를 구성합니다. 자세한 내용은 VPC에서 DB 인스턴스에 액세스하는 시나리오 및 VPC에서 DB 인스턴스를 사용한 작업 단원을 참조하십시오.

Oracle OLAP 옵션 추가

OLAP 옵션을 DB 인스턴스에 추가하는 일반적인 프로세스는 다음과 같습니다.

1. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 복사 또는 수정합니다.
2. [] 옵션을 옵션 그룹에 추가합니다.
3. 옵션 그룹을 DB 인스턴스에 연동시킵니다.

Oracle Java Virtual Machine(JVM)이 DB 인스턴스에 설치되어 있지 않은 경우 OLAP 옵션이 추가되는 동안 잠시 중단이 발생합니다. Oracle Java Virtual Machine(JVM)이 DB 인스턴스에 이미 설치되어 있으면 중단이 발생하지 않습니다. 옵션을 추가한 후 DB 인스턴스를 재시작할 필요가 없습니다. 옵션 그룹이 활성화되는 즉시 Oracle OLAP를 사용할 수 있습니다.

OLAP 옵션을 DB 인스턴스에 추가하려면

1. 사용할 옵션 그룹을 결정합니다. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 사용합니다. 기존 옵션 그룹을 사용하려면 다음 단계로 건너뛰십시오. 그렇지 않으면 다음 설정을 사용하여 사용자 지정 DB 옵션을 생성합니다.

   - 엔진에서 DB 인스턴스의 Oracle 에디션을 선택합니다.
   - 메이저 엔진 버전에서 DB 인스턴스의 버전을 선택합니다.

   자세한 내용은 옵션 그룹 생성 섹션을 참조하세요.
2. OLAP 옵션을 옵션 그룹에 추가합니다. 옵션 추가에 대한 자세한 내용은 옵션 그룹에 옵션 추가 섹션을 참조하세요.
3. 옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스에 적용합니다:

   - 새 DB 인스턴스의 경우, 인스턴스를 시작할 때 옵션 그룹을 적용하십시오. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
   - 기존 DB 인스턴스의 경우, 해당 인스턴스를 수정하고 새 옵션 그룹을 연결하여 옵션 그룹을 적용하십시오. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

Oracle OLAP 사용

Oracle OLAP 옵션을 활성화한 후에 사용할 수 있습니다. Oracle OLAP에 대해 지원되는 기능 목록은 Oracle 설명서를 참조하십시오.

Oracle OLAP 옵션 제거

OLAP 옵션에서 제공하는 데이터 유형을 사용하는 모든 객체를 삭제한 후 DB 인스턴스에서 옵션을 삭제할 수 있습니다. Oracle Java Virtual Machine(JVM)이 DB 인스턴스에 설치되어 있지 않은 경우 OLAP 옵션이 제거되는 동안 잠시 중단이 발생합니다. Oracle Java Virtual Machine(JVM)이 DB 인스턴스에 이미 설치되어 있으면 중단이 발생하지 않습니다. OLAP 옵션을 제거한 후 DB 인스턴스를 재시작할 필요가 없습니다.

OLAP 옵션을 삭제하려면

1. 데이터를 백업합니다.

   > **Warning**
   >
   > 인스턴스에서 옵션의 일부로 활성화된 데이터 유형을 사용하고 OLAP 옵션을 제거하면 데이터가 손실될 수 있습니다. 자세한 내용은 데이터 백업, 복원 및 내보내기 섹션을 참조하세요.

2. 기존 객체가 OLAP 옵션의 데이터 유형 또는 기능을 참조하는지 확인합니다.
3. OLAP 옵션의 데이터 유형 또는 기능을 참조하는 모든 객체를 삭제합니다.
4. 다음 중 하나를 수행하십시오.

   - 소속 옵션 그룹에서 OLAP 옵션을 제거합니다. 이 변경은 해당 옵션 그룹을 사용하는 모든 DB 인스턴스에 영향을 미칩니다. 자세한 내용은 옵션 그룹에서 옵션 제거 섹션을 참조하세요.
   - DB 인스턴스를 수정하고 OLAP 옵션이 포함되지 않은 다른 옵션 그룹을 지정합니다. 이 변경은 단일 DB 인스턴스에 영향을 미칩니다. 기본(빈) 옵션 그룹을 지정하거나 다른 사용자 지정 옵션 그룹을 지정할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

Oracle 보안 소켓 Layer

RDS for Oracle DB 인스턴스에 대해 SSL 암호화를 활성화하려면 DB 인스턴스와 연결된 옵션 그룹에 Oracle SSL 옵션을 추가합니다. Amazon RDS는 Oracle에서 요구하는 대로 SSL 연결을 위해 두 번째 포트를 사용합니다. 이 접근 방식에서는 DB 인스턴스와 SQL*Plus 간에 클리어 텍스트 통신과 SSL로 암호화된 통신이 동시에 발생할 수 있습니다. 예를 들어 이 포트를 클리어 텍스트 통신에 사용하여 VPC 내의 다른 리소스와 통신하면서 동일한 포트를 SSL로 암호화된 통신에 사용하여 VPC 외부의 리소스와 통신할 수 있습니다.

> **Note**
>
> 동일한 RDS for Oracle DB 인스턴스에서 SSL 또는 NNE(기본 네트워크 암호화)를 사용할 수 있지만 동시에 사용할 수는 없습니다. SSL 암호화를 사용할 경우 다른 연결 암호화는 모두 해제해야 합니다. 자세한 내용은 Oracle 기본 네트워크 암호화 단원을 참조하십시오.

SSL/TLS 및 NNE는 더 이상 Oracle Advanced Security의 일부가 아닙니다. RDS for Oracle에서는 다음 데이터베이스 버전의 모든 정식 에디션에서 SSL 암호화를 사용할 수 있습니다.

- Oracle Database 21c(21.0.0)
- Oracle Database 19c(19.0.0)

주제

- Oracle SSL 옵션에 대한 TLS 버전
- Oracle SSL 옵션에 대한 암호 그룹
- FIPS 지원
- SSL 옵션 추가
- RDS for Oracle DB 인스턴스에 SSL을 사용하도록 SQL*Plus 구성
- SSL을 사용하여 RDS for Oracle DB 인스턴스에 연결
- JDBC를 통한 SSL 연결 설정
- SSL 연결과 DN 일치 적용
- SSL 연결 문제 해결

Oracle SSL 옵션에 대한 TLS 버전

Amazon RDS for Oracle은 TLS(전송 계층 보안) 버전 1.0 및 1.2를 지원합니다. 새 Oracle SSL 옵션을 추가할 경우, 유효한 값에 SQLNET.SSL_VERSION을 명시적으로 설정합니다. 다음은 이 옵션 설정으로 허용되는 값입니다.

- "1.0" – 클라이언트는 TLS 버전 1.0만 사용하여 DB 인스턴스에 연결할 수 있습니다. 기존 Oracle SSL 옵션에서 SQLNET.SSL_VERSION은 "1.0"으로 자동 설정됩니다. 필요할 경우 설정을 변경할 수 있습니다.
- "1.2" – 클라이언트는 TLS 1.2만 사용하여 DB 인스턴스에 연결할 수 있습니다.
- "1.2 or 1.0" – 클라이언트는 TLS 1.2 또는 1.0을 사용하여 DB 인스턴스에 연결할 수 있습니다.

Oracle SSL 옵션에 대한 암호 그룹

Amazon RDS for Oracle은 여러 개의 SSL 암호 그룹을 지원합니다. 기본적으로 Oracle SSL은 SSL_RSA_WITH_AES_256_CBC_SHA 암호 그룹을 사용하도록 구성되어 있습니다. SSL 연결을 통해 사용할 다른 암호 그룹을 지정하려면 SQLNET.CIPHER_SUITE 옵션 설정을 사용하십시오.

SQLNET.CIPHER_SUITE에 대해 여러 값을 지정할 수 있습니다. 이 기법은 DB 인스턴스 간에 데이터베이스 링크가 있고 암호 그룹을 업데이트하려는 경우에 유용합니다.

다음 표에는 Oracle 데이터베이스 19c 및 21c의 모든 버전에서 RDS for Oracle에 대한 SSL 지원이 요약되어 있습니다.

암호 그룹(SQLNET.CIPHER_SUITE)	TLS 버전 지원(SQLNET.SSL_VERSION)	FIPS 지원	FedRAMP 규정 준수
SSL_RSA_WITH_AES_256_CBC_SHA(기본)	1.0 및 1.2	예	아니요
SSL_RSA_WITH_AES_256_CBC_SHA256	1.2	예	아니요
SSL_RSA_WITH_AES_256_GCM_SHA384	1.2	예	아니요
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	1.2	예	예
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	1.2	예	예

암호 그룹(SQLNET.CIPHER_SUITE)	TLS 버전 지원(SQLNET.SSL_VERSION)	FIPS 지원	FedRAMP 규정 준수
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	1.2	예	예
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	1.2	예	예
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	1.2	예	예
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	1.2	예	예

FIPS 지원

RDS for Oracle을 사용하면 140-2에 대해 FIPS(Federal Information Processing Standard) 표준을 사용할 수 있습니다. FIPS 140-2는 암호 모듈 보안 요구 사항을 정의하는 미국 정부 표준입니다. Oracle SSL 옵션에 대해 `FIPS.SSLFIPS_140`을 `TRUE`로 설정하여 FIPS 표준을 활성화합니다. FIPS 140-2가 SSL용으로 구성된 경우 암호화 라이브러리는 클라이언트와 RDS for Oracle DB 인스턴스 간의 데이터를 암호화합니다.

클라이언트는 FIPS를 준수하는 암호 제품군을 사용해야 합니다. 연결을 설정할 때 클라이언트와 RDS for Oracle DB 인스턴스는 메시지를 주고받을 때 사용할 암호 제품군을 협상합니다. Oracle SSL 옵션에 대한 암호 그룹의 테이블에서는 각 TLS 버전에 대한 FIPS 호환 SSL 암호 그룹을 보여줍니다. 자세한 내용은 Oracle Database 설명서의 Oracle Database FIPS 140-2 설정을 참조하세요.

SSL 옵션 추가

SSL을 사용하려면 RDS for Oracle DB 인스턴스가 `SSL` 옵션을 포함하는 옵션 그룹과 연결되어 있어야 합니다.

콘솔

SSL 옵션을 옵션 그룹에 추가하려면

1. 새 옵션 그룹을 만들거나 `SSL` 옵션을 추가할 수 있는 기존 옵션 그룹을 식별하십시오.

옵션 그룹의 생성에 대한 자세한 내용은 옵션 그룹 생성 단원을 참조하십시오.

2. [SSL] 옵션을 옵션 그룹에 추가합니다.

   SSL 연결에 FIPS 확인 암호 제품군만 사용하려면 `FIPS.SSLFIPS_140` 옵션을 `TRUE`로 설정합니다. FIPS 표준에 대한 자세한 내용은 FIPS 지원 단원을 참조하십시오.

   옵션 그룹에 옵션을 추가하는 방법에 대한 자세한 내용은 옵션 그룹에 옵션 추가 단원을 참조하십시오.

3. 새 RDS for Oracle DB 인스턴스를 생성하고 옵션 그룹을 연결하거나, RDS for Oracle DB 인스턴스를 수정하여 옵션 그룹을 연결합니다.

   DB 인스턴스 생성에 대한 자세한 내용은 Amazon RDS DB 인스턴스 생성 단원을 참조하세요.

   DB 인스턴스 수정에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하세요.

AWS CLI

SSL 옵션을 옵션 그룹에 추가하려면

1. 새 옵션 그룹을 만들거나 SSL 옵션을 추가할 수 있는 기존 옵션 그룹을 식별하십시오.

   옵션 그룹의 생성에 대한 자세한 내용은 옵션 그룹 생성 단원을 참조하십시오.

2. [SSL] 옵션을 옵션 그룹에 추가합니다.

   다음 옵션 설정을 지정합니다.

   - `Port` – SSL 포트 번호입니다.
   - `VpcSecurityGroupMemberships` – 옵션이 활성화되는 VPC 보안 그룹입니다.
   - `SQLNET.SSL_VERSION` – 클라이언트가 DB 인스턴스에 연결할 때 사용할 수 있는 TLS 버전입니다.

   예를 들어 다음 AWS CLI 명령은 SSL 옵션을 `ora-option-group`이라는 옵션 그룹에 추가합니다.

   Example

   대상 LinuxmacOS, 또는Unix:

```
aws rds add-option-to-option-group --option-group-name ora-option-group \
  --options
 'OptionName=SSL,Port=2484,VpcSecurityGroupMemberships="sg-68184619",OptionSettings=[{Name=
```

Windows의 경우:

```
aws rds add-option-to-option-group --option-group-name ora-option-group ^
  --options
 'OptionName=SSL,Port=2484,VpcSecurityGroupMemberships="sg-68184619",OptionSettings=[{Name=
```

3. 새 RDS for Oracle DB 인스턴스를 생성하고 옵션 그룹을 연결하거나, RDS for Oracle DB 인스턴스를 수정하여 옵션 그룹을 연결합니다.

   DB 인스턴스 생성에 대한 자세한 내용은 Amazon RDS DB 인스턴스 생성 단원을 참조하세요.

   DB 인스턴스 수정에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하세요.

RDS for Oracle DB 인스턴스에 SSL을 사용하도록 SQL*Plus 구성

Oracle SSL 옵션을 사용하는 RDS for Oracle DB 인스턴스에 연결하려면 우선 연결 전에 SQL*Plus를 구성해야 합니다.

> **Note**
>
> 해당 클라이언트에서 DB 인스턴스에 액세스하는 것을 허용하려면 보안 그룹을 올바르게 구성해야 합니다. 자세한 내용은 보안 그룹을 통한 액세스 제어 섹션을 참조하세요. 또한 이러한 지침은 Oracle 홈을 직접 사용하는 SQL*Plus 및 기타 클라이언트에 대한 것입니다. JDBC 연결에 대한 자세한 정보는 JDBC를 통한 SSL 연결 설정 단원을 참조하십시오.

SSL을 사용하여 RDS for Oracle DB 인스턴스에 연결하도록 SQL*Plus를 구성하려면

1. `ORACLE_HOME` 환경 변수를 Oracle 홈 디렉터리의 위치로 설정합니다.

   Oracle 홈 디렉터리 경로는 설치에 따라 달라집니다. 다음은 `ORACLE_HOME` 환경 변수를 설정하는 예제입니다.

```
prompt>export ORACLE_HOME=/home/user/app/user/product/19.0.0/dbhome_1
```

Oracle 환경 변수 설정에 대한 자세한 정보는 Oracle 설명서의 SQL*Plus Environment Variables 및 운영 체제에 해당하는 Oracle 설치 안내서를 참조하십시오.

2. $ORACLE_HOME/lib를 LD_LIBRARY_PATH 환경 변수에 추가합니다.

   다음은 LD_LIBRARY_PATH 환경 변수를 설정하는 예제입니다.

   ```
   prompt>export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib
   ```

3. $ORACLE_HOME/ssl_wallet에 Oracle Wallet을 위한 디렉터리를 만듭니다.

   다음은 Oracle Wallet 디렉터리를 생성하는 예제입니다.

   ```
   prompt>mkdir $ORACLE_HOME/ssl_wallet
   ```

4. 모든 AWS 리전에서 작동하는 인증서 번들 .pem 파일을 다운로드하고 파일을 ssl_wallet 디렉터리에 저장합니다. 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화을 참조하세요.

5. $ORACLE_HOME/network/admin 디렉터리에서 tnsnames.ora 파일을 수정하거나 생성하고 다음 항목을 포함합니다.

   ```
   net_service_name =
     (DESCRIPTION =
       (ADDRESS_LIST =
         (ADDRESS =
           (PROTOCOL = TCPS)
           (HOST = endpoint)
           (PORT = ssl_port_number)
         )
       )
       (CONNECT_DATA =
         (SID = database_name)
       )
       (SECURITY =
         (SSL_SERVER_CERT_DN =
    "C=US,ST=Washington,L=Seattle,O=Amazon.com,OU=RDS,CN=endpoint")
       )
     )
   ```

6. 동일한 디렉터리에서 sqlnet.ora 파일을 수정하거나 만들고 다음 파라미터를 포함합니다.

> **Note**
>
> TLS 보안 연결을 통해 개체와 통신하기 위해 Oracle이 인증에 필요한 인증서가 있는 wallet을 요구합니다. 7단계에서와 같이 Oracle의 ORAPKI 유틸리티를 사용하여 Oracle Wallet을 만들고 유지 관리할 수 있습니다. 자세한 내용은 Oracle 설명서의 Setting up Oracle wallet using ORAPKI를 참조하십시오.

```
WALLET_LOCATION = (SOURCE = (METHOD = FILE) (METHOD_DATA = (DIRECTORY =
 $ORACLE_HOME/ssl_wallet)))
SSL_CLIENT_AUTHENTICATION = FALSE
SSL_VERSION = 1.0
SSL_CIPHER_SUITES = (SSL_RSA_WITH_AES_256_CBC_SHA)
SSL_SERVER_DN_MATCH = ON
```

> **Note**
>
> DB 인스턴스가 지원하는 경우 SSL_VERSION을 더 높은 값으로 설정할 수 있습니다.

7. 다음 명령을 실행하여 Oracle Wallet을 만듭니다.

```
prompt>orapki wallet create -wallet $ORACLE_HOME/ssl_wallet -auto_login_only
```

8. OS 유틸리티를 사용하여 .pem 번들 파일의 각 인증서를 별도의 .pem 파일로 추출합니다.
9. .pem 파일의 절대 파일 이름으로 *certificate-pem-file*을 대체하여 개별 orapki 명령을 통해 Wallet에 각 인증서를 추가합니다.

```
prompt>orapki wallet add -wallet $ORACLE_HOME/ssl_wallet -trusted_cert -cert
      certificate-pem-file -auto_login_only
```

자세한 내용은 SSL/TLS 인증서 교체 단원을 참조하십시오.

SSL을 사용하여 RDS for Oracle DB 인스턴스에 연결

앞에서 설명한 대로 SSL을 사용하도록 SQL*Plus를 구성한 후에는 SSL 옵션을 통해 RDS for Oracle DB 인스턴스에 연결할 수 있습니다. 선택적으로, tnsnames.ora 및 sqlnet.ora 파일이 들어있는 디렉터

리를 가리키는 TNS_ADMIN 값을 먼저 내보낼 수 있습니다. 그러면 SQL *Plus가 이러한 파일을 일관되게 찾을 수 있습니다. 다음 예에서는 TNS_ADMIN 값을 내보냅니다.

```
export TNS_ADMIN = ${ORACLE_HOME}/network/admin
```

DB 인스턴스에 연결합니다. 예를 들어 SQL*Plus와 tnsnames.ora 파일의 *<net_service_name>*을 사용하여 연결할 수 있습니다.

```
sqlplus mydbuser@net_service_name
```

다음 명령을 사용하면 tnsnames.ora 파일 사용 없이 SQL*Plus를 사용하여 DB 인스턴스에 연결할 수도 있습니다.

```
sqlplus 'mydbuser@(DESCRIPTION = (ADDRESS = (PROTOCOL = TCPS)(HOST = endpoint) (PORT
 = ssl_port_number))(CONNECT_DATA = (SID = database_name)))'
```

SSL을 사용하지 않고 RDS for Oracle DB 인스턴스에 연결할 수도 있습니다. 예를 들어 다음 명령을 실행하면 SSL 암호화 없이 클리어 텍스트 포트를 통해 DB 인스턴스에 연결됩니다.

```
sqlplus 'mydbuser@(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = endpoint) (PORT
 = port_number))(CONNECT_DATA = (SID = database_name)))'
```

TCP(Transmission Control Protocol) 포트 액세스를 닫으려는 경우 IP 주소 수신이 없는 보안 그룹을 만들어 인스턴스에 추가합니다. 이렇게 하면 TCP 포트를 통한 연결이 닫히지만 SSL 옵션 보안 그룹에 의해 허용되는 범위 내에 있는 IP 주소를 사용하여 지정되는 SSL 포트를 통해 계속 연결할 수 있습니다.

JDBC를 통한 SSL 연결 설정

JDBC를 통한 SSL 연결을 사용하려면 키 스토어를 만들고 Amazon RDS 루트 CA 인증서를 신뢰하고 아래에 지정된 코드 조각을 사용해야 합니다.

키 스토어를 JKS 형식으로 만들려면 다음 명령을 사용하면 됩니다. 키 스토어 만들기에 대한 자세한 내용은 Oracle 설명서의 키 스토어 생성을 참조하세요. 참조 정보는 Java 플랫폼, 표준 에디션 도구 참조의 keytool을 참조하세요.

```
keytool -genkey -alias client -validity 365 -keyalg RSA -keystore clientkeystore
```

다음 단계에 따라 Amazon RDS 루트 CA 인증서를 신뢰합니다.

Amazon RDS 루트 CA 인증서를 신뢰하려면

1. 모든 AWS 리전에서 작동하는 인증서 번들 .pem 파일을 다운로드하고 파일을 ssl_wallet 디렉터리에 저장합니다.

   인증서 다운로드에 대한 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 단원을 참조하세요.
2. OS 유틸리티를 사용하여 .pem 파일의 각 인증서를 별도의 파일로 추출합니다.
3. 별도의 openssl 명령을 사용하여 각 인증서를 .der 형식으로 변환하고 *certificate-pem-file*을 인증서 .pem 파일 이름(.pem 확장자 제외)으로 대체합니다.

   ```
   openssl x509 -outform der -in certificate-pem-file.pem -out certificate-pem-file.der
   ```
4. 다음 명령을 사용하여 각 인증서를 키 스토어로 가져옵니다.

   ```
   keytool -import -alias rds-root -keystore clientkeystore.jks -file certificate-pem-file.der
   ```

   자세한 내용은 SSL/TLS 인증서 교체 단원을 참조하십시오.
5. 키 스토어가 성공적으로 만들어졌는지 확인하십시오.

   ```
   keytool -list -v -keystore clientkeystore.jks
   ```

   메시지가 표시되면 키 스토어 암호를 입력하십시오.

다음은 JDBC를 사용하여 SSL 연결을 설정하는 방법을 보여 주는 코드 예입니다.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

public class OracleSslConnectionTest {
    private static final String DB_SERVER_NAME = "dns-name-provided-by-amazon-rds";
    private static final Integer SSL_PORT = "ssl-option-port-configured-in-option-group";
    private static final String DB_SID = "oracle-sid";
```

```
    private static final String DB_USER = "user-name";
    private static final String DB_PASSWORD = "password";
    // This key store has only the prod root ca.
    private static final String KEY_STORE_FILE_PATH = "file-path-to-keystore";
    private static final String KEY_STORE_PASS = "keystore-password";

    public static void main(String[] args) throws SQLException {
        final Properties properties = new Properties();
        final String connectionString = String.format(
                "jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)(HOST=%s)(PORT=
%d))(CONNECT_DATA=(SID=%s)))",
                DB_SERVER_NAME, SSL_PORT, DB_SID);
        properties.put("user", DB_USER);
        properties.put("password", DB_PASSWORD);
        properties.put("oracle.jdbc.J2EE13Compliant", "true");
        properties.put("javax.net.ssl.trustStore", KEY_STORE_FILE_PATH);
        properties.put("javax.net.ssl.trustStoreType", "JKS");
        properties.put("javax.net.ssl.trustStorePassword", KEY_STORE_PASS);
        final Connection connection = DriverManager.getConnection(connectionString,
 properties);
        // If no exception, that means handshake has passed, and an SSL connection can
 be opened
    }
}
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

SSL 연결과 DN 일치 적용

Oracle 파라미터 SSL_SERVER_DN_MATCH를 사용하여 데이터베이스 서버에 대한 DN(고유 이름)을 서비스 이름과 일치하도록 적용할 수 있습니다. 일치 확인을 적용한 경우 SSL에서 인증서가 서버에서 가져온 것인지를 확인합니다. 일치 확인을 적용하지 않은 경우 SSL에서 확인을 수행하지만 일치 여부에 상관없이 연결을 허용합니다. 일치를 적용하지 않은 경우 서버 ID가 위조될 수 있습니다.

DN 일치를 적용하려면 DN 일치 속성을 추가하고 아래 지정된 연결 문자열을 사용합니다.

다음과 같이 속성을 클라이언트 연결에 추가하여 DN 일치를 적용합니다.

```
properties.put("oracle.net.ssl_server_dn_match", "TRUE");
```

SSL을 사용할 때 DN 일치를 적용하려면 다음 연결 문자열을 사용합니다.

```
final String connectionString = String.format(
    "jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)(HOST=%s)(PORT=%d))" +
    "(CONNECT_DATA=(SID=%s))" +
    "(SECURITY = (SSL_SERVER_CERT_DN =
\"C=US,ST=Washington,L=Seattle,O=Amazon.com,OU=RDS,CN=%s\")))",
    DB_SERVER_NAME, SSL_PORT, DB_SID, DB_SERVER_NAME);
```

SSL 연결 문제 해결

데이터베이스를 쿼리한 후 ORA-28860 오류가 표시될 수 있습니다.

```
ORA-28860: Fatal SSL error
28860. 00000 - "Fatal SSL error"
*Cause: An error occurred during the SSL connection to the peer. It is likely that this
 side sent data which the peer rejected.
*Action: Enable tracing to determine the exact cause of this error.
```

이 오류는 클라이언트가 서버에서 지원하지 않는 TLS 버전을 사용하여 연결을 시도할 때 발생합니다. 이 오류를 방지하려면 sqlnet.ora를 편집하고 SSL_VERSION을 올바른 TLS 버전으로 설정하세요. 자세한 내용은 My Oracle Support 문서의 Oracle Support Document 2748438.1을 참조하세요.

Oracle Spatial

Amazon RDS는 SPATIAL 옵션 사용을 통해 Oracle Spatial을 지원합니다. Oracle Spatial에는 Oracle 데이터베이스에서 공간 데이터의 저장, 검색, 업데이트 및 쿼리를 신속하게 실행할 수 있는 SQL 스키마 및 기능이 있습니다. 자세한 내용은 Oracle 설명서에서 Spatial Concepts를 참조하십시오. Amazon RDS는 모든 지원되는 릴리스의 모든 에디션에서 Oracle Spatial을 지원합니다.

공간 패치 번들(SPB) 작동 방식

분기마다 RDS for Oracle은 각 지원되는 메이저 엔진에 대해 새로운 마이너 엔진 버전을 릴리스합니다. 릴리스 업데이트(RU) 엔진 버전은 지정된 분기에 대한 RU 패치를 포함하여 Oracle의 버그 수정을 통합합니다. 공간 패치 번들(SPB) 엔진 버전에는 Oracle Spatial과 관련된 RU 패치와 패치가 포함되어 있습니다. 예를 들어 19.0.0.0.ru-2025-01.spb-1.r1은 엔진 버전 19.0.0.0.ru-2025-01.rur-2025-01.r1에 RU 패치와 공간 패치가 포함된 마이너 엔진 버전입니다. SPB는 Oracle Database 19c에서만 지원됩니다.

SPB는 RU와 동일한 방식으로 작동하지만 이름이 다릅니다. RU는 이름 지정 형식 19.0.0.0.ru-2025-01.rur-2025-01.r1을 사용합니다. SPB 이름에는 19.0.0.0.ru-2025-01.spb-1.r1과 같은 'spb' 텍스트가 포함됩니다. 일반적으로 SPB는 해당 분기별 RU가 있고서 2~3주 후에 릴리스됩니다. 예를 들어 19.0.0.0.ru-2025-01.spb-1.r1은 19.0.0.0.ru-2025-01.rur-2025-01.r1 이후에 릴리스됩니다.

RDS for Oracle은 RU 및 SPB의 마이너 버전 자동 업그레이드에 대한 별도 경로를 갖추고 있습니다. DB 인스턴스가 RU를 사용하는 경우 RDS는 인스턴스를 RU로 자동으로 업그레이드합니다. DB 인스턴스가 SPB를 사용하는 경우 RDS는 인스턴스를 SPB로 업그레이드합니다. DB 인스턴스에 Oracle Spatial 옵션을 추가하고 자동 업그레이드를 사용 설정하면 RDS는 인스턴스를 SPB의 업그레이드 경로에 배치합니다.

RU 및 SPB에 대한 자세한 내용은 Oracle 마이너 버전 업그레이드 섹션을 참조하시기 바랍니다. Oracle Database 19c에 대해 지원되는 RU 및 SPB 목록은 Amazon RDS for Oracle 릴리스 정보에 나와 있는 Amazon RDS for Oracle Database 19c(19.0.0.0)를 참조하시기 바랍니다.

Oracle Spatial의 사전 요구 사항

Oracle Spatial 사용을 위한 사전 요구 사항은 다음과 같습니다.

- DB 인스턴스가 충분한 인스턴스 클래스인지 확인합니다. Oracle Spatial은 db.t3.small DB 인스턴스 클래스에 대해 지원되지 않습니다. 자세한 내용은 RDS for Oracle DB 인스턴스 클래스 단원을 참조하십시오.
- DB 인스턴스에 자동 마이너 버전 업그레이드가 활성화되어 있는지 확인합니다. 이 옵션을 사용하면 DB 인스턴스를 활성화하여 사용 가능할 때 마이너 DB 엔진 버전 업그레이드를 자동으로 받을 수 있

으며, Oracle Java Virtual Machine(JVM)을 설치하는 옵션에 필요합니다. Amazon RDS는 이 옵션을 사용하여 DB 인스턴스를 최신 Oracle Patch Set Update(PSU) 또는 Release Update(RU)로 업데이트합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

Oracle Spatial의 모범 사례

다음은 Oracle Spatial 사용에 관한 모범 사례입니다.

- 보안을 극대화하기 위해 Secure Sockets Layer(SSL)와 함께 SPATIAL 옵션을 사용합니다. 자세한 내용은 Oracle 보안 소켓 Layer 섹션을 참조하세요.
- DB 인스턴스에 대한 액세스를 제한하도록 DB 인스턴스를 구성합니다. 자세한 내용은 VPC에서 DB 인스턴스에 액세스하는 시나리오 및 VPC에서 DB 인스턴스를 사용한 작업 단원을 참조하십시오.

Oracle Spatial 옵션 추가

SPATIAL 옵션을 DB 인스턴스에 추가하는 일반적인 프로세스는 다음과 같습니다.

1. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 복사 또는 수정합니다.
2. [] 옵션을 옵션 그룹에 추가합니다.
3. 옵션 그룹을 DB 인스턴스에 연동시킵니다.

Oracle Java Virtual Machine(JVM)이 DB 인스턴스에 설치되어 있지 않은 경우 SPATIAL 옵션이 추가되는 동안 잠시 중단이 발생합니다. Oracle Java Virtual Machine(JVM)이 DB 인스턴스에 이미 설치되어 있으면 중단이 발생하지 않습니다. 옵션을 추가한 후 DB 인스턴스를 재시작할 필요가 없습니다. 옵션 그룹이 활성화되는 즉시 Oracle Spatial을 사용할 수 있습니다.

> **Note**
>
> 이 중단 기간 동안에는 암호 확인 기능이 잠시 비활성화됩니다. 중단 기간 중에 암호 확인 기능과 관련된 이벤트를 볼 수도 있습니다. Oracle DB 인스턴스를 사용하기 전에 암호 확인 기능이 다시 활성화됩니다.

SPATIAL 옵션을 DB 인스턴스에 추가하려면

1. 사용할 옵션 그룹을 결정합니다. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 사용합니다. 기존 옵션 그룹을 사용하려면 다음 단계로 건너뛰십시오. 그렇지 않으면 다음 설정을 사용하여 사용자 지정 DB 옵션을 생성합니다.

   a. 엔진에서 DB 인스턴스의 Oracle 에디션을 선택합니다.

   b. 메이저 엔진 버전에서 DB 인스턴스의 버전을 선택합니다.

   자세한 내용은 옵션 그룹 생성 섹션을 참조하세요.
2. [SPATIAL] 옵션을 옵션 그룹에 추가합니다. 옵션 추가에 대한 자세한 내용은 옵션 그룹에 옵션 추가 섹션을 참조하세요.
3. 옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스에 적용합니다:

   - 새 DB 인스턴스의 경우, 인스턴스를 시작할 때 옵션 그룹을 적용합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
   - 기존 DB 인스턴스의 경우, 해당 인스턴스를 수정하고 새 옵션 그룹을 연결하여 옵션 그룹을 적용합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

Oracle Spatial 옵션 제거

SPATIAL 옵션에서 제공하는 데이터 유형을 사용하는 모든 객체를 삭제한 후 DB 인스턴스에서 옵션을 삭제할 수 있습니다. Oracle Java Virtual Machine(JVM)이 DB 인스턴스에 설치되어 있지 않은 경우 SPATIAL 옵션이 제거되는 동안 잠시 중단이 발생합니다. Oracle Java Virtual Machine(JVM)이 DB 인스턴스에 이미 설치되어 있으면 중단이 발생하지 않습니다. SPATIAL 옵션을 제거한 후 DB 인스턴스를 재시작할 필요가 없습니다.

SPATIAL 옵션을 삭제하려면

1. 데이터를 백업합니다.

> **Warning**
>
> 인스턴스에서 옵션의 일부로 활성화된 데이터 유형을 사용하고 SPATIAL 옵션을 제거하면 데이터가 손실될 수 있습니다. 자세한 내용은 데이터 백업, 복원 및 내보내기 섹션을 참조하세요.

2. 기존 객체가 SPATIAL 옵션의 데이터 유형 또는 기능을 참조하는지 확인합니다.

   SPATIAL 옵션이 있는 경우 SPATIAL 옵션이 없는 새 옵션 그룹을 적용할 때 인스턴스가 중단될 수 있습니다. 다음 쿼리를 사용하여 객체를 식별할 수 있습니다.

```
SELECT OWNER, SEGMENT_NAME, TABLESPACE_NAME, BYTES/1024/1024 mbytes
FROM   DBA_SEGMENTS
WHERE  SEGMENT_TYPE LIKE '%TABLE%'
AND    (OWNER, SEGMENT_NAME) IN
       (SELECT DISTINCT OWNER, TABLE_NAME
        FROM   DBA_TAB_COLUMNS
        WHERE  DATA_TYPE='SDO_GEOMETRY'
        AND    OWNER <> 'MDSYS')
ORDER BY 1,2,3,4;

SELECT OWNER, TABLE_NAME, COLUMN_NAME
FROM   DBA_TAB_COLUMNS
WHERE  DATA_TYPE = 'SDO_GEOMETRY'
AND    OWNER <> 'MDSYS'
ORDER BY 1,2,3;
```

3. SPATIAL 옵션의 데이터 유형 또는 기능을 참조하는 모든 객체를 삭제합니다.
4. 다음 중 하나를 수행하십시오.

   - 소속 옵션 그룹에서 SPATIAL 옵션을 제거합니다. 이 변경은 해당 옵션 그룹을 사용하는 모든 DB 인스턴스에 영향을 미칩니다. 자세한 내용은 옵션 그룹에서 옵션 제거 섹션을 참조하세요.
   - DB 인스턴스를 수정하고 SPATIAL 옵션이 포함되지 않은 다른 옵션 그룹을 지정합니다. 이 변경은 단일 DB 인스턴스에 영향을 미칩니다. 기본(빈) 옵션 그룹을 지정하거나 다른 사용자 지정 옵션 그룹을 지정할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

Oracle SQLT

Amazon RDS는 SQLT 옵션 사용을 통해 Oracle SQLTXPLAIN(SQLT)을 지원합니다. 모든 버전의 Oracle Database 19c 이상에서 SQLT를 사용할 수 있습니다.

Oracle `EXPLAIN PLAN` 문은 SQL 문의 실행 계획을 확인할 수 있습니다. 이 문은 Oracle 옵티마이저가 특정 실행 계획(예: 중첩 루프 조인)을 선택했는지 확인할 수 있습니다. 또한 옵티마이저의 결정(예: 해시 조인이 아니라 중첩 루프 조인을 선택한 이유)을 이해하도록 돕습니다. 따라서 `EXPLAIN PLAN`을 통해 문의 성능을 파악할 수 있습니다.

SQLT는 보고서를 생성하는 Oracle 유틸리티입니다. 이 보고서에는 객체 통계, 객체 메타데이터, 옵티마이저 관련 초기화 파라미터와 데이터베이스 관리자가 최적의 성능을 위해 SQL 문을 튜닝하는 데 사용할 수 있는 기타 정보가 포함됩니다. SQLT는 모든 섹션에 대한 하이퍼링크가 포함된 HTML 보고서를 생성합니다.

Automatic Workload Repository 또는 Statspack 보고서와 달리 SQLT는 개별 SQL 문에서 작동합니다. SQLT는 성능 데이터를 수집, 저장 및 표시하는 SQLT, PL/SQL 및 SQL*Plus 파일의 모음입니다.

다음은 각 SQLT 버전에 지원되는 Oracle 버전입니다.

SQLT 버전	Oracle Database 21c	Oracle Database 19c
2018-07-25.v1	지원	지원
2018-03-31.v1	지원되지 않음	지원되지 않음
2016-04-29.v1	지원되지 않음	지원되지 않음

이를 사용하기 위해 SQLT 및 액세스 지침을 다운로드하려면:

- My Oracle 지원 계정에 로그인하고 다음 문서를 여십시오.
- SQLT를 다운로드하려면: 문서 215187.1
- SQLT 사용 지침: 문서 1614107.1
- SQLT에 대한 FAQ: 문서 1454160.1
- SQLT 출력 읽기에 대한 자세한 내용: 문서 1456176.1
- 주요 보고서: 문서 1922234.1를 해석하려면

Amazon RDS는 다음 SQLT 메서드를 지원하지 않습니다.

- XPLORE
- XHUME

SQLT의 사전 조건

다음은 SQLT 사용을 위한 사전 조건입니다.

- SQLT에 필요한 사용자 및 역할을 제거해야 합니다(있는 경우).

  SQLT 옵션이 DB 인스턴스에 다음 사용자 및 역할을 생성합니다.

  - SQLTXPLAIN user
  - SQLTXADMIN user
  - SQLT_USER_ROLE 역할

  DB 인스턴스에 이러한 사용자 또는 역할이 있는 경우 SQL 클라이언트를 사용해 DB 인스턴스에 로그인하여 다음 문을 사용하여 삭제하십시오.

  ```
  DROP USER SQLTXPLAIN CASCADE;
  DROP USER SQLTXADMIN CASCADE;
  DROP ROLE SQLT_USER_ROLE CASCADE;
  ```

- SQLT에 필요한 테이블 공간을 제거해야 합니다(있는 경우).

  SQLT 옵션이 DB 인스턴스에 다음 테이블 공간을 생성합니다.

  - RDS_SQLT_TS
  - RDS_TEMP_SQLT_TS

  DB 인스턴스에 이러한 테이블 공간이 있는 경우 SQL 클라이언트를 사용해 DB 인스턴스에 로그인하여 삭제하십시오.

SQLT 옵션 설정

SQLT에서는 Oracle Tuning Pack 및 Oracle Diagnostics Pack에서 제공하는 사용이 허가된 기능을 사용할 수 있습니다. Oracle Tuning Pack에는 SQL Tuning Advisor가 들어 있고, Oracle Diagnostics Pack에는 Automatic Workload Repository가 들어 있습니다. SQLT 설정에 따라 SQLT에서 이러한 기능에 대한 액세스가 활성화 또는 비활성화됩니다.

Amazon RDS는 SQLT 옵션에 대해 다음 설정을 지원합니다.

옵션 설정	유효한 값	기본값	설명
LICENSE_PACK	T, D, N	N	SQLT를 사용하여 액세스하려고 하는 Oracle Management Pack입니다. 다음 값 중 하나를 입력합니다. • T는 Oracle Tuning Pack 및 Oracle Diagnostics Pack에 대한 라이선스가 있고 SQLT에서 SQL Tuning Advisor 및 Automatic Workload Repository에 액세스하려고 함을 나타냅니다. • D는 Oracle Diagnostics Pack에 대한 라이선스가 있고 SQLT에서 Automatic Workload Repository에 액세스하려고 함을 나타냅니다. • N은 Oracle Tuning Pack 및 Oracle Diagnostics Pack에 대한 라이선스가 없거나 둘 중 하나 또는 둘 다에 대한 라이선스가 있지만 SQLT에서 이러한 팩에 액세스하려고 하지 않음을 나타냅니다. **Note** Amazon RDS는 Oracle Management Pack에 대한 라이선스를 제공하지 않습니다. DB에 포함되어 있는 않은 팩을 사용하려고 하는 경우에는 DB 인스턴스에서 SQLT를 사용할 수 있습니다. 그러나 SQLT에서 해당 팩에 액세스할 수 없고 SQLT 보고서에 해당 팩에 대한 데이터가 포함되지 않습니다.

옵션 설정	유효한 값	기본값	설명
			예를 들어, T를 지정한 경우 DB 인스턴스에는 Oracle Tuning Pack이 포함되지 않고, SQLT가 DB 인스턴스에서 작동하지만 SQLT 보고서에는 Oracle Tuning Pack과 관련된 데이터가 들어 있지 않습니다.
VERSION	2016-04-29.v1 2018-03-31.v1 2018-07-25.v1	2016-04-29.v1	설치하려는 SQLT 버전 **Note** Oracle Database 19c 및 21c의 경우 지원되는 유일한 버전은 2018-07-25.v1 입니다. 이 버전은 이러한 릴리스의 기본값입니다.

SQLT 옵션 추가

SQLT 옵션을 DB 인스턴스에 추가하는 일반적인 프로세스는 다음과 같습니다.

1. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 복사 또는 수정합니다.
2. SQLT 옵션을 옵션 그룹에 추가합니다.
3. 옵션 그룹을 DB 인스턴스에 연동시킵니다.

SQLT 옵션을 추가하면 옵션 그룹이 활성화되는 순간, SQLT가 활성화됩니다.

DB 인스턴스에 SQLT 옵션을 추가하려면

1. 사용할 옵션 그룹을 결정합니다. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 사용합니다. 기존 옵션 그룹을 사용하려면 다음 단계로 건너뛰십시오. 그렇지 않으면 다음 설정을 사용하여 사용자 지정 DB 옵션을 생성합니다.

a. [Engine]에서 사용할 Oracle 버전을 선택합니다. 모든 버전에서 SQLT 옵션이 지원됩니다.

b. 메이저 엔진 버전에서 DB 인스턴스의 버전을 선택합니다.

자세한 내용은 옵션 그룹 생성 섹션을 참조하세요.

2. [SQLT] 옵션을 옵션 그룹에 추가합니다. 옵션 추가에 대한 자세한 내용은 옵션 그룹에 옵션 추가 섹션을 참조하세요.
3. 옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스에 적용합니다:

   - 새 DB 인스턴스의 경우, 인스턴스를 시작할 때 옵션 그룹을 적용합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
   - 기존 DB 인스턴스의 경우, 해당 인스턴스를 수정하고 새 옵션 그룹을 연결하여 옵션 그룹을 적용합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

4. (선택 사항) SQLT 옵션을 사용하여 각 DB 인스턴스에서 SQLT 설치를 확인합니다.

   a. SQL 클라이언트를 사용하여 마스터 관리자로 DB 인스턴스에 연결합니다.

   SQL 클라이언트를 사용하여 Oracle DB 인스턴스에 연결하는 방법에 대한 자세한 내용은 Oracle DB 인스턴스에 연결 단원을 참조하십시오.

   b. 다음 쿼리를 실행합니다.

```
SELECT sqltxplain.sqlt$a.get_param('tool_version') sqlt_version FROM DUAL;
```

   이 쿼리는 Amazon RDS에 있는 SQLT 옵션의 현재 버전을 반환합니다. `12.1.160429`는 Amazon RDS에서 사용 가능한 SQLT 버전의 예입니다.

5. SQLT 옵션에서 생성한 사용자의 암호를 변경합니다.

   a. SQL 클라이언트를 사용하여 마스터 관리자로 DB 인스턴스에 연결합니다.

   b. 다음 SQL 문을 실행하여 `SQLTXADMIN` 사용자의 암호를 변경합니다.

```
ALTER USER SQLTXADMIN IDENTIFIED BY new_password ACCOUNT UNLOCK;
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

c. 다음 SQL 문을 실행하여 SQLTXPLAIN 사용자의 암호를 변경합니다.

```
ALTER USER SQLTXPLAIN IDENTIFIED BY new_password ACCOUNT UNLOCK;
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

> **Note**
>
> SQLT를 업그레이드하려면 이전 버전의 SQLT를 제거한 다음 새 버전을 설치해야 합니다. 따라서 SQLT를 업그레이드하면 모든 SQLT 메타데이터가 손실될 수 있습니다. 데이터베이스의 메이저 버전 업그레이드 역시 SQLT를 제거한 후 다시 설치합니다. 메이저 버전 업그레이드의 예로는 Oracle Database 19c에서 Oracle Database 21c로의 업그레이드가 있습니다.

SQLT 사용

SQLT는 Oracle SQL*Plus 유틸리티와 함께 작동합니다.

SQLT를 사용하려면

1. My Oracle Support 사이트의 문서 215187.1에서 SQLT .zip 파일을 다운로드합니다.

> **Note**
>
> My Oracle Support 사이트에서 SQLT 12.1.160429를 다운로드할 수 없습니다. Oracle에는 이 구 버전이 사용되지 않습니다.

2. SQLT .zip 파일의 압축을 풉니다.
3. 명령 프롬프트에서 파일 시스템의 `sqlt/run` 디렉터리로 변경합니다.
4. 명령 프롬프트에서 SQL*Plus를 열고 DB 인스턴스에 마스터 사용자로 연결합니다.

   SQL*Plus를 사용한 DB 인스턴스 연결에 대한 자세한 내용은 Oracle DB 인스턴스에 연결 단원을 참조하십시오.
5. SQL 문의 SQL ID를 가져옵니다.

```
SELECT SQL_ID FROM V$SQL WHERE SQL_TEXT='sql_statement';
```

   다음과 유사하게 출력됩니다.

```
SQL_ID
-------------
chvsmttqjzjkn
```

6. SQLT를 사용하여 SQL 문을 분석합니다.

```
START sqltxtract.sql sql_id sqltxplain_user_password
```

   예를 들어 SQL ID `chvsmttqjzjkn`의 경우 다음과 같이 입력합니다.

```
START sqltxtract.sql chvsmttqjzjkn sqltxplain_user_password
```

   SQLT가 SQLT 명령이 실행된 디렉터리에 HTML 보고서 및 관련 리소스를 .zip 파일로 생성합니다.
7. (선택 사항) 애플리케이션 사용자가 SQLT를 사용하여 SQL 문을 진단하도록 하려면 다음 문을 사용하여 각 애플리케이션 사용자에게 `SQLT_USER_ROLE`을 부여하십시오.

```
GRANT SQLT_USER_ROLE TO application_user_name;
```

> **Note**
>
> Oracle에서는 SYS 사용자 또는 DBA 역할을 가진 사용자가 SQLT를 실행하는 것을 권장하지 않습니다. 애플리케이션 사용자에게 SQLT_USER_ROLE을 부여해 애플리케이션 사용자의 계정으로 SQLT 진단을 실행하는 것이 가장 좋습니다.

SQLT 옵션 업그레이드

Amazon RDS for Oracle을 사용하면 SQLT 옵션을 기존 버전에서 상위 버전으로 업그레이드할 수 있습니다. SQLT 옵션을 업그레이드하려면 SQLT 새 버전에 대한 SQLT 사용의 1단계–3단계를 완료하십시오. 또한, 이 섹션의 7단계에서 SQLT 이전 버전에 대한 권한을 부여한 경우 새 SQLT 버전에 대한 이 권한을 다시 부여하십시오.

SQLT 옵션을 업그레이드하면 이전 SQLT 버전의 메타데이터가 손실됩니다. 이전 SQLT 버전의 스키마 및 관련 객체는 삭제되고 최신 버전의 SQLT가 설치됩니다. 최신 SQLT 버전의 변경 사항에 대한 자세한 내용은 My Oracle Support 사이트의문서 1614201.1을 참조하십시오.

> **Note**
>
> 버전 다운그레이드는 지원되지 않습니다.

SQLT 설정 수정

SQLT를 활성화한 후에는 이 옵션에 대한 LICENSE_PACK 및 VERSION 설정을 수정할 수 있습니다.

옵션 설정을 변경하는 방법에 대한 자세한 내용은 옵션 설정 수정을(를) 참조하십시오. 각 설정에 대한 자세한 내용은 SQLT 옵션 설정 단원을 참조하십시오.

SQLT 옵션 제거

DB 인스턴스에서 SQLT를 제거할 수 있습니다.

DB 인스턴스에서 SQLT를 제거하려면 다음 중 하나를 수행합니다.

- 여러 DB 인스턴스에서 SQLT를 제거하려면 DB 인스턴스가 속한 옵션 그룹에서 해당 SQLT 옵션을 제거합니다. 이 변경은 해당 옵션 그룹을 사용하는 모든 DB 인스턴스에 영향을 미칩니다. 자세한 내용은 옵션 그룹에서 옵션 제거 섹션을 참조하세요.

- 단일 DB 인스턴스에서 SQLT를 제거하려면 DB 인스턴스를 수정하고 SQLT 옵션이 포함되지 않은 다른 옵션 그룹을 지정합니다. 기본(빈) 옵션 그룹을 지정하거나 다른 사용자 지정 옵션 그룹을 지정할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정을 참조하세요.

Oracle Statspack

Oracle 옵션(STATSPACK)은 Oracle Statspack 성능 통계 기능을 설치하고 활성화합니다. Oracle Statspack은 성능 데이터를 수집, 저장 및 표시하는 SQL, PL/SQL 및 SQL*Plus 스크립트의 모음입니다. Oracle Statspack 사용에 대한 자세한 정보는 Oracle 설명서의 Oracle Statspack 단원을 참조하십시오.

> **Note**
>
> Oracle Statspack은 Oracle에서 더 이상 지원되지 않으며 어드밴스 Automatic Workload Repository(AWR)로 대체되었습니다. AWR은 Diagnostics Pack을 구입한 Oracle Enterprise Edition 고객만 사용할 수 있습니다. Amazon RDS의 Oracle DB 엔진과 함께 Oracle Statspack을 사용할 수 있습니다. Amazon RDS 읽기 전용 복제본에서는 Oracle Statspack을 실행할 수 없습니다.

Oracle Statspack 설정

Statspack 스크립트를 실행하려면 Statspack 옵션을 추가해야 합니다.

Oracle Statspack을 설정하려면

1. SQL 클라이언트에서 관리 계정으로 Oracle DB에 로그인합니다.
2. Statspack이 설치되어 있는지 여부에 따라 다음 작업 중 하나를 수행합니다.
   - Statspack이 설치되어 있고 PERFSTAT 계정이 Statspack과 연결된 경우 4단계로 건너뜁니다.
   - Statspack이 설치되어 있지 않고 PERFSTAT 계정이 있는 경우 다음과 같이 계정을 삭제합니다.

     ```
     DROP USER PERFSTAT CASCADE;
     ```

     그렇지 않으면 Statspack 옵션을 추가하려고 하면 오류와 RDS-Event-0058이 생성됩니다.
3. Statspack 옵션을 옵션 그룹에 추가합니다. 옵션 그룹에 옵션 추가 단원을 참조하십시오.

   Amazon RDS는 자동으로 DB 인스턴스에 Statspack 스크립트를 설치한 다음 PERFSTAT 계정을 설정합니다.
4. 다음 SQL 문을 사용하여 암호를 재설정하고 pwd를 새 암호로 바꿉니다.

   ```
   ALTER USER PERFSTAT IDENTIFIED BY pwd ACCOUNT UNLOCK;
   ```

PERFSTAT 사용자 계정을 사용하여 로그인하고 Statspack 스크립트를 실행할 수 있습니다.

5. 다음 명령문을 사용하여 CREATE JOB 권한을 PERFSTAT 계정에 부여합니다.

```
GRANT CREATE JOB TO PERFSTAT;
```

6. PERFSTAT.STATS$IDLE_EVENT 테이블의 유휴 대기 이벤트가 채워져 있는지 확인합니다.

   Oracle 버그 28523746 때문에 PERFSTAT.STATS$IDLE_EVENT의 유휴 대기 이벤트가 채워지지 않을 수 있습니다. 모든 유휴 이벤트를 사용할 수 있도록 하려면 다음 명령문을 실행합니다.

```
INSERT INTO PERFSTAT.STATS$IDLE_EVENT (EVENT)
SELECT NAME FROM V$EVENT_NAME WHERE WAIT_CLASS='Idle'
MINUS
SELECT EVENT FROM PERFSTAT.STATS$IDLE_EVENT;
COMMIT;
```

Statspack 보고서 생성

Statspack 보고서는 두 개의 스냅샷을 비교합니다.

Statspack 보고서를 생성하려면

1. SQL 클라이언트에서 PERFSTAT 계정으로 Oracle DB에 로그인합니다.
2. 다음 방법 중 하나를 사용하여 스냅샷을 생성합니다.

   - Statspack 스냅샷을 수동으로 생성합니다.
   - 지정된 시간 간격 후에 Statspack 스냅샷을 만드는 작업을 생성합니다. 예를 들어, 다음 작업은 Statspack 스냅샷을 1시간마다 생성합니다.

```
VARIABLE jn NUMBER;
exec dbms_job.submit(:jn, 'statspack.snap;',SYSDATE,'TRUNC(SYSDATE
+1/24,''HH24'')');
COMMIT;
```

3. 다음 쿼리를 사용하여 스냅샷을 봅니다.

```
SELECT SNAP_ID, SNAP_TIME FROM STATS$SNAPSHOT ORDER BY 1;
```

4. Amazon RDS 프로시저 `rdsadmin.rds_run_spreport`를 실행하여 begin_snap 및 end_snap을 스냅샷 ID로 바꿉니다.

```
exec rdsadmin.rds_run_spreport(begin_snap,end_snap);
```

예를 들어, 다음 명령은 Statspack 스냅샷 1과 2 사이의 간격을 기반으로 보고서를 생성합니다.

```
exec rdsadmin.rds_run_spreport(1,2);
```

Statspack 보고서의 파일 이름에는 두 스냅샷의 번호가 포함됩니다. 예를 들어, Statspack 스냅샷 1과 2를 사용하여 생성한 보고서 파일의 이름은 `ORCL_spreport_1_2.lst`가 됩니다.

5. 출력을 모니터링하여 오류가 있는지 확인합니다.

Oracle Statspack은 보고서를 실행하기 전에 검사를 수행합니다. 따라서 명령 출력에 오류 메시지가 표시될 수도 있습니다. 예를 들어 Statspack 스냅샷 시작 값이 종료 값보다 큰 잘못된 범위를 기반으로 보고서를 생성하려고 할 수 있습니다. 이 경우 출력에 오류 메시지가 표시되지만 DB 엔진은 오류 파일을 생성하지 않습니다.

```
exec rdsadmin.rds_run_spreport(2,1);
*
ERROR at line 1:
ORA-20000: Invalid snapshot IDs. Find valid ones in perfstat.stats$snapshot.
```

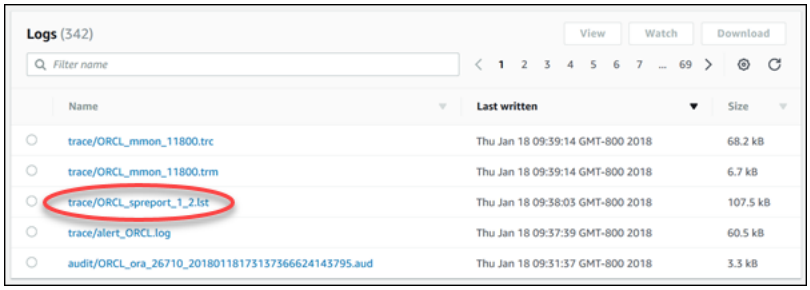
Statspack 스냅샷에 잘못된 숫자를 사용하는 경우 출력에 오류가 표시됩니다. 예를 들어 스냅샷 1과 50에 대한 보고서를 생성하려고 하지만 스냅샷 50이 없는 경우 출력에 오류가 표시됩니다.

```
exec rdsadmin.rds_run_spreport(1,50);
*
ERROR at line 1:
ORA-20000: Could not find both snapshot IDs
```

6. (선택 사항)

보고서를 검색하려면 Oracle 추적 파일을 사용한 작업의 설명과 같이 추적 파일 프로시저를 호출합니다.

또는 RDS 콘솔에서 Statspack 보고서를 다운로드합니다. DB 인스턴스 세부 정보의 로그 섹션으로 이동하여 다운로드를 선택합니다. 다음 예제는 `trace/ORCL_spreport_1_2.lst`를 보여줍니다.



보고서를 생성하는 동안 오류가 발생하면 DB 엔진은 보고서와 동일한 명명 규칙을 사용하지만, `.err`이라는 확장명을 지정합니다. 예를 들어, Statspack 스냅샷 1과 7을 사용하여 보고서를 생성하는 동안 오류가 발생한 경우 보고서 파일의 이름은 `ORCL_spreport_1_7.err`이 됩니다. 표준 스냅샷 보고서와 동일한 기법을 사용하여 오류 보고서를 다운로드할 수 있습니다.

Statspack 스냅샷 제거

여러 Oracle Statspack 스냅샷을 제거하려면 다음 명령을 사용합니다.

```
exec statspack.purge(begin snap, end snap);
```

Oracle 시간대

Oracle DB 인스턴스에서 사용하는 시스템 시간대를 변경하려면 시간대 옵션을 사용합니다. 예를 들면 온프레미스 환경 또는 기존 애플리케이션과 시간을 호환하기 위해 DB 인스턴스의 시간대를 변경할 수 있습니다. 시간대 옵션은 호스트 레벨에서 시간대를 변경합니다. 시간대를 변경하면 SYSDATE 및 SYSTIMESTAMP를 비롯한 모든 날짜 열과 값이 영향을 받습니다.

시간대 옵션은 rdsadmin_util.alter_db_time_zone 명령과 다릅니다. alter_db_time_zone 명령은 특정 데이터 유형의 시간대만 변경합니다. 시간대 옵션은 모든 날짜 열과 값의 시간대를 변경합니다. For more information about alter_db_time_zone, see 데이터베이스 시간대 설정. 업그레이드 고려 사항에 대한 자세한 내용은 시간대 고려 사항 단원을 참조하십시오.

표준 시간대 설정에 대한 제한 사항

시간대 옵션은 영구적이고 지속적인 옵션입니다. 따라서 다음을 수행할 수 없습니다.

- 옵션을 추가한 후에는 옵션 그룹에서 표준 시간대 옵션을 제거합니다.
- 그룹을 추가한 후에는 DB 인스턴스에서 이 옵션 그룹을 제거합니다.
- 옵션의 시간대 설정을 다른 시간대로 수정합니다.

표준 시간대 설정을 위한 권장 사항

시간대 옵션을 프로덕션 데이터베이스에 추가하기 전에 다음을 수행하는 것이 좋습니다.

- DB 인스턴스의 스냅샷을 만듭니다. 실수로 표준 시간대를 잘못 설정한 경우 DB 인스턴스를 이전 표준 시간대 설정으로 복구해야 합니다. 자세한 내용은 Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성 단원을 참조하십시오.
- 테스트 DB 인스턴스에 시간대 옵션을 추가합니다. 시간대 옵션을 추가하면 시스템 날짜를 이용해 날짜나 시간을 추가하는 테이블에 문제가 발생할 수 있습니다. 테스트 인스턴스에서 데이터와 애플리케이션을 분석하는 것이 좋습니다. 이렇게 하면 표준 시간대 변경이 프로덕션 인스턴스에 미치는 영향을 평가할 수 있습니다.

DB 인스턴스가 기본 옵션 그룹을 사용한다면 다음 단계를 따르세요.

1. DB 인스턴스의 스냅샷을 만듭니다.
2. DB 인스턴스에 표준 시간대 옵션을 추가합니다.

현재 DB 인스턴스가 기본이 아닌 옵션 그룹을 사용한다면 다음 단계를 따르세요.

1. DB 인스턴스의 스냅샷을 만듭니다.
2. 새 옵션 그룹을 생성합니다.
3. 기존 옵션 그룹과 현재 연결되어 있는 다른 모든 옵션과 함께 시간대 옵션을 여기에 추가합니다.

   이렇게 하면 시간대 옵션을 활성화하는 동안 기존 옵션이 제거되는 것을 방지할 수 있습니다.
4. 옵션 그룹을 DB 인스턴스에 추가합니다.

시간대 옵션 설정

Amazon RDS는 시간대 옵션에 대해 다음 설정을 지원합니다.

옵션 설정	유효한 값	설명
TIME_ZONE	사용 가능한 시간대 중 하나입니다. 전체 목록은 사용 가능한 시간대 단원을 참조하십시오.	DB 인스턴스에 대한 새 시간대를 선택합니다.

시간대 옵션 추가

DB 인스턴스에 표준 시간대 옵션을 추가하려면 다음 단계를 완료하세요.

1. (권장) DB 인스턴스의 스냅샷을 만듭니다.
2. 다음 태스크 중 하나를 수행합니다.
   - 처음부터 새 옵션 그룹을 만듭니다. 자세한 내용은 옵션 그룹 생성 단원을 참조하십시오.
   - AWS CLI 또는 API를 사용하여 기존 옵션 그룹을 복사합니다. 자세한 내용은 옵션 그룹 생성 단원을 참조하십시오.
   - 기본값이 아닌 기존 옵션 그룹을 재사용합니다. 가장 좋은 방법은 현재 DB 인스턴스나 스냅샷과 연결되어 있지 않은 옵션 그룹을 사용하는 것입니다.
3. 이전 단계의 옵션 그룹에 새 옵션을 추가합니다.
4. 현재 DB 인스턴스와 연결된 옵션 그룹에 활성화된 옵션이 있는 경우 이러한 옵션을 새 옵션 그룹에 추가합니다. 이 전략은 새 옵션을 활성화하는 동안 기존 옵션이 제거되는 것을 방지합니다.
5. 새 옵션 그룹을 DB 인스턴스에 추가합니다.

시간대 옵션을 추가하는 경우 DB 인스턴스를 자동으로 다시 시작하는 동안 인스턴스가 잠시 중단됩니다.

콘솔

시간대 옵션을 옵션 그룹에 추가하고 DB 인스턴스와 연결하는 방법

1. RDS 콘솔에서 옵션 그룹을 선택합니다.
2. 옵션을 추가할 옵션 그룹의 이름을 선택합니다.
3. 옵션 추가를 선택합니다.
4. 옵션 이름에서 표준 시간대를 선택한 다음 옵션 설정을 구성합니다.
5. 옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스와 연결합니다.

   - 새 DB 인스턴스의 경우, 인스턴스를 시작할 때 옵션 그룹을 적용하십시오. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
   - 기존 DB 인스턴스의 경우, 해당 인스턴스를 수정하고 새 옵션 그룹을 연결하여 옵션 그룹을 적용하십시오. 기존 DB 인스턴스에 새 옵션을 추가하는 경우 DB 인스턴스를 자동으로 다시 시작하는 동안 인스턴스가 잠시 중단됩니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

AWS CLI

다음 예에서는 AWS CLI add-option-to-option-group 명령을 사용하여 `Timezone` 옵션 및 `TIME_ZONE` 옵션 설정을 `myoptiongroup`이라는 옵션 그룹에 추가합니다. 표준 시간대는 `Africa/Cairo`로 설정되어 있습니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds add-option-to-option-group \
    --option-group-name "myoptiongroup" \
    --options "OptionName=Timezone,OptionSettings=[{Name=TIME_ZONE,Value=Africa/
Cairo}]" \
    --apply-immediately
```

Windows의 경우:

```
aws rds add-option-to-option-group ^
    --option-group-name "myoptiongroup" ^
```

```
    --options "OptionName=Timezone,OptionSettings=[{Name=TIME_ZONE,Value=Africa/
Cairo}]" ^
    --apply-immediately
```

시간대 설정 수정

시간대 옵션은 영구적이고 지속적인 옵션입니다. 추가한 후에는 옵션 그룹에서 이 옵션을 제거할 수 없습니다. 추가한 후에는 DB 인스턴스에서 옵션 그룹을 제거할 수 없습니다. 옵션의 시간대 설정을 다른 시간대로 수정할 수 없습니다. 시간대가 잘못 설정된 경우, 시간대 옵션을 추가하기 전에 DB 인스턴스의 스냅샷을 복구합니다.

시간대 옵션 제거

시간대 옵션은 영구적이고 지속적인 옵션입니다. 추가한 후에는 옵션 그룹에서 이 옵션을 제거할 수 없습니다. 추가한 후에는 DB 인스턴스에서 옵션 그룹을 제거할 수 없습니다. 시간대 옵션을 제거하려면, 시간대 옵션을 추가하기 전에 DB 인스턴스의 스냅샷을 복구합니다.

사용 가능한 시간대

시간대 옵션에 사용할 수 있는 값은 다음과 같습니다.

영역	시간대
아프리카	Africa/Cairo, Africa/Casablanca, Africa/Harare, Africa/Lagos, Africa/Luanda, Africa/Monrovia, Africa/Nairobi, Africa/Tripoli, Africa/Windhoek
아메리카	America/Araguaina, America/Argentina/Buenos_Aires, America/Asuncion, America/Bogota, America/Caracas, America/Chicago, America/Chihuahua, America/Cuiaba, America/Denver, America/Detroit, America/Fortaleza, America/Godthab, America/Guatemala, America/Halifax, America/Lima, America/Los_Angeles, America/Manaus, America/Matamoros, America/Mexico_City, America/Monterrey, America/Montevideo, America/New_York, America/Phoenix, America/Santiago, America/Sao_Paulo, America/Tijuana, America/Toronto
아시아	Asia/Amman, Asia/Ashgabat, Asia/Baghdad, Asia/Baku, Asia/Bangkok, Asia/Beirut, Asia/Calcutta, Asia/Damascus, Asia/Dhaka, Asia/Hong_Kong, Asia/Irkutsk, Asia/Jakarta, Asia/Jerusalem, Asia/Kabul, Asia/Karachi, Asia/Kathmandu, Asia/Kolkata, Asia/Krasnoyarsk, Asia/Magadan, Asia/Manila, Asia/

영역	시간대
	Muscat, Asia/Novosibirsk, Asia/Rangoon, Asia/Riyadh, Asia/Seoul, Asia/Shanghai, Asia/Singapore, Asia/Taipei, Asia/Tehran, Asia/Tokyo, Asia/Ulaanbaatar, Asia/Vladivostok, Asia/Yakutsk, Asia/Yerevan
대서양	Atlantic/Azores, Atlantic/Cape_Verde
호주	Australia/Adelaide, Australia/Brisbane, Australia/Darwin, Australia/Eucla, Australia/Hobart, Australia/Lord_Howe, Australia/Perth, Australia/Sydney
브라질	Brazil/DeNoronha, Brazil/East
캐나다	Canada/Newfoundland, Canada/Saskatchewan
기타	Etc/GMT-3
유럽	Europe/Amsterdam, Europe/Athens, Europe/Berlin, Europe/Dublin, Europe/Helsinki, Europe/Kaliningrad, Europe/London, Europe/Madrid, Europe/Moscow, Europe/Paris, Europe/Prague, Europe/Rome, Europe/Sarajevo
태평양	Pacific/Apia, Pacific/Auckland, Pacific/Chatham, Pacific/Fiji, Pacific/Guam, Pacific/Honolulu, Pacific/Kiritimati, Pacific/Marquesas, Pacific/Samoa, Pacific/Tongatapu, Pacific/Wake
US	US/Alaska, US/Central, US/East-Indiana, US/Eastern, US/Pacific
UTC	UTC

Oracle 시간대 파일 자동 업그레이드

TIMEZONE_FILE_AUTOUPGRADE 옵션을 사용하면 현재 시간대 파일을 RDS for Oracle DB 인스턴스의 최신 버전으로 업그레이드할 수 있습니다.

주제

- Oracle 시간대 파일 개요
- 시간대 파일 업데이트 전략
- 시간대 파일 업데이트 중 가동 중지 시간
- 시간대 파일 업데이트 준비
- 시간대 파일 자동 업그레이드 옵션 추가
- 시간대 파일을 업데이트 한 후 데이터 확인

Oracle 시간대 파일 개요

Oracle 데이터베이스 시간대 파일은 다음 정보를 저장합니다.

- 협정 세계표준시(UTC)의 오프셋
- 일광 절약 시간(DST) 전환 시간
- 표준 시간 및 DST의 약어

Oracle Database는 여러 버전의 시간대 파일을 제공합니다. 온프레미스 환경에서 Oracle 데이터베이스를 만들 때 시간대 파일 버전을 선택합니다. 자세한 내용은 Oracle Database Globalization Support Guide의 Choosing a Time Zone File(시간대 파일 선택)을 참조하세요.

DST에 대한 규칙이 변경되면 Oracle은 새 표준 시간대 파일을 게시합니다. Oracle은 분기별 Release Updates(RUs) 및 Release Update Revisions(RURs) 일정과 별도로 이러한 새 표준 시간대 파일을 릴리스합니다. 시간대 파일은 데이터베이스 호스트의 $ORACLE_HOME/oracore/zoneinfo/ 디렉터리에 있습니다. 시간대 파일 이름은 DSTv35와 같이 DSTv*version* 형식을 사용합니다.

시간대 파일이 데이터 전송에 미치는 영향

Oracle Database에서 TIMESTAMP WITH TIME ZONE 데이터 형식은 타임스탬프 및 시간대 데이터를 저장합니다. TIMESTAMP WITH TIME ZONE 데이터 형식의 데이터는 연결된 시간대 파일 버전의 규칙을 사용합니다. 따라서 표준 시간대 파일을 업데이트하면 기존 TIMESTAMP WITH TIME ZONE 데이터가 영향을 받습니다.

다른 버전의 표준 시간대 파일을 사용하는 데이터베이스 간에 데이터를 전송할 때 문제가 발생할 수 있습니다. 예를 들어, 대상 데이터베이스보다 높은 시간대 파일 버전을 사용하는 소스 데이터베이스에서 데이터를 가져오려고 하면 데이터베이스에 ORA-39405 오류가 발생합니다. 이전에는 다음 방법 중 하나를 사용하여 이 오류를 해결해야 했습니다.

- 원하는 시간대 파일을 사용하여 RDS for Oracle DB 인스턴스를 생성하고 소스 데이터베이스에서 데이터를 내보낸 다음 새 데이터베이스로 가져옵니다.
- AWS DMS 또는 논리적 복제를 사용하여 데이터를 마이그레이션합니다.

TIZONE_파일_AUTOUPGRADE 옵션을 사용한 자동 업데이트

RDS for Oracle DB 인스턴스에 연결된 옵션 그룹에 TIMEZONE_FILE_AUTOUPGRADE 옵션이 포함되어 있으면 RDS가 시간대 파일을 자동으로 업데이트합니다. Oracle 데이터베이스에서 동일한 시간대 파일 버전을 사용하도록 하면 서로 다른 환경 간에 데이터를 이동할 때 시간이 많이 걸리는 수동 기술을 사용하지 않아도 됩니다. TIMEZONE_FILE_AUTOUPGRADE 옵션에서는 컨테이너 데이터베이스(CDB)와 비 CDB가 모두 지원됩니다.

TIMEZONE_FILE_AUTOUPGRADE 옵션을 추가할지 또는 유지 관리 기간에 추가할지 또는 유지 관리 기간에 추가할지 선택할 수 있습니다. DB 인스턴스에서 새 옵션을 적용한 후 RDS는 최신 DSTv*version* 파일을 설치할 수 있는지 확인합니다. 대상 DSTv ### 다음에 따라 달라집니다.

- DB 인스턴스가 현재 실행 중인 마이너 엔진 버전
- DB 인스턴스를 업그레이드하려는 마이너 엔진 버전

예를 들어, 현재 시간대 파일 버전은 DSTv33일 수 있습니다. RDS가 옵션 그룹에 업데이트를 적용할 때, DB 인스턴스 파일 시스템에서 현재 DSTv34를 사용 가능하다고 판단할 수 있습니다. 그러면 RDS가 표준 시간대 파일을 DSTv34로 자동 업데이트합니다.

지원되는 RDS 릴리스 업데이트에서 사용 가능한 DST 버전을 찾으려면 Amazon Relational Database Service(Amazon RDS) for Oracle 릴리스 정보의 패치를 살펴보세요. 예를 들어 버전 19.0.0.0.ru-2022-10.rur-2022-10.r1은 패치 34533061: RDBMS - DSTV39 UPDATE - TZDATA2022C를 나열합니다.

시간대 파일 업데이트 전략

DB 엔진 업그레이드와 옵션 그룹에 TIMEZONE_FILE_AUTOUPGRADE 옵션을 추가하는 작업은 별개입니다. TIMEZONE_FILE_AUTOUPGRADE 옵션을 추가하면 더 최신 파일을 사용할 수 있는 경우 시간대

파일의 업데이트가 시작됩니다. 다음 명령(관련 옵션만 표시됨)을 즉시 또는 다음 유지 관리 기간에 실행합니다.

- 다음 RDS CLI 명령을 사용해서만 DB 엔진을 업그레이드합니다.

```
modify-db-instance --engine-version name ...
```

- 다음 CLI 명령만 사용하여 TIMEZONE_FILE_AUTOUPGRADE 옵션을 추가합니다.

```
add-option-to-option-group --option-group-name name --options
 OptionName=TIMEZONE_FILE_AUTOUPGRADE ...
```

- 다음 CLI 명령을 사용하여 DB 엔진을 업그레이드하고 인스턴스에 새 옵션 그룹을 추가합니다.

```
modify-db-instance --engine-version name --option-group-name name ...
```

업데이트 전략은 데이터베이스와 시간대 파일을 함께 업그레이드할지 아니면 이러한 작업 중 하나만 수행할지 결정합니다. 옵션 그룹을 업데이트한 다음 별도의 API 작업으로 DB 엔진을 업그레이드하는 경우 DB 엔진을 업그레이드할 때 시간대 파일 업데이트가 현재 진행 중일 수 있다는 점에 유의하세요.

이 섹션의 예에서는 다음과 같이 가정합니다.

- TIMEZONE_FILE_AUTOUPGRADE를 DB 인스턴스와 현재 연결되어 있는 옵션 그룹에 아직 추가하지 않았습니다.
- DB 인스턴스가 데이터베이스 버전 19.0.0.0.ru-2019-07.rur-2019-07.r1 및 표준 시간대 파일 DSTv33을 사용합니다.
- DB 인스턴스 파일 시스템에 DSTv34 파일이 포함되어 있습니다.
- 릴리스 업데이트 19.0.0.0.ru-2022-10.rur-2022-10.r1에 DSTv35가 포함되어 있습니다.

다음 전략을 사용하여 시간대 파일을 업데이트할 수 있습니다.

주제

- 엔진을 업그레이드하지 않고 시간대 파일 업데이트
- 시간대 파일 및 DB 엔진 버전 업그레이드
- 시간대 파일 업데이트 없이 DB 엔진 버전 업그레이드

엔진을 업그레이드하지 않고 시간대 파일 업데이트

이 시나리오에서 데이터베이스는 DSTv33을 사용하고 있지만, DB 인스턴스 파일 시스템에서 DSTv34를 사용할 수 있습니다. DB 인스턴스에서 사용하는 표준 시간대 파일을 DSTv33에서 DSTv34로 업데이트하고 싶지만 엔진은 새 마이너 버전(DSTv35 포함됨)으로 업그레이드하지 않으려고 합니다.

`add-option-to-option-group` 명령으로 `TIMEZONE_FILE_AUTOUPGRADE`를 DB 인스턴스가 사용하는 옵션 그룹에 추가합니다. 옵션을 즉시 추가할지 아니면 유지 관리 기간으로 연기할지 지정합니다. RDS는 `TIMEZONE_FILE_AUTOUPGRADE` 옵션을 적용한 후 다음을 수행합니다.

1. 새 DST 버전이 있는지 확인합니다.
2. 파일 시스템에서 DSTv34를 사용할 수 있는지 확인합니다.
3. 시간대 파일을 즉시 업데이트합니다.

시간대 파일 및 DB 엔진 버전 업그레이드

이 시나리오에서 데이터베이스는 DSTv33을 사용하고 있지만, DB 인스턴스 파일 시스템에서 DSTv34를 사용할 수 있습니다. 엔진 업그레이드 중 DB 엔진을 마이너 버전 19.0.0.0.ru-2022-10.rur-2022-10.r1(DSTv35 포함됨)로 업그레이드하고, 표준 시간대 파일을 DSTv35로 업데이트하려고 합니다. 따라서 목표는 DSTv34를 건너뛰고 표준 시간대 파일을 DSTv35로 직접 업데이트하는 것입니다.

엔진과 시간대 파일을 함께 업그레이드하려면 `--option-group-name` 및 `--engine-version` 옵션과 함께 `modify-db-instance`를 실행합니다. 명령을 즉시 실행하거나 유지 관리 기간으로 연기할 수 있습니다. In `--option-group-name`에 `TIMEZONE_FILE_AUTOUPGRADE` 옵션이 포함된 옵션 그룹을 지정하세요. 예:

```
aws rds modify-db-instance
    --db-instance-identifier my-instance \
    --engine-version new-version \
    ----option-group-name og-with-timezone-file-autoupgrade \
    --apply-immediately
```

RDS는 엔진을 19.0.0.0.ru-2022-10.rur-2022-10.r1으로 업그레이드하기 시작합니다. `TIMEZONE_FILE_AUTOUPGRADE` 옵션이 적용된 후 RDS는 새 DST 버전을 확인하고, 19.0.0.0.ru-2022-10.rur-2022-10.r1에서 DSTv35를 사용할 수 있는지 확인한 후 즉시 DSTv35로 업데이트를 시작합니다.

엔진을 즉시 업그레이드한 다음 시간대 파일을 업그레이드하려면 작업을 순서대로 수행합니다.

1. 다음 CLI 명령을 사용해서만 DB 엔진을 업그레이드합니다.

```
aws rds modify-db-instance \
    --db-instance-identifier my-instance \
    --engine-version new-version \
    --apply-immediately
```

2. 다음 CLI 명령을 사용하여 인스턴스에 연결된 옵션 그룹에 TIMEZONE_FILE_AUTOUPGRADE 옵션을 추가합니다.

```
aws rds add-option-to-option-group \
    --option-group-name og-in-use-by-your-instance \
    --options OptionName=TIMEZONE_FILE_AUTOUPGRADE \
    --apply-immediately
```

시간대 파일 업데이트 없이 DB 엔진 버전 업그레이드

이 시나리오에서 데이터베이스는 DSTv33을 사용하고 있지만, DB 인스턴스 파일 시스템에서 DSTv34를 사용할 수 있습니다. DB 엔진을 버전 19.0.0.0.ru-2022-10.rur-2022-10.r1(DSTv35 포함됨)로 업그레이드하되, 표준 시간대 파일 DSTv33은 유지하려고 합니다. 다음과 같은 이유로 이 전략을 선택할 수 있습니다.

- 데이터가 TIMESTAMP WITH TIME ZONE 데이터 형식을 사용하지 않습니다.
- 데이터가 TIMESTAMP WITH TIME ZONE 데이터 형식을 사용하지만 시간대 변경의 영향을 받지 않습니다.
- 추가 가동 중지 시간을 허용할 수 없기 때문에 시간대 파일 업데이트를 연기하려고합니다.

전략은 다음 가능성 중 어느 것에 해당하는지에 따라 달라집니다.

- DB 인스턴스가 TIMEZONE_FILE_AUTOUPGRADE를 포함하는 옵션 그룹과 연결되어 있지 않습니다. RDS가 시간대 파일을 업데이트하지 않도록 modify-db-instance 명령으로 새 옵션 그룹을 지정하지 마세요.
- DB 인스턴스가 현재 TIMEZONE_FILE_AUTOUPGRADE를 포함하는 옵션 그룹과 연결되어 있습니다. 단일 modify-db-instance 명령 내에서 DB 인스턴스를 TIMEZONE_FILE_AUTOUPGRADE를 포함하지 않는 옵션 그룹에 연결하고 DB 엔진을 19.0.0.0.ru-2022-10.rur-2022-10.r1로 업그레이드합니다.

시간대 파일 업데이트 중 가동 중지 시간

RDS가 시간대 파일을 업데이트하면 `TIMESTAMP WITH TIME ZONE`을 사용하는 기존 데이터가 변경될 수 있습니다. 이 경우 주요 고려 사항은 가동 중지 시간입니다.

> **Warning**
>
> `TIMEZONE_FILE_AUTOUPGRADE` 옵션을 추가하면 엔진 업그레이드의 가동 중지 시간이 길어질 수 있습니다. 대규모 데이터베이스의 시간대 데이터를 업데이트하는 데에는 몇 시간 또는 며칠이 걸릴 수 있습니다.

시간대 파일 업데이트에 소요되는 시간은 다음과 같은 요인에 따라 달라집니다.

- 데이터베이스의 `TIMESTAMP WITH TIME ZONE` 데이터 양
- DB 인스턴스 구성
- DB 인스턴스 클래스
- 스토리지 구성
- 데이터베이스 구성
- 데이터베이스 파라미터 설정

다음을 수행할 때 추가 가동 중지 시간이 발생할 수 있습니다.

- DB 인스턴스가 오래된 시간대 파일을 사용할 때 옵션 그룹에 옵션 추가
- 새 엔진 버전에 시간대 파일의 새 버전이 포함되어 있을 때 Oracle 데이터베이스 엔진 업그레이드

> **Note**
>
> 시간대 파일을 업데이트하는 동안 RDS for Oracle는 `PURGE DBA_RECYCLEBIN`을 호출합니다.

시간대 파일 업데이트 준비

시간대 파일 업그레이드에는 준비와 업그레이드라는 두 가지 단계가 있습니다. 필수는 아니지만 준비 단계를 수행하는 것이 좋습니다. 이 단계에서는 PL/SQL 프로시저 `DBMS_DST.FIND_AFFECTED_TABLES`를 실행하여 영향을 받게 될 데이터를 찾습니다. 준비 기간에

대한 자세한 내용은 Oracle 데이터베이스 설명서에서 시간대 데이터를 사용하여 시간대 파일 및 타임스탬프 업그레이드를 참조하세요.

시간대 파일 업데이트를 준비하려면

1. SQL 클라이언트를 사용하여 Oracle 데이터베이스에 연결합니다.
2. 사용된 현재 시간대 파일 버전을 확인합니다.

```
SELECT * FROM V$TIMEZONE_FILE;
```

3. DB 인스턴스에 사용 가능한 최신 시간대 파일 버전을 확인합니다.

```
SELECT DBMS_DST.GET_LATEST_TIMEZONE_VERSION FROM DUAL;
```

4. TIMESTAMP WITH LOCAL TIME ZONE 또는 TIMESTAMP WITH TIME ZONE 유형의 열이 있는 테이블의 총 크기를 결정합니다.

```
SELECT SUM(BYTES)/1024/1024/1024 "Total_size_w_TSTZ_columns_GB"
FROM   DBA_SEGMENTS
WHERE  SEGMENT_TYPE LIKE 'TABLE%'
AND    (OWNER, SEGMENT_NAME) IN
         (SELECT OWNER, TABLE_NAME
          FROM   DBA_TAB_COLUMNS
          WHERE  DATA_TYPE LIKE 'TIMESTAMP%TIME ZONE');
```

5. TIMESTAMP WITH LOCAL TIME ZONE 또는 TIMESTAMP WITH TIME ZONE 유형의 열이 있는 세그먼트의 이름과 크기를 결정합니다.

```
SELECT OWNER, SEGMENT_NAME, SUM(BYTES)/1024/1024/1024
 "SEGMENT_SIZE_W_TSTZ_COLUMNS_GB"
FROM   DBA_SEGMENTS
WHERE  SEGMENT_TYPE LIKE 'TABLE%'
AND    (OWNER, SEGMENT_NAME) IN
         (SELECT OWNER, TABLE_NAME
          FROM   DBA_TAB_COLUMNS
          WHERE  DATA_TYPE LIKE 'TIMESTAMP%TIME ZONE')
GROUP BY OWNER, SEGMENT_NAME;
```

6. 준비 단계를 실행합니다.

   - DBMS_DST.CREATE_AFFECTED_TABLE 프로시저는 영향을 받는 데이터를 저장할 테이블을 생성합니다. 이 테이블의 이름을 DBMS_DST.FIND_AFFECTED_TABLES 프로시저에 전달합니다.

자세한 내용은 Oracle 데이터베이스 설명서에서 CREATE_AFFECTED_TABLE 프로시저를 참조하세요.

- 이 프로시저 CREATE_ERROR_TABLE은 오류를 기록할 테이블을 생성합니다. 자세한 내용은 Oracle 데이터베이스 설명서에서 CREATE_ERROR_TABLE 프로시저를 참조하세요.

다음 예에서는 영향을 받는 데이터 및 오류 테이블을 만들고 영향을 받는 테이블을 모두 찾습니다.

```
EXEC DBMS_DST.CREATE_ERROR_TABLE('my_error_table')
EXEC DBMS_DST.CREATE_AFFECTED_TABLE('my_affected_table')

EXEC DBMS_DST.BEGIN_PREPARE(new_version);
EXEC DBMS_DST.FIND_AFFECTED_TABLES('my_affected_table', TRUE, 'my_error_table');
EXEC DBMS_DST.END_PREPARE;

SELECT * FROM my_affected_table;
SELECT * FROM my_error_table;
```

7. 영향을 받는 테이블과 오류 테이블을 쿼리합니다.

```
SELECT * FROM my_affected_table;
SELECT * FROM my_error_table;
```

시간대 파일 자동 업그레이드 옵션 추가

옵션을 옵션 그룹에 추가하면 옵션 그룹은 다음 상태 중 하나가 됩니다.

- 기존 옵션 그룹은 현재 하나 이상의 DB 인스턴스에 연결되어 있습니다. 옵션을 추가하면 이 옵션 그룹을 사용하는 모든 DB 인스턴스가 자동으로 다시 시작됩니다. 이로 인해 잠시 중단됩니다.
- 기존 옵션 그룹은 DB 인스턴스에 연결되어 있지 않습니다. 옵션을 추가한 다음 기존 옵션 그룹을 기존 DB 인스턴스 또는 새 DB 인스턴스와 연결할 계획입니다.
- 새 옵션 그룹을 생성하고 옵션을 추가합니다. 새 옵션 그룹을 기존 DB 인스턴스 또는 새 DB 인스턴스와 연결할 계획입니다.

콘솔

DB 인스턴스에 시간대 파일 자동 업그레이드 옵션을 추가하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 옵션 그룹을 선택합니다.
3. 사용할 옵션 그룹을 결정합니다. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 사용합니다. 기존 옵션 그룹을 사용하려면 다음 단계로 건너뛰십시오. 그렇지 않으면 다음 설정을 사용하여 사용자 지정 DB 옵션을 생성합니다.

   a. Engine(엔진)에서 DB 인스턴스에 대한 Oracle Database 에디션을 선택합니다.

   b. 메이저 엔진 버전에서 DB 인스턴스의 버전을 선택합니다.

   자세한 내용은 옵션 그룹 생성 단원을 참조하십시오.
4. 수정하려는 옵션 그룹을 선택한 다음 옵션 추가를 선택합니다.
5. 옵션 추가 창에서 다음과 같이 합니다.

   a. TIMEZONE_FILE_AUTOUPGRADE를 선택합니다.

   b. 옵션을 추가하는 즉시 연동된 모든 DB 인스턴스에서 옵션을 활성화하려면 Apply Immediately에서 Yes를 선택합니다. No(기본 설정)를 선택하면 다음 유지 관리 기간에 연동된 모든 DB 인스턴스에서 옵션이 활성화됩니다.
6. 원하는 대로 설정이 되었으면 옵션 추가를 선택합니다.

AWS CLI

다음 예에서는 AWS CLI add-option-to-option-group 명령을 사용하여 `TIMEZONE_FILE_AUTOUPGRADE` 옵션을 `myoptiongroup`이라는 옵션 그룹에 추가합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds add-option-to-option-group \
    --option-group-name "myoptiongroup" \
    --options "OptionName=TIMEZONE_FILE_AUTOUPGRADE" \
    --apply-immediately
```

Windows의 경우:

```
aws rds add-option-to-option-group ^
    --option-group-name "myoptiongroup" ^
    --options "OptionName=TIMEZONE_FILE_AUTOUPGRADE" ^
    --apply-immediately
```

시간대 파일을 업데이트 한 후 데이터 확인

시간대 파일을 업데이트한 후 데이터를 확인하는 것이 좋습니다. 준비 단계 동안 RDS for Oracle는 다음 테이블을 자동으로 생성합니다.

- `rdsadmin.rds_dst_affected_tables` - 업데이트의 영향을 받는 데이터가 포함된 테이블을 나열합니다.
- `rdsadmin.rds_dst_error_table` - 업데이트 중에 발생한 오류를 나열합니다.

이러한 테이블은 준비 기간에 생성하는 테이블과는 별개입니다. 업데이트 결과를 보려면 다음과 같이 테이블을 쿼리합니다.

```
SELECT * FROM rdsadmin.rds_dst_affected_tables;
SELECT * FROM rdsadmin.rds_dst_error_table;
```

영향을 받는 데이터 및 오류 테이블의 스키마에 대한 자세한 내용은 Oracle 설명서에서 FIND_AFFECTED_TABLES Procedure를 참조하세요.

Oracle Transparent Data Encryption

Amazon RDS는 Oracle Enterprise Edition에서 지원되는 Oracle Advanced Security 옵션의 한 가지 기능인 Oracle Transparent Data Encryption(TDE)을 지원합니다. 이 기능은 스토리지에 데이터를 쓰기 전에 자동으로 데이터를 암호화한 뒤에 데이터를 스토리지에서 읽을 때 다시 자동으로 해독합니다. 이 옵션은 기존 보유 라이선스 사용(BYOL) 모델에만 지원됩니다.

TDE는 제3자가 데이터 파일 및 백업을 가져올 경우 민감한 데이터를 암호화해야 하는 시나리오에서 유용합니다. TDE는 보안 관련 규정을 준수해야 하는 경우에도 유용합니다.

Oracle Database의 TDE에 대한 자세한 설명은 이 가이드의 범위를 벗어납니다. 자세한 내용은 다음 Oracle Database 리소스를 참조하세요.

- Oracle Database 설명서의 투명한 데이터 암호화 소개
- Oracle Database 설명서의 Oracle 고급 보안
- Oracle 백서의 Oracle 고급 보안 투명한 데이터 암호화 모범 사례

RDS for Oracle과 TDE를 사용하는 방법에 대한 자세한 내용은 다음 블로그를 참조하세요.

- Amazon RDS의 Oracle Database 암호화 옵션
- AWS DMS를 통해 가동 중지를 줄여 계정 간 TDE를 지원하는 Amazon RDS for Oracle DB 인스턴스 마이그레이션

TDE 암호화 모델

Oracle Transparent Data Encryption은 TDE 테이블스페이스 암호화 및 TDE 열 암호화의 두 가지 모드를 지원합니다. TDE 테이블스페이스 암호화는 전체 애플리케이션 테이블을 암호화하는 데 사용됩니다. TDE 열 암호화는 중요 데이터를 포함하는 개별 데이터 요소를 암호화하는 데 사용됩니다. TDE 테이블스페이스 암호화와 열 암호화를 모두 사용하는 하이브리드 암호화 솔루션을 적용할 수도 있습니다.

> **Note**
>
> Amazon RDS가 DB 인스턴스의 Oracle Wallet 및 TDE 마스터 키를 관리합니다. [`ALTER SYSTEM set encryption key`] 명령을 사용하여 암호화 키를 설정하지 않아도 됩니다.

TDE 옵션을 활성화한 후 다음 명령을 사용하여 Oracle Wallet의 상태를 확인할 수 있습니다.

```
SELECT * FROM v$encryption_wallet;
```

암호화된 테이블스페이스를 생성하려면 다음 명령을 사용합니다.

```
CREATE TABLESPACE encrypt_ts ENCRYPTION DEFAULT STORAGE (ENCRYPT);
```

암호화 알고리즘을 지정하려면 다음 명령을 사용하십시오.

```
CREATE TABLESPACE encrypt_ts ENCRYPTION USING 'AES256' DEFAULT STORAGE (ENCRYPT);
```

테이블스페이스 암호화에 대한 이전 문은 온프레미스 Oracle 데이터베이스에서 사용하는 것과 동일합니다.

TDE 옵션에 대한 제한 사항

TDE 옵션은 영구적이고 지속적입니다. TDE 옵션이 활성화된 옵션 그룹에 DB 인스턴스를 연결한 후에는 다음 작업을 수행할 수 없습니다.

- 현재 연결된 옵션 그룹에서 TDE 옵션을 비활성화합니다.
- DB 인스턴스를 TDE 옵션이 포함되지 않은 다른 옵션 그룹에 연결합니다.
- TDE 옵션을 사용하는 DB 스냅샷을 공유합니다. DB 스냅샷 공유에 대한 자세한 내용은 Amazon RDS의 DB 스냅샷 공유 단원을 참조하십시오.

지속 옵션 및 영구 옵션에 대한 자세한 내용은 지속적이거나 영구적인 옵션 섹션을 참조하세요.

DB 인스턴스가 TDE를 사용하는지 여부 결정

DB 인스턴스가 TDE 옵션을 활성화한 옵션 그룹과 연결되어 있는지 확인할 수 있습니다. DB 인스턴스와 연동되어 있는 옵션 그룹은 RDS 콘솔, describe-db-instance AWS CLI 명령 또는 API 작업 DescribeDBInstances를 사용하여 확인할 수 있습니다.

TDE 옵션 추가

DB 인스턴스에 TDE 옵션을 추가하려면 다음 단계를 완료하세요.

1. (권장) DB 인스턴스의 스냅샷을 만듭니다.
2. 다음 태스크 중 하나를 수행합니다.

- 처음부터 새 옵션 그룹을 만듭니다. 자세한 내용은 옵션 그룹 생성 단원을 참조하십시오.
- AWS CLI 또는 API를 사용하여 기존 옵션 그룹을 복사합니다. 자세한 내용은 옵션 그룹 생성 단원을 참조하십시오.
- 기본값이 아닌 기존 옵션 그룹을 재사용합니다. 가장 좋은 방법은 현재 DB 인스턴스나 스냅샷과 연결되어 있지 않은 옵션 그룹을 사용하는 것입니다.

3. 이전 단계의 옵션 그룹에 새 옵션을 추가합니다.
4. 현재 DB 인스턴스와 연결된 옵션 그룹에 활성화된 옵션이 있는 경우 이러한 옵션을 새 옵션 그룹에 추가합니다. 이 전략은 새 옵션을 활성화하는 동안 기존 옵션이 제거되는 것을 방지합니다.
5. 새 옵션 그룹을 DB 인스턴스에 추가합니다.

콘솔

TDE 옵션을 옵션 그룹에 추가하고 DB 인스턴스와 연결하는 방법

1. RDS 콘솔에서 옵션 그룹을 선택합니다.
2. 옵션을 추가할 옵션 그룹의 이름을 선택합니다.
3. 옵션 추가를 선택합니다.
4. 옵션 이름에서 TDE를 선택한 다음 옵션 설정을 구성합니다.
5. 옵션 추가를 선택합니다.

> **Important**
>
> 하나 이상의 DB 인스턴스에 현재 연결되어 있는 옵션 그룹에 TDE 옵션을 추가하면 모든 DB 인스턴스가 자동으로 다시 시작되는 동안 인스턴스가 잠시 중단됩니다.

옵션 추가에 대한 자세한 내용은 옵션 그룹에 옵션 추가 섹션을 참조하세요.

6. 옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스와 연결합니다.
   - 새 DB 인스턴스의 경우, 인스턴스를 시작할 때 옵션 그룹을 적용하십시오. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
   - 기존 DB 인스턴스의 경우, 해당 인스턴스를 수정하고 새 옵션 그룹을 연결하여 옵션 그룹을 적용하십시오. DB 인스턴스는 이 작업의 일부로 다시 시작되지 않습니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

AWS CLI

다음 예제에서는 AWS CLI add-option-to-option-group 명령을 사용하여 TDE 옵션을 myoptiongroup이라는 옵션 그룹에 추가합니다. 자세한 내용은 시작하기: Flink 1.13.2를 참조하세요.

대상 LinuxmacOS, 또는Unix:

```
aws rds add-option-to-option-group \
    --option-group-name "myoptiongroup" \
    --options "OptionName=TDE" \
    --apply-immediately
```

Windows의 경우:

```
aws rds add-option-to-option-group ^
    --option-group-name "myoptiongroup" ^
    --options "OptionName=TDE" ^
    --apply-immediately
```

TDE 옵션이 포함되지 않은 DB 인스턴스로 데이터 복사

TDE 옵션을 DB 인스턴스에서 제거하거나 TDE 옵션을 포함하지 않는 옵션 그룹과 연결할 수 없습니다. TDE 옵션이 포함되지 않은 인스턴스로 데이터를 마이그레이션하려면 다음 작업을 수행하세요.

1. DB 인스턴스의 데이터를 복호화합니다.
2. TDE가 활성화된 옵션 그룹과 연결되지 않은 새 DB 인스턴스로 데이터를 복사합니다.
3. 원본 DB 인스턴스를 삭제합니다.

새 인스턴스에는 이전 DB 인스턴스와 동일한 이름을 사용할 수 있습니다.

Oracle Data Pump와 함께 TDE 사용 시 고려 사항

Oracle Data Pump를 사용하여 암호화된 덤프 파일을 가져오거나 내보낼 수 있습니다. Amazon RDS는 Oracle Data Pump에 대한 암호 암호화 모드 (ENCRYPTION_MODE=PASSWORD)를 지원합니다. Amazon RDS는 Oracle Data Pump에 대해 투명 암호화 모드 (ENCRYPTION_MODE=TRANSPARENT)를 지원하지 않습니다. 자세한 내용은 Oracle Data Pump를 사용한 가져오기 단원을 참조하십시오.

Oracle UTL_MAIL

Amazon RDS는 UTL_MAIL 옵션 및 SMTP 서버 사용을 통해 Oracle UTL_MAIL을 지원합니다. UTL_MAIL 패키지를 사용하여 데이터베이스에서 직접 이메일을 전송할 수 있습니다. Amazon RDS는 다음 Oracle 버전에 대해 UTL_MAIL을 지원합니다.

- Oracle Database 21c(21.0.0.0), 모든 버전
- Oracle Database 19c(19.0.0.0), 모든 버전

다음은 UTL_MAIL을 사용할 때 적용되는 몇 가지 제한 사항입니다.

- UTL_MAIL에서는 TLS(전송 계층 보안)를 지원하지 않으므로 이메일이 암호화되지 않습니다.

  사용자 지정 Oracle wallet을 생성하고 업로드하여 원격 SSL/TLS 리소스에 안전하게 연결하려면 인증서 및 Oracle Wallet을 사용하여 UTL_HTTP 액세스 구성의 지침을 따르십시오.

  wallet에 필요한 특정 인증서는 서비스별로 다릅니다. AWS 서비스의 경우 일반적으로 Amazon Trust Services 리포지토리에서 이 정보를 확인할 수 있습니다.
- UTL_MAIL은 SMTP 서버를 통한 인증을 지원하지 않습니다.
- 단일 첨부만 이메일로 보낼 수 있습니다.
- 32K를 초과하는 첨부를 보낼 수 없습니다.
- ASCII 및 EBCDIC(Extended Binary Coded Decimal Interchange Code) 문자 인코딩만 사용할 수 있습니다.
- SMTP 포트(25)는 탄력적 네트워크 인터페이스 소유자의 정책에 따라 조절됩니다.

UTL_MAIL을 활성화하는 경우 DB 인스턴스의 마스터 사용자에게만 실행 권한이 부여됩니다. 필요한 경우, 마스터 사용자는 다른 사용자에게 UTL_MAIL을 사용할 수 있는 실행 권한을 부여할 수 있습니다.

> **Important**
>
> UTL_MAIL 절차 사용을 추적하려면 Oracle의 기본 제공 감사 기능을 사용하는 것이 좋습니다.

Oracle UTL_MAIL 필수 선행 조건

다음은 Oracle UTL_MAIL 사용을 위한 필수 선행 조건입니다.

- 하나 이상의 SMTP 서버와 해당 IP 주소 또는 퍼블릭 또는 프라이빗 DNS(Domain Name Server) 이름. 사용자 지정 DNS 서버를 통해 확인되는 프라이빗 DNS 이름에 대한 자세한 내용은 사용자 지정 DNS 서버 설정 단원을 참조하십시오.

Oracle UTL_MAIL 옵션 추가

Oracle UTL_MAIL 옵션을 DB 인스턴스에 추가하는 일반적인 프로세스는 다음과 같습니다.

1. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 복사 또는 수정합니다.
2. [] 옵션을 옵션 그룹에 추가합니다.
3. 옵션 그룹을 DB 인스턴스에 연동시킵니다.

UTL_MAIL 옵션을 추가하면 옵션 그룹이 활성화되고 UTL_MAIL이 활성화됩니다.

DB 인스턴스에 UTL_MAIL 옵션을 추가하려면

1. 사용할 옵션 그룹을 결정합니다. 새 옵션 그룹을 생성하거나 기존 옵션 그룹을 사용합니다. 기존 옵션 그룹을 사용하려면 다음 단계로 건너뛰십시오. 그렇지 않으면 다음 설정을 사용하여 사용자 지정 DB 옵션을 생성합니다.

   a. [Engine]에서 사용할 Oracle 버전을 선택합니다.
   b. 메이저 엔진 버전에서 DB 인스턴스의 버전을 선택합니다.

   자세한 내용은 옵션 그룹 생성 섹션을 참조하세요.
2. [UTL_MAIL] 옵션을 옵션 그룹에 추가합니다. 옵션 추가에 대한 자세한 내용은 옵션 그룹에 옵션 추가 섹션을 참조하세요.
3. 옵션 그룹을 새 DB 인스턴스 또는 기존 DB 인스턴스에 적용합니다:

   - 새 DB 인스턴스의 경우, 인스턴스를 시작할 때 옵션 그룹을 적용합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
   - 기존 DB 인스턴스의 경우, 해당 인스턴스를 수정하고 새 옵션 그룹을 연결하여 옵션 그룹을 적용합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

Oracle UTL_MAIL 사용

UTL_MAIL 옵션을 활성화한 후 사용하기 전에 SMTP 서버를 구성해야 합니다.

SMTP_OUT_SERVER 파라미터를 유효한 IP 주소 또는 퍼블릭 DNS 이름으로 설정하여 SMTP 서버를 구성합니다. SMTP_OUT_SERVER 파라미터에 대해 여러 서버 주소를 쉼표로 구분된 목록으로 지정할 수 있습니다. 첫 번째 서버를 사용할 수 없는 경우 UTL_MAIL에서는 다음 서버를 순서대로 시도합니다.

DB 파라미터 그룹을 사용하여 DB 인스턴스에 대한 기본 SMTP_OUT_SERVER를 설정할 수 있습니다. DB 인스턴스에서 데이터베이스에 대해 다음 코드를 실행하여 세션에 대한 SMTP_OUT_SERVER 파라미터를 설정할 수 있습니다.

```
ALTER SESSION SET smtp_out_server = mailserver.domain.com:25;
```

UTL_MAIL 옵션을 활성화하고 SMTP_OUT_SERVER를 구성한 후 SEND 절차를 사용하여 메일을 보낼 수 있습니다. 자세한 내용은 Oracle 설명서의 UTL_MAIL을 참조하십시오.

Oracle UTL_MAIL 옵션 제거

DB 인스턴스에서 Oracle UTL_MAIL을 제거할 수 있습니다.

DB 인스턴스에서 UTL_MAIL을 제거하려면 다음 중 하나를 수행합니다.

- 여러 DB 인스턴스에서 UTL_MAIL을 제거하려면 UTL_MAIL이 속한 옵션 그룹에서 해당 UTL_MAIL 옵션을 제거합니다. 이 변경은 해당 옵션 그룹을 사용하는 모든 DB 인스턴스에 영향을 미칩니다. 자세한 내용은 옵션 그룹에서 옵션 제거 섹션을 참조하세요.
- 단일 DB 인스턴스에서 UTL_MAIL을 제거하려면 DB 인스턴스를 수정하고 UTL_MAIL 옵션이 포함되지 않은 다른 옵션 그룹을 지정합니다. 기본(빈) 옵션 그룹을 지정하거나 다른 사용자 지정 옵션 그룹을 지정할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

문제 해결

다음은 Amazon RDS에서 UTL_MAIL을 사용할 때 발생할 수 있는 문제입니다.

- Throttling. SMTP 포트(25)는 탄력적 네트워크 인터페이스 소유자의 정책에 따라 조절됩니다. UTL_MAIL을 사용하여 이메일을 발송할 수 있지만 오류 `ORA-29278: SMTP transient error: 421 Service not available`이 표시될 경우 포트가 조절되는 것일 수 있습니다. 이메일 발송 시 조절을 경험할 경우 백오프 알고리즘을 구현하는 것이 좋습니다. 백오프 알고리즘에 대한 자세한 내용은 AWS의 오류 재시도 횟수 및 지수 백오프 섹션과 '제한 - 최대 송신률 초과' 오류를 처리하는 방법 섹션을 참조하세요.

이러한 조절은 제거를 요청할 수 있습니다. 자세한 내용은 내 EC2 인스턴스에서 포트 25에 대한 조절을 제거하려면 어떻게 해야 하나요? 단원을 참조하세요.

Oracle XML DB

Oracle XML DB는 DB 인스턴스에 기본 XML 지원을 추가합니다. XML DB를 사용하면 관계형 데이터베이스뿐 아니라 정형 또는 비정형 XML을 저장 및 검색할 수 있습니다. XML DB 프로토콜 서버는 RDS for Oracle에서 지원되지 않습니다.

XML DB는 Oracle 데이터베이스 12c 이상에 사전 설치되어 있습니다. 따라서 추가 특성으로 XML DB를 명시적으로 설치하기 위해 옵션 그룹을 사용할 필요가 없습니다.

XML DB를 구성 및 사용하는 방법을 알아보려면 Oracle 데이터베이스 설명서의 Oracle XML DB 개발자 안내서를 참조하세요.

RDS for Oracle DB 엔진 업그레이드

Amazon RDS에서 새 Oracle Database 버전을 지원하는 경우, DB 인스턴스를 새 버전으로 업그레이드할 수 있습니다. Amazon RDS에서 사용할 수 있는 Oracle 버전에 대한 자세한 내용은 Amazon RDS for Oracle 릴리스 정보를 참조하세요.

> **Important**
>
> RDS for Oracle Database 11g, 12c 및 18c는 더 이상 지원되지 않습니다. Oracle Database 11g, 12c 또는 18c 스냅샷을 유지하는 경우 이후 릴리스로 업그레이드하세요. 자세한 내용은 Oracle DB 스냅샷 업그레이드 단원을 참조하십시오.

주제

- RDS for Oracle 엔진 업그레이드 개요
- Oracle 메이저 버전 업그레이드
- Oracle 마이너 버전 업그레이드
- Oracle 데이터베이스 업그레이드에 대한 고려 사항
- Oracle DB 업그레이드 테스트
- RDS for Oracle DB 인스턴스 버전 업그레이드
- Oracle DB 스냅샷 업그레이드

RDS for Oracle 엔진 업그레이드 개요

RDS for Oracle DB 인스턴스를 업그레이드하기 전에 다음 개념을 숙지하세요.

주제

- 메이저 및 마이너 버전 업그레이드
- RDS for Oracle 지원 날짜 및 필수 업그레이드
- Oracle 엔진 버전 관리
- 엔진 업그레이드 중 자동 스냅샷
- 다중 AZ 배포에서 Oracle 업그레이드
- 읽기 전용 복제본의 Oracle 업그레이드

메이저 및 마이너 버전 업그레이드

메이저 버전은 1~2년마다 출시되는 Oracle Database의 메이저 릴리스입니다. Oracle Database 19c와 Oracle Database 21c는 메이저 릴리스입니다.

분기마다 RDS for Oracle은 각 지원되는 메이저 엔진에 대해 새로운 마이너 엔진 버전을 릴리스합니다. 릴리스 업데이트(RU) 엔진 버전은 지정된 분기에 대한 RU 패치를 포함하여 Oracle의 버그 수정을 통합합니다. 예를 들어 21.0.0.0.ru-2024-10.rur-2024-10.r1은 2024년 10월 RU를 통합한 Oracle Database 21c의 마이너 버전입니다.

공간 패치 번들(SPB) 엔진 버전에는 RU 패치와 Oracle Spatial과 관련된 패치가 포함되어 있습니다. 예를 들어 19.0.0.0.ru-2025-01.spb-1.r1은 엔진 버전 19.0.0.0.ru-2025-01.rur-2025-01.r1에 RU 패치와 공간 패치가 포함된 마이너 엔진 버전입니다. 일반적으로 RDS for Oracle은 해당 RU가 있고서 2~3주 후에 SPB를 릴리스합니다. RU와 SPB의 차이점에 대한 설명은 릴리스 업데이트(RU) 및 공간 패치 번들(SPB) 섹션을 참조하시기 바랍니다. 자세한 내용은 Release notes for Amazon Relational Database Service (Amazon RDS) for Oracle을 참조하시기 바랍니다.

RDS for Oracle은 DB 인스턴스에 대해 다음과 같은 업그레이드를 지원합니다.

업그레이드 유형	애플리케이션 호환성	업그레이드 메서드	샘플 업그레이드 경로
메이저 버전	메이저 버전 업그레이드로 기존 애플리케이션과 호환되지 않는 변경 사항이 도입될 수 있습니다.	수동 전용	Oracle Database 19c에서 Oracle Database 21c로
마이너 버전	마이너 버전 업그레이드에는 기존 애플리케이션과 호환되는 변경 사항만 포함됩니다.	자동 또는 수동	21.0.0.0.ru-2023-07.rur-2022-07.r1에서 21.0.0.0.ru-2023-10.rur-2022-10.r1로

> **Important**
>
> DB 엔진을 업데이트할 때 운영 중단이 발생합니다. 운영 중단 지속 시간은 엔진 버전 및 DB 인스턴스 크기에 따라 다릅니다.

프로덕션 데이터베이스에 업그레이드를 적용하기 전에 철저하게 테스트하여 애플리케이션이 올바르게 작동하는지 확인해야 합니다. 자세한 내용은 Oracle DB 업그레이드 테스트 섹션을 참조하세요.

RDS for Oracle 지원 날짜 및 필수 업그레이드

RDS for Oracle의 데이터베이스 버전에는 예상 지원 날짜가 있습니다. RDS for Oracle DB 엔진의 메이저 또는 마이너 버전이 지원 종료에 가까워지면 RDS는 강제 업그레이드라고도 하는 필수 업그레이드를 시작합니다. RDS는 다음 정보를 게시합니다.

- 더 이상 사용되지 않는 버전의 인스턴스를 지원되는 버전으로 수동으로 업그레이드하기 시작하라는 권장 사항
- 지원되지 않는 버전에서 더 이상 인스턴스를 만들 수 없는 날짜
- 유지 관리 기간 동안 RDS가 인스턴스를 지원되는 버전으로 자동 업그레이드하기 시작하는 날짜
- 유지 관리 기간이 아닐 때 RDS가 인스턴스를 지원되는 버전으로 자동 업그레이드하기 시작하는 날짜

Important

강제 업그레이드는 AWS CloudFormation 스택에 예기치 않은 결과를 초래할 수 있습니다. RDS를 사용하여 DB 인스턴스를 자동으로 업그레이드하는 경우 AWS CloudFormation에 문제가 발생할 수 있습니다.

이 섹션은 다음 주제를 포함합니다:

주제

- RDS for Oracle 메이저 릴리스에 대한 지원 날짜
- RDS for Oracle 마이너 버전에 대한 지원 날짜

RDS for Oracle 메이저 릴리스에 대한 지원 날짜

Oracle에 대한 RDS 메이저 버전은 해당 Oracle 데이터베이스 릴리스 버전에 대한 지원 수명이 끝날 때까지 사용할 수 있습니다. 다음 날짜를 사용하여 테스트 및 업그레이드 주기를 계획할 수 있습니다. 이 날짜는 최신 버전 업그레이드가 필요할 수 있는 가장 빠른 날짜를 나타냅니다. Amazon이 RDS for

Oracle 버전에 대한 지원을 원래 명시일보다 오래 연장할 경우, 이 표를 이후 날짜를 반영하도록 업데이트할 계획입니다.

> **Note**
>
> describe-db-major-engine-versions AWS CLI 명령을 실행하거나 DescribeDBMajorEngineVersions RDS API 작업을 사용하여 Oracle 데이터베이스의 메이저 버전을 볼 수 있습니다.

Oracle 데이터베이스 메이저 릴리스 버전	최신 버전으로 업그레이드할 예정일
Oracle Database 19c	2029년 12월 31일(BYOL 프리미어 지원 포함(추가 지원 수수료 면제)) 2032년 12월 31일(BYOL 추가 지원(추가 비용) 또는 무제한 라이선스 계약 포함) 2029년 12월 31일(라이선스 포함(LI))
Oracle Database 21c	2027년 7월 31일(추가 지원에는 사용할 수 없음)

RDS는 새 메이저 버전으로 업그레이드해야 하기 최소 12개월 전에 사용자에게 알립니다. 알림을 통해 중요한 일정의 타이밍, 사용자의 DB 인스턴스에 미치는 영향, 권장 조치 등 업그레이드 프로세스를 알려줍니다. 메이저 버전으로 업그레이드를 수행하기 전에 새 RDS for Oracle 버전으로 애플리케이션을 철저히 테스트하는 것이 좋습니다.

이 사전 알림 기간이 지나면 후속 주요 버전으로의 자동 업그레이드가 여전히 이전 버전을 실행 중인 모든 RDS for Oracle DB 인스턴스에 적용될 수 있습니다. 이 경우 예약된 유지 관리 기간에 업그레이드가 시작됩니다.

자세한 내용은 My Oracle Support의 현재 데이터베이스 릴리스 릴리스 일정을 참조하세요.

RDS for Oracle 마이너 버전에 대한 지원 날짜

Oracle Database 19c의 다음 마이너 버전은 지원 종료가 다가오고 있습니다.

- 19.0.0.0.ru-2019-07.rur-2019-07.r1

- 19.0.0.0.ru-2019-10.rur-2019-10.r1
- 19.0.0.0.ru-2020-01.rur-2020-01.r1

Oracle Database 21c의 다음 마이너 버전은 지원 종료가 다가오고 있습니다.

- 21.0.0.0.ru-2022-01.rur-2022-01.r1
- 21.0.0.0.ru-2022-04.rur-2022-04.r1
- 21.0.0.0.ru-2022-07.rur-2022-07.r1
- 21.0.0.0.ru-2022-10.rur-2022-10.r1
- 21.0.0.0.ru-2023-01.rur-2023-01.r1
- 21.0.0.0.ru-2023-01.rur-2023-01.r2

다음 표에는 이전의 더 이상 사용되지 않는 버전에 대한 필수 마이너 버전 업그레이드 일정이 나와 있습니다.

날짜	더 이상 사용되지 않는 버전에 대한 작업
2025년 7월 2일	이제 더 이상 사용되지 않는 버전을 실행하는 DB 인스턴스를 만들 수 없습니다. DB 인스턴스를 수동으로 최신 릴리스 업데이트(RU)로 업그레이드하거나 RDS가 자동으로 업그레이드할 때까지 기다립니다.
2025년 8월 1일 ~2025년 9월 15일	RDS는 예약된 유지 관리 기간의 DB 인스턴스를 최신 RU로 강제 업그레이드합니다. 또한 RDS는 더 이상 사용되지 않는 버전의 스냅샷에서 복원된 모든 인스턴스를 업그레이드합니다.
2025년 9월 15일	RDS는 예약된 유지 관리 기간에 관계없이 DB 인스턴스를 최신 RU로 강제 업그레이드하기 시작합니다. 모든 인스턴스가 지원되는 마이너 버전을 실행할 때까지 강제 업그레이드가 진행됩니다.

RDS가 더 이상 사용되지 않는 버전을 실행하는 인스턴스를 자동으로 업그레이드하지 않도록 하려면 다음을 수행합니다.

- Oracle Database 19c 및 21c 마이너 버전을 수동으로 최신 RU 또는 필수 업그레이드가 예약되지 않은 모든 RU로 업그레이드합니다. 예를 들어 21.0.0.0.ru-2023-01.rur-2023-01.r2를

21.0.0.0.ru-2023-04.rur-2023-04.r2로 업그레이드하거나 19.0.0.0.ru-2019-07.rur-2019-07.r1을 19.0.0.0.ru-2025-01.rur-2025-01.r2로 업그레이드할 수 있습니다.

- 더 이상 사용되지 않는 마이너 버전을 사용하는 DB 스냅샷을 업그레이드한 다음 스냅샷을 복원합니다.

마이너 버전에 대한 자세한 내용은 Release notes for Amazon Relational Database Service (Amazon RDS) for Oracle을 참조하세요.

Oracle 엔진 버전 관리

DB 엔진 버전 관리를 통해 데이터베이스 엔진의 패치 및 업그레이드 시기와 방법을 제어할 수 있습니다. 데이터베이스 엔진 패치 버전과의 호환성을 유연하게 유지할 수 있습니다. 또한 RDS for Oracle의 새 패치 버전을 테스트하여 프로덕션에 배포하기 전에 애플리케이션과 작동하는지 확인할 수 있습니다. 또한 자신의 조건과 일정에 따라 버전을 업그레이드합니다.

> **Note**
>
> Amazon RDS는 Amazon RDS 관련 DB 엔진 버전을 사용하여 공식 Oracle 데이터베이스 패치를 정기적으로 수집합니다. Amazon RDS Oracle 관련 엔진 버전에 포함되는 Oracle 패치의 목록은 Amazon RDS for Oracle 릴리스 정보를 참조하세요.

엔진 업그레이드 중 자동 스냅샷

Oracle DB 인스턴스를 업그레이드하는 동안 스냅샷은 업그레이드 문제로부터 인스턴스를 보호합니다. DB 인스턴스의 백업 보존 기간이 0보다 크면 Amazon RDS는 업그레이드 중에 다음과 같은 DB 스냅샷을 생성합니다.

1. 업그레이드 변경이 수행되기 전 DB 인스턴스의 스냅샷입니다. 업그레이드가 실패하면 이 스냅샷을 복원하여 이전 버전을 실행하는 DB 인스턴스를 생성할 수 있습니다.
2. 업그레이드가 완료된 후 DB 인스턴스의 스냅샷입니다.

> **Note**
>
> 백업 보존 기간을 변경하려면 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

업그레이드 후에는 이전 엔진 버전으로 되돌릴 수 없습니다. 하지만 업그레이드 전 스냅샷을 복원하여 새 Oracle DB 인스턴스를 생성할 수 있습니다.

다중 AZ 배포에서 Oracle 업그레이드

DB 인스턴스가 다중 AZ 배포에 있는 경우 Amazon RDS는 기본 복제본과 대기 복제본을 모두 업그레이드합니다. 운영 체제 업데이트가 필요하지 않은 경우 기본 업그레이드와 대기 업그레이드가 동시에 수행됩니다. 업그레이드가 완료될 때까지 인스턴스를 사용할 수 없습니다.

다중 AZ 배포에서 운영 체제 업데이트가 필요한 경우 데이터베이스 업그레이드를 요청할 때 Amazon RDS가 업데이트를 적용합니다. Amazon RDS는 다음 단계를 수행합니다.

1. 현재 대기 DB 인스턴스의 운영 체제를 업데이트합니다.
2. 기본 DB 인스턴스를 대기 DB 인스턴스로 장애 조치합니다.
3. 이전에 대기 인스턴스였던 새 기본 DB 인스턴스의 데이터베이스 버전을 업그레이드합니다. 업그레이드 중에는 기본 데이터베이스를 사용할 수 없습니다.
4. 이전에 기본 DB 인스턴스였던 새로운 대기 DB 인스턴스에서 운영 체제를 업데이트합니다.
5. 새로운 대기 DB 인스턴스에서 데이터베이스 버전을 업그레이드합니다.
6. 새 기본 DB 인스턴스를 원래 기본 DB 인스턴스로 장애 조치하고 새 대기 DB 인스턴스를 원래 대기 DB 인스턴스로 다시 장애 조치합니다. 따라서 Amazon RDS는 복제 구성을 원래 상태로 되돌립니다.

읽기 전용 복제본의 Oracle 업그레이드

원본 DB 인스턴스의 Oracle DB 엔진 버전과 모든 읽기 전용 복제본은 동일해야 합니다. Amazon RDS는 다음과 같은 단계로 업그레이드를 수행합니다.

1. 원본 DB 인스턴스를 업그레이드합니다. 이 단계 중에 읽기 전용 복제본을 사용할 수 있습니다.
2. 복제본 유지 관리 기간과 상관없이 읽기 전용 복제본을 병렬로 업그레이드합니다. 이 단계 중에 원본 DB를 사용할 수 있습니다.

리전 간 읽기 전용 복제본의 메이저 버전 업그레이드의 경우 Amazon RDS는 다음과 같은 추가 작업을 수행합니다.

- 대상 버전에 대한 옵션 그룹을 자동으로 생성합니다.
- 모든 옵션과 옵션 설정을 원래 옵션 그룹에서 새 옵션 그룹으로 복사합니다.

- 업그레이드된 리전 간 읽기 전용 복제본을 새 옵션 그룹과 연결합니다.

Oracle 메이저 버전 업그레이드

메이저 버전 업그레이드를 수행하려면 DB 인스턴스를 수동으로 수정합니다. 메이저 버전 업그레이드는 자동으로 수행되지 않습니다.

> **Important**
>
> 프로덕션 데이터베이스에 업그레이드를 적용하기 전에 철저하게 테스트하여 애플리케이션이 올바르게 작동하는지 확인해야 합니다. 자세한 내용은 Oracle DB 업그레이드 테스트 단원을 참조하십시오.

주제

- 메이저 업그레이드에 지원되는 버전
- 메이저 업그레이드에 지원되는 인스턴스 클래스
- 메이저 업그레이드 전에 통계 수집
- 메이저 업그레이드 허용

메이저 업그레이드에 지원되는 버전

Amazon RDS는 다음과 같은 메이저 버전 업그레이드를 지원합니다.

현재 버전	지원하는 업그레이드
CDB 아키텍처 사용 19.0.0.0	21.0.0.0

Oracle Database의 메이저 버전 업그레이드는 같은 달 또는 그 이후에 릴리스된 RU(릴리스 업데이트)로 업그레이드해야 합니다. Oracle Database 버전에 메이저 버전 다운그레이드는 지원되지 않습니다.

메이저 업그레이드에 지원되는 인스턴스 클래스

현재 Oracle DB 인스턴스가 업그레이드하려는 버전에서 지원되지 않는 DB 인스턴스 클래스에서 실행될 수 있습니다. 이 경우 업그레이드하기 전에 DB 인스턴스를 지원되는 DB 인스턴스 클래스로 마이그

레이션합니다. 각 Amazon RDS for Oracle 버전 및 에디션에서 지원되는 DB 인스턴스 클래스에 대한 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요.

메이저 업그레이드 전에 통계 수집

메이저 버전 업그레이드를 수행하기 전에 업그레이드하려는 DB 인스턴스에서 최적화 프로그램 통계를 수집하는 것이 좋습니다. 이렇게 하면 업그레이드 중 DB 인스턴스 가동 중지 시간을 줄일 수 있습니다.

최적화 프로그램 통계를 수집하려면 다음 예제와 같이 마스터 사용자로 DB 인스턴스에 연결하고, `DBMS_STATS.GATHER_DICTIONARY_STATS` 프로시저를 실행합니다.

```
EXEC DBMS_STATS.GATHER_DICTIONARY_STATS;
```

자세한 내용은 Oracle 설명서의 GATHER_DICTIONARY_STATS 프로시저를 참조하세요.

메이저 업그레이드 허용

메이저 엔진 버전 업그레이드가 애플리케이션과 호환되지 않을 수 있습니다. 업그레이드는 되돌릴 수 없습니다. EngineVersion 파라미터에 현재 메이저 버전과 다른 메이저 버전을 지정하는 경우 메이저 버전 업그레이드를 허용해야 합니다.

CLI 명령 modify-db-instance를 사용하여 메이저 버전을 업그레이드하는 경우 `--allow-major-version-upgrade`를 지정합니다. 이 설정은 영구적이지 않으므로 메이저 업그레이드를 수행할 때마다 `--allow-major-version-upgrade`를 지정해야 합니다. 이 파라미터는 마이너 엔진 버전의 업그레이드에 영향을 주지 않습니다. 자세한 내용은 DB 인스턴스 엔진 버전 업그레이드 단원을 참조하십시오.

콘솔을 사용하여 메이저 버전을 업그레이드하는 경우 업그레이드를 허용하는 옵션을 선택할 필요가 없습니다. 대신 콘솔에 메이저 업그레이드를 되돌릴 수 없다는 경고가 표시됩니다.

Oracle 마이너 버전 업그레이드

RDS for Oracle에서 마이너 버전 업그레이드는 메이저 DB 엔진 버전에 대한 업데이트를 의미합니다. RDS에서 마이너 엔진 버전은 릴리스 업데이트(RU) 또는 공간 패치 번들(SPB)입니다. 예를 들어 DB 인스턴스에서 메이저 버전 Oracle Database 21c와 마이너 버전 21.0.0.0.ru-2024-10.rur-2024-10.r1을 실행하는 경우 마이너 버전 21.0.0.0.ru-2025-01.rur-2025-01.r1로 업그레이드할 수 있습니다. RDS for Oracle은 마이너 버전 다운그레이드를 지원하지 않습니다.

수동 또는 자동으로 DB 엔진을 마이너 버전으로 업그레이드할 수 있습니다. 수동 업그레이드하는 방법에 대한 자세한 내용은 엔진 버전 수동 업그레이드 섹션을 참조하세요. 자동 업그레이드를 구성하는 방법에 대한 자세한 내용은 마이너 엔진 버전 자동 업그레이드 섹션을 참조하세요. 수동 업그레이드인지, 자동 업그레이드인지와 관계없이 마이너 버전 업그레이드에는 자동 중지 시간이 수반됩니다. 업그레이드를 계획할 때 이 가동 중지 시간을 고려합니다.

> **Important**
>
> 프로덕션 데이터베이스에 업그레이드를 적용하기 전에 철저하게 테스트하여 애플리케이션이 올바르게 작동하는지 확인해야 합니다. 자세한 내용은 Oracle DB 업그레이드 테스트 섹션을 참조하세요.

주제

- 릴리스 업데이트(RU) 및 공간 패치 번들(SPB)
- Oracle용 마이너 버전 자동 업그레이드 활성화
- RDS for Oracle의 자동 마이너 버전 업그레이드에 대한 알림
- RDS가 RDS for Oracle용 자동 마이너 버전 업그레이드 일정을 예약할 때
- RDS for Oracle용 자동 마이너 버전 업그레이드 관리

릴리스 업데이트(RU) 및 공간 패치 번들(SPB)

RDS에서 릴리스 업데이트(RU)는 Oracle Database에 대한 보안 수정, 버그 수정 및 새로운 기능을 포함하는 분기별 마이너 엔진 버전입니다. 공간 패치 번들(SPB)은 Oracle Spatial 옵션용으로 설계된 패치가 포함된 RU 엔진 버전입니다. 예를 들어 19.0.0.0.ru-2025-01.spb-1.r1이라는 SPB에는 해당 RU 19.0.0.0.ru-2025-01.rur-2025-01.r1에 있는 모든 패치와 공간에 고유한 패치가 포함됩니다. SPB는 Oracle Database 19c에서만 지원됩니다.

인스턴스가 자동 마이너 버전 업그레이드로 구성된 경우 RU와 SPB는 별도의 업그레이드 경로에 존재합니다. 일반적으로 SPB는 해당 RU가 있고서 2~3주 후에 릴리스됩니다. 다음 표에는 Oracle Database 19c용 샘플 마이너 버전이 나와 있습니다.

표준 RU 업그레이드 경로	SPB 업그레이드 경로
19.0.0.0.ru-2025-01.rur-2025-01.r1	19.0.0.0.ru-2025-01.spb-1.r1
19.0.0.0.ru-2025-04.rur-2025-04.r1	19.0.0.0.ru-2025-04.spb-1.r1

표준 RU 업그레이드 경로	SPB 업그레이드 경로
19.0.0.0.ru-2025-07.rur-2025-07.r1	19.0.0.0.ru-2025-07.spb-1.r1
19.0.0.0.ru-2025-10.rur-2025-10.r1	19.0.0.0.ru-2025-10.spb-1.r1

DB 인스턴스가 자동 업그레이드용으로 구성된 경우 인스턴스는 현재 버전에 해당하는 업그레이드 경로에 존재합니다. 예를 들어 DB 인스턴스가 버전 19.0.0.0.ru-2025-01.rur-2025-01.r1을 실행하는 경우 19.0.0.0.ru-2025-04.rur-2025-04.r1이 릴리스되면 인스턴스가 이 RU로 자동으로 업그레이드됩니다. 마찬가지로 DB 인스턴스가 19.0.0.0.ru-2025-01.spb-1.r1을 실행하는 경우 19.0.0.0.ru-2025-04.spb-1.r1이 릴리스되면 인스턴스가 이 SPB로 자동으로 업그레이드됩니다. RU인 19.0.0.0.ru-2025-01.rur-2025-01.r1을 실행하는 인스턴스는 별도의 업그레이드 경로에 있는 SPB인 19.0.0.0.ru-2025-04.spb-1.r1로 자동으로 업그레이드되지 않습니다.

인스턴스가 공간을 사용하지 않더라도 DB 인스턴스를 SPB로 업그레이드할 수 있지만, 공간 패치는 Oracle Spatial에만 적용됩니다. RU에서 동일한 엔진 버전 이상의 SPB로 수동으로 업그레이드할 수 있습니다. 예를 들어 19.0.0.0.ru-2025-01.rur-2025-01.r1에서 다음 엔진 버전 중 하나로 인스턴스를 업그레이드할 수 있습니다.

- 19.0.0.0.ru-2025-01.spb-1.r1
- 19.0.0.0.ru-2025-04.spb-1.r1

RU가 더 높은 엔진 버전인 경우에만 SPB에서 RU로 인스턴스를 업그레이드할 수 있습니다. 예를 들어 SPB 버전 19.0.0.0.ru-2025-04.spb-1.r1에서 상위 RU 버전 19.0.0.0.ru-2025-07.rur-2025-07.r1로 업그레이드할 수 있지만 동일한 RU 버전 19.0.0.0.ru-2025-04.rur-2025-04.r1로 업그레이드할 수 없습니다.

DB 인스턴스가 자동 마이너 버전 업그레이드용으로 구성되어 있고 RU에서 SPB로 또는 SPB에서 RU로 수동으로 업그레이드하는 경우 자동 업그레이드 경로가 변경됩니다. RU 버전 19.0.0.0.ru-2025-01.rur-2025-01.r1에서 SPB 버전 19.0.0.0.ru-2025-01.spb-1.r1로 수동으로 업그레이드한다고 가정해 보겠습니다. 다음 마이너 버전 자동 업그레이드는 SPB 버전 19.0.0.0.ru-2025-04.spb-1.r1입니다.

SPB가 RU처럼 작동하므로 인스턴스를 RU 및 SPB로 업그레이드하는 데 사용하는 RDS API는 동일합니다. 다음 명령은 RU 및 SPB로 업그레이드하는 것을 보여줍니다.

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --engine-version 19.0.0.0.ru-2025-01.rur-2025-01.r1
```

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --engine-version 19.0.0.0.ru-2025-01.spb-1.r1
```

Oracle Spatial 옵션에 대한 자세한 내용은 공간 패치 번들(SPB) 작동 방식 섹션을 참조하시기 바랍니다. Oracle Database 19c에 대해 지원되는 RU 및 SPB에 대해 알아보려면 Amazon RDS for Oracle Database 19c(19.0.0.0)를 참조하시기 바랍니다.

Oracle용 마이너 버전 자동 업그레이드 활성화

자동 마이너 버전 업그레이드에서 RDS는 수동 개입 없이 사용 가능한 최신 마이너 버전을 Oracle 데이터베이스에 적용합니다. Amazon RDS Oracle DB 인스턴스는 다음과 같은 상황일 때 다음번 유지 관리 기간 중에 업그레이드 일정을 예약합니다.

- DB 클러스터에 자동 마이너 버전 업그레이드 옵션이 활성화되어 있는 경우
- DB 인스턴스가 아직 최신 마이너 DB 엔진 버전을 실행하고 있지 않은 경우
- DB 인스턴스에 보류 중인 업그레이드가 예약되어 있지 않은 경우

자동 업그레이드를 활성화하는 방법에 대한 자세한 내용은 마이너 엔진 버전 자동 업그레이드 섹션을 참조하세요.

RDS for Oracle의 자동 마이너 버전 업그레이드에 대한 알림

RDS는 자동 업그레이드 일정을 예약하기 전에 사전 공지를 게시합니다. 데이터베이스 세부 정보 페이지의 유지 관리 및 백업 탭에서 알림을 찾을 수 있습니다. 메시지의 형식은 다음과 같습니다.

```
An automatic minor version upgrade to engine version will become available
 on availability-date and will be applied during a subsequent maintenance window.
```

사전 공지의 *availability-date*는 RDS가 AWS 리전에서 DB 인스턴스에 대한 업그레이드 예약을 시작하는 날짜입니다. 지금은 DB 인스턴스 업그레이드가 수행되도록 예정된 날짜가 아닙니다. 예를 들어 *availability-date*가 3월 1일인 경우 이 날짜에 RDS가 4월 14일로 업그레이드를 예약할 수 있습니다.

다음 예와 같이 AWS CLI의 `describe-pending-maintenance-actions` 명령을 사용하여 업그레이드 가능 날짜를 확인할 수도 있습니다.

```
aws rds describe-pending-maintenance-actions
```

```
{
    "PendingMaintenanceActions": [
        {
            "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:db:orclinst1",
            "PendingMaintenanceActionDetails": [
                {
                    "Action": "db-upgrade",
                    "Description": "Automatic minor version upgrade to
 21.0.0.0.ru-2024-07.rur-2024-07.r1",
                    "CurrentApplyDate": "2024-12-02T08:10:00Z",
                    "OptInStatus": "next-maintenance"
                }
            ]
        }, ...
```

다음 표에서는 보류 중인 유지 관리 작업 메시지의 각 유형에 따른 옵션을 설명합니다.

보류 중인 유지 관리 조치 메시지	메시지가 나타나는 시점	다음 유지 관리 기간에 적용할 수 있는지 여부	즉시 적용할 수 있는지 여부	옵트인을 취소할 수 있는지 여부
engine-version 으로의 자동 마이너 버전 업그레이드는 *availability-date* 에 제공될 예정이며, 이후 유지 관리 기간에 적용해야 합니다.	자동 업그레이드 일정이 잡히기 4~6주 전	예	예	예
마이너 버전을 *engine-version* 으로 자동 업그레이드	*availability-date* 당일 또는 이후 RDS는 DB 인스턴스의 다음 유지 관리 기간에 이 업그레이드를 자동으로 적용합니다.	예	예	아니요

describe-pending-maintenance-actions에 대한 자세한 내용은 AWS CLI 명령 참조를 참조하세요.

RDS가 RDS for Oracle용 자동 마이너 버전 업그레이드 일정을 예약할 때

자동 업그레이드 가능 날짜가 도래하면 RDS는 업그레이드 일정을 예약하기 시작합니다. 대부분의 AWS 리전에서는 RDS가 가용 날짜 4~6주 후에 최신 분기별 RU로 업그레이드하도록 업그레이드 일정을 예약합니다. 예정 날짜는 AWS 리전 및 기타 요인에 따라 달라집니다. RURs 및 RU에 대한 자세한 내용은 Amazon RDS for Oracle 릴리스 정보를 참조하세요.

RDS에서 업그레이드를 예약하면 데이터베이스 세부 정보 페이지의 유지 관리 및 백업 탭에 다음과 같은 알림이 표시됩니다.

```
Automatic minor version upgrade to engine-version
```

위 메시지는 RDS가 다음 유지 관리 기간에 DB 엔진을 업그레이드하도록 예약했음을 나타냅니다.

때로는 RDS가 이전 마이너 버전을 적용하기 전에 새 마이너 버전을 사용할 수 있게 됩니다. 예를 들어, *minor-version-2*와 *minor-version-3*을 업그레이드 대상으로 사용할 수 있을 때 인스턴스가 *minor-version-1*을 실행하고 있습니다. 이 경우 DB 인스턴스의 불필요한 가동 중지 시간을 방지하기 위해 RDS는 자동 마이너 버전 업그레이드를 최신 버전으로 예약하여 이전 버전으로의 업그레이드를 건너뜁니다. 이 예시에서 RDS는 인스턴스를 *minor-version-1*에서 *minor-version-3*으로 직접 업그레이드합니다.

마이너 버전 업그레이드가 특정 주기로 이루어지도록 하기 위해 자동 업그레이드 메커니즘을 사용하는 대신 인스턴스를 수동으로 업그레이드할 수 있습니다. 다음 유지 관리 기간의 업그레이드를 예약하려면 `modify-db-instance`를 사용하여 마이너 버전으로 업그레이드할 때 `--no-apply-immediately`를 지정합니다. 즉시 업그레이드하려면 대신 `--apply-immediately`를 지정합니다. 자세한 내용은 엔진 버전 수동 업그레이드 섹션을 참조하세요.

RDS for Oracle용 자동 마이너 버전 업그레이드 관리

새로운 마이너 버전이 출시되면 이 버전으로 DB 인스턴스를 수동 업그레이드할 수 있습니다. 다음 예시 업그레이드에서는 `orclinst1`이라는 DB 인스턴스를 즉시 업그레이드합니다.

```
aws rds apply-pending-maintenance-action \
    --resource-identifier arn:aws:rds:us-east-1:123456789012:db:orclinst1 \
    --apply-action db-upgrade \
    --opt-in-type immediate
```

아직 예약되지 않은 자동 마이너 버전 업그레이드를 옵트아웃하려면 다음 예시와 같이 `--opt-in-type`을 `undo-opt-in`으로 설정합니다.

```
aws rds apply-pending-maintenance-action \
    --resource-identifier arn:aws:rds:us-east-1:123456789012:db:orclinst1 \
    --apply-action db-upgrade \
    --opt-in-type undo-opt-in
```

RDS에서 이미 DB 인스턴스 업그레이드를 예약한 경우 `apply-pending-maintenance-action`을 사용하여 취소할 수 있습니다. 하지만 DB 인스턴스를 수정하고 자동 마이너 업그레이드 기능을 비활성화하면 업그레이드 일정이 취소됩니다.

자동 마이너 버전 업그레이드를 비활성화하는 방법에 대한 자세한 내용은 마이너 엔진 버전 자동 업그레이드 섹션을 참조하세요. apply-pending-maintenance-action에 대한 자세한 내용은 AWS CLI 명령 참조를 참조하세요.

Oracle 데이터베이스 업그레이드에 대한 고려 사항

Oracle 인스턴스를 업그레이드하기 전에 다음 정보를 검토합니다.

주제

- Oracle 멀티테넌트 고려 사항
- 옵션 그룹 고려 사항
- 파라미터 그룹 고려 사항
- 시간대 고려 사항
- 공간 패치 번들(SPB) 고려 사항

Oracle 멀티테넌트 고려 사항

다음 표에서는 다양한 릴리스에서 지원되는 Oracle Database 아키텍처에 대해 설명합니다.

Oracle Database 릴리스	RDS 지원 상태	아키텍처
Oracle Database 21c	지원	CDB만
Oracle Database 19c	지원	CDB 또는 비 CDB

다음 표에서는 지원되는 업그레이드 경로와 지원되지 않는 업그레이드 경로에 대해 설명합니다.

업그레이드 경로	지원?
CDB에서 CDB로	예
비 CDB에서 CDB로	아니요, 하지만 비CDB를 CDB로 변환한 다음 업그레이드할 수 있습니다.
CDB에서 비 CDB로	아니요

RDS for Oracle의 Oracle 멀티테넌트에 대한 자세한 내용은 CDB 아키텍처의 단일 테넌트 구성 단원을 참조하세요.

옵션 그룹 고려 사항

DB 인스턴스가 사용자 지정 옵션 그룹을 사용하는 경우 Amazon RDS가 새 옵션 그룹을 자동으로 할당하지 못할 수 있습니다. 예를 들어, 이 상황은 새로운 메이저 버전으로 업그레이드할 경우 발생합니다. 이 경우 업그레이드할 때 새 옵션 그룹을 지정합니다. 새 옵션 그룹을 생성하고 동일한 옵션을 기존 사용자 지정 옵션 그룹에 추가하는 것이 좋습니다.

자세한 내용은 옵션 그룹 생성 또는 옵션 그룹 생성 섹션을 참조하세요.

DB 인스턴스가 APEX 및 APEX-DEV 옵션이 포함된 사용자 지정 옵션 그룹을 사용하는 경우 업그레이드 시간이 단축될 수 있습니다. 이렇게 하려면 DB 인스턴스와 동시에 Oracle APEX 버전을 업그레이드하세요. 자세한 내용은 Oracle APEX 버전 업그레이드 섹션을 참조하세요.

파라미터 그룹 고려 사항

DB 인스턴스에서 사용자 지정 파라미터 그룹을 사용할 경우 Amazon RDS에서 DB 인스턴스에 새 파라미터 그룹을 자동으로 할당할 수 없는 경우도 있습니다. 예를 들어, 이 상황은 새로운 메이저 버전으로 업그레이드할 경우 발생합니다. 이 경우 업그레이드할 때 새 파라미터 그룹을 지정해야 합니다. 새 파라미터 그룹을 생성하고 기존 사용자 지정 파라미터 그룹에서와 같은 방법으로 파라미터를 구성하는 것이 좋습니다.

자세한 내용은 Amazon RDS에서 DB 파라미터 그룹 생성 또는 Amazon RDS에서 DB 파라미터 그룹 복사 섹션을 참조하세요.

시간대 고려 사항

시간대 옵션을 사용하면 Oracle DB 인스턴스에서 사용하는 시스템 시간대를 변경할 수 있습니다. 예를 들면 온프레미스 환경 또는 기존 애플리케이션과 시간을 호환하기 위해 DB 인스턴스의 시간대를 변경

할 수 있습니다. 시간대 옵션은 호스트 레벨에서 시간대를 변경합니다. Amazon RDS for Oracle은 연중 내내 시스템 시간대를 자동으로 업데이트합니다. 시스템 시간대에 대한 자세한 내용은 Oracle 시간대 단원을 참조하십시오.

Oracle DB 인스턴스를 생성하면 데이터베이스가 데이터베이스 시간대를 자동으로 설정합니다. 데이터베이스 시간대는 일광 절약 시간(DST) 시간대라고도 합니다. 데이터베이스 시간대는 시스템 시간대와 완전히 다릅니다.

Oracle Database 릴리스 간에 패치 세트 또는 개별 패치에 새 DST 버전이 포함될 수 있습니다. 이러한 패치는 다양한 시간대 리전에 대한 전환 규칙의 변경 사항을 반영합니다. 예를 들어 DST가 적용되면 정부가 변경될 수 있습니다. DST 규칙을 변경하면 `TIMESTAMP WITH TIME ZONE` 데이터 형식의 기존 데이터에 영향을 줄 수 있습니다.

RDS for Oracle DB 인스턴스를 업그레이드하더라도 Amazon RDS는 데이터베이스 시간대 파일은 자동으로 업그레이드되지 않습니다. 시간대 파일을 자동으로 업그레이드하려면 엔진 버전 업그레이드 도중 또는 이후에 DB 인스턴스와 연결된 옵션 그룹에 `TIMEZONE_FILE_AUTOUPGRADE` 옵션을 포함하면 됩니다. 자세한 내용은 Oracle 시간대 파일 자동 업그레이드을 참조하세요.

또는 데이터베이스 시간대 파일을 수동으로 업그레이드하려면 원하는 DST 패치가 있는 새 Oracle DB 인스턴스를 생성합니다. 하지만 `TIMEZONE_FILE_AUTOUPGRADE` 옵션을 사용하여 데이터베이스 시간대 파일을 업그레이드하는 것이 좋습니다.

시간대 파일을 업그레이드한 후 현재 인스턴스에서 새 인스턴스로 데이터를 마이그레이션합니다. 다음을 비롯한 여러 기술을 사용하여 데이터를 마이그레이션할 수 있습니다.

- AWS Database Migration Service
- Oracle GoldenGate
- Oracle Data Pump
- 원본 내보내기/가져오기(일반 용도로 지원되지 않음)

> **Note**
>
> Oracle Data Pump를 사용하여 데이터를 마이그레이션할 때 유틸리티는 대상 시간대 버전이 소스 시간대 버전보다 낮은 경우 ORA-39405 오류를 발생시킵니다.

자세한 내용은 Oracle 설명서에서 시간대 제한이 있는 타임스탬프를 참조하세요.

공간 패치 번들(SPB) 고려 사항

RDS for Oracle에서 릴리스 업데이트(RU)는 Oracle Database에 대한 보안 수정, 버그 수정 및 새로운 기능을 포함하는 마이너 엔진 버전입니다. 또한 공간 패치 번들(SPB)은 Oracle Spatial 옵션용으로 설계된 패치를 포함하는 마이너 엔진 버전입니다. 예를 들어 19.0.0.0.ru-2025-01.spb-1.r1은 엔진 버전 19.0.0.0.ru-2025-01.rur-2025-01.r1에 RU 패치와 공간 패치가 포함된 마이너 엔진 버전입니다.

데이터베이스를 SPB로 업그레이드하려면 다음 사항을 고려합니다.

- SPB는 Oracle Database 19c에서만 지원됩니다.
- 일반적으로 SPB는 해당 분기별 RU가 있고서 2~3주 후에 릴리스됩니다.
- 인스턴스가 Oracle Spatial 옵션을 사용하지 않더라도 DB 인스턴스를 SPB로 업그레이드할 수 있지만 엔진 버전의 Spatial 패치는 Oracle Spatial에만 적용됩니다. SPB에서 새 인스턴스를 만들고 나중에 Oracle Spatial 옵션을 설치할 수 있습니다.
- DB 인스턴스에 대해 마이너 버전 자동 업그레이드를 사용 설정하는 경우 업그레이드 경로는 인스턴스가 현재 SPB를 사용하는지 RU를 사용하는지에 따라 달라집니다. 인스턴스가 SPB를 사용하는 경우 RDS는 인스턴스를 최신 SPB로 자동으로 업그레이드합니다. 인스턴스가 RU를 사용하는 경우 RDS는 인스턴스를 최신 RU로 자동으로 업그레이드합니다.
- SPB가 현재 RU와 동일한 엔진 버전 또는 그 이상일 경우에만 DB 인스턴스를 RU에서 SPB로 수동으로 업그레이드할 수 있습니다.
- RU가 상위 버전인 경우에만 SPB에서 RU로 DB 인스턴스를 수동으로 업그레이드할 수 있습니다.

Oracle DB 업그레이드 테스트

DB 인스턴스를 메이저 버전으로 업그레이드하기 전에 데이터베이스와 해당 데이터베이스에 액세스하는 모든 애플리케이션을 철저히 테스트하여 새 버전과의 호환성을 확인합니다. 다음 절차를 참조하는 것이 좋습니다.

메이저 버전 업그레이드를 테스트하려면

1. 다음과 같이 새 버전의 데이터베이스 엔진에 대한 Oracle 업그레이드 문서를 검토하여 데이터베이스나 애플리케이션에 영향을 끼칠 수도 있는 호환성 문제가 있는지 살펴봅니다. 자세한 내용은 Oracle 문서의 Database Upgrade Guide 단원을 참조하십시오.
2. DB 인스턴스에서 사용자 지정 옵션 그룹을 사용할 경우 업그레이드하려는 새 버전과 호환되는 새 옵션 그룹을 생성합니다. 자세한 내용은 옵션 그룹 고려 사항 섹션을 참조하세요.
3. DB 인스턴스에서 사용자 지정 파라미터 그룹을 사용할 경우 업그레이드하려는 새 버전과 호환되는 새 파라미터 그룹을 생성합니다. 자세한 내용은 파라미터 그룹 고려 사항 섹션을 참조하세요.

4. 업그레이드할 DB 인스턴스의 DB 스냅샷을 생성합니다. 자세한 내용은 Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성 섹션을 참조하세요.
5. DB 스냅샷을 복구하여 새로운 테스트 DB 인스턴스를 생성합니다. 자세한 내용은 DB 인스턴스 복원 섹션을 참조하세요.
6. 다음 방법 중 한 가지를 사용하여 이 새로운 테스트 DB 인스턴스를 변경하고 새로운 버전으로 업그레이드합니다.

   - 콘솔
   - AWS CLI
   - RDS API
7. 테스트 수행:

   - 업그레이드한 DB 인스턴스와 관련하여 데이터베이스 및 애플리케이션과 새로운 버전의 호환성을 보장하는 데 필요하다면 최대한 많은 수의 품질 보증 테스트를 실행합니다.
   - 또한 1단계에서 발견된 호환성 문제의 영향을 평가하는 데 필요한 새로운 테스트도 모두 실행합니다.
   - 저장 프로시저와 함수, 트리거를 모두 테스트합니다.
   - 업그레이드한 DB 인스턴스에 대해 애플리케이션의 테스트 버전을 실행합니다. 새 버전에서 애플리케이션이 올바로 작동하는지 확인합니다.
   - 업그레이드한 인스턴스에서 사용할 스토리지를 평가하여 업그레이드 시 추가 스토리지의 필요 여부를 결정합니다. 프러덕션에서 새 버전을 지원하려면 더 큰 인스턴스 클래스를 선택해야 할 수도 있습니다. 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요.
8. 모든 테스트가 통과되면 프로덕션 DB 인스턴스로 업그레이드하세요. DB 인스턴스에 대한 쓰기 작업을 허용하기 전에 DB 인스턴스가 제대로 작동하는지 확인하는 것이 좋습니다.

RDS for Oracle DB 인스턴스 버전 업그레이드

RDS for Oracle DB 인스턴스의 DB 엔진 버전을 수동으로 업그레이드하려면 AWS Management Console, AWS CLI 또는 RDS API를 사용하세요. RDS에서 데이터베이스 업그레이드에 대한 일반적인 정보는 RDS for Oracle DB 인스턴스 버전 업그레이드 섹션을 참조하세요. 유효한 업그레이드 대상을 가져오려면 AWS CLI describe-db-engine-versions 명령을 사용하세요.

콘솔

콘솔을 사용하여 RDS for Oracle DB 인스턴스의 엔진 버전을 업그레이드하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 다음 업그레이드하려는 DB 인스턴스를 선택합니다.
3. 수정을 선택합니다.
4. DB 엔진 버전에서 더 높은 데이터베이스 버전을 선택합니다.
5. 계속해서 수정 사항을 요약한 내용을 확인합니다. 데이터베이스 버전 업그레이드의 영향을 이해해야 합니다. 업그레이드된 DB 인스턴스를 이전 버전으로 다시 변환할 수 없습니다. 계속하기 전에 새 버전으로 데이터베이스와 애플리케이션을 모두 테스트하세요.
6. DB 인스턴스 업그레이드 일정을 결정합니다. 변경 사항을 즉시 적용하려면 즉시 적용을 선택합니다. 일부의 경우 이 옵션을 선택하면 중단이 발생할 수 있습니다. 자세한 내용은 수정 예약 설정 사용 섹션을 참조하세요.
7. 확인 페이지에서 변경 내용을 검토합니다. 변경 내용이 정확할 경우 DB 인스턴스 수정을 선택하여 변경 내용을 저장합니다.

   그렇지 않으면 [Back]을 선택하여 변경 내용을 편집하거나 [Cancel]을 선택하여 변경 내용을 취소합니다.

AWS CLI

RDS for Oracle DB 인스턴스의 엔진 버전을 업그레이드하려면 CLI modify-db-instance 명령을 사용합니다. 다음 파라미터를 지정합니다.

- `--db-instance-identifier` - RDS for Oracle DB 인스턴스의 이름입니다.
- `--engine-version` – 업그레이드할 데이터베이스 엔진의 버전 번호입니다.

  유효한 엔진 버전에 대한 정보를 보려면 AWS CLI describe-db-engine-versions 명령을 사용합니다.
- `--allow-major-version-upgrade` - DB 엔진 버전을 업그레이드합니다.
- `--no-apply-immediately` – 변경 사항이 다음 유지 관리 기간에 적용됩니다. 변경 사항을 바로 적용하려면 `--apply-immediately`를 사용합니다.

Example

다음 예시는 이름이 myorainst인 CDB 인스턴스를 현재 버전인 19.0.0.0.ru-2024-01.rur-2024-01.r1에서 버전 21.0.0.0.ru-2024-04.rur-2024-04.r1로 업그레이드합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier myorainst \
    --engine-version 21.0.0.0.ru-2024-04.rur-2024-04.r1 \
    --allow-major-version-upgrade \
    --no-apply-immediately
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier myorainst ^
    --engine-version 21.0.0.0.ru-2024-04.rur-2024-04.r1 ^
    --allow-major-version-upgrade ^
    --no-apply-immediately
```

RDS API

RDS for Oracle DB 인스턴스를 업그레이드하려면 ModifyDBInstance 작업을 사용합니다. 다음 파라미터를 지정합니다.

- DBInstanceIdentifier – DB 인스턴스의 이름입니다(예:*myorainst*).
- EngineVersion – 업그레이드할 데이터베이스 엔진의 버전 번호입니다. 유효한 엔진 버전에 대한 정보를 보려면 DescribeDBEngineVersions 작업을 사용합니다.
- AllowMajorVersionUpgrade – 메이저 버전 업그레이드를 허용하는지 여부입니다. 그렇게 하려면 값을 true로 설정합니다.
- ApplyImmediately – 변경 사항을 즉시 적용하거나 다음 유지 관리 기간에 적용합니다. 변경 사항을 바로 적용하려면 값을 true로 설정합니다. 변경 사항을 다음 유지 관리 기간에 적용하려면 값을 false로 설정합니다.

Oracle DB 스냅샷 업그레이드

Amazon RDS에서 Oracle DB 스냅샷을 업그레이드하면 데이터베이스가 안전하고 호환 가능해지며 완벽하게 지원됩니다. 이전 Oracle 버전에서 패치 지원이 종료되면 이러한 버전에 연결된 수동 DB 스냅샷을 업그레이드하여 잠재적 취약성이나 서비스 제한을 피할 수 있습니다. 자세한 내용은 Oracle 엔진 버전 관리 단원을 참조하십시오.

Amazon RDS는 모든 AWS 리전에서 스냅샷 업그레이드를 지원합니다.

콘솔

Oracle DB 스냅샷을 업그레이드하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택한 다음 업그레이드하려는 DB 스냅샷을 선택합니다.
3. 작업에서 Upgrade snapshot(스냅샷 업그레이드)을 선택합니다. Upgrade snapshot(스냅샷 업그레이드) 페이지가 표시됩니다.
4. 스냅샷을 업그레이드할 새 엔진 버전을 선택합니다.
5. (선택 사항) 옵션 그룹에서 업그레이드된 DB 스냅샷의 옵션 그룹을 선택합니다. DB 스냅샷을 업그레이드할 때 고려할 옵션 그룹은 DB 인스턴스를 업그레이드할 때와 동일합니다. 자세한 내용은 옵션 그룹 고려 사항 섹션을 참조하세요.
6. 변경 사항을 저장하려면 변경 사항 저장을 선택합니다.

   업그레이드 중에는 이 DB 스냅샷의 모든 스냅샷 작업이 비활성화됩니다. 또한 DB 스냅샷 상태가 사용 가능에서 업그레이드 중으로 바뀐 다음 완료되면 활성으로 바뀝니다 스냅샷 손상 문제로 인해 DB 스냅샷을 업그레이드할 수 없는 경우, 상태가 사용할 수 없음으로 바뀝니다. 이 상태로부터 스냅샷을 복구할 수는 없습니다.

> **Note**
>
> DB 스냅샷 업그레이드에 실패하면 스냅샷이 원래 버전의 원래 상태로 롤백됩니다.

AWS CLI

AWS CLI를 사용하여 Oracle DB 스냅샷을 업그레이드하려면 다음 파라미터와 함께 modify-db-snapshot 명령을 호출합니다.

- `--db-snapshot-identifier` – DB 스냅샷의 이름입니다.
- `--engine-version` – 스냅샷을 업그레이드할 버전입니다.

또한 다음 파라미터를 포함해야 할 수 있습니다. DB 스냅샷을 업그레이드할 때 고려할 옵션 그룹은 DB 인스턴스를 업그레이드할 때와 동일합니다. 자세한 내용은 옵션 그룹 고려 사항 섹션을 참조하세요.

- `--option-group-name` – 업그레이드된 DB 스냅샷에 대한 옵션 그룹입니다.

Example

다음 예제는 DB 스냅샷을 업그레이드합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-snapshot \
    --db-snapshot-identifier mydbsnapshot \
    --engine-version 19.0.0.0.ru-2020-10.rur-2020-10.r1 \
    --option-group-name default:oracle-se2-19
```

Windows의 경우:

```
aws rds modify-db-snapshot ^
    --db-snapshot-identifier mydbsnapshot ^
    --engine-version 19.0.0.0.ru-2020-10.rur-2020-10.r1 ^
    --option-group-name default:oracle-se2-19
```

RDS API

Amazon RDS API를 사용하여 Oracle DB 스냅샷을 업그레이드하려면 다음 파라미터와 함께 ModifyDBSnapshot 작업을 호출하십시오.

- `DBSnapshotIdentifier` – DB 스냅샷의 이름입니다.
- `EngineVersion` – 스냅샷을 업그레이드할 버전입니다.

`OptionGroupName` 파라미터를 포함해야 할 수도 있습니다. DB 스냅샷을 업그레이드할 때 고려할 옵션 그룹은 DB 인스턴스를 업그레이드할 때와 동일합니다. 자세한 내용은 옵션 그룹 고려 사항 섹션을 참조하세요.

RDS for Oracle DB 인스턴스에 서드 파티 소프트웨어 사용

도구 및 서드 파티 소프트웨어를 지원하는 RDS for Oracle DB 인스턴스를 호스팅할 수 있습니다.

주제

- Amazon RDS for Oracle과 함께 Oracle GoldenGate 사용
- Oracle용 RDS에서 Oracle Repository Creation Utility 사용
- Amazon EC2 인스턴스에서 Oracle Connection Manager 구성
- Amazon RDS에서 Oracle에 Siebel Database 설치

Amazon RDS for Oracle과 함께 Oracle GoldenGate 사용

Oracle GoldenGate는 데이터베이스 간 트랜잭션 데이터를 수집, 복제 및 관리합니다. 온라인 트랜잭션 처리(OLTP) 시스템에서 데이터베이스와 함께 사용하는 로그 기반 변경 데이터 캡처(CDC) 및 복제 소프트웨어 패키지입니다. Oracle GoldenGate는 원본 데이터베이스에서 가장 최근에 변경된 데이터를 포함하는 추적 파일을 생성합니다. 그런 다음 이러한 파일을 서버로 푸시하고, 서버에서는 트레일 파일을 표준 SQL로 변환하여 대상 데이터베이스에 적용하는 프로세스가 진행됩니다.

RDS for Oracle을 사용하는 Oracle GoldenGate는 다음 기능을 지원합니다.

- 활성/활성 데이터베이스 복제
- 재해 복구
- 데이터 보호
- 리전 내 및 리전 간 복제
- 제로 가동 중지 마이그레이션 및 업그레이드
- RDS for Oracle DB 인스턴스와 비 Oracle 데이터베이스 간 데이터 복제

> **Note**
>
> 지원되는 데이터베이스 목록은 Oracle 설명서에서 Oracle Fusion Middleware 지원 시스템 구성을 참조하십시오.

RDS for Oracle이 포함된 Oracle GoldenGate를 사용하여 Oracle Database의 메이저 버전으로 업그레이드할 수 있습니다. 예를 들어 Oracle GoldenGate를 사용하여 Oracle Database 11g 온프레미스 데이터베이스를 Amazon RDS DB 인스턴스의 Oracle Database 19c로 업그레이드할 수 있습니다.

주제

- Oracle GoldenGate에 대해 지원되는 버전 및 라이선스 옵션
- Oracle GoldenGate의 요구 사항 및 제한 사항
- Oracle GoldenGate 아키텍처
- Oracle GoldenGate 설정
- Oracle GoldenGate의 EXTRACT 및 REPLICAT 유틸리티 작업
- Oracle GoldenGate 모니터링
- Oracle GoldenGate 문제 해결

Oracle GoldenGate에 대해 지원되는 버전 및 라이선스 옵션

Oracle GoldenGate 포함 RDS for Oracle 버전 12c 이상의 Standard Edition 2(SE2) 또는 Enterprise Edition(EE)을 사용할 수 있습니다. 다음 Oracle GoldenGate 기능을 사용할 수 있습니다.

- Oracle GoldenGate Remote Capture(추출)가 지원됩니다.
- Capture(추출)는 기존 비 CDB 데이터베이스 아키텍처를 사용하는 RDS for Oracle DB 인스턴스에서 지원됩니다. Oracle GoldenGate 원격 PDB 캡처는 Oracle Database 21c 또는 Oracle Database 19c 버전 19.0.0.0.ru-2024-04.rur-2024-04.r1 이상을 실행하는 CDB에서 지원됩니다.
- Oracle GoldenGate Remote Delivery(복제)는 CDB가 아닌 아키텍처나 CDB 아키텍처를 사용하는 RDS for Oracle DB 인스턴스에서 지원됩니다. Remote Delivery는 Integrated Replicat(통합 복제), Parallel Replicat(병렬 복제), Coordinated Replicat(조정 복제) 및 클래식 Replicat(복제)를 지원합니다.
- RDS for Oracle은 Oracle GoldenGate의 Classic(클래식) 및 Microservices(마이크로서비스) 아키텍처를 지원합니다.
- Oracle GoldenGate DDL과 Sequence(시퀀스) 값 복제는 Integrated(통합) 캡처 모드를 사용할 때 지원됩니다.

사용자는 모든 AWS 리전에서 Amazon RDS와 함께 사용하는 데 필요한 Oracle GoldenGate 라이선싱(BYOL)을 관리해야 합니다. 자세한 내용은 RDS for Oracle 라이선스 옵션 단원을 참조하십시오.

Oracle GoldenGate의 요구 사항 및 제한 사항

Oracle GoldenGate 및 RDS for Oracle을 작업할 때는 다음 요구 사항과 제한 사항을 고려해야 합니다.

- 사용자는 RDS for Oracle과 함께 사용할 수 있도록 Oracle GoldenGate를 설정하고 관리해야 합니다.
- 소스 및 대상 데이터베이스를 이용해 인증한 Oracle GoldenGate 버전을 설정하는 것은 사용자의 책임입니다. 자세한 내용은 Oracle 설명서에서 Oracle Fusion Middleware 지원 시스템 구성을 참조하십시오.
- Oracle GoldenGate는 다양한 AWS 환경에서 다양한 사용 사례에 사용할 수 있습니다. Oracle GoldenGate와 관련된 지원 관련 문제가 있다면 Oracle Support Services에 문의하십시오.
- Oracle Transparent Data Encryption(TDE)을 사용하는 RDS for Oracle DB 인스턴스에서 Oracle GoldenGate를 사용할 수 있습니다. 복제된 데이터의 무결성을 유지 관리하려면 Amazon EBS 암호화된 볼륨 또는 추적 파일 암호화를 사용하여 Oracle GoldenGate 허브에서 암호화를 구성합니다. 또한 Oracle GoldenGate 허브와 원본 및 대상 데이터베이스 인스턴스 간에 전송되는 데이터에 대한 암

호화를 구성합니다. RDS for Oracle DB 인스턴스는 Oracle 보안 소켓 Layer 또는 Oracle 기본 네트워크 암호화을 사용하여 암호화를 지원합니다.

Oracle GoldenGate 아키텍처

Amazon RDS용 OracleGoldenGate 아키텍처는 다음과 같은 분리된 모듈로 구성됩니다.

원본 데이터베이스

원본 데이터베이스는 온프레미스 Oracle 데이터베이스, Amazon EC2 인스턴스의 Oracle 데이터베이스 또는 Amazon RDS DB 인스턴스의 Oracle 데이터베이스입니다.

Oracle GoldenGate 허브

Oracle GoldenGate 허브는 트랜잭션 정보를 원본 데이터베이스에서 대상 데이터베이스로 이동합니다. 허브는 다음 중 하나일 수 있습니다.

- Oracle 데이터베이스 및 Oracle GoldenGate가 설치된 Amazon EC2 인스턴스
- 온프레미스 Oracle 설치

2개 이상의 Amazon EC2 허브가 있을 수 있습니다. 리전 간 복제에 Oracle GoldenGate를 사용하는 경우 허브를 2개 사용하는 것이 좋습니다.

대상 데이터베이스:

대상 데이터베이스는 Amazon RDS DB 인스턴스, Amazon EC2 인스턴스 또는 온프레미스 위치에 있을 수 있습니다.

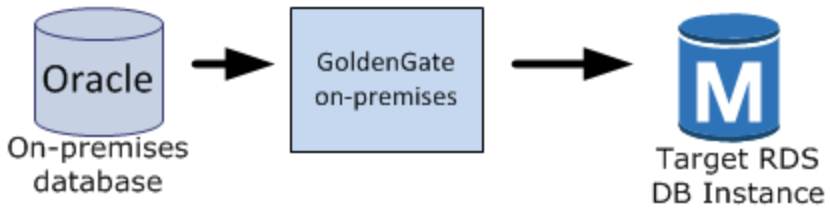
다음 섹션에서는 Amazon RDS의 Oracle GoldenGate에 대한 일반적인 시나리오를 설명합니다.

주제

- 온프레미스 원본 데이터베이스 및 Oracle GoldenGate 허브
- 온프레미스 원본 데이터베이스 및 Amazon EC2 허브
- Amazon RDS 원본 데이터베이스 및 Amazon EC2 허브
- Amazon EC2 원본 데이터베이스 및 Amazon EC2 허브
- 서로 다른 여러 AWS 리전의 Amazon EC2 허브

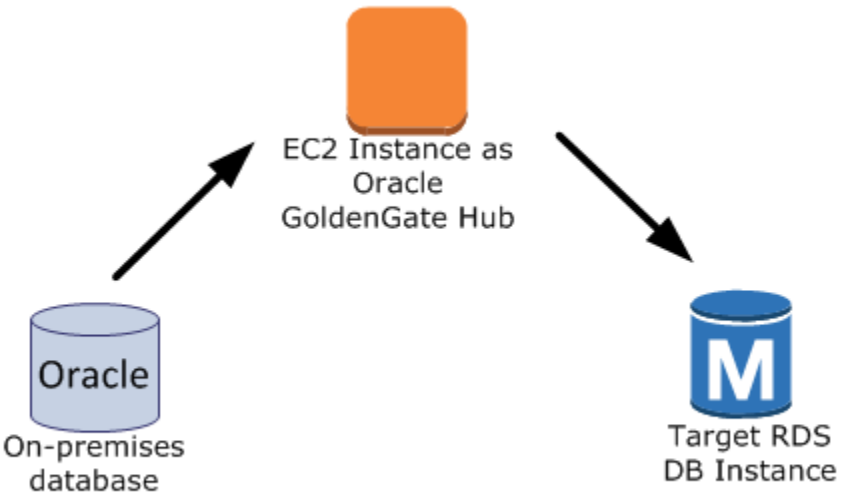
온프레미스 원본 데이터베이스 및 Oracle GoldenGate 허브

이 시나리오에서는 온프레미스 Oracle 원본 데이터베이스와 온프레미스 Oracle GoldenGate 허브가 대상 Amazon RDS DB 인스턴스에 데이터를 제공합니다.



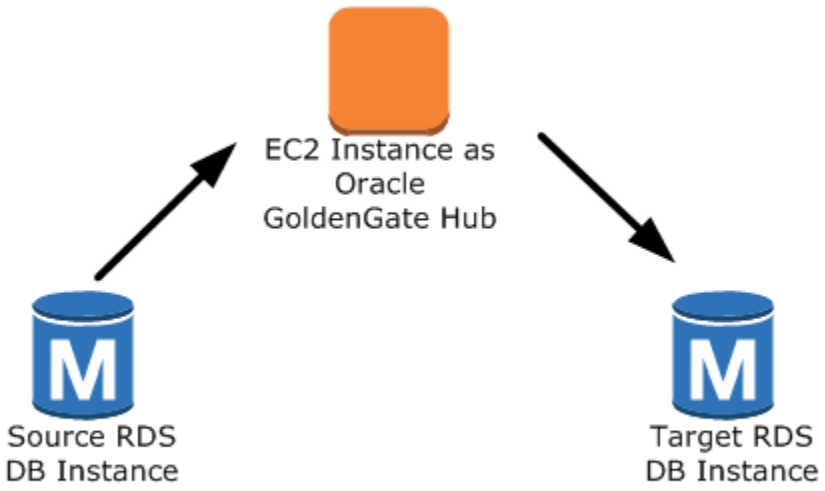
온프레미스 원본 데이터베이스 및 Amazon EC2 허브

이 시나리오에서는 온프레미스 Oracle 데이터베이스가 원본 데이터베이스 역할을 합니다. 이 데이터베이스가 Amazon EC2 인스턴스 허브에 연결됩니다. 이 허브는 대상 RDS for Oracle DB 인스턴스에 데이터를 제공합니다.



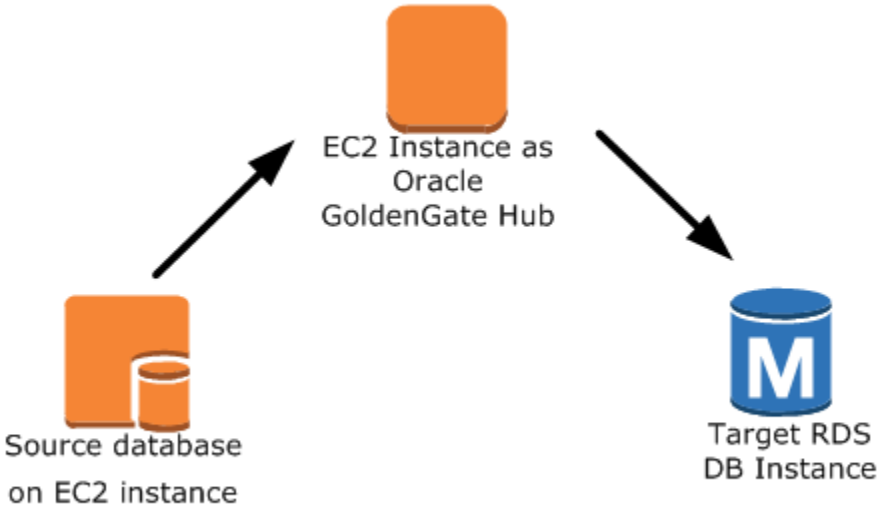
Amazon RDS 원본 데이터베이스 및 Amazon EC2 허브

이 시나리오에서는 RDS for Oracle DB 인스턴스가 원본 데이터베이스 역할을 합니다. 이 데이터베이스가 Amazon EC2 인스턴스 허브에 연결됩니다. 이 허브는 대상 RDS for Oracle DB 인스턴스에 데이터를 제공합니다.



Amazon EC2 원본 데이터베이스 및 Amazon EC2 허브

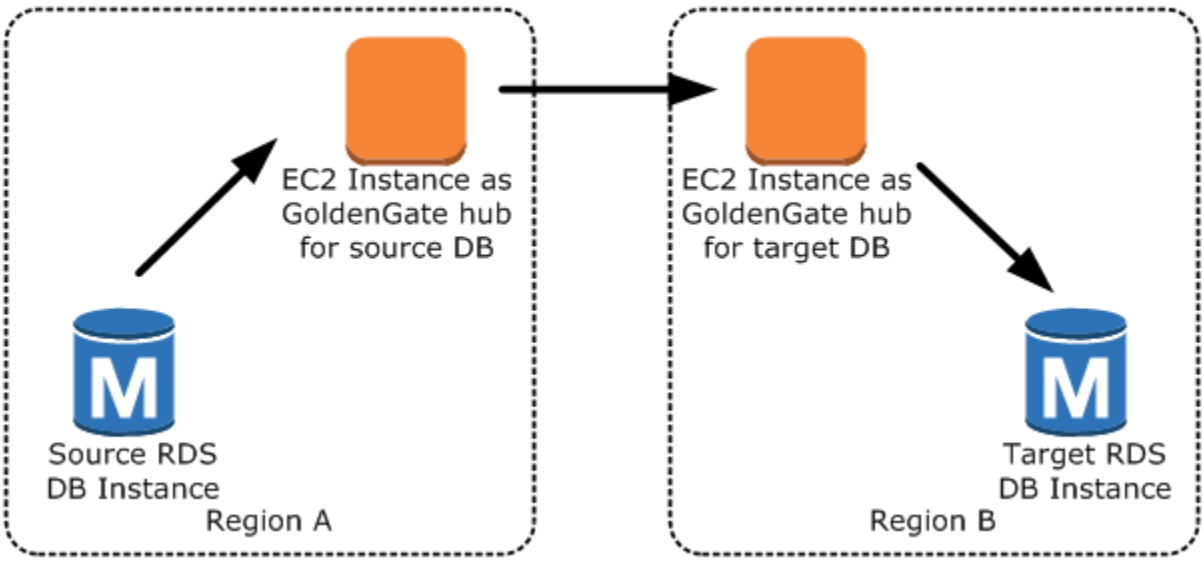
이 시나리오에서는 Amazon EC2 인스턴스의 Oracle 데이터베이스가 소스 데이터베이스 역할을 합니다. 이 데이터베이스가 Amazon EC2 인스턴스 허브에 연결됩니다. 이 허브는 대상 RDS for Oracle DB 인스턴스에 데이터를 제공합니다.



서로 다른 여러 AWS 리전의 Amazon EC2 허브

이 시나리오에서는 Amazon RDS DB 인스턴스의 Oracle 데이터베이스가 동일한 AWS 리전의 Amazon EC2 인스턴스 허브에 연결되어 있습니다. 허브는 다른 AWS 리전의 Amazon EC2 인스턴스

허브에 연결되어 있습니다. 이 두 번째 허브는 두 번째 Amazon EC2 인스턴스 허브와 동일한 AWS 리전의 대상 RDS for Oracle DB 인스턴스에 데이터를 제공합니다.



> **Note**
>
> 온프레미스 환경에서 Oracle GoldenGate를 실행하는 데 영향을 미치는 문제는 AWS에서 Oracle GoldenGate를 실행하는 데도 영향을 미칩니다. Oracle GoldenGate 허브를 모니터링하여 장애 조치가 발생하는 경우 `EXTRACT`와 `REPLICAT`가 재개되는지 확인하는 것이 좋습니다. Oracle GoldenGate 허브가 Amazon EC2 인스턴스에서 실행되어 Amazon RDS에서 Oracle GoldenGate 허브를 관리하지 않으므로 이 허브가 실행 중인지 확인할 수 없습니다.

Oracle GoldenGate 설정

Amazon RDS를 사용하는 Oracle GoldenGate를 설정하려면 Amazon EC2 인스턴스에서 허브를 구성한 후 소스 및 대상 데이터베이스를 구성해야 합니다. 다음 섹션에서는 Amazon RDS for Oracle과 함께 사용하도록 Oracle GoldenGate를 설치하는 방법을 보여줍니다.

주제

- Amazon EC2에 Oracle GoldenGate 허브 설정
- Amazon RDS에서 Oracle GoldenGate와 함께 사용할 소스 데이터베이스 설정
- Amazon RDS에서 Oracle GoldenGate와 함께 사용할 대상 데이터베이스 설정

Amazon EC2에 Oracle GoldenGate 허브 설정

Amazon EC2 인스턴스에서 Oracle GoldenGate 허브를 생성하려면 먼저 Oracle DBMS의 전체 클라이언트 설치가 완료된 Amazon EC2 인스턴스를 생성해야 합니다. Amazon EC2 인스턴스에는 Oracle GoldenGate 소프트웨어도 설치되어 있어야 합니다. Oracle GoldenGate 소프트웨어 버전은 소스 및 대상 데이터베이스 버전에 따라 달라집니다. Oracle GoldenGate 설치에 대한 자세한 내용은 Oracle GoldenGate 설명서를 참조하십시오.

Oracle GoldenGate 허브 역할을 하는 Amazon EC2 인스턴스는 원본 데이터베이스의 트랜잭션 정보를 추적 파일로 저장 및 처리합니다. 이 프로세스를 지원하려면 다음 요구 사항을 충족해야 합니다.

- 트레일 파일을 위한 충분한 저장 공간을 할당한 상태여야 합니다.
- Amazon EC2 인스턴스에 해당 데이터 용량을 관리할 수 있는 충분한 처리 능력이 있어야 합니다.
- 트랜잭션 정보를 추적 파일에 기록하기 전에 해당 정보를 저장할 충분한 메모리가 EC2 인스턴스에 있어야 합니다.

Amazon EC2 인스턴스에서 Oracle GoldenGate 클래식 아키텍처 허브를 설정하려면

1. Oracle GoldenGate 디렉터리에서 하위 디렉터리를 만듭니다.

   Amazon EC2 명령줄 쉘에서 Oracle GoldenGate 명령 인터프리터인 `ggsci`를 시작합니다. `CREATE SUBDIRS` 명령은 `/gg` 디렉터리에 파라미터, 보고서 및 체크포인트 파일용 하위 디렉터리를 만듭니다.

   ```
   prompt$ cd /gg
   prompt$ ./ggsci

   GGSCI> CREATE SUBDIRS
   ```

2. `mgr.prm` 파일을 구성합니다.

   다음 예에서는 `$GGHOME/dirprm/mgr.prm` 파일에 라인을 추가합니다.

   ```
   PORT 8199
   PurgeOldExtracts ./dirdat/*, UseCheckpoints, MINKEEPDAYS 5
   ```

3. 관리자를 시작합니다.

   다음 예제에서는 `ggsci`를 시작하고 `start mgr` 명령을 실행합니다.

```
GGSCI> start mgr
```

이제 Oracle GoldenGate 허브를 사용할 수 있습니다.

Amazon RDS에서 Oracle GoldenGate와 함께 사용할 소스 데이터베이스 설정

다음 작업을 완료하여 Oracle GoldenGate에서 사용할 소스 데이터베이스를 설정합니다.

설정 단계

- 1단계: 소스 데이터베이스에서 보충 로깅 켜기
- 2단계: ENABLE_GOLDENGATE_REPLICATION 초기화 파라미터를 true로 설정
- 3단계: 소스 데이터베이스에서 로그 보존 기간 설정
- 4단계: 소스 데이터베이스에서 Oracle GoldenGate 사용자 계정을 생성합니다.
- 5단계: 소스 데이터베이스에 사용자 계정 권한 부여
- 6단계: 소스 데이터베이스의 TNS 별칭 추가

1단계: 소스 데이터베이스에서 보충 로깅 켜기

최소 데이터베이스 수준 보충 로깅을 설정하려면 다음 PL/SQL 프로시저를 실행합니다.

```
EXEC rdsadmin.rdsadmin_util.alter_supplemental_logging(p_action => 'ADD')
```

2단계: ENABLE_GOLDENGATE_REPLICATION 초기화 파라미터를 true로 설정

`ENABLE_GOLDENGATE_REPLICATION` 초기화 파라미터를 `true`로 설정하면 데이터베이스 서비스가 논리적 복제를 지원할 수 있습니다. 소스 데이터베이스가 Amazon RDS DB 인스턴스에 있는 경우 파라미터 그룹을 DB 인스턴스에 할당하고 `ENABLE_GOLDENGATE_REPLICATION` 초기화 파라미터를 `true`로 설정해야 합니다. `ENABLE_GOLDENGATE_REPLICATION` 초기화 파라미터에 대한 자세한 내용은 Oracle 데이터베이스 설명서를 참조하세요.

3단계: 소스 데이터베이스에서 로그 보존 기간 설정

보관된 다시 실행 로그를 유지하도록 원본 데이터베이스를 구성했는지 확인합니다. 다음 지침을 참고하세요.

- 로그 보존 기간을 시간 단위로 지정합니다. 최소값은 1시간입니다.

- 소스 DB 인스턴스의 잠재적 가동 중지 시간, 잠재적 통신 시간, 소스 인스턴스의 네트워킹 문제가 발생할 잠재적 기간보다 긴 기간을 설정합니다. 이러한 기간을 통해 Oracle GoldenGate는 필요에 따라 원본 인스턴스에서 로그를 복구할 수 있습니다.
- 인스턴스에 파일을 저장할 스토리지가 충분한지 확인합니다.

예를 들어 보관된 다시 실행 로그의 보존 기간을 24시간으로 설정합니다.

```
EXEC rdsadmin.rdsadmin_util.set_configuration('archivelog retention hours',24)
```

로그 보존을 활성화하지 않거나 보존 값이 너무 작으면 다음과 비슷한 오류 메시지가 표시됩니다.

```
2022-03-06 06:17:27  ERROR   OGG-00446  error 2 (No such file or directory)
opening redo log /rdsdbdata/db/GGTEST3_A/onlinelog/o1_mf_2_9k4bp1n6_.log for sequence
 1306
Not able to establish initial position for begin time 2022-03-06 06:16:55.
```

DB 인스턴스는 보관된 다시 실행 로그를 보관하니, 파일을 저장할 공간이 충분한지 확인해야 합니다. 지난 *num_hours*시간 동안 사용한 공간의 양을 확인하려면 다음 쿼리를 사용하여 *num_hours*를 시간 숫자로 바꿔야 합니다.

```
SELECT SUM(BLOCKS * BLOCK_SIZE) BYTES FROM V$ARCHIVED_LOG
   WHERE NEXT_TIME>=SYSDATE-num_hours/24 AND DEST_ID=1;
```

4단계: 소스 데이터베이스에서 Oracle GoldenGate 사용자 계정을 생성합니다.

Oracle GoldenGate는 데이터베이스 사용자로 실행되며 소스 데이터베이스에 대한 다시 실행 및 저장된 다시 실행 로그에 액세스할 수 있는 적절한 데이터베이스 권한이 필요합니다. 이러한 권한을 제공하려면 소스 데이터베이스에서 사용자 계정을 생성해야 합니다. Oracle GoldenGate 사용자 계정 권한에 대한 자세한 내용은 Oracle 설명서를 참조하세요.

다음 문은 oggadm1이라는 사용자 계정을 생성합니다.

```
CREATE TABLESPACE administrator;
CREATE USER oggadm1  IDENTIFIED BY "password"
   DEFAULT TABLESPACE ADMINISTRATOR TEMPORARY TABLESPACE TEMP;
ALTER USER oggadm1 QUOTA UNLIMITED ON administrator;
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

5단계: 소스 데이터베이스에 사용자 계정 권한 부여

이 작업에서는 소스 데이터베이스에서 데이터베이스 사용자에게 필요한 계정 권한을 부여합니다.

소스 데이터베이스에서 계정 권한을 부여하려면

1. SQL 명령 `grant`와 `rdsadmin.rdsadmin_util` 프로시저 `grant_sys_object`를 사용하여 Oracle GoldenGate 사용자 계정에 필요한 권한을 부여합니다. 다음 문은 `oggadm1`이라는 사용자에게 권한을 부여합니다.

   ```
   GRANT CREATE SESSION, ALTER SESSION TO oggadm1;
   GRANT RESOURCE TO oggadm1;
   GRANT SELECT ANY DICTIONARY TO oggadm1;
   GRANT FLASHBACK ANY TABLE TO oggadm1;
   GRANT SELECT ANY TABLE TO oggadm1;
   GRANT SELECT_CATALOG_ROLE TO rds_master_user_name WITH ADMIN OPTION;
   EXEC rdsadmin.rdsadmin_util.grant_sys_object ('DBA_CLUSTERS', 'OGGADM1');
   GRANT EXECUTE ON DBMS_FLASHBACK TO oggadm1;
   GRANT SELECT ON SYS.V_$DATABASE TO oggadm1;
   GRANT ALTER ANY TABLE TO oggadm1;
   ```

2. Oracle GoldenGate 관리자가 되려면 사용자 계정에 필요한 권한을 부여합니다. 다음 PL/SQL 프로그램을 실행합니다.

   ```
   EXEC rdsadmin.rdsadmin_dbms_goldengate_auth.grant_admin_privilege (
       grantee                 => 'OGGADM1',
       privilege_type          => 'capture',
       grant_select_privileges => true,
       do_grants               => TRUE);
   ```

   권한을 취소하려면 동일한 패키지의 `revoke_admin_privilege` 프로시저를 사용합니다.

6단계: 소스 데이터베이스의 TNS 별칭 추가

Oracle 홈의 $ORACLE_HOME/network/admin/tnsnames.ora에 다음 항목을 추가하여 EXTRACT 프로세스에서 사용할 수 있습니다. tnsnames.ora 파일에 대한 자세한 내용은 Oracle 설명서를 참조하세요.

```
OGGSOURCE=
   (DESCRIPTION=
        (ENABLE=BROKEN)
        (ADDRESS_LIST=
            (ADDRESS=(PROTOCOL=TCP)(HOST=goldengate-source.abcdef12345.us-
west-2.rds.amazonaws.com)(PORT=8200)))
        (CONNECT_DATA=(SERVICE_NAME=ORCL))
    )
```

Amazon RDS에서 Oracle GoldenGate와 함께 사용할 대상 데이터베이스 설정

이 작업에서는 Oracle GoldenGate와 함께 사용할 대상 DB 인스턴스를 설정합니다.

설정 단계

- 1단계: ENABLE_GOLDENGATE_REPLICATION 초기화 파라미터를 true로 설정
- 2단계: 대상 데이터베이스에서 Oracle GoldenGate 사용자 계정을 생성합니다.
- 3단계: 대상 데이터베이스에서 계정 권한 부여
- 4단계: 대상 데이터베이스의 TNS 별칭 추가

1단계: ENABLE_GOLDENGATE_REPLICATION 초기화 파라미터를 true로 설정

ENABLE_GOLDENGATE_REPLICATION 초기화 파라미터를 true로 설정하면 데이터베이스 서비스가 논리적 복제를 지원할 수 있습니다. 소스 데이터베이스가 Amazon RDS DB 인스턴스에 있는 경우 파라미터 그룹을 DB 인스턴스에 할당하고 ENABLE_GOLDENGATE_REPLICATION 초기화 파라미터를 true로 설정해야 합니다. ENABLE_GOLDENGATE_REPLICATION 초기화 파라미터에 대한 자세한 내용은 Oracle 데이터베이스 설명서를 참조하세요.

2단계: 대상 데이터베이스에서 Oracle GoldenGate 사용자 계정을 생성합니다.

Oracle GoldenGate는 데이터베이스 사용자로 실행되며 적절한 데이터베이스 권한이 필요합니다. 이러한 권한을 부여하려면 대상 데이터베이스에서 사용자 계정을 생성해야 합니다.

다음 문은 oggadm1이라는 사용자를 생성합니다.

```
CREATE TABLESPSACE administrator;
CREATE USER oggadm1  IDENTIFIED BY "password"
   DEFAULT TABLESPACE administrator
   TEMPORARY TABLESPACE temp;
ALTER USER oggadm1 QUOTA UNLIMITED ON administrator;
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

3단계: 대상 데이터베이스에서 계정 권한 부여

이 작업에서는 대상 데이터베이스에서 데이터베이스 사용자에게 필요한 계정 권한을 부여합니다.

대상 데이터베이스에서 계정 권한 부여

1. 대상 데이터베이스에서 Oracle GoldenGate 사용자 계정에 필요한 권한을 부여합니다. 다음 예제에서는 oggadm1에 권한을 부여합니다.

   ```
   GRANT CREATE SESSION        TO oggadm1;
   GRANT ALTER SESSION         TO oggadm1;
   GRANT CREATE CLUSTER        TO oggadm1;
   GRANT CREATE INDEXTYPE      TO oggadm1;
   GRANT CREATE OPERATOR       TO oggadm1;
   GRANT CREATE PROCEDURE      TO oggadm1;
   GRANT CREATE SEQUENCE       TO oggadm1;
   GRANT CREATE TABLE          TO oggadm1;
   GRANT CREATE TRIGGER        TO oggadm1;
   GRANT CREATE TYPE           TO oggadm1;
   GRANT SELECT ANY DICTIONARY TO oggadm1;
   GRANT CREATE ANY TABLE      TO oggadm1;
   GRANT ALTER ANY TABLE       TO oggadm1;
   GRANT LOCK ANY TABLE        TO oggadm1;
   GRANT SELECT ANY TABLE      TO oggadm1;
   GRANT INSERT ANY TABLE      TO oggadm1;
   GRANT UPDATE ANY TABLE      TO oggadm1;
   GRANT DELETE ANY TABLE      TO oggadm1;
   ```

2. Oracle GoldenGate 관리자가 되려면 사용자 계정에 필요한 권한을 부여합니다. 다음 PL/SQL 프로그램을 실행합니다.

```
EXEC rdsadmin.rdsadmin_dbms_goldengate_auth.grant_admin_privilege (
    grantee                 => 'OGGADM1',
    privilege_type          => 'apply',
    grant_select_privileges => true,
    do_grants               => TRUE);
```

권한을 취소하려면 동일한 패키지의 `revoke_admin_privilege` 프로시저를 사용합니다.

4단계: 대상 데이터베이스의 TNS 별칭 추가

Oracle 홈의 `$ORACLE_HOME/network/admin/tnsnames.ora`에 다음 항목을 추가하여 REPLICAT 프로세스에서 사용할 수 있습니다. Oracle 멀티테넌트 데이터베이스의 경우 TNS 별칭이 PDB의 서비스 이름을 가리키는지 확인하세요. `tnsnames.ora` 파일에 대한 자세한 내용은 Oracle 설명서를 참조하세요.

```
OGGTARGET=
    (DESCRIPTION=
        (ENABLE=BROKEN)
        (ADDRESS_LIST=
            (ADDRESS=(PROTOCOL=TCP)(HOST=goldengate-target.abcdef12345.us-
west-2.rds.amazonaws.com)(PORT=8200)))
        (CONNECT_DATA=(SERVICE_NAME=ORCL))
    )
```

Oracle GoldenGate의 EXTRACT 및 REPLICAT 유틸리티 작업

Oracle GoldenGate 유틸리티인 EXTRACT와 REPLICAT는 함께 작동하여 트레일 파일을 사용한 증분형 트랜잭션 복제를 통해 소스 및 대상 데이터베이스의 동기 상태를 유지합니다. 원본 데이터베이스에 대해 발생하는 모든 변경 내용은 EXTRACT에 의해 자동으로 감지되고 서식이 지정되어 Oracle GoldenGate 온프레미스 또는 Amazon EC2 인스턴스 허브의 트레일 파일로 전송됩니다. 초기 로드가 완료된 후에는 REPLICAT 유틸리티에서 이러한 파일의 데이터를 읽어 대상 데이터베이스로 복제합니다.

Oracle GoldenGate EXTRACT 유틸리티 실행

EXTRACT 유틸리티는 소스 데이터베이스에서 데이터를 검색 및 변환하고 트레일 파일로 출력합니다. 기본 프로세스는 다음과 같습니다.

1. EXTRACT가 트랜잭션 세부 정보를 메모리나 임시 디스크 스토리지의 대기열에 저장합니다.

2. 소스 데이터베이스가 트랜잭션을 커밋합니다.
3. EXTRACT가 트랜잭션 세부 정보를 트레일 파일에 기록합니다.
4. 추적 파일은 이러한 세부 정보를 Oracle GoldenGate 온프레미스 또는 Amazon EC2 인스턴스 허브로 라우팅한 다음 대상 데이터베이스로 라우팅합니다.

다음 단계는 EXTRACT 유틸리티를 시작하고, 소스 데이터베이스 OGGSOURCE의 EXAMPLE.TABLE에서 데이터를 캡처하고, 트레일 파일을 생성합니다.

EXTRACT 유틸리티를 실행하려면

1. Oracle GoldenGate 허브(온프레미스 또는 Amazon EC2 인스턴스)에서 EXTRACT 파라미터 파일을 구성합니다. 다음 목록은 $GGHOME/dirprm/eabc.prm이라는 예제 EXTRACT 파라미터 파일을 보여 줍니다.

```
EXTRACT EABC

USERID oggadm1@OGGSOURCE, PASSWORD "my-password"
EXTTRAIL /path/to/goldengate/dirdat/ab

IGNOREREPLICATES
GETAPPLOPS
TRANLOGOPTIONS EXCLUDEUSER OGGADM1

TABLE EXAMPLE.TABLE;
```

2. Oracle GoldenGate 허브에서 소스 데이터베이스에 로그인하고 Oracle GoldenGate 명령줄 인터페이스 ggsci를 시작합니다. 다음 예제에서는 로그인 형식을 보여 줍니다.

```
dblogin oggadm1@OGGSOURCE
```

3. 데이터베이스 테이블에 대한 보충 로깅을 사용하기 위해 트랜잭션 데이터를 추가합니다.

```
add trandata EXAMPLE.TABLE
```

4. ggsci 명령줄에서 다음 명령을 사용하여 EXTRACT 유틸리티를 활성화합니다.

```
add extract EABC tranlog, INTEGRATED tranlog, begin now
add exttrail /path/to/goldengate/dirdat/ab
   extract EABC,
   MEGABYTES 100
```

5. EXTRACT 유틸리티를 데이터베이스에 등록하여 보관 로그가 삭제되지 않도록 합니다. 이 작업을 사용하면 필요한 경우 이전의 커밋되지 않은 트랜잭션을 복구할 수 있습니다. EXTRACT 유틸리티를 데이터베이스에 등록하려면 다음 명령을 사용합니다.

```
register EXTRACT EABC, DATABASE
```

6. 다음 명령을 이용해 EXTRACT 유틸리티를 시작합니다.

```
start EABC
```

Oracle GoldenGate REPLICAT 유틸리티 실행

REPLICAT 유틸리티는 트레일 파일의 트랜잭션 정보를 대상 데이터베이스로 "푸시"합니다.

다음 단계는 캡처된 데이터를 대상 데이터베이스 OGGTARGET의 EXAMPLE.TABLE 테이블에 복제할 수 있도록 REPLICAT 유틸리티를 활성화하고 시작합니다.

REPLICATE 유틸리티를 실행하려면

1. Oracle GoldenGate 허브(온프레미스 또는 EC2 인스턴스)에서 REPLICAT 파라미터 파일을 구성합니다. 다음 목록은 $GGHOME/dirprm/rabc.prm이라는 예제 REPLICAT 파라미터 파일을 보여 줍니다.

```
REPLICAT RABC

USERID oggadm1@OGGTARGET, password "my-password"

ASSUMETARGETDEFS
MAP EXAMPLE.TABLE, TARGET EXAMPLE.TABLE;
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

2. 대상 데이터베이스에 로그인하고 Oracle GoldenGate 명령줄 인터페이스(ggsci)를 시작합니다. 다음 예제에서는 로그인 형식을 보여 줍니다.

```
dblogin userid oggadm1@OGGTARGET
```

3. ggsci 명령줄을 사용하여 체크포인트 테이블을 추가합니다. 표시된 사용자는 대상 테이블 스키마 소유자가 아니라 Oracle GoldenGate 사용자 계정이어야 합니다. 다음 예제에서는 gg_checkpoint라는 체크포인트 테이블을 생성합니다.

```
add checkpointtable oggadm1.oggchkpt
```

4. REPLICAT 유틸리티를 활성화하려면 다음 명령을 사용합니다.

```
add replicat RABC EXTTRAIL /path/to/goldengate/dirdat/ab CHECKPOINTTABLE
 oggadm1.oggchkpt
```

5. 다음 명령을 사용하여 REPLICAT 유틸리티를 시작합니다.

```
start RABC
```

Oracle GoldenGate 모니터링

Oracle GoldenGate를 복제에 사용하는 경우 Oracle GoldenGate 프로세스가 실행 중이고 소스 및 대상 데이터베이스가 동기화되었는지 확인하세요. 다음 모니터링 도구를 사용할 수 있습니다.

- Amazon CloudWatch는 GoldenGate 오류 로그를 모니터링하는 데 이 패턴으로 사용되는 모니터링 서비스입니다.
- Amazon SNS는 이메일 알림을 보내는 데 이 패턴으로 사용되는 메시지 알림 서비스입니다.

자세한 지침은 Amazon CloudWatch를 사용한 Oracle GoldenGate 로그 모니터링을 참조하세요.

Oracle GoldenGate 문제 해결

이 섹션에서는 Amazon RDS for Oracle과 함께 Oracle GoldenGate를 사용할 때 가장 자주 발생하는 문제를 설명합니다.

주제

- 온라인 다시 실행 로그를 여는 중 발생하는 오류
- Oracle GoldenGate는 제대로 구성된 것으로 보이나 복제가 작동하지 않음
- SYS."_DBA_APPLY_CDR_INFO"에 대한 쿼리 때문에 통합 REPLICAT가 느려집니다.

온라인 다시 실행 로그를 여는 중 발생하는 오류

보관된 다시 실행 로그를 유지하도록 소스 데이터베이스를 구성했는지 확인합니다. 다음 지침을 참고하세요.

- 로그 보존 기간을 시간 단위로 지정합니다. 최소값은 1시간입니다.
- 소스 DB 인스턴스의 잠재적 가동 중지 시간, 잠재적 통신 시간, 소스 DB 인스턴스의 네트워킹 문제가 발생할 잠재적 기간보다 긴 기간을 설정합니다. 이러한 기간을 통해 Oracle GoldenGate는 필요에 따라 소스 DB 인스턴스에서 로그를 복구할 수 있습니다.
- 인스턴스에 파일을 저장할 스토리지가 충분한지 확인합니다.

로그 보존을 활성화하지 않거나 보존 값이 너무 작으면 다음과 비슷한 오류 메시지가 표시됩니다.

```
2022-03-06 06:17:27  ERROR   OGG-00446  error 2 (No such file or directory)
opening redo log /rdsdbdata/db/GGTEST3_A/onlinelog/o1_mf_2_9k4bp1n6_.log for sequence
 1306
Not able to establish initial position for begin time 2022-03-06 06:16:55.
```

Oracle GoldenGate는 제대로 구성된 것으로 보이나 복제가 작동하지 않음

기존 테이블의 경우 Oracle GoldenGate이 작동할 SCN을 지정해야 합니다.

이 문제를 해결하려면

1. 소스 데이터베이스에 로그인하고 Oracle GoldenGate 명령줄 인터페이스(ggsci)를 시작합니다. 다음 예제에서는 로그인 형식을 보여 줍니다.

   ```
   dblogin userid oggadm1@OGGSOURCE
   ```

2. ggsci 명령줄을 사용하여 EXTRACT 프로세스에 대한 시작 SCN을 설정합니다. 다음 예제에서는 EXTRACT을 위한 SCN을 223274로 설정합니다.

   ```
   ALTER EXTRACT EABC SCN 223274
   start EABC
   ```

3. 대상 데이터베이스에 로그인합니다. 다음 예제에서는 로그인 형식을 보여 줍니다.

   ```
   dblogin userid oggadm1@OGGTARGET
   ```

4. ggsci 명령줄을 사용하여 REPLICAT 프로세스에 대한 시작 SCN을 설정합니다. 다음 예제에서는 REPLICAT을 위한 SCN을 223274로 설정합니다.

```
start RABC atcsn 223274
```

SYS."_DBA_APPLY_CDR_INFO"에 대한 쿼리 때문에 통합 REPLICAT가 느려집니다.

Oracle GoldenGate 충돌 감지 및 해결(CDR) 기능은 기본적인 충돌 해결 루틴을 제공합니다. 예를 들어 CDR은 INSERT 문의 고유한 충돌 문제를 해결할 수 있습니다.

CDR이 충돌을 해결할 때, 일시적으로 예외 테이블 _DBA_APPLY_CDR_INFO에 레코드를 삽입할 수 있습니다. 통합 REPLICAT는 나중에 이러한 레코드를 삭제합니다. 드문 경우에, 통합 REPLICAT는 많은 수의 충돌을 처리할 수 있지만 새로운 통합 REPLICAT는 이를 대체하지 않습니다. _DBA_APPLY_CDR_INFO의 기존 행은 제거되지 않고 분리됩니다. 새로운 통합 REPLICAT 프로세스는 _DBA_APPLY_CDR_INFO에서 분리된 행을 쿼리하기 때문에 속도가 느려집니다.

_DBA_APPLY_CDR_INFO에서 모든 행을 제거하려면 Amazon RDS 프로시저 rdsadmin.rdsadmin_util.truncate_apply$_cdr_info를 사용합니다. 이 프로시저는 2020년 10월 릴리스 및 패치 업데이트의 일부로 릴리스되었습니다. 이 프로시저는 다음 데이터베이스 버전에서 사용할 수 있습니다.

- 버전 21.0.0.0.ru-2022-01.rur-2022-01.r1 이상
- 버전 19.0.0.0.ru-2020-10.rur-2020-10.r1 이상

다음 예제에서는 _DBA_APPLY_CDR_INFO 테이블을 자릅니다.

```
SET SERVEROUTPUT ON SIZE 2000
EXEC rdsadmin.rdsadmin_util.truncate_apply$_cdr_info;
```

Oracle용 RDS에서 Oracle Repository Creation Utility 사용

Amazon RDS를 사용하여 Oracle Fusion Middleware 구성 요소를 지원하는 스키마를 보관하는 Oracle DB 인스턴스용 RDS를 호스팅할 수 있습니다. Oracle Fusion Middleware 구성 요소를 사용하려면 데이터베이스에 이 구성 요소의 스키마를 생성하고 채웁니다. 스키마는 Oracle Repository Creation Utility(RCU)를 사용하여 만들고 채웁니다.

RCU에 대해 지원되는 버전 및 라이선스 옵션

Amazon RDS는 Oracle Repository Creation Utility(RCU) 버전 12c만 지원합니다. 다음 구성에서 RCU를 사용할 수 있습니다.

- RCU 12c, Oracle Database 21c
- RCU 12c, Oracle Database 19c

RCU를 사용하려면 다음을 수행해야 합니다.

- Oracle Fusion Middleware용 라이선스를 취득합니다
- 또한 리포지토리를 호스팅하는 Oracle Database용 Oracle 라이선싱 지침을 따릅니다. 자세한 내용은 Oracle 설명서에서 Oracle Fusion Middleware 라이선싱 정보 사용자 매뉴얼을 참조하십시오.

Fusion MiddleWare는 Oracle Database Enterprise Edition 및 Standard Edition 2의 리포지토리를 지원합니다. Oracle은 분할이 필요한 프로덕션 설치 및 온라인 인덱스 재구축이 필요한 설치에 Enterprise Edition을 권장합니다.

RDS for Oracle 인스턴스를 생성하기 전에 배포할 구성 요소를 지원해야 하는 Oracle Database 버전을 확인하세요. 인증표에서 배포할 Fusion Middleware 구성 요소 및 버전에 대한 요구 사항을 확인하십시오. 자세한 내용은 Oracle 설명서에서 Oracle Fusion Middleware 지원 시스템 구성을 참조하십시오.

Amazon RDS는 필요한 경우 Oracle Database 버전 업그레이드를 지원합니다. 자세한 정보는 DB 인스턴스 엔진 버전 업그레이드의 내용을 참조하세요.

RCU에 대한 요구 사항 및 제한 사항

RCU를 사용하려면 Amazon VPC가 필요합니다. Amazon RDS DB 인스턴스는 Fusion Middleware 구성 요소에서만 사용할 수 있고, 퍼블릭 인터넷에서는 사용할 수 없습니다. 따라서 보안 강화를 위해 프라이빗 서브넷에서 Amazon RDS DB 인스턴스를 호스팅합니다. RDS for Oracle DB 인스턴스도 필요합니다. 자세한 내용은 Oracle DB 인스턴스 생성 및 해당 인스턴스에 연결 단원을 참조하십시오.

Fusion Middleware 구성 요소에 대한 스키마를 Amazon RDS DB 인스턴스에 저장할 수 있습니다. 다음 스키마는 제대로 설치되는 것으로 확인되었습니다.

- 분석(ACTIVITIES)
- Audit Services(IAU)
- Audit Services Append(IAU_APPEND)
- Audit Services Viewer(IAU_VIEWER)
- Discussions(DISCUSSIONS)
- Metadata Services(MDS)
- Oracle Business Intelligence(BIPLATFORM)
- Oracle Platform Security Services(OPSS)
- 포털 및 서비스(WEBCENTER)
- 포틀릿 생산자(PORTLET)
- 서비스 테이블(STB)
- SOA 인프라(SOAINFRA)
- User Messaging Service(UCSUMS)
- WebLogic Services(WLS)

RCU 사용 지침

다음은 이 시나리오에서 DB 인스턴스를 작업하는 경우 알아 두어야 할 권장 사항입니다.

- 프로덕션 워크로드에는 다중 다중 AZ를 사용하는 것이 좋습니다. 다중 가용 영역 작업에 대한 자세한 내용은 리전, 가용 영역 및 로컬 영역 단원을 참조하세요.
- 보안을 추가하려면 유휴 데이터를 암호화할 수 있는 TDE(Transparent Data Encryption)를 사용하는 것이 좋습니다. 고급 보안 옵션이 제공되는 Enterprise Edition 라이선스를 보유하고 있는 경우 TDE 옵션을 사용하여 유휴 시 암호화를 활성화할 수 있습니다. 자세한 내용은 Oracle Transparent Data Encryption 섹션을 참조하세요.

  Amazon RDS는 모든 데이터베이스 버전에 유휴 시 암호화 옵션도 제공합니다. 자세한 내용은 Amazon RDS 리소스 암호화 섹션을 참조하세요.
- 애플리케이션 서버와 Amazon RDS DB 인스턴스 간에 통신할 수 있도록 VPC 보안 그룹을 구성합니다. Fusion Middleware 구성 요소를 호스트하는 애플리케이션 서버는 Amazon EC2 또는 온프레미스에 있을 수 있습니다.

RCU 실행

Fusion Middleware 구성 요소를 지원하기 위해 스키마를 생성하고 채우려면 Oracle Repository Creation Utility(RCU)를 사용합니다. 다양한 방법으로 RCU를 실행할 수 있습니다.

주제

- 한 단계로 명령줄을 사용하여 RCU 실행
- 여러 단계에 걸쳐 명령줄을 사용하여 RCU 실행
- 대화형 모드에서 RCU 실행

한 단계로 명령줄을 사용하여 RCU 실행

스키마를 채우기 전에 편집할 필요가 없는 경우 RCU를 한 단계로 실행할 수 있습니다. 편집이 필요한 경우 다음 단원에서 여러 단계에 걸쳐 RCU 실행에 대한 내용을 참조하세요.

명령줄 파라미터 `-silent`를 사용하면 자동 모드에서 RCU를 실행할 수 있습니다. 자동 모드에서 RCU를 실행하는 경우 비밀번호가 포함된 텍스트 파일을 만들어 명령줄에 비밀번호를 입력하지 않아도 됩니다. 첫 행에 `dbUser`의 암호가 있고 다음 행에 각 구성 요소에 대한 암호가 나오는 텍스트 파일을 만듭니다. RCU 명령의 마지막 파라미터로 암호 파일의 이름을 지정합니다.

Example

다음은 한 단계로 SOA 인프라 구성 요소(및 종속 항목)의 스키마를 만들어 채우는 예제입니다.

대상 LinuxmacOS, 또는Unix:

```
export ORACLE_HOME=/u01/app/oracle/product/12.2.1.0/fmw
export JAVA_HOME=/usr/java/jdk1.8.0_65
${ORACLE_HOME}/oracle_common/bin/rcu \
-silent \
-createRepository \
-connectString ${dbhost}:${dbport}:${dbname} \
-dbUser ${dbuser} \
-dbRole Normal \
-honorOMF \
-schemaPrefix ${SCHEMA_PREFIX} \
-component MDS \
-component STB \
-component OPSS \
-component IAU \
-component IAU_APPEND \
```

```
-component IAU_VIEWER \
-component UCSUMS \
-component WLS \
-component SOAINFRA \
-f < /tmp/passwordfile.txt
```

자세한 내용은 Oracle 설명서에서 명령줄에서 Repository Creation Utility 실행을 참조하세요.

여러 단계에 걸쳐 명령줄을 사용하여 RCU 실행

스키마 스크립트를 수동으로 편집하려면 여러 단계에 걸쳐 RCU를 실행합니다.

1. 스키마에 대한 스크립트를 생성하려면 `-generateScript` 명령줄 파라미터를 사용하여 Prepare Scripts for System Load 모드에서 RCU를 실행합니다.
2. 생성된 스크립트 `script_systemLoad.sql`을 수동으로 편집하고 실행합니다.
3. 스키마를 채우려면 `-dataLoad` 명령줄 파라미터를 사용하여 Perform Product Load 모드에서 RCU를 다시 실행합니다.
4. 생성된 정리 스크립트 `script_postDataLoad.sql`을 실행합니다.

자동 모드에서 RCU를 실행하려면 명령줄 파라미터 `-silent`를 지정합니다. 자동 모드로 RCU를 실행하는 경우 암호가 들어 있는 텍스트 파일을 만들면 명령줄에 암호를 입력하지 않아도 됩니다. 첫 행에 `dbUser`의 암호가 있고 다음 행에 각 구성 요소에 대한 암호가 나오는 텍스트 파일을 만듭니다. RCU 명령의 마지막 파라미터로 암호 파일의 이름을 지정합니다.

Example

다음은 SOA 인프라 구성 요소 및 종속 항목의 스키마 스크립트를 만드는 예제입니다.

대상 LinuxmacOS, 또는Unix:

```
export ORACLE_HOME=/u01/app/oracle/product/12.2.1.0/fmw
export JAVA_HOME=/usr/java/jdk1.8.0_65
${ORACLE_HOME}/oracle_common/bin/rcu \
-silent \
-generateScript \
-connectString ${dbhost}:${dbport}:${dbname} \
-dbUser ${dbuser} \
-dbRole Normal \
-honorOMF \
[-encryptTablespace true] \
-schemaPrefix ${SCHEMA_PREFIX} \
```

```
-component MDS \
-component STB \
-component OPSS \
-component IAU \
-component IAU_APPEND \
-component IAU_VIEWER \
-component UCSUMS \
-component WLS \
-component SOAINFRA \
-scriptLocation /tmp/rcuscripts \
-f < /tmp/passwordfile.txt
```

이제 생성된 스크립트를 편집하고, Oracle DB 인스턴스에 연결하고, 스크립트를 실행할 수 있습니다. 생성된 스크립트의 이름은 `script_systemLoad.sql`입니다. Oracle DB 인스턴스 연결에 대한 자세한 내용은 3단계: SQL 클라이언트를 Oracle DB 인스턴스에 연결 단원을 참조하세요.

다음은 SOA 인프라 구성 요소(및 종속 항목)의 스키마를 채우는 예제입니다.

대상 LinuxmacOS, 또는Unix:

```
export JAVA_HOME=/usr/java/jdk1.8.0_65
${ORACLE_HOME}/oracle_common/bin/rcu \
-silent \
-dataLoad \
-connectString ${dbhost}:${dbport}:${dbname} \
-dbUser ${dbuser} \
-dbRole Normal \
-honorOMF \
-schemaPrefix ${SCHEMA_PREFIX} \
-component MDS \
-component STB \
-component OPSS \
-component IAU \
-component IAU_APPEND \
-component IAU_VIEWER \
-component UCSUMS \
-component WLS \
-component SOAINFRA \
-f < /tmp/passwordfile.txt
```

완료하려면 Oracle DB 인스턴스에 연결하고 정리 스크립트를 실행합니다. 생성된 스크립트의 이름은 `script_postDataLoad.sql`입니다.

자세한 내용은 Oracle 설명서에서 명령줄에서 Repository Creation Utility 실행을 참조하세요.

대화형 모드에서 RCU 실행

RCU 그래픽 사용자 인터페이스를 사용하려면 대화형 모드로 RCU를 실행합니다. `-interactive` 파라미터를 포함하고 `-silent` 파라미터를 생략합니다. 자세한 내용은 Oracle 설명서에서 Repository Creation Utility 화면 이해를 참조하세요.

Example

다음은 대화형 모드에서 RCU를 시작하고 연결 정보를 미리 채우는 예제입니다.

대상 LinuxmacOS, 또는Unix:

```
export ORACLE_HOME=/u01/app/oracle/product/12.2.1.0/fmw
export JAVA_HOME=/usr/java/jdk1.8.0_65
${ORACLE_HOME}/oracle_common/bin/rcu \
-interactive \
-createRepository \
-connectString ${dbhost}:${dbport}:${dbname} \
-dbUser ${dbuser} \
-dbRole Normal
```

RCU 문제 해결

다음 문제에 주의하십시오.

주제

- Oracle Managed Files(OMF)
- 객체 권한
- Enterprise Scheduler Service

Oracle Managed Files(OMF)

Amazon RDS는 OMF 데이터 파일을 사용하여 스토리지 관리를 간소화합니다. 크기 및 익스텐트 관리와 같은 테이블스페이스 속성을 사용자 지정할 수 있습니다. 하지만 RCU를 실행할 때 데이터 파일 이름을 지정하면 테이블스페이스 코드에 `ORA-20900` 오류가 발생합니다. 다음과 같은 방법으로 OMF와 함께 RCU를 사용할 수 있습니다.

- RCU 12.2.1.0 이상에서 `-honorOMF` 명령줄 파라미터를 사용합니다.

- RCU 12.1.0.3 이상에서 여러 단계를 사용하여 생성된 스크립트를 편집합니다. 자세한 내용은 여러 단계에 걸쳐 명령줄을 사용하여 RCU 실행 단원을 참조하십시오.

객체 권한

Amazon RDS는 관리형 서비스이므로 RDS for Oracle 인스턴스에 대한 전체 SYSDBA 액세스 권한이 없습니다. 하지만 RCU 12c는 낮은 권한을 가진 사용자를 지원합니다. 대부분의 경우 마스터 사용자 권한으로 리포지토리를 생성할 수 있습니다.

마스터 계정은 `WITH GRANT OPTION`으로 이미 부여된 권한을 직접 부여할 수 있습니다. 경우에 따라 SYS 객체 권한을 부여하려고 할 때 RCU에 `ORA-01031` 오류가 발생할 수 있습니다. 다음 예와 같이 `rdsadmin_util.grant_sys_object` 저장 프로시저를 다시 시도하고 실행할 수 있습니다.

```
BEGIN
  rdsadmin.rdsadmin_util.grant_sys_object('GV_$SESSION','MY_DBA','SELECT');
END;
/
```

객체 SCHEMA_VERSION_REGISTRY에 대한 SYS 권한을 부여하려고 할 때, 작업에`ORA-20199: Error in rdsadmin_util.grant_sys_object` 오류가 발생할 수도 있습니다. SCHEMA_VERSION_REGISTRY$ 테이블과 SCHEMA_VERSION_REGISTRY 보기를 스키마 소유자 이름인 SYSTEM으로 정규화하고 작업을 재시도할 수 있습니다. 또는 동의어를 생성할 수 있습니다 마스터 사용자로 로그인하고 다음 명령문을 사용하십시오.

```
CREATE OR REPLACE VIEW SYSTEM.SCHEMA_VERSION_REGISTRY
  AS SELECT * FROM SYSTEM.SCHEMA_VERSION_REGISTRY$;
CREATE OR REPLACE PUBLIC SYNONYM SCHEMA_VERSION_REGISTRY FOR
 SYSTEM.SCHEMA_VERSION_REGISTRY;
CREATE OR REPLACE PUBLIC SYNONYM SCHEMA_VERSION_REGISTRY$ FOR SCHEMA_VERSION_REGISTRY;
```

Enterprise Scheduler Service

RCU를 사용하여 Enterprise Scheduler Service 리포지토리를 삭제하는 경우 RCU에 `Error: Component drop check failed` 오류가 발생할 수 있습니다.

Amazon EC2 인스턴스에서 Oracle Connection Manager 구성

Oracle Connection Manager(CMAN)는 데이터베이스 서버 또는 다른 프록시 서버에 연결 요청을 전달하는 프록시 서버입니다. CMAN으로 다음을 구성할 수 있습니다.

액세스 제어

사용자 지정 클라이언트 요청을 필터링하고 다른 요청을 수락하는 규칙을 생성할 수 있습니다.

세션 멀티플렉싱

네트워크 연결을 통해 공유 서버 대상에 여러 클라이언트 세션을 보낼 수 있습니다.

일반적으로 CMAN은 데이터베이스 서버 및 클라이언트 호스트와 별도의 호스트에 있습니다. 자세한 내용은 Oracle Database 설명서의 Configuring Oracle Connection Manager(Oracle Connection Manager 구성)를 참조하세요.

주제

- CMAN에 대해 지원되는 버전 및 라이선스 옵션
- CMAN에 대한 요구 사항 및 제한 사항
- CMAN 구성

CMAN에 대해 지원되는 버전 및 라이선스 옵션

CMAN은 Amazon RDS에서 지원하는 모든 Oracle Database 버전의 Enterprise Edition을 지원합니다. 자세한 정보는 RDS for Oracle 릴리스의 내용을 참조하세요.

Oracle Database가 설치된 호스트와 별도의 호스트에 Oracle Connection Manager를 설치할 수 있습니다. CMAN을 실행하는 호스트에는 별도의 라이선스가 필요하지 않습니다.

CMAN에 대한 요구 사항 및 제한 사항

완전 관리형 환경을 제공하기 위해 Amazon RDS는 운영 체제에 대한 액세스를 제한합니다. 운영 체제 액세스가 필요한 데이터베이스 파라미터는 수정할 수 없습니다. 따라서 Amazon RDS는 운영 체제에 로그인해야 하는 CMAN의 기능을 지원하지 않습니다.

CMAN 구성

CMAN을 구성할 때 RDS for Oracle 데이터베이스 외부에서 대부분의 작업을 수행합니다.

주제

- 1단계: RDS for Oracle 인스턴스와 동일한 VPC의 Amazon EC2 인스턴스에서 CMAN 구성
- 2단계: CMAN에 대한 데이터베이스 파라미터 구성
- 3단계: DB 인스턴스를 파라미터 그룹과 연결

1단계: RDS for Oracle 인스턴스와 동일한 VPC의 Amazon EC2 인스턴스에서 CMAN 구성

CMAN을 설정하는 방법을 알아보려면 블로그 게시물, Configuring and using Oracle Connection Manager on Amazon EC2 for Amazon RDS for Oracle(Amazon RDS for Oracle용 Amazon EC2에서 Oracle Connection Manager 구성 및 사용)의 자세한 지침을 따르세요.

2단계: CMAN에 대한 데이터베이스 파라미터 구성

Traffic Director 모드 및 세션 멀티플렉싱과 같은 CMAN 기능의 경우 `REMOTE_LISTENER` 파라미터를 DB 파라미터 그룹의 CMAN 인스턴스 주소로 설정합니다. 다음 시나리오를 고려해 보세요.

- CMAN 인스턴스는 IP 주소가 `10.0.159.100`인 호스트에 있으며 포트 `1521`을 사용합니다.
- 데이터베이스 `orcla`, `orclb` 및 `orclc`는 별도의 RDS for Oracle DB 인스턴스에 있습니다.

다음 표에서는 `REMOTE_LISTENER` 값을 설정하는 방법을 보여줍니다. `LOCAL_LISTENER` 값은 Amazon RDS에 의해 자동으로 설정됩니다.

DB 인스턴스 이름	DB 인스턴스 IP	로컬 리스너 값(자동 설정)	원격 리스너 값(사용자가 설정)
`orcla`	`10.0.159.200`	`( address= (protocol=tcp) (host=10.0.159.200) (port=1521) )`	`10.0.159.100:1521`
`orclb`	`10.0.159.300`	`( address= (protocol=tcp) (host=10.0.159.300) (port=1521) )`	`10.0.159.100:1521`

DB 인스턴스 이름	DB 인스턴스 IP	로컬 리스너 값(자동 설정)	원격 리스너 값(사용자가 설정)
orclc	10.0.159.400	(address= (protocol=tcp) (host=10.0.159.400) (port=1521))	10.0.159.100:1521

3단계: DB 인스턴스를 파라미터 그룹과 연결

2단계: CMAN에 대한 데이터베이스 파라미터 구성에서 구성한 파라미터 그룹을 사용하도록 DB 인스턴스를 생성하거나 수정합니다. 자세한 정보는 Amazon RDS의 DB 인스턴스에 DB 파라미터 그룹 연결의 내용을 참조하세요.

Amazon RDS에서 Oracle에 Siebel Database 설치

Amazon RDS를 사용하여 Oracle DB 인스턴스에서 Siebel Database를 호스팅할 수 있습니다. Siebel Database는 Siebel 고객 관계 관리(CRM) 애플리케이션 아키텍처의 일부입니다. 예시는 Generic Architecture of Siebel Business Application을 참조하십시오.

다음 주제는 Amazon RDS의 Oracle DB 인스턴스에서 Siebel Database를 설정하는 데 도움이 됩니다. 또한 Amazon Web Services를 사용하여 Siebel CRM 애플리케이션 아키텍처에서 요구하는 다른 구성 요소를 지원하는 방법도 설명합니다.

> **Note**
>
> Amazon RDS에서 Oracle에 Siebel Database를 설치하려면 마스터 사용자 계정을 사용해야 합니다. SYSDBA 권한은 필요 없으며, 마스터 사용자 권한으로 충분합니다. 자세한 내용은 마스터 사용자 계정 권한 섹션을 참조하세요.

라이선스 및 버전

Amazon RDS에 Siebel Database를 설치하려면 Oracle 데이터베이스 라이선스와 Siebel 라이선스를 사용해야 합니다. DB 인스턴스 클래스와 Oracle 데이터베이스 에디션에 적합한 Oracle 데이터베이스 라이선스(소프트웨어 업데이트 라이선스 및 지원 포함)가 있어야 합니다. 자세한 내용은 RDS for Oracle 라이선스 옵션 섹션을 참조하세요.

이 시나리오의 경우 Oracle Database Enterprise Edition은 Siebel이 인증한 유일한 버전입니다. Amazon RDS는 Siebel CRM 15.0 또는 16.0 버전을 지원합니다.

Amazon RDS는 데이터베이스 버전 업그레이드를 지원합니다. 자세한 내용은 DB 인스턴스 엔진 버전 업그레이드 섹션을 참조하세요.

시작하기 전에

시작하려면 Amazon VPC가 필요합니다. Amazon RDS DB 인스턴스는 퍼블릭 인터넷이 아닌 Siebel Enterprise Server에서만 사용 가능해야 하므로 Amazon RDS DB 인스턴스는 프라이빗 서브넷에서 호스팅되고 이로써 보안이 향상됩니다. Siebel CRM과 함께 사용할 Amazon VPC를 만드는 방법에 대한 정보는 Oracle DB 인스턴스 생성 및 해당 인스턴스에 연결 단원을 참조하십시오.

시작하려면 Oracle DB 인스턴스도 필요합니다. Siebel CRM과 함께 사용하도록 Oracle DB 인스턴스를 만드는 방법에 대한 자세한 내용은 Amazon RDS DB 인스턴스 생성 단원을 참조하십시오.

Siebel Database 설치 및 구성

Oracle DB 인스턴스를 만든 후에 Siebel Database를 설치할 수 있습니다. 테이블 소유자 및 관리자 계정을 만들어 데이터베이스를 설치하고, 저장된 절차와 기능을 설치한 다음 Siebel Database 구성 마법사를 실행합니다. 자세한 내용은 Installing the Siebel Database on the RDBMS를 참조하십시오.

Siebel Database 구성 마법사를 실행하려면 마스터 사용자 계정을 사용해야 합니다. SYSDBA 권한은 필요 없으며, 마스터 사용자 권한으로 충분합니다. 자세한 내용은 마스터 사용자 계정 권한 섹션을 참조하세요.

Siebel Database에 다른 Amazon RDS 기능 사용

Oracle DB 인스턴스를 만든 후, 다른 Amazon RDS 기능을 사용하여 Siebel Database를 사용자 지정할 수 있습니다.

Oracle Statspack 통계 수집 옵션

DB 옵션 그룹의 옵션을 사용해 Oracle DB 인스턴스에 기능을 추가할 수 있습니다. Oracle DB 인스턴스를 만들 때는 기본 DB 옵션 그룹을 사용했습니다. 데이터베이스에 기능을 추가하고 싶다면 DB 인스턴스를 위한 새로운 옵션 그룹을 만들 수 있습니다.

Siebel Database에서 성능 통계를 수집하고 싶다면 Oracle Statspack 기능을 추가할 수 있습니다. 자세한 내용은 Oracle Statspack 섹션을 참조하세요.

즉시 적용되는 옵션 변경 사항도 있고, DB 인스턴스의 다음 번 유지 관리 기간 중에 적용되는 옵션 변경 사항도 있습니다. 자세한 내용은 옵션 그룹 작업 섹션을 참조하세요. 사용자 지정 옵션 그룹을 생성한 후에는 DB 인스턴스를 수정해 옵션 그룹에 연결하십시오. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

파라미터로 성능 튜닝

DB 파라미터 그룹의 파라미터 사용을 통해 DB 엔진 구성을 관리합니다. Oracle DB 인스턴스를 만들 때는 기본 DB 파라미터 그룹을 사용했습니다. 데이터베이스 구성을 사용자 지정하고 싶다면 DB 인스턴스를 위한 새로운 파라미터 그룹을 만들 수 있습니다.

파라미터를 변경하는 경우, 파라미터 유형에 따라 변경 사항이 즉시 적용되거나 DB 인스턴스를 수동으로 재부팅한 후에 적용될 수 있습니다. 자세한 내용은 Amazon RDS의 파라미터 그룹 섹션을 참조하세요. 사용자 지정 파라미터 그룹을 생성한 후에는 DB 인스턴스를 수정해 파라미터 그룹에 연결하십시오. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

Siebel CRM에 맞게 Oracle DB 인스턴스를 최적화하기 위해 일부 파라미터를 사용자 지정할 수 있습니다. 다음 표에는 몇 가지 권장되는 파라미터 설정이 나와 있습니다. Sieble CRM 성능 튜닝에 대한 자세한 내용은 Siebel CRM Performance Tuning Guide를 참조하세요.

파라미터 이름	기본값	Siebel CRM 성능 최적화 지침
_always_semi_join	CHOOSE	OFF
_b_tree_bitmap_plans	TRUE	FALSE
_like_with_bind_as_equality	FALSE	TRUE
_no_or_expansion	FALSE	FALSE
_optimizer_join_sel_sanity_check	TRUE	TRUE
_optimizer_max_permutations	2000	100
_optimizer_sortmerge_join_enabled	TRUE	FALSE
_partition_view_enabled	TRUE	FALSE

파라미터 이름	기본값	Siebel CRM 성능 최적화 지침
open_cursors	300	최소 **2000**.

스냅샷 생성

Siebel Database를 만든 후 Amazon RDS의 스냅샷 기능을 이용해 데이터베이스를 복사할 수 있습니다. 자세한 내용은 Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성 및 DB 인스턴스 복원 단원을 참조하십시오.

다른 Siebel CRM 구성 요소에 대한 지원

Amazon Web Services를 사용하여 Siebel Database 외에 Siebel CRM 애플리케이션 아키텍처의 다른 구성 요소도 지원할 수 있습니다. 다음 표에 추가 Siebel CRM 구성 요소를 위한 Amazon AWS의 지원 옵션이 설명되어 있습니다.

Siebel CRM 구성 요소	Amazon AWS Support
Siebel Enterprise (Siebel 서버 1개 이상)	Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스에서 Siebel 서버를 호스팅할 수 있습니다. Amazon EC2를 사용하여 필요한 만큼 많거나 적은 수의 가상 서버를 시작할 수 있습니다. Amazon EC2를 사용하여 손쉽게 규모를 확장 또는 축소하여 수요 변화에 대처할 수 있습니다. 자세한 내용은 Amazon EC2란 무엇입니까?를 참조하세요. 서버를 DB 인스턴스와 같은 VPC에 배치하고 VPC 보안 그룹을 사용해 데이터베이스에 액세스할 수 있습니다. 자세한 내용은 VPC에서 DB 인스턴스를 사용한 작업 섹션을 참조하세요.
웹 서버 (Siebel Web Server Extensions 사용)	복수의 EC2 인스턴스에 복수의 웹 서버를 설치할 수 있습니다. 그런 다음 Elastic Load Balancing 을 사용해 수신 트래픽을 인스턴스에 분산할 수 있습니다. 자세한 내용은 Elastic Load Balancing란 무엇입니까?를 참조하십시오.
Siebel Gateway Name Server	EC2 인스턴스에서 Siebel Gateway Name Server를 호스팅할 수 있습니다. 그러면 서버를 DB 인스턴스와 같은

Siebel CRM 구성 요소	Amazon AWS Support
	VPC에 배치하고 VPC 보안 그룹을 사용해 데이터베이스에 액세스할 수 있습니다. 자세한 내용은 VPC에서 DB 인스턴스를 사용한 작업 섹션을 참조하세요.

Oracle 데이터베이스 엔진 릴리스 정보

Amazon RDS for Oracle DB 인스턴스 업데이트로 인스턴스를 최신으로 유지합니다. 업데이트를 적용하면 Oracle과 Amazon 모두에서 테스트를 거친 버전의 데이터베이스 소프트웨어를 DB 인스턴스에서 실행할 수 있습니다. 개별 RDS for Oracle DB 인스턴스에 대한 일회용 패치 적용은 지원하지 않습니다.

새 DB 인스턴스를 생성할 때 현재 지원되는 모든 Oracle 데이터베이스 버전을 지정할 수 있습니다. Oracle 데이터베이스 19c와 같은 메이저 버전과 지정된 메이저 버전에 대해 지원되는 모든 마이너 버전을 설정할 수 있습니다. 버전이 지정되지 않은 경우 Amazon RDS는 지원되는 버전(보통 최신 버전)을 기본값으로 설정합니다. 메이저 버전이 지정되었지만 마이너 버전이 지정되지 않은 경우, Amazon RDS는 고객이 지정한 메이저 버전의 최근 릴리스를 기본값으로 설정합니다. 지원되는 버전 목록과 새로 만든 DB 인스턴스의 기본값을 보려면 `describe-db-engine-versions` AWS CLI 명령을 사용합니다.

Amazon RDS에서 지원하는 Oracle 데이터베이스 버전에 대한 자세한 내용은 Amazon RDS for Oracle 릴리스 정보를 참조하십시오.

Amazon RDS for PostgreSQL

Amazon RDS는 여러 PostgreSQL 버전을 실행하는 DB 인스턴스를 지원합니다. 사용 가능한 버전 목록은 사용 가능한 PostgreSQL 데이터베이스 버전 섹션을 참조하세요.

DB 인스턴스 및 DB 스냅샷, 특정 시점으로 복원 및 백업을 만들 수 있습니다. PostgreSQL을 실행하는 DB 인스턴스는 다중 AZ 배포, 읽기 전용 복제본,프로비저닝된 IOPS를 지원하며 Virtual Private Cloud(VPC) 내부에서 생성될 수 있습니다. 또한, SSL(Secure Socket Layer)을 사용하여 PostgreSQL을 실행하는 DB 인스턴스에 연결할 수 있습니다.

DB 인스턴스를 생성하기 전에 Amazon RDS 환경 설정 단계를 완료해야 합니다.

클라이언트 컴퓨터에서 표준 SQL 클라이언트 애플리케이션을 사용하여 인스턴스에 대한 명령을 실행할 수 있습니다. 이런 애플리케이션으로는 PostgreSQL용으로 널리 사용되는 오픈 소스 관리 및 개발 도구인 pgAdmin 또는 PostgreSQL 설치 시 포함되는 명령줄 유틸리티인 psql이 포함됩니다. 관리형 서비스 환경을 제공하기 위해 Amazon RDS는 DB 인스턴스에 대해 호스트 액세스를 제공하지 않습니다. 또한 고급 권한이 필요한 특정 시스템 절차와 테이블에 대한 액세스를 제한합니다. Amazon RDS는 표준 SQL 클라이언트 애플리케이션을 사용하여 DB 인스턴스의 데이터베이스에 대한 액세스를 지원합니다. Amazon RDS에서는 Telnet 또는 SSH(Secure Shell)를 사용하여 DB 인스턴스에 대한 직접 호스트 액세스를 허용하지 않습니다.

Amazon RDS for PostgreSQL은 다수의 산업 표준을 준수합니다. 예를 들어 Amazon RDS for PostgreSQL 데이터베이스를 사용하여 HIPAA 준수 애플리케이션을 구축하고 의료 관련 정보를 저장할 수 있습니다. 여기에는 AWS와 체결한 비즈니스 제휴 계약(BAA)에 따라 보호 대상 건강 정보(PHI)를 위한 스토리지가 포함됩니다. 또한 Amazon RDS for PostgreSQL은 또한 연방 위험 및 인증 관리 프로그램(FedRAMP) 보안 요건을 충족합니다. Amazon RDS for PostgreSQL은 FedRAMP 공동 승인 위원회(JAB)로부터 AWS GovCloud (US) 리전 내에 FedRAMP High Baseline 수준의 잠정적 운영 권한(P-ATO)을 받았습니다. 지원되는 규정 준수 표준에 대한 자세한 내용은 AWS 클라우드 규정 준수를 참조하세요.

PostgreSQL 데이터를 DB 인스턴스로 가져오려면 Amazon RDS에서 PostgreSQL로 데이터 가져오기 섹션의 정보를 따릅니다.

> **Important**
>
> RDS for PostgreSQL DB 인스턴스에 문제가 발생하면 AWS Support 에이전트가 데이터베이스 상태에 대한 추가 정보가 필요할 수 있습니다. AWS Support가 가능한 한 빨리 필요한 정보를 확보하는 것이 목표입니다.

PG 컬렉터를 사용하여 통합 HTML 파일에서 중요한 데이터베이스 정보를 수집할 수 있습니다. PG 컬렉터, 실행 방법 및 HTML 보고서 다운로드 방법에 대한 자세한 내용은 PG 컬렉터를 참조하세요.
스크립트가 성공적으로 완료되고 달리 명시되지 않는 한, 스크립트는 읽기 가능한 HTML 형식으로 출력을 반환합니다. 스크립트는 업무에 영향을 미칠 수 있는 모든 데이터 또는 보안 세부 정보를 HTML에서 제외하도록 설계되었습니다. 또한 스크립트는 데이터베이스나 해당 환경을 수정하지 않습니다. 하지만 HTML에서 공유하기에 부적합한 정보를 발견한 경우, HTML을 업로드하기 전에 문제가 있는 정보를 얼마든지 제거해도 됩니다. HTML이 허용되는 경우, 지원 사례의 사례 세부 정보에 있는 첨부 파일 섹션을 사용하여 업로드합니다.

주제

- Amazon RDS for PostgreSQL의 일반적인 관리 태스크
- 데이터베이스 미리 보기 환경 작업
- 사용 가능한 PostgreSQL 데이터베이스 버전
- RDS for PostgreSQL 증분 릴리스 프로세스 이해
- 지원되는 PostgreSQL 확장 버전
- Amazon RDS for PostgreSQL에서 지원되는 PostgreSQL 기능 작업
- PostgreSQL 데이터베이스 엔진을 실행하는 DB 인스턴스에 연결
- SSL/TLS를 사용한 RDS for PostgreSQL 연결 보안
- Amazon RDS for PostgreSQL과 함께 Kerberos 인증 사용
- 아웃바운드 네트워크 액세스에 사용자 지정 DNS 서버 사용
- RDS for PostgreSQL DB 엔진 업그레이드
- PostgreSQL DB 스냅샷 엔진 버전 업그레이드
- Amazon RDS for PostgreSQL의 읽기 전용 복제본 작업
- Amazon RDS Optimized Reads로 RDS for PostgreSQL 쿼리 성능 개선
- Amazon RDS에서 PostgreSQL로 데이터 가져오기
- RDS for PostgreSQL DB 인스턴스에서 Amazon S3로 데이터 내보내기
- RDS for PostgreSQL DB 인스턴스에서 AWS Lambda 함수 호출
- Amazon RDS for PostgreSQL의 일반적인 DBA 태스크
- RDS for PostgreSQL의 대기 이벤트를 사용한 튜닝
- Amazon DevOps Guru의 사전 예방 인사이트를 활용하여 RDS for PostgreSQL 튜닝

- Amazon RDS for PostgreSQL로 PostgreSQL 확장 사용
- Amazon RDS for PostgreSQL용 지원되는 외부 데이터 래퍼 작업
- PostgreSQL용 신뢰할 수 있는 언어 확장 작업

Amazon RDS for PostgreSQL의 일반적인 관리 태스크

다음은 Amazon RDS for PostgreSQL DB 인스턴스로 수행하는 일반적인 관리 작업과 각 작업에 해당하는 문서 링크입니다.

작업 영역	관련 설명서
처음 사용 시 Amazon RDS 설정 DB 인스턴스를 생성하기 전에 몇 가지 사전 요구 사항을 완료해야 합니다. 예를 들어, DB 인스턴스는 기본적으로 인스턴스에 대한 액세스를 막는 방화벽도 함께 생성됩니다. 따라서 DB 인스턴스에 액세스하기 위한 알맞은 IP 주소와 네트워크 구성으로 보안 그룹을 만들어야 합니다.	Amazon RDS 환경 설정
Amazon RDS DB 인스턴스 이해 프로덕션 목적으로 DB 인스턴스를 만들 경우에는 Amazon RDS에서 인스턴스 클래스, 스토리지 유형 및 프로비저닝된 IOPS이 작동하는 방식을 이해해야 합니다.	DB 인스턴스 클래스 Amazon RDS 스토리지 유형 프로비저닝된 IOPS SSD 스토리지
사용 가능한 PostgreSQL 버전 찾기 Amazon RDS는 여러 버전의 PostgreSQL을 지원합니다.	사용 가능한 PostgreSQL 데이터베이스 버전
고가용성 및 장애 조치 지원 설정 프로덕션 DB 인스턴스에서는 다중 AZ 배포를 사용해야 합니다. 다중 AZ 배포는 DB 인스턴스를 위해 향상된 가용성, 데이터 내구성 및 내결함성을 제공합니다.	Amazon RDS에 대한 다중 AZ 배포 구성 및 관리
Amazon Virtual Private Cloud(VPC) 네트워크 이해	VPC에서 DB 인스턴스를 사용한 작업

작업 영역	관련 설명서
AWS 계정에 기본 VPC가 있는 경우에는 DB 인스턴스가 기본 VPC 내부에 자동으로 생성됩니다. 경우에 따라 계정에 기본 VPC가 없을 수 있으며 VPC에 DB 인스턴스가 필요할 수 있습니다. 이러한 경우에는 DB 인스턴스를 생성하기 전에 VPC 및 서브넷 그룹을 생성하세요.	
Amazon RDS PostgreSQL로 데이터 가져오기 다양한 도구를 사용하여 Amazon RDS의 PostgreSQL DB 인스턴스로 데이터를 가져올 수 있습니다.	Amazon RDS에서 PostgreSQL로 데이터 가져오기
읽기 전용 복제본(기본 및 스탠바이) 설정 RDS for PostgreSQL는 프라이머리 인스턴스와 동일한 AWS 리전과 다른 AWS 리전 모두에서 읽기 전용 복제본을 지원합니다.	DB 인스턴스 읽기 전용 복제본 작업 Amazon RDS for PostgreSQL의 읽기 전용 복제본 작업 다른 AWS 리전에서 읽기 전용 복제본 생성
보안 그룹 이해 기본적으로, DB 인스턴스와 함께 인스턴스에 대한 액세스를 막는 방화벽도 생성됩니다. 해당 방화벽을 통한 액세스를 제공하기 위해 DB 인스턴스를 호스팅하는 VPC와 연결된 VPC 보안 그룹의 인바운드 규칙을 편집합니다.	보안 그룹을 통한 액세스 제어
파라미터 그룹 및 기능 설정 DB 인스턴스의 기본 파라미터를 변경하려면 사용자 지정 DB 파라미터 그룹을 생성하고 설정을 변경하세요. DB 인스턴스를 생성하기 전인 경우, 인스턴스를 생성할 때 사용자 지정 DB 파라미터 그룹을 선택할 수 있습니다.	Amazon RDS의 파라미터 그룹

작업 영역	관련 설명서
PostgreSQL DB 인스턴스에 연결 보안 그룹을 만들고 이를 DB 인스턴스에 연결한 후 `psql` 또는 `pgAdmin`과 같은 스탠더드 SQL 클라이언트 애플리케이션을 사용하여 DB 인스턴스에 연결할 수 있습니다.	PostgreSQL 데이터베이스 엔진을 실행하는 DB 인스턴스에 연결 PostgreSQL DB 인스턴스와 함께 SSL 사용
DB 인스턴스 백업 및 복원 DB 인스턴스를 구성하여 자동 백업을 생성하거나 수동 스냅샷을 생성한 다음 백업 또는 스냅샷에서 인스턴스를 복원할 수 있습니다.	데이터 백업, 복원 및 내보내기
DB 인스턴스 활동 및 성능 모니터링 CloudWatch Amazon RDS 측정치, 이벤트 및 향상된 모니터링 기능을 통해 PostgreSQL DB 인스턴스를 모니터링할 수 있습니다.	Amazon RDS 콘솔에서 지표 보기 Amazon RDS 이벤트 보기
PostgreSQL 데이터베이스 버전 업그레이드 PostgreSQL DB 인스턴스의 메이저 버전과 마이너 버전을 모두 업그레이드할 수 있습니다.	RDS for PostgreSQL DB 엔진 업그레이드 RDS for PostgreSQL 업그레이드에 대한 메이저 버전 선택
로그 파일 작업 PostgreSQL DB 인스턴스의 로그 파일에 액세스할 수 있습니다.	RDS for PostgreSQL 데이터베이스 로그 파일
PostgreSQL DB 인스턴스에 대한 모범 사례 이해 Amazon RDS의 PostgreSQL 사용에 대한 몇 가지 모범 사례를 찾아 보십시오.	PostgreSQL로 작업하기 위한 모범 사례

다음은 PostgreSQL용 RDS의 주요 기능을 이해하고 사용하는 데 도움이 되는 이 가이드의 다른 섹션 목록입니다.

- PostgreSQL 역할 및 권한 이해

- PostgreSQL 데이터베이스에 대한 사용자 액세스 제어
- RDS for PostgreSQL DB 인스턴스에 파라미터로 작업
- RDS for PostgreSQL에서 지원되는 로깅 메커니즘 이해
- Amazon RDS for PostgreSQL에서 PostgreSQL 자동 정리 사용
- 아웃바운드 네트워크 액세스에 사용자 지정 DNS 서버 사용

데이터베이스 미리 보기 환경 작업

PostgreSQL 커뮤니티는 베타 버전을 포함한 새로운 PostgreSQL 버전과 확장을 지속적으로 릴리스합니다. 이를 통해 PostgreSQL 사용자는 새로운 PostgreSQL 버전을 조기에 체험할 수 있습니다. PostgreSQL 커뮤니티 베타 릴리스 프로세스에 대해 자세히 알아보려면 PostgreSQL 설명서의 베타 정보를 참조하세요. 마찬가지로 Amazon RDS는 특정 PostgreSQL 베타 버전을 미리 보기 릴리스로 제공합니다. 이를 통해 사용자는 미리 보기 버전을 사용하여 DB 인스턴스를 만들고 데이터베이스 미리 보기 환경에서 관련 기능을 테스트할 수 있습니다.

데이터베이스 미리 보기 환경의 RDS for PostgreSQL DB 인스턴스는 다른 RDS for PostgreSQL 인스턴스와 기능적으로 유사합니다. 그러나 프로덕션에는 미리 보기 버전을 사용할 수 없습니다.

이 방식을 사용하는 경우 다음과 같은 중요 제한 사항에 유의하십시오.

- 모든 DB 인스턴스는 생성 60일 후 백업 및 스냅샷과 함께 삭제됩니다.
- Amazon VPC 서비스 기반의 Virtual Private Cloud(VPC)에서만 DB 인스턴스를 생성할 수 있습니다.
- 범용 SSD와 프로비저닝된 IOPS SSD 스토리지만 사용할 수 있습니다.
- DB 인스턴스와 관련해서는 AWS Support의 지원을 받을 수 없습니다. 대신 AWS 관리형 Q&A 커뮤니티인 AWS re:Post에 질문을 게시할 수 있습니다.
- DB 인스턴스의 스냅샷을 프로덕션 환경으로 복제할 수 없습니다.

미리 보기에서 지원되는 옵션은 다음과 같습니다.

- DB 인스턴스는 M6i, R6i, M6g, M5, T3, R6g 및 R5인스턴스 유형으로만 만들 수 있습니다. RDS 인스턴스 클래스에 대한 자세한 내용은 DB 인스턴스 클래스 단원을 참조하세요.
- 단일 AZ 배포와 다중 AZ 배포를 모두 사용할 수 있습니다.
- 표준 PostgreSQL 덤프 및 로드 함수를 사용하여 데이터베이스 미리 보기 환경에서 데이터베이스를 내보내거나 데이터베이스 미리 보기 환경으로 데이터베이스를 가져올 수 있습니다.

주제

- 데이터베이스 미리 보기 환경에서 지원하지 않는 기능
- 데이터베이스 미리 보기 환경의 PostgreSQL 버전 17
- 데이터베이스 미리 보기 환경에서 새 DB 인스턴스 생성

데이터베이스 미리 보기 환경에서 지원하지 않는 기능

다음 기능은 데이터베이스 미리 보기 환경에서 사용할 수 없습니다.

- 리전 간 스냅샷 복제
- 리전 간 읽기 전용 복제본

데이터베이스 미리 보기 환경의 PostgreSQL 버전 17

> **Note**
>
> 이 문서는 Amazon RDS PostgreSQL 버전 17의 미리 보기 설명서입니다. 내용은 변경될 수 있습니다.

이제 PostgreSQL 버전 17.0을 Amazon RDS Database Preview Environment에서 사용할 수 있습니다. PostgreSQL 버전 17.0에는 여러 개선 사항이 포함되어 있으며, 이에 관한 설명은 다음 PostgreSQL 설명서에 나와 있습니다. PostgreSQL 17 릴리스

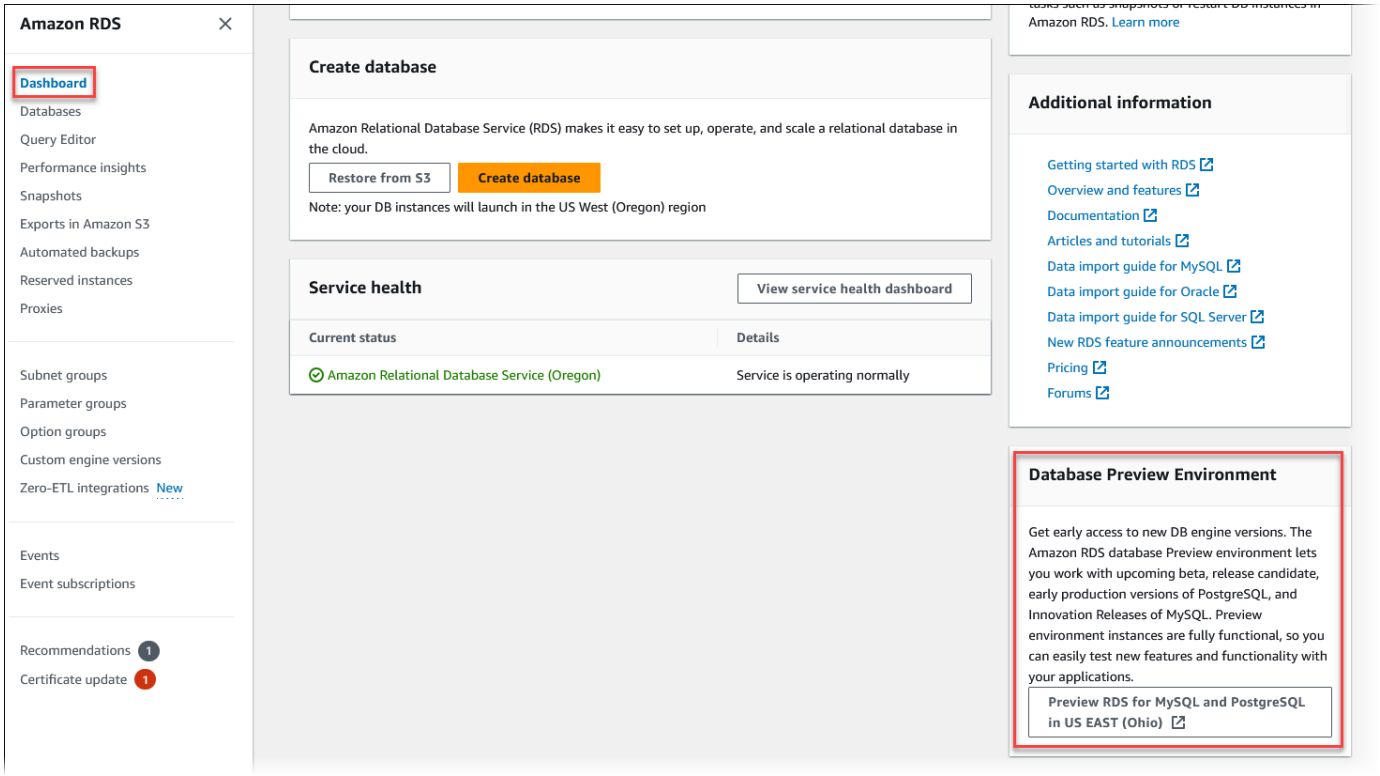
데이터베이스 미리 보기 환경에 대한 자세한 내용은 the section called “데이터베이스 미리 보기 환경 작업” 섹션을 참조하세요. 콘솔에서 미리 보기 환경에 액세스하려면 https://console.aws.amazon.com/rds-preview/를 선택합니다.

데이터베이스 미리 보기 환경에서 새 DB 인스턴스 생성

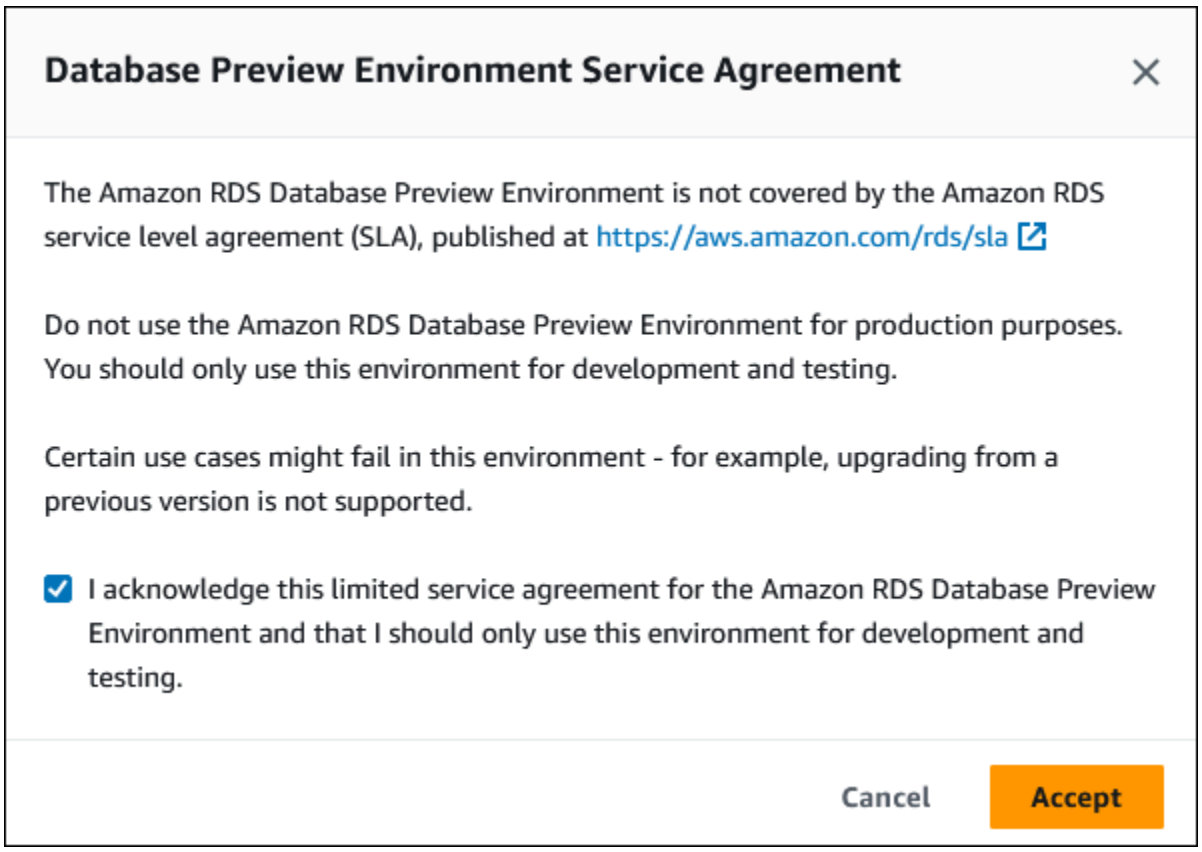
다음 절차를 이용하여 미리 보기 환경에서 DB 인스턴스를 새로 만듭니다.

데이터베이스 미리 보기 환경에서 DB 인스턴스를 생성하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 대시보드를 선택합니다.
3. 다음 그림과 같이 Dashboard(대시보드) 페이지에서 Database Preview Environment(데이터베이스 미리 보기 환경) 섹션을 찾습니다.



데이터베이스 미리 보기 환경으로 바로 이동할 수도 있습니다. 계속하려면 먼저 제한 사항을 확인하고 수락해야 합니다.



4. RDS for PostgreSQL DB 인스턴스를 생성하려면 아무 Amazon RDS DB 인스턴스를 생성할 때와 동일한 프로세스를 따릅니다. 자세한 내용은 DB 인스턴스 생성의 콘솔 절차를 참조하세요.

RDS API 또는 AWS CLI를 사용하여 데이터베이스 미리 보기 환경에 인스턴스를 생성하려면 다음 엔드포인트를 사용합니다.

```
rds-preview.us-east-2.amazonaws.com
```

사용 가능한 PostgreSQL 데이터베이스 버전

Amazon RDS는 여러 PostgreSQL 에디션을 실행하는 DB 인스턴스를 지원합니다. 새 DB 인스턴스를 생성할 때는 현재 사용 가능한 모든 PostgreSQL 버전을 지정할 수 있습니다. 메이저 버전(예: PostgreSQL 14) 및 지정된 메이저 버전에 대해 사용 가능한 모든 마이너 버전을 지정할 수 있습니다. 버전이 지정되지 않은 경우 Amazon RDS는 사용 가능한 버전(보통 최신 버전)을 기본값으로 설정합니다. 메이저 버전이 지정되었지만 마이너 버전이 지정되지 않은 경우, Amazon RDS는 고객이 지정한 메이저 버전의 최근 릴리스를 기본값으로 설정합니다.

사용 가능한 버전 목록과 새로 만든 DB 인스턴스의 기본값을 보려면 `describe-db-engine-versions` AWS CLI 명령을 사용합니다. 예를 들어 기본 PostgreSQL 엔진 버전을 표시하려면 다음 명령을 사용합니다.

```
aws rds describe-db-engine-versions --default-only --engine postgres
```

Amazon RDS에서 지원되는 PostgreSQL 버전에 대한 자세한 내용은 Amazon RDS for PostgreSQL 릴리스 정보를 참조하세요. describe-db-major-engine-versions AWS CLI 명령을 실행하거나 DescribeDBMajorEngineVersions RDS API 작업을 사용하여 메이저 엔진 버전의 지원 날짜에 대한 정보를 볼 수도 있습니다.

RDS 표준 지원 종료일 이전에 새 메이저 엔진 버전으로 수동 업그레이드할 준비가 되지 않은 경우 Amazon RDS에서 RDS 표준 지원 종료일 이후에 데이터베이스를 Amazon RDS 추가 지원에 자동 등록합니다. 그런 다음 RDS for PostgreSQL 버전 11 이상을 계속 실행할 수 있습니다. 자세한 내용은 Amazon RDS를 사용한 Amazon RDS 추가 지원 및 Amazon RDS 요금을 참조하세요.

더 이상 사용되지 않는 Amazon RDS for PostgreSQL 버전

다음 버전은 지원 중단되었다는 점에 유의하세요.

- RDS for PostgreSQL 10은 2023년 2월부로 지원 중단되었습니다.
- RDS for PostgreSQL 9.6은 2022년 3월부로 지원 중단되었습니다.
- RDS for PostgreSQL 9.5는 2021년 3월부로 지원 중단되었습니다.

RDS for PostgreSQL의 지원 중단 정책에 대한 자세한 내용은 Amazon RDS FAQ를 참조하세요. PostgreSQL 버전에 대한 자세한 내용은 PostgreSQL 설명서의 Versioning Policy(버전 관리 정책)을 참조하세요.

RDS for PostgreSQL 증분 릴리스 프로세스 이해

RDS for PostgreSQL은 마이너 버전 호환성을 유지하면서 증분 릴리스를 통해 보안 수정, 성능 개선 및 새로운 기능을 제공합니다. 이러한 릴리스에는 R1, R2, R3 등으로 레이블이 지정됩니다.

릴리스 버전 이름 지정 규칙

- 마이너 버전의 초기 릴리스는 R1입니다. 여기에는 때때로 새로운 기능, 확장 또는 기존 확장 프로그램에 대한 업그레이드가 포함됩니다.
- 후속 릴리스 버전(R2, R3 이상)에는 다음이 포함됩니다.
  - 보안 업데이트
  - 성능 개선
  - 버그 수정
  - 확장 프로그램 업데이트

RDS for PostgreSQL 증분 릴리스 프로세스의 이점

증분 릴리스 프로세스는 다음과 같은 이점을 제공합니다.

- 후속 릴리스를 통해 RDS별 개선 사항을 별도로 관리하면서 새로운 PostgreSQL 커뮤니티 릴리스를 빠르게 도입합니다. 이렇게 하면 릴리스 프로세스가 간소화되고 중요한 업데이트를 더 빠르게 제공할 수 있습니다.
- PostgreSQL 마이너 버전과의 호환성을 유지하면서 버그 수정, 새 기능, 보안 업데이트 및 확장 프로그램 업데이트에 액세스할 수 있습니다.

릴리스 업데이트 관리

Amazon RDS는 AWS Management Console에서 보류 중인 유지 관리 작업을 통해 새 증분 릴리스에 대해 알립니다. 다음 방법 중 하나를 사용하여 데이터베이스를 업데이트할 수 있습니다.

- 예약된 유지 관리 기간 동안 자동 업데이트를 활성화합니다.
- 보류 중인 유지 관리 작업을 통해 업데이트를 수동으로 적용합니다.
- 물리적 복제와 함께 블루/그린 배포를 사용하여 가동 중지 시간을 최소화합니다. 자세한 내용은 Amazon RDS 블루/그린 배포, RDS for PostgreSQL의 마이너 버전 업그레이드 지원을 참조하세요.

데이터베이스를 업데이트하기 전에 다음과 같은 중요한 점을 고려합니다.

- 물리적 복제와 함께 블루/그린 배포를 사용하지 않는 한 업데이트에는 데이터베이스 재부팅이 필요합니다.
- 일부 증분 릴리스, 특히 보안 수정 사항이 포함된 릴리스는 필수입니다.

Amazon RDS DB 인스턴스 업데이트에 대한 자세한 내용은 PostgreSQL 신뢰할 수 있는 확장 및 apply-pending-maintenance-action을 참조하세요.

지원되는 PostgreSQL 확장 버전

RDS for PostgreSQL는 여러 PostgreSQL 확장을 지원합니다. PostgreSQL 커뮤니티에서는 이러한 확장을 모듈이라고 부르기도 합니다. 여기에서 확장이란 PostgreSQL 엔진에서 제공하는 기능이 더욱 확장된 것을 말합니다. 해당 PostgreSQL 버전의 기본 DB 파라미터 그룹에서 Amazon RDS가 지원하는 확장 기능 목록을 확인할 수 있습니다. 또한 다음 예제에서처럼 `psql` 파라미터를 표시하면 `rds.extensions`을 사용하여 현재 확장 기능 목록을 확인할 수 있습니다.

```
SHOW rds.extensions;
```

> **Note**
>
> `rds.extensions`에서 `psql` 파라미터를 사용하면 마이너 버전 릴리스에서 추가된 파라미터가 잘못 표시될 수 있습니다.

RDS for PostgreSQL 13부터 특정 확장은 `rds_superuser` 이외의 데이터베이스 사용자가 설치할 수 있습니다. 신뢰할 수 있는 확장이라고 합니다. 자세한 내용은 PostgreSQL 신뢰할 수 있는 확장을 참조하십시오.

RDS for PostgreSQL의 특정 버전에서 `rds.allowed_extensions` 파라미터를 지원합니다. 이 파라미터를 사용하면 RDS for PostgreSQL DB 인스턴스에 설치할 수 있는 확장을 `rds_superuser` 제한할 수 있습니다. 자세한 내용은 PostgreSQL 확장의 설치 제한을 참조하세요.

사용 가능한 각 RDS for PostgreSQL 버전에서 지원되는 PostgreSQL 확장 및 버전 목록은 Amazon RDS for PostgreSQL 릴리스 정보의 Amazon RDS에서 지원되는 PostgreSQL 확장을 참조하세요.

PostgreSQL 확장의 설치 제한

PostgreSQL DB 인스턴스에 설치할 수 있는 확장을 제한할 수 있습니다. 기본적으로 이 파라미터는 설정되지 않으므로 사용자에게 권한이 있는 경우 지원되는 확장 프로그램을 추가할 수 있습니다. 이렇게 하려면 `rds.allowed_extensions` 파라미터를 쉼표로 구분된 확장 이름의 문자열로 설정합니다. 이 파라미터에 확장 프로그램 목록을 추가하면 RDS for PostgreSQL DB 인스턴스가 사용할 수 있는 확장을 명시적으로 식별할 수 있습니다. 그런 다음 해당 확장만 PostgreSQL DB 인스턴스에 설치할 수 있습니다.

`rds.allowed_extensions` 파라미터의 기본 문자열은 '*'입니다. 즉, 해당 엔진 버전에 사용할 수 있는 모든 확장을 설치할 수 있습니다. `rds.allowed_extensions` 파라미터는 동적 파라미터이므로 변경해도 데이터베이스를 다시 시작할 필요가 없습니다.

rds.allowed_extensions 파라미터를 사용하려면 PostgreSQL DB 인스턴스 엔진이 다음 버전 중 하나여야 합니다.

- 모든 PostgreSQL 16 버전
- PostgreSQL 15 이상의 모든 버전
- PostgreSQL 14 이상의 모든 버전
- PostgreSQL 13.3 이상의 마이너 버전
- PostgreSQL 12.7 이상의 마이너 버전

허용되는 확장 설치를 확인하려면 다음 psql 명령을 사용합니다.

```
postgres=> SHOW rds.allowed_extensions;
 rds.allowed_extensions
------------------------
 *
```

확장이 rds.allowed_extensions 파라미터의 목록에서 제외되기 전에 설치된 경우, 해당 확장은 정상적으로 사용할 수 있으며 ALTER EXTENSION 및 DROP EXTENSION 같은 명령이 계속 작동합니다. 하지만 확장이 제한되고 나면 제한된 확장에 대한 CREATE EXTENSION 명령이 실패합니다.

CREATE EXTENSION CASCADE를 사용한 확장 종속 구성 요소 설치도 제한됩니다. 확장과 해당 종속 구성 요소는 rds.allowed_extensions에 지정해야 합니다. 확장 종속 구성 요소 설치가 실패하면 전체 CREATE EXTENSION CASCADE 문이 실패합니다.

확장이 rds.allowed_extensions 파라미터에 포함되어 있지 않은 경우, 해당 확장을 설치하려고 하면 다음과 같은 오류가 표시됩니다.

```
ERROR: permission denied to create extension "extension-name"
HINT: This extension is not specified in "rds.allowed_extensions".
```

PostgreSQL 신뢰할 수 있는 확장

대부분의 PostgreSQL 확장을 설치하려면 rds_superuser 권한이 필요합니다. PostgreSQL 13에서는 rds_superuser 권한을 일반 사용자에게 부여해야 할 필요성을 줄이는 신뢰할 수 있는 확장 기능이 도입되었습니다. 이 기능을 사용하면 현재 데이터베이스에 대한 CREATE 권한이 있는 사용자가 rds_superuser 역할 없이도 여러 확장을 설치할 수 있습니다. 자세한 내용은 PostgreSQL 설명서에서 SQL CREATE EXTENSION 명령을 참조하세요.

다음 목록에는 현재 데이터베이스에 대한 CREATE 권한이 있는 사용자가 rds_superuser역할 없이 설치할 수 있는 확장이 나열되어 있습니다.

- bool_plperl
- btree_gin
- btree_gist
- citext
- cube
- dict_int
- fuzzystrmatch
- hstore
- intarray
- isn
- jsonb_plperl
- ltree
- pg_trgm
- pgcrypto
- plperl
- plpgsql
- pltcl
- tablefunc
- tsm_system_rows
- tsm_system_time
- unaccent
- uuid-ossp

사용 가능한 각 RDS for PostgreSQL 버전에서 지원되는 PostgreSQL 확장 및 버전 목록은 Amazon RDS for PostgreSQL 릴리스 정보의 Amazon RDS에서 지원되는 PostgreSQL 확장을 참조하세요.

Amazon RDS for PostgreSQL에서 지원되는 PostgreSQL 기능 작업

Amazon RDS for PostgreSQL 가장 일반적인 기타 기능을 많이 지원합니다. 예를 들어 PostgreSQL에는 데이터베이스에서 일상적인 유지 관리를 수행하는 자동 autovacuum 기능이 있습니다. 자동 정리 실행 기능은 기본적으로 활성화되어 있습니다. 이 기능을 끌 수 있지만 계속 사용하는 것이 좋습니다. 이 기능을 이해하고 제대로 작동하는지 확인하기 위해 수행할 수 있는 작업은 모든 DBA의 기본 작업입니다. autovacuum에 대한 자세한 내용은 Amazon RDS for PostgreSQL에서 PostgreSQL 자동 정리 사용 섹션을 참조하세요. 기타 일반적인 DBA 작업에 대해 자세히 알아보려면 Amazon RDS for PostgreSQL의 일반적인 DBA 태스크 섹션을 참조하세요.

RDS for PostgreSQL은 DB 인스턴스에 중요한 기능을 추가하는 확장도 지원합니다. 예를 들어 PostGIS 확장을 사용하여 공간 데이터로 작업하거나 pg_cron 확장을 사용하여 인스턴스 내에서 유지 관리를 예약할 수 있습니다. PostgreSQL 확장에 대한 자세한 내용은 Amazon RDS for PostgreSQL로 PostgreSQL 확장 사용 섹션을 참조하세요.

외부 데이터 래퍼는 RDS for PostgreSQL DB 인스턴스가 다른 상용 데이터베이스 또는 데이터 유형과 함께 작동하도록 설계된 특정 유형의 확장입니다. RDS for PostgreSQL에서 지원되는 외부 데이터 래퍼에 대한 자세한 내용은 Amazon RDS for PostgreSQL용 지원되는 외부 데이터 래퍼 작업 섹션을 참조하세요.

다음에서 RDS for PostgreSQL이 지원하는 일부 다른 기능에 대한 정보를 확인할 수 있습니다.

주제

- RDS for PostgreSQL을 사용한 사용자 지정 데이터 유형 및 열거
- RDS for PostgreSQL용 이벤트 트리거
- RDS for PostgreSQL용 방대한 페이지
- Amazon RDS for PostgreSQL에 대한 논리적 복제 수행
- stats_temp_directory에 대한 RAM 디스크
- RDS for PostgreSQL용 테이블스페이스
- EBCDIC 및 기타 메인프레임 마이그레이션을 위한 RDS for PostgreSQL 데이터 정렬
- RDS for PostgreSQL에 대한 논리적 슬롯 동기화 관리

RDS for PostgreSQL을 사용한 사용자 지정 데이터 유형 및 열거

PostgreSQL은 사용자 지정 데이터 형식을 만들고 열거형 작업을 지원합니다. 열거형 및 기타 데이터 유형을 생성하고 사용하는 방법에 대한 자세한 내용은 PostgreSQL 문서의 열거 형식을 참조하세요.

다음은 유형을 열거형으로 만든 다음 테이블에 값을 삽입하는 예입니다.

```
CREATE TYPE rainbow AS ENUM ('red', 'orange', 'yellow', 'green', 'blue', 'purple');
CREATE TYPE
CREATE TABLE t1 (colors rainbow);
CREATE TABLE
INSERT INTO t1 VALUES ('red'), ( 'orange');
INSERT 0 2
SELECT * from t1;
colors
--------
red
orange
(2 rows)
postgres=> ALTER TYPE rainbow RENAME VALUE 'red' TO 'crimson';
ALTER TYPE
postgres=> SELECT * from t1;
colors
---------
crimson
orange
(2 rows)
```

RDS for PostgreSQL용 이벤트 트리거

현재 모든 PostgreSQL 버전은 이벤트 트리거를 지원하며 사용 가능한 모든 RDS for PostgreSQL 버전도 지원합니다. 기본 사용자 계정(기본값, `postgres`)을 사용하여 이벤트 트리거를 생성, 수정, 이름 변경 및 삭제할 수 있습니다. 이벤트 트리거는 DB 인스턴스 레벨에서 수행되므로 인스턴스에 대한 모든 데이터베이스에 적용될 수 있습니다.

예를 들어 다음 코드는 모든 데이터 정의 언어(DDL) 명령의 끝에 현재 사용자를 인쇄하는 이벤트 트리거를 생성합니다.

```
CREATE OR REPLACE FUNCTION raise_notice_func()
    RETURNS event_trigger
    LANGUAGE plpgsql AS
$$
BEGIN
    RAISE NOTICE 'In trigger function: %', current_user;
END;
$$;
```

```
CREATE EVENT TRIGGER event_trigger_1
    ON ddl_command_end
EXECUTE PROCEDURE raise_notice_func();
```

PostgreSQL 이벤트 트리거에 대한 자세한 내용은 PostgreSQL 문서의 Event Triggers 단원을 참조하세요.

Amazon RDS에서 PostgreSQL 이벤트 트리거를 사용하는 데 대한 몇 가지 제한 사항이 있습니다. 여기에는 다음이 포함됩니다.

- 읽기 전용 복제본에 대한 이벤트 트리거를 생성할 수 없지만, 읽기 전용 복제 원본에 대한 이벤트 트리거는 생성할 수 있습니다. 그러면 이벤트 트리거가 읽기 전용 복제본으로 복사됩니다. 읽기 전용 복제본으로 복사된 이벤트 트리거는 변경 내용이 원본에서 푸시되더라도 읽기 전용 복제본에서 발생하지 않습니다. 하지만 읽기 전용 복제본이 승격되면 데이터베이스 작업이 있을 때 기존 이벤트 트리거가 발생합니다.
- 이벤트 트리거를 사용하는 PostgreSQL DB 인스턴스로의 메이저 버전 업그레이드를 수행하려면 인스턴스를 업그레이드하기 전에 이벤트 트리거를 삭제해야 합니다.

RDS for PostgreSQL용 방대한 페이지

방대한 페이지는 DB 인스턴스가 공유 버퍼에서 사용하는 것과 같은 대규모 연속 메모리 청크로 작업할 때 오버헤드를 줄이는 메모리 관리 기능입니다. 이 PostgreSQL 기능은 현재 사용 가능한 모든 RDS for PostgreSQL 버전에서 지원됩니다. 애플리케이션에 방대한 페이지를 할당하려면 `mmap` 또는 `SYSV` 공유 메모리 호출을 사용합니다. RDS for PostgreSQL은 4KB 페이지 크기와 2MB 페이지 크기를 모두 지원합니다.

`huge_pages` 파라미터의 값을 변경하여 방대한 페이지를 활성화하거나 비활성화할 수 있습니다. 이 기능은 마이크로, 스몰, 미디엄 DB 인스턴스 클래스를 제외한 모든 DB 인스턴스 클래스에 대해 기본적으로 사용 설정되어 있습니다.

RDS for PostgreSQL는 사용 가능한 공유 메모리를 기반으로 방대한 페이지를 사용합니다. DB 인스턴스가 공유 메모리 제약 때문에 방대한 페이지를 사용할 수 없을 경우 Amazon RDS는 DB 인스턴스 시작을 금지합니다. 이 경우 Amazon RDS는 DB 인스턴스 상태를 파라미터 호환 불가 상태로 설정합니다. 이렇게 되면 `huge_pages` 파라미터를 `off`로 설정하여 Amazon RDS가 DB 인스턴스를 시작하도록 허용할 수 있습니다.

`shared_buffers` 파라미터가 방대한 페이지를 사용하는 데 필요한 공유 메모리 풀을 설정하는 관건입니다. `shared_buffers` 파라미터의 기본값은 데이터베이스 파라미터 매크로를 사용합니다. 이 매

크로는 DB 인스턴스 메모리에서 사용 가능한 총 8KB 페이지 중 백분율을 설정합니다. 방대한 페이지를 사용할 때 이러한 페이지는 방대한 페이지에 할당됩니다. 공유 메모리 파라미터를 DB 인스턴스 메모리의 90% 이상을 요구하도록 설정할 경우 Amazon RDS가 DB 인스턴스를 파라미터 호환 장애 상태로 설정하므로 유의해야 합니다.

PostgreSQL 메모리 관리에 대한 자세한 내용은 PostgreSQL 설명서의 Resource Consumption(리소스 소비)을 참조하세요.

Amazon RDS for PostgreSQL에 대한 논리적 복제 수행

버전 10.4부터 RDS for PostgreSQL는 PostgreSQL 10에 도입된 게시 및 구독 SQL 구문을 지원합니다. 자세한 내용은 PostgreSQL의 논리적 복제 문서를 참조하세요.

> **Note**
>
> RDS for PostgreSQL은 PostgreSQL 10에 도입된 기본 PostgreSQL 논리적 복제 기능 외에 `pglogical` 확장도 지원합니다. 자세한 내용은 pglogical을 사용하여 인스턴스 간 데이터 동기화 섹션을 참조하세요.

다음에서 RDS for PostgreSQL DB 인스턴스의 논리적 복제 설정에 관한 정보를 확인할 수 있습니다.

주제

- 논리적 복제 및 논리적 디코딩 이해
- 논리적 복제 슬롯 작업
- 논리적 복제를 사용하여 테이블 수준 데이터 복제

논리적 복제 및 논리적 디코딩 이해

RDS for PostgreSQL은 PostgreSQL의 논리적 복제 슬롯을 사용하여 미리 쓰기 로그(WAL) 변경 사항 스트리밍을 지원합니다. 또한 논리적 디코딩 사용을 지원합니다. 인스턴스에 대한 논리적 복제 슬롯을 설정하고 해당 슬롯을 통해 데이터베이스 변경을 클라이언트(예: `pg_recvlogical`)에 스트리밍할 수 있습니다. 논리적 복제 슬롯은 데이터베이스 수준에서 생성되며 단일 데이터베이스에 대한 복제 연결을 지원합니다.

가장 일반적인 PostgreSQL 논리적 복제용 클라이언트는 Amazon EC2 인스턴스의 사용자 지정 관리형 호스트 또는 AWS Database Migration Service입니다. 논리적 복제 슬롯에는 스트림 수신기에 대한

정보가 없습니다. 또한 대상이 복제본 데이터베이스일 필요는 없습니다. 논리적 복제 슬롯을 설정하고 슬롯에서 데이터를 읽지 않을 경우 데이터가 DB 인스턴스의 스토리지에 기록되고 빠르게 가득 찰 수 있습니다.

Amazon RDS의 PostgreSQL 논리적 복제 및 논리적 디코딩은 파라미터, 복제 연결 유형 및 보안 역할에 의해 활성화됩니다. PostgreSQL DB 인스턴스의 데이터베이스에 대한 복제 연결을 설정할 수 있는 모든 클라이언트는 논리적 디코딩용 클라이언트가 될 수 있습니다.

RDS for PostgreSQL DB 인스턴스에 논리적 디코딩 사용 설정

1. 사용 중인 사용자 계정에 다음과 같은 역할이 있는지 확인합니다.
   - 논리 복제를 설정할 수 있는 `rds_superuser` 역할
   - 논리적 슬롯을 관리하고 논리적 슬롯을 사용하여 데이터를 스트리밍할 수 있는 권한을 부여하는 `rds_replication` 역할
2. `rds.logical_replication` 정적 파라미터를 1로 설정합니다. 이 파라미터를 적용하는 중에 `wal_level`, `max_wal_senders`, `max_replication_slots`, `max_connections` 파라미터 또한 설정합니다. 이러한 파라미터 변경은 WAL 생성을 강화하므로 논리적 슬롯을 사용할 때 `rds.logical_replication` 파라미터만 설정하면 됩니다.
3. 정적 `rds.logical_replication` 파라미터가 적용되도록 DB 인스턴스를 재부팅합니다.
4. 다음 섹션에 설명된 대로 논리적 복제 슬롯을 생성합니다. 이 프로세스에서는 디코딩 플러그인을 지정해야 합니다. 현재 RDS for PostgreSQL은 PostgreSQL과 함께 배송되는 test_decoding 및 wal2json 출력 플러그 인을 지원합니다.

PostgreSQL 논리적 디코딩에 대한 자세한 내용은 PostgreSQL 설명서를 참조하세요.

논리적 복제 슬롯 작업

SQL 명령을 사용하여 논리적 슬롯 작업을 수행할 수 있습니다. 예를 들어 다음 명령은 기본 PostgreSQL 출력 플러그인인 `test_slot`을 사용하여 `test_decoding` 논리적 슬롯을 생성합니다.

```
SELECT * FROM pg_create_logical_replication_slot('test_slot', 'test_decoding');
slot_name    | xlog_position
-----------------+---------------
regression_slot | 0/16B1970
(1 row)
```

논리적 슬롯 목록을 보려면 다음 명령을 사용합니다.

```
SELECT * FROM pg_replication_slots;
```

논리적 슬롯을 삭제하려면 다음 명령을 사용합니다.

```
SELECT pg_drop_replication_slot('test_slot');
pg_drop_replication_slot
-----------------------
(1 row)
```

논리적 복제 슬롯 작업에 대한 더 많은 예제는 PostgreSQL 문서의 Logical Decoding Examples를 참조하세요.

논리적 복제 슬롯을 생성한 후 스트리밍을 시작할 수 있습니다. 다음 예에서는 스트리밍 복제 프로토콜을 통해 논리적 디코딩을 제어하는 방법을 보여 줍니다. 이 예에서는 PostgreSQL 배포판에 포함된 pg_recvlogical 프로그램을 사용합니다. 이렇게 하려면 복제 연결을 허용하도록 클라이언트 인증을 설정해야 합니다.

```
pg_recvlogical -d postgres --slot test_slot -U postgres
    --host -instance-name.111122223333.aws-region.rds.amazonaws.com
    -f -  --start
```

`pg_replication_origin_status` 보기의 내용을 보려면 `pg_show_replication_origin_status` 함수를 쿼리합니다.

```
SELECT * FROM pg_show_replication_origin_status();
local_id | external_id | remote_lsn | local_lsn
----------+-------------+------------+-----------
(0 rows)
```

논리적 복제를 사용하여 테이블 수준 데이터 복제

RDS for PostgreSQL에서 논리적 복제를 사용하여 소스 테이블의 데이터를 대상 테이블로 복제할 수 있습니다. 논리적 복제는 먼저 소스 테이블에서 기존 데이터의 초기 로드를 수행한 후 진행 중인 변경 사항을 계속 복제합니다.

1. 소스 테이블 생성

   RDS for PostgreSQL DB 인스턴스의 소스 데이터베이스에 연결합니다.

```
source=> CREATE TABLE testtab (slno int primary key);
CREATE TABLE
```

2. 소스 테이블에 데이터 삽입

```
source=> INSERT INTO testtab VALUES (generate_series(1,1000));
INSERT 0 1000
```

3. 소스 테이블에 대한 게시 생성

   - 소스 테이블에 대한 게시를 생성합니다.

     ```
     source=> CREATE PUBLICATION testpub FOR TABLE testtab;
     CREATE PUBLICATION
     ```

   - SELECT 쿼리를 사용하여 생성된 게시의 세부 정보를 확인합니다.

     ```
     source=> SELECT * FROM pg_publication;
       oid   | pubname | pubowner | puballtables | pubinsert | pubupdate | pubdelete |
      pubtruncate | pubviaroot
     --------+---------+----------+--------------+-----------+-----------+-----------
     +-------------+------------
      115069 | testpub |    16395 | f            | t         | t         | t         |
      t           | f
     (1 row)
     ```

   - 소스 테이블이 게시에 추가되었는지 확인합니다.

     ```
     source=> SELECT * FROM pg_publication_tables;
     pubname | schemaname | tablename
     ---------+------------+-----------
      testpub | public     | testtab
     (1 rows)
     ```

- 데이터베이스의 모든 테이블을 복제하려면 다음을 사용합니다.

```
CREATE PUBLICATION testpub FOR ALL TABLES;
```

- 개별 테이블에 대한 게시가 이미 생성되어 있고 새 테이블을 추가해야 하는 경우 아래 쿼리를 실행하여 기존 게시에 새 테이블을 추가할 수 있습니다.

```
ALTER PUBLICATION <publication_name> add table <new_table_name>;
```

4. 대상 데이터베이스에 연결 및 대상 테이블 생성

- 대상 DB 인스턴스의 대상 데이터베이스에 연결합니다. 소스 테이블과 이름이 동일한 대상 테이블을 생성합니다.

```
target=> CREATE TABLE testtab (slno int primary key);
CREATE TABLE
```

- 대상 테이블에서 SELECT 쿼리를 실행하여 대상 테이블에 데이터가 없는지 확인합니다.

```
target=> SELECT count(*) FROM testtab;
 count
-------
     0
(1 row)
```

5. 대상 데이터베이스에서 구독 생성 및 확인

- 대상 데이터베이스에서 구독을 생성합니다.

```
target=> CREATE SUBSCRIPTION testsub
CONNECTION 'host=<source RDS/host endpoint> port=5432 dbname=<source_db_name>
 user=<user> password=<password>'
```

```
PUBLICATION testpub;
NOTICE:  Created replication slot "testsub" on publisher
CREATE SUBSCRIPTION
```

- SELECT 쿼리를 사용하여 구독이 활성화되어 있는지 확인합니다.

```
target=> SELECT oid, subname, subenabled, subslotname, subpublications FROM
 pg_subscription;
  oid  | subname | subenabled | subslotname | subpublications
-------+---------+------------+-------------+-----------------
 16434 | testsub | t          | testsub     | {testpub}
(1 row)
```

- 구독이 생성되면 소스 테이블의 모든 데이터가 대상 테이블로 로드됩니다. 대상 테이블에서 SELECT 쿼리를 실행하여 초기 데이터가 로드되는지 확인합니다.

```
target=> SELECT count(*) FROM testtab;
 count
-------
  1000
(1 row)
```

6. 소스 데이터베이스의 복제 슬롯 확인

대상 데이터베이스에서 구독을 생성하면 소스 데이터베이스에 복제 슬롯이 생성됩니다. 소스 데이터베이스에서 다음 SELECT 쿼리를 실행하여 복제 슬롯 세부 정보를 확인합니다.

```
source=> SELECT * FROM pg_replication_slots;

slot_name |  plugin  | slot_type | datoid | database | temporary | active |
 active_pid | xmin | catalog_xmin | restart_lsn | confirmed_flush_lsn | wal_status
 | safe_wal_size
----------+----------+-----------+--------+----------+-----------+--------
+------------+------+--------------+-------------+---------------------
+------------+---------------
testsub   | pgoutput | logical   | 115048 | source   | f         | t      |
 846 |      |         6945 | 58/B4000568 | 58/B40005A0         | reserved   |
```

```
(1 row)
```

7. 복제 테스트

   - 소스 테이블에 행을 삽입하여 소스 테이블의 데이터 변경 사항이 대상 테이블에 복제되는지 테스트합니다.

```
source=> INSERT INTO testtab VALUES(generate_series(1001,2000));
INSERT 0 1000

source=> SELECT count(*) FROM testtab;
 count
-------
  2000
(1 row)
```

   - 대상 테이블의 행 수를 확인하여 새 삽입 항목이 복제되는지 확인합니다.

```
target=> SELECT count(*) FROM testtab;
 count
-------
  2000
(1 row)
```

8. 테이블 추가 후 구독 새로 고침

   - 기존 게시에 새 테이블을 추가할 때 변경 사항을 적용하려면 구독을 새로 고쳐야 합니다.

```
ALTER SUBSCRIPTION <subscription_name> REFRESH PUBLICATION;
```

   - 이 명령은 게시자에서 누락된 테이블 정보를 가져오고 구독이 생성되거나 마지막으로 새로 고쳐진 이후 구독 대상 게시에 추가된 테이블에 대한 복제를 시작합니다.

stats_temp_directory에 대한 RAM 디스크

RDS for PostgreSQL의 파라미터 `rds.pg_stat_ramdisk_size`를 사용하여 PostgreSQL `stats_temp_directory` 저장용 RAM 디스크에 할당되는 시스템 메모리를 지정할 수 있습니다. RAM 디스크 파라미터는 RDS for PostgreSQL 버전 14 이하에서만 사용할 수 있습니다.

특정 워크로드에서 이 파라미터를 설정하면 성능이 향상되고 I/O 요구 사항이 감소될 수 있습니다. `stats_temp_directory`에 대한 자세한 내용은 PostgreSQL 문서 단원을 참조하세요.

`stats_temp_directory`에 대한 RAM 디스크를 설정하려면 DB 인스턴스에 사용되는 파라미터 그룹에서 `rds.pg_stat_ramdisk_size` 파라미터를 정수 리터럴 값으로 설정합니다. 이 파라미터는 MB를 나타내므로 정수 값을 사용해야 합니다. 표현식, 공식 및 함수는 `rds.pg_stat_ramdisk_size` 파라미터에 유효하지 않습니다. 변경 사항을 적용하려면 DB 인스턴스를 재부팅해야 합니다. 파라미터 설정에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹을 참조하세요.

예를 들어 다음 AWS CLI 명령은 RAM 디스크 파라미터를 256MB로 설정합니다.

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name pg-95-ramdisk-testing \
    --parameters "ParameterName=rds.pg_stat_ramdisk_size, ParameterValue=256,
 ApplyMethod=pending-reboot"
```

재부팅 후 다음 명령을 실행하여 `stats_temp_directory`의 상태를 확인합니다.

```
postgres=> SHOW stats_temp_directory;
```

명령은 다음을 반환합니다.

```
stats_temp_directory
----------------------------
/rdsdbramdisk/pg_stat_tmp
(1 row)
```

RDS for PostgreSQL용 테이블스페이스

RDS for PostgreSQL은 호환성을 위해 테이블스페이스를 지원합니다. 모든 스토리지가 단일 논리 볼륨에 있으므로 I/O 분할 또는 격리를 위해 테이블스페이스를 사용할 수 없습니다. Amazon의 벤치마크 결과와 실용적 경험에 따르면 단일 논리 볼륨이 대부분의 사용 사례에 최선의 설정입니다.

RDS for PostgreSQL DB 인스턴스로 테이블스페이스를 생성하고 사용하려면 rds_superuser 역할이 필요합니다. RDS for PostgreSQL DB 인스턴스의 기본 사용자 계정(기본 이름, postgres)은 이 역할의 멤버입니다. 자세한 정보는 PostgreSQL 역할 및 권한 이해의 내용을 참조하세요.

테이블스페이스를 생성할 때 파일 이름을 지정하는 경우, 경로 접두사는 /rdsdbdata/db/base/tablespace입니다. 다음 예제에서는 테이블스페이스 파일을 /rdsdbdata/db/base/tablespace/data에 배치합니다. 이 예에서는 dbadmin 사용자(역할)가 존재하고 테이블스페이스 작업에 필요한 rds_superuser 역할이 부여되었다고 가정합니다.

```
postgres=> CREATE TABLESPACE act_data
  OWNER dbadmin
  LOCATION '/data';
CREATE TABLESPACE
```

PostgreSQL 테이블스페이스에 대한 자세한 내용은 Tablespaces(테이블스페이스)를 참조하세요.

EBCDIC 및 기타 메인프레임 마이그레이션을 위한 RDS for PostgreSQL 데이터 정렬

PostgreSQL용 RDS 버전 10 이상에는 유니코드 10.0을 기반으로 하고 유니코드 공통 로케일 데이터 저리포지토리(CLDR 32)의 데이터 정렬이 포함된 ICU 버전 60.2가 포함되어 있습니다. 이러한 소프트웨어 국제화 라이브러리는 운영 체제나 플랫폼에 관계없이 문자 인코딩이 일관된 방식으로 표시되도록 합니다. 유니코드 CLDR-32에 대한 자세한 내용은 유니코드 CLDR 웹 사이트의 CLDR 32 릴리스 노트에서 확인할 수 있습니다. ICU 기술 위원회(ICU-TC) 웹 사이트에서 유니코드(ICU)의 국제화 구성 요소에 대해 자세히 알아볼 수 있습니다. ICU-60에 대한 정보는 ICU 60 다운로드를 참조하세요.

14.3 버전부터 PostgreSQL용 RDS에는 EBCDIC 기반 시스템에서 데이터 통합 및 변환을 지원하는 데이터 정렬도 포함됩니다. 확장 이진 코딩 십진 교환 코드 또는 EBCDIC 인코딩은 일반적으로 메인프레임 운영 체제에서 사용됩니다. Amazon RDS에서 제공하는 이러한 데이터 정렬은 EBCDIC 코드 페이지에 직접 매핑되는 유니코드 문자만 정렬하도록 좁게 정의되어 있습니다. 문자는 변환 후 데이터 유효성 검사가 가능하도록 EBCDIC 코드 포인트 순서로 정렬됩니다. 이러한 데이터 정렬에는 비정규화된 형식이 포함되지 않으며 소스 EBCDIC 코드 페이지의 문자에 직접 매핑되지 않는 유니코드 문자도 포함되지 않습니다.

EBCDIC 코드 페이지와 유니코드 코드 포인트 간의 문자 매핑은 IBM에서 게시한 테이블을 기반으로 합니다. 전체 세트는 다운로드용 압축 파일로 IBM에서 사용할 수 있습니다. PostgreSQL용 RDS는 ICU에서 제공하는 도구와 함께 이러한 매핑을 사용하여 이 섹션의 표에 나열된 데이터 정렬을 만들었습니다. 데이터 정렬 이름에는 ICU에서 요구하는 언어 및 국가가 포함됩니다. 그러나 EBCDIC 코드 페이지는 언어를 지정하지 않으며 일부 EBCDIC 코드 페이지는 여러 국가를 포함합니다. 따라서 테이블에 있

는 데이터 정렬 이름의 언어 및 국가 부분이 임의적이며 현재 로케일과 일치할 필요가 없습니다. 코드 페이지 번호는 이 표의 데이터 정렬 이름에서 가장 중요한 부분입니다. RDS for PostgreSQL 데이터베이스에서 다음 표에 나열된 모든 데이터 정렬을 사용할 수 있습니다.

- Unicode to EBCDIC collations table - 일부 메인프레임 데이터 마이그레이션 툴은 내부적으로 LATIN1 또는 LATIN9 을 사용하여 데이터를 인코딩하고 처리합니다. 이러한 도구는 왕복 방식을 사용하여 데이터 무결성을 유지하고 역변환을 지원합니다. 이 표의 데이터 정렬은 특별한 처리가 필요하지 않은 LATIN1 인코딩을 사용하여 데이터를 처리하는 도구에서 사용할 수 있습니다.
- Unicode to LATIN9 collations table - 모든 RDS for PostgreSQL 데이터베이스에서 이러한 데이터 정렬을 사용할 수 있습니다.

다음 표에는 EBCDIC 코드 페이지를 유니코드 코드 포인트에 매핑하는 RDS for PostgreSQL에서 사용할 수 있는 데이터 정렬이 나와 있습니다. IBM 코드 페이지의 순서를 기준으로 정렬해야 하는 애플리케이션 개발에는 이 테이블의 데이터 정렬을 사용하는 것이 좋습니다.

PostgreSQL 데이터 정렬 이름	코드 페이지 매핑 및 정렬 순서에 대한 설명
da-DK-cp277-x-icu	IBM EBCDIC 코드 페이지 277에 직접 매핑되는 유니코드 문자(변환 테이블당)는 IBM CP 277 코드 포인트 순서로 정렬됩니다.
de-DE-cp273-x-icu	IBM EBCDIC 코드 페이지 273에 직접 매핑되는 유니코드 문자(변환 테이블당)는 IBM CP 273 코드 포인트 순서로 정렬됩니다.
en-GB-cp285-x-icu	IBM EBCDIC 코드 페이지 285에 직접 매핑되는 유니코드 문자(변환 테이블당)는 IBM CP 285 코드 포인트 순서로 정렬됩니다.
en-US-cp037-x-icu	IBM EBCDIC 코드 페이지 037에 직접 매핑되는 유니코드 문자(변환 테이블당)는 IBM CP 37 코드 포인트 순서로 정렬됩니다.
es-ES-cp284-x-icu	IBM EBCDIC 코드 페이지 284에 직접 매핑되는 유니코드 문자(변환 테이블당)는 IBM CP 284 코드 포인트 순서로 정렬됩니다.

PostgreSQL 데이터 정렬 이름	코드 페이지 매핑 및 정렬 순서에 대한 설명
fi-FI-cp278-x-icu	IBM EBCDIC 코드 페이지 278에 직접 매핑되는 유니코드 문자(변환 테이블당)는 IBM CP 278 코드 포인트 순서로 정렬됩니다.
fr-FR-cp297-x-icu	IBM EBCDIC 코드 페이지 297에 직접 매핑되는 유니코드 문자(변환 테이블당)는 IBM CP 297 코드 포인트 순서로 정렬됩니다.
it-IT-cp280-x-icu	IBM EBCDIC 코드 페이지 280에 직접 매핑되는 유니코드 문자(변환 테이블당)는 IBM CP 280 코드 포인트 순서로 정렬됩니다.
nl-BE-cp500-x-icu	IBM EBCDIC 코드 페이지 500에 직접 매핑되는 유니코드 문자(변환 테이블당)는 IBM CP 500 코드 포인트 순서로 정렬됩니다.

Amazon RDS는 소스 데이터의 EBCDIC 코드 페이지에 따라 원래 코드 포인트의 순서로 IBM에서 게시한 테이블을 사용하여 LATIN9 문자에 매핑되는 유니코드 코드 포인트를 정렬하는 추가 데이터 정렬 세트를 제공합니다.

PostgreSQL 데이터 정렬 이름	코드 페이지 매핑 및 정렬 순서에 대한 설명
da-DK-cp1142b-x-icu	IBM EBCDIC 코드 페이지 1142(변환 테이블당)에서 원래 변환된 LATIN9 문자에 매핑되는 유니코드 문자는 IBM CP 1142 코드 포인트 순서로 정렬됩니다.
de-DE-cp1141m-x-icu	IBM EBCDIC 코드 페이지 1141(변환 테이블당)에서 원래 변환된 LATIN9 문자에 매핑되는 유니코드 문자는 IBM CP 1141 코드 포인트 순서로 정렬됩니다.
en-GB-cp1146m-x-icu	IBM EBCDIC 코드 페이지 1146(변환 테이블당)에서 원래 변환된 LATIN9 문자에 매핑되는 유니

PostgreSQL 데이터 정렬 이름	코드 페이지 매핑 및 정렬 순서에 대한 설명
	코드 문자는 IBM CP 1146 코드 포인트 순서로 정렬됩니다.
en-US-cp1140m-x-icu	IBM EBCDIC 코드 페이지 1140(변환 테이블당)에서 원래 변환된 LATIN9 문자에 매핑되는 유니코드 문자는 IBM CP 1140 코드 포인트 순서로 정렬됩니다.
es-ES-cp1145m-x-icu	IBM EBCDIC 코드 페이지 1145(변환 테이블당)에서 원래 변환된 LATIN9 문자에 매핑되는 유니코드 문자는 IBM CP 1145 코드 포인트 순서로 정렬됩니다.
fi-FI-cp1143m-x-icu	IBM EBCDIC 코드 페이지 1143(변환 테이블당)에서 원래 변환된 LATIN9 문자에 매핑되는 유니코드 문자는 IBM CP 1143 코드 포인트 순서로 정렬됩니다.
fr-FR-cp1147m-x-icu	IBM EBCDIC 코드 페이지 1147(변환 테이블당)에서 원래 변환된 LATIN9 문자에 매핑되는 유니코드 문자는 IBM CP 1147 코드 포인트 순서로 정렬됩니다.
it-IT-cp1144m-x-icu	IBM EBCDIC 코드 페이지 1144(변환 테이블당)에서 원래 변환된 LATIN9 문자에 매핑되는 유니코드 문자는 IBM CP 1144 코드 포인트 순서로 정렬됩니다.
nl-BE-cp1148m-x-icu	IBM EBCDIC 코드 페이지 1148(변환 테이블당)에서 원래 변환된 LATIN9 문자에 매핑되는 유니코드 문자는 IBM CP 1148 코드 포인트 순서로 정렬됩니다.

RDS for PostgreSQL 데이터 정렬 사용 예를 찾아볼 수 있습니다.

```
db1=> SELECT pg_import_system_collations('pg_catalog');
```

```
 pg_import_system_collations
-----------------------------
                            36
db1=> SELECT '¤' < 'a' col1;
 col1
------
 t
db1=> SELECT '¤' < 'a' COLLATE "da-DK-cp277-x-icu" col1;
 col1
------
 f
```

IBM 코드 페이지의 순서를 기반으로 정렬해야 하는 애플리케이션 개발의 경우 Unicode to EBCDIC collations table 및 Unicode to LATIN9 collations table의 데이터 정렬을 사용하는 것이 좋습니다. 다음 데이터 정렬(문자 "b" 접미사)도 pg_collation에서 볼 수 있지만 특정 코드 포인트 이동이 있는 코드 페이지를 매핑하고 데이터 정렬에서 특별한 처리가 필요한 AWS의 메인프레임 데이터 통합 및 마이그레이션 도구에서 사용하기 위한 것입니다. 즉, 다음과 같은 데이터 정렬 데이터 정렬 방식은 권장되지 않습니다.

- da-DK-cp1142b-x-icu
- da-DK-cp1142b-x-icu
- de-DE-cp273b-x-icu
- de-DE-cp1141b-x-icu
- en-GB-cp1146b-x-icu
- en-GB-cp285b-x-icu
- en-US-cp037b-x-icu
- en-US-cp1140b-x-icu
- es-ES-cp1145b-x-icu
- es-ES-cp284b-x-icu
- fi-FI-cp1143b-x-icu
- fr-FR-cp1147b-x-icu
- fr-FR-cp297b-x-icu
- it-IT-cp1144b-x-icu
- it-IT-cp280b-x-icu
- nl-BE-cp1148b-x-icu

- nl-BE-cp500b-x-icu

메인프레임 환경에서 AWS로 애플리케이션을 마이그레이션하는 방법에 대한 자세한 내용은 AWS Mainframe Modernization 무엇입니까?를 참조하세요.

PostgreSQL의 데이터 정렬에 대한 자세한 내용은 PostgreSQL 설명서의 데이터 정렬 지원을 참조하세요.

RDS for PostgreSQL에 대한 논리적 슬롯 동기화 관리

커뮤니티 PostgreSQL 17부터 실행 시 슬롯을 수동으로 동기화하는 파라미터 `sync_replication_slots` 또는 관련 함수 `pg_sync_replication_slots()`를 통해 기본 서버에서 대기 서버로 논리적 복제 슬롯을 자동으로 동기화하는 새로운 기능이 도입되었습니다.

이러한 기능은 RDS for PostgreSQL 17부터 사용할 수 있습니다. 일반적인 설정에는 기본 인스턴스와 읽기 전용 복제본 및 프라이머리 인스턴스에 대한 논리적 복제 구독자가 포함됩니다.

장애 조치 옵션이 true로 설정된 상태에서 구독이 생성되었는지 확인합니다.

```
CREATE SUBSCRIPTION subname CONNECTION 'host=...' PUBLICATION pubname WITH (failover =
 true);
```

이렇게 하면 장애 조치가 활성화된 상태로 게시자에 논리적 슬롯이 생성됩니다.

```
postgres=> SELECT slot_name, slot_type, failover FROM pg_catalog.pg_replication_slots;
 slot_name | slot_type | failover
-----------+-----------+----------
 subname   | logical   | t
(1 row)
```

슬롯 동기화를 활성화하면 프라이머리 인스턴스의 모든 장애 조치 논리적 복제 슬롯이 물리적 대기에 자동으로 생성되고 주기적으로 동기화됩니다. 파라미터 그룹을 통해 다음 값이 설정되었는지 확인합니다.

- 논리적 복제를 활성화하려면 `rds.logical_replication`이 1이어야 합니다.
- 대기 상태에서 `hot_standby_feedback`이 1이어야 합니다.
- 대기 상태의 `rds.logical_slot_sync_dbname`을 유효한 데이터베이스 이름으로 설정해야 합니다.

파라미터의 기본값은 postgres입니다. 논리적 게시 인스턴스에 postgres 데이터베이스가 있는 경우 기본 파라미터를 변경할 필요가 없습니다.

- 프라이머리 인스턴스의 synchronized_standby_slots는 동기화되도록 의도된 대기의 물리적 복제 슬롯으로 설정되어야 합니다.
- 자동 동기화를 활성화하려면 sync_replication_slots가 1이어야 합니다.

장애 조치 활성화 구독 슬롯과 위의 파라미터 값을 사용하면 대기가 승격될 때 구독자는 새로 승격된 이 인스턴스에 대한 구독을 변경하고 논리적 복제를 원활하게 계속할 수 있습니다.

PostgreSQL 데이터베이스 엔진을 실행하는 DB 인스턴스에 연결

Amazon RDS가 DB 인스턴스를 프로비저닝한 후에는 표준 SQL 클라이언트 애플리케이션을 사용해 인스턴스에 연결할 수 있습니다. 연결하려면 먼저 DB 인스턴스를 사용할 수 있고 액세스할 수 있어야 합니다. VPC 외부에서 인스턴스에 연결할 수 있는지 여부는 Amazon RDS DB 인스턴스를 생성한 방법에 따라 달라집니다.

- DB 인스턴스를 퍼블릭으로 생성한 경우 VPC 외부의 디바이스 및 Amazon EC2 인스턴스를 데이터베이스에 연결할 수 있습니다.
- DB 인스턴스를 프라이빗으로 생성한 경우 Amazon VPC 내의 Amazon EC2 인스턴스 및 디바이스만 데이터베이스에 연결할 수 있습니다.

DB 인스턴스가 퍼블릭인지 프라이빗인지 확인하려면 AWS Management Console을 사용하여 인스턴스의 연결 & 보안(Connectivity & security) 탭을 봅니다. 보안(Security) 아래에서 "퍼블릭 액세스 가능(Publicly accessible)" 값이 아니요(No)이면 프라이빗이고 예(Yes)이면 퍼블릭입니다.

다양한 Amazon RDS 및 Amazon VPC 구성과 이러한 구성이 접근성에 미치는 영향에 대한 자세한 내용은 VPC에서 DB 인스턴스에 액세스하는 시나리오 섹션을 참조하세요.

목차

- psql 클라이언트 설치
- RDS for PostgreSQL DB 인스턴스에 대한 연결 정보 찾기
- pgAdmin을 사용하여 RDS for PostgreSQL DB 인스턴스에 연결
- psql을 사용하여 RDS for PostgreSQL DB 인스턴스에 연결
- Amazon Web Services(AWS) JDBC 드라이버를 사용하여 RDS for PostgreSQL에 연결
- Amazon Web Services(AWS) Python 드라이버를 사용하여 RDS for PostgreSQL에 연결
- RDS for PostgreSQL 인스턴스에 대한 연결 문제 해결
  - Error – FATAL: database name does not exist
  - Error – Could not connect to server: Connection timed out
  - 보안 그룹 액세스 규칙 오류

psql 클라이언트 설치

EC2 인스턴스에서 DB 인스턴스에 연결하려면 EC2 인스턴스에 PostgreSQL 클라이언트를 설치하면 됩니다. 최신 버전의 psql 클라이언트를 Amazon Linux 2023에 설치하려면 다음 명령을 실행하세요.

```
sudo dnf install postgresql<version number>
```

최신 버전의 psql 클라이언트를 Amazon Linux 2에 설치하려면 다음 명령을 실행하세요.

```
sudo yum install -y postgresql
```

최신 버전의 psql 클라이언트를 Ubuntu에 설치하려면 다음 명령을 실행하세요.

```
sudo apt install -y postgresql-client
```

RDS for PostgreSQL DB 인스턴스에 대한 연결 정보 찾기

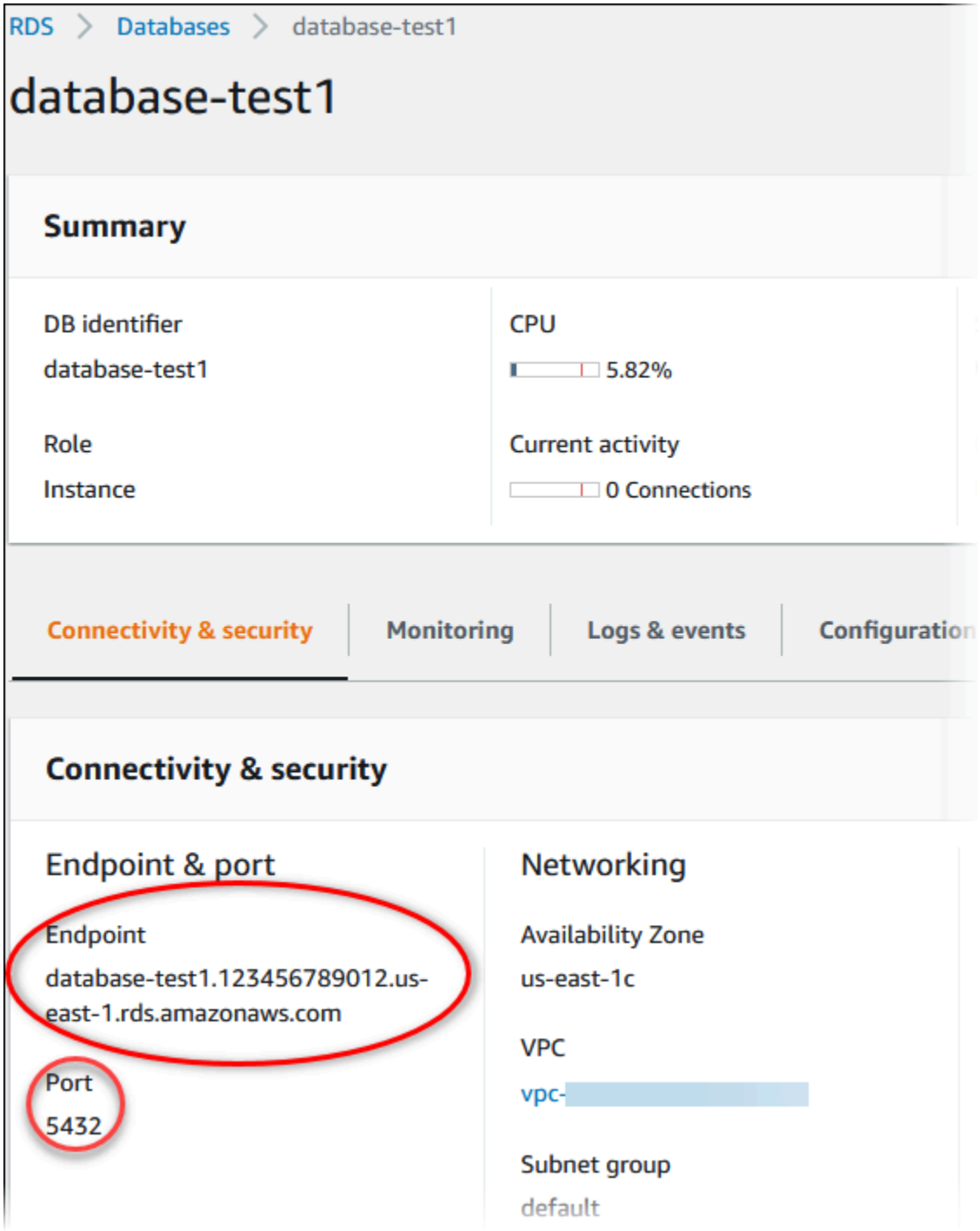
DB 인스턴스를 사용할 수 있고 액세스할 수 있는 경우 SQL 클라이언트 애플리케이션에 다음 정보를 제공하여 연결할 수 있습니다.

- 인스턴스의 호스트 이름(DNS 이름) 역할을 하는 DB 인스턴스 엔드포인트입니다.
- DB 인스턴스 서버가 수신 대기하는 포트입니다. PostgreSQL의 경우 기본 포트는 5432입니다.
- DB 인스턴스의 사용자 이름 및 암호입니다. 기본으로 설정된 PostgreSQL의 '기본 사용자 이름'은 `postgres`입니다.
- 데이터베이스의 이름 및 암호입니다(DB 이름).

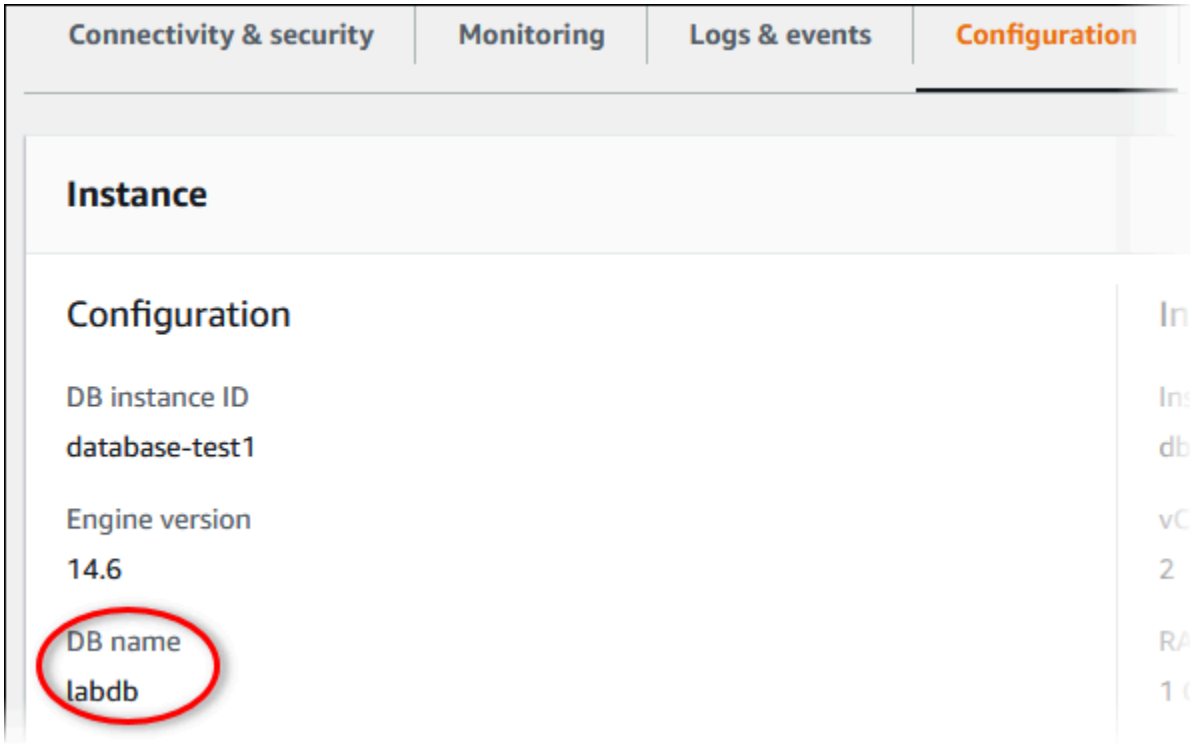
이러한 세부 정보는 AWS Management Console, AWS CLI describe-db-instances 명령 또는 Amazon RDS API DescribeDBInstances 작업을 사용하여 확인할 수 있습니다.

AWS Management Console을 사용하여 엔드포인트, 포트 번호, DB 이름을 찾는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. RDS 콘솔을 연 다음 데이터베이스를 선택해 DB 인스턴스의 목록을 표시합니다.
3. 세부 정보를 표시하고자 하는 PostgreSQL DB 인스턴스 이름을 선택합니다.
4. Connectivity & security(연결 및 보안) 탭에서 엔드포인트를 복사합니다. 또한 포트 번호를 적어 둡니다. DB 인스턴스에 연결하려면 엔드포인트와 포트 번호가 모두 필요합니다.



5. 구성 탭에 나와 있는 DB 이름을 적어 둡니다. RDS for PostgreSQL 인스턴스를 만들 때 데이터베이스를 생성한 경우 DB 이름 아래에 이름이 표시됩니다. 데이터베이스를 만들지 않은 경우 DB 이름에 대시(-)가 표시됩니다.



다음은 PostgreSQL DB 인스턴스에 연결하는 두 가지 방법입니다. 첫 번째 예에서는 PostgreSQL의 잘 알려진 오픈 소스 관리 및 개발 도구인 pgAdmin을 사용합니다. 두 번째 예에서는 PostgreSQL 설치에 속하는 명령행 유틸리티인 psql을 사용합니다.

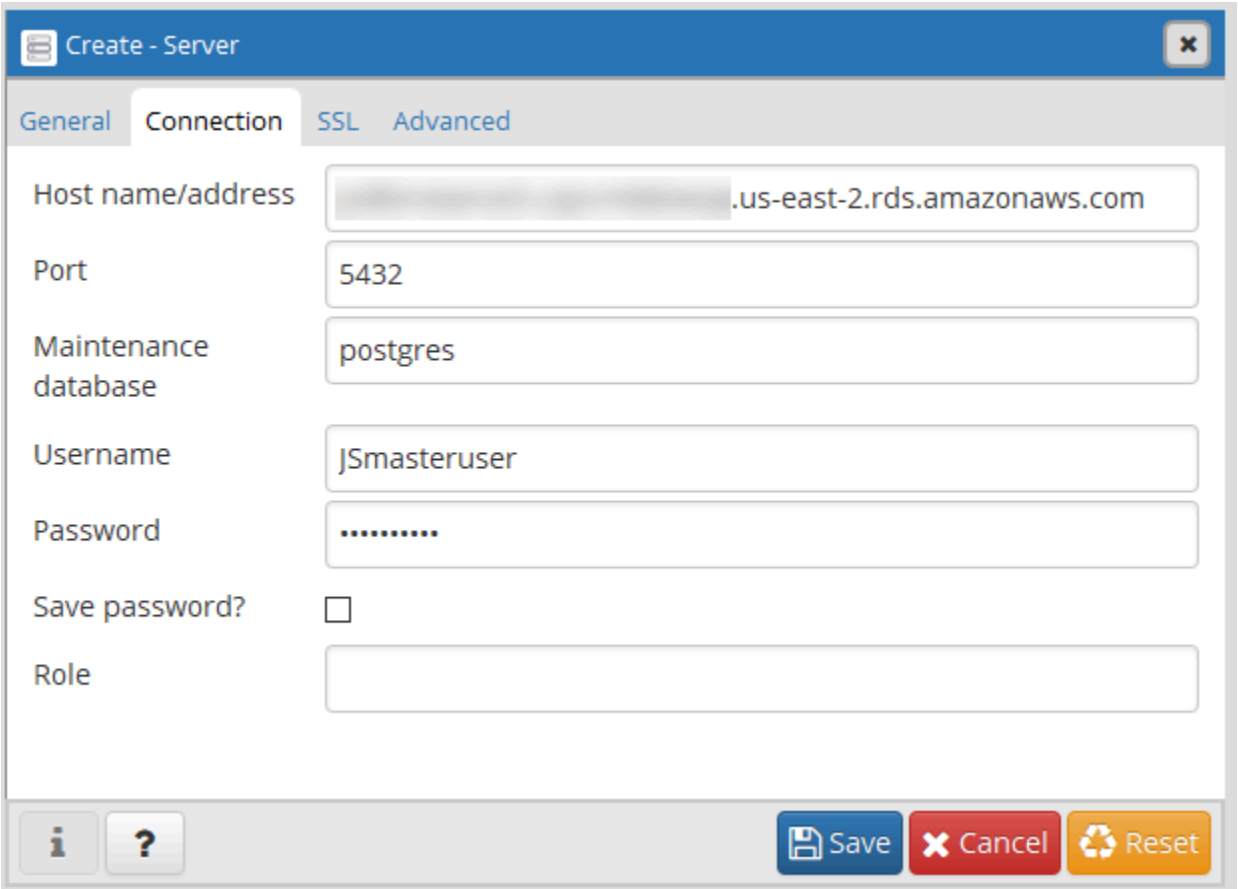
pgAdmin을 사용하여 RDS for PostgreSQL DB 인스턴스에 연결

오픈 소스 도구인 pgAdmin을 사용하여 RDS for PostgreSQL DB 인스턴스에 연결할 수 있습니다. 클라이언트 컴퓨터에 로컬 PostgreSQL 인스턴스가 없어도 http://www.pgadmin.org/에서 pgAdmin을 다운로드하여 설치할 수 있습니다.

pgAdmin을 사용하여 RDS for PostgreSQL DB 인스턴스에 연결하는 방법

1. 클라이언트 컴퓨터에서 pgAdmin 애플리케이션을 실행합니다.
2. [Dashboard] 탭에서 [Add New Server]를 선택합니다.
3. [Create - Server] 대화 상자에서 pgAdmin의 서버를 식별하기 위해 [General] 탭에 이름을 입력합니다.
4. [Connection] 탭에서 사용자 DB 인스턴스에 있는 다음 정보를 입력합니다.
   - Host(호스트)에 엔드포인트를 입력합니다(예: `mypostgresql.c6c8dntfzzhgv0.us-east-2.rds.amazonaws.com`).

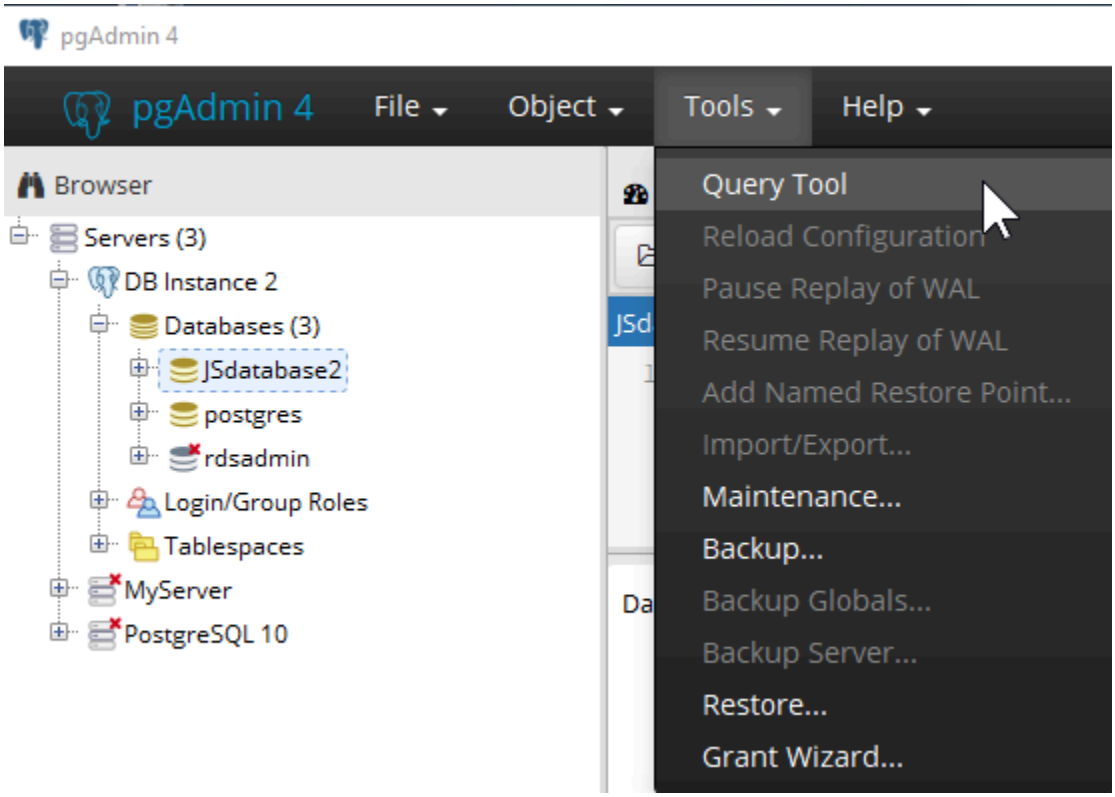
- 포트에 할당된 포트를 입력합니다.
- 사용자 이름(Username)에 DB 인스턴스를 생성할 때 입력한 사용자 이름을 입력합니다('기본 사용자 이름'을 기본값에서 변경한 경우 `postgres`).
- 암호에 DB 인스턴스를 생성할 때 입력했던 암호를 입력합니다.



5. Save(저장)를 선택합니다.

   연결 문제는 RDS for PostgreSQL 인스턴스에 대한 연결 문제 해결를 참조하십시오.

6. pgAdmin 브라우저에서 데이터베이스에 액세스 하려면, [Servers], DB 인스턴스, [Databases]를 확장합니다. DB 인스턴스의 데이터베이스 이름을 선택합니다.



7. SQL 명령을 입력할 수 있는 패널을 열려면, 도구, 쿼리 도구를 선택합니다.

psql을 사용하여 RDS for PostgreSQL DB 인스턴스에 연결

psql 명령줄 유틸리티의 로컬 인스턴스를 사용하여 RDS for PostgreSQL DB 인스턴스에 연결할 수 있습니다. 클라이언트 컴퓨터에 PostgreSQL 또는 psql 클라이언트를 설치해야 할 수도 있습니다.

PostgreSQL 웹 사이트에서 PostgreSQL 클라이언트를 다운로드할 수 있습니다. psql을 설치하려면 운영 체제 버전별 지침을 따르세요.

psql을 사용하여 RDS for PostgreSQL DB 인스턴스에 연결하려면 호스트(DNS) 정보, 액세스 자격 증명 및 데이터베이스 이름을 제공해야 합니다.

다음 형식 중 하나를 사용하여 RDS for PostgreSQL DB 인스턴스에 연결합니다. 연결이 되면 암호를 입력하라는 메시지가 표시됩니다. 배치 작업이나 스크립트에는 `--no-password` 옵션을 사용합니다. 이 옵션은 전체 세션에 대해 설정됩니다.

> **Note**
>
> 서버에 암호 인증이 필요하고 다른 소스에서 암호를 사용할 수 없는 경우 `--no-password`를 사용한 연결 시도가 실패합니다. 자세한 내용은 psql 설명서를 참조하세요.

이 DB 인스턴스에 처음 연결하는 경우 또는 RDS for PostgreSQL 인스턴스에 대한 데이터베이스를 아직 생성하지 않은 경우 '기본 사용자 이름' 및 암호를 사용하여 postgres 데이터베이스에 연결할 수 있습니다.

Unix는 다음 형식을 사용합니다.

```
psql \
   --host=<DB instance endpoint> \
   --port=<port> \
   --username=<master username> \
   --password \
   --dbname=<database name>
```

Windows는 다음 형식을 사용합니다.

```
psql ^
   --host=<DB instance endpoint> ^
   --port=<port> ^
   --username=<master username> ^
   --password ^
   --dbname=<database name>
```

예를 들어 다음 명령은 가상 자격 증명을 사용해 `mypgdb`이라는 PostgreSQL DB 인스턴스에서 `mypostgresql`라는 데이터베이스에 연결합니다.

```
psql --host=mypostgresql.c6c8mwvfdgv0.us-west-2.rds.amazonaws.com --port=5432 --
username=awsuser --password --dbname=mypgdb
```

Amazon Web Services(AWS) JDBC 드라이버를 사용하여 RDS for PostgreSQL에 연결

Amazon Web Services(AWS) JDBC 드라이버는 고급 JDBC 래퍼로 설계되었습니다. 이 래퍼는 기존 JDBC 드라이버의 기능을 보완하고 확장합니다. 이 드라이버는 커뮤니티 pgJDBC 드라이버와 드롭인 호환됩니다.

AWS JDBC 드라이버를 설치하려면 CLASSPATH 애플리케이션에 있는 AWS JDBC 드라이버 .jar 파일을 추가하고 해당 커뮤니티 드라이버에 대한 참조를 보관해 두세요. 다음과 같이 해당 연결 URL 접두사를 업데이트하세요.

- `jdbc:postgresql://~jdbc:aws-wrapper:postgresql://`

AWS JDBC 드라이버에 대한 자세한 내용 및 사용 방법에 대한 전체 지침은 Amazon Web Services (AWS) JDBC Driver GitHub repository를 참조하세요.

Amazon Web Services(AWS) Python 드라이버를 사용하여 RDS for PostgreSQL에 연결

Amazon Web Services(AWS) Python 드라이버는 고급 Python 래퍼로 설계되었습니다. 이 래퍼는 오픈 소스 Psycopg 드라이버의 기능을 보완하고 확장합니다. AWS Python 드라이버는 Python 버전 3.8 이상을 지원합니다. `pip` 명령을 사용하여 `psycopg` 오픈 소스 패키지와 함께 `aws-advanced-python-wrapper` 패키지를 설치할 수 있습니다.

AWS Python 드라이버에 대한 자세한 내용 및 사용 방법에 대한 전체 지침은 Amazon Web Services (AWS) JDBC Python GitHub repository를 참조하세요.

RDS for PostgreSQL 인스턴스에 대한 연결 문제 해결

주제

- Error – FATAL: database name does not exist
- Error – Could not connect to server: Connection timed out
- 보안 그룹 액세스 규칙 오류

Error – FATAL: database *name* does not exist

연결할 때 `FATAL: database name does not exist` 같은 오류가 발생하면 `--dbname` 옵션에 기본 데이터베이스 이름 postgres를 사용해봅니다.

Error – Could not connect to server: Connection timed out

DB 인스턴스에 연결할 수 없도록 만드는 가장 많은 오류가 `Could not connect to server: Connection timed out.`입니다. 이 오류 메시지가 표시되면 다음을 확인합니다.

- 사용한 호스트 이름이 DB 인스턴스 엔드포인트이고 포트 번호가 올바른지 확인하십시오.
- 외부 연결을 허용하도록 DB 인스턴스의 퍼블릭 액세스 가능성이 예(Yes)로 설정되어 있는지 확인합니다. 퍼블릭 액세스(Public access) 설정을 수정하려면 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.
- 데이터베이스에 연결하는 사용자에게 CONNECT 액세스 권한이 있는지 확인합니다. 다음 쿼리를 사용하여 데이터베이스에 대한 연결 액세스를 제공할 수 있습니다.

```
GRANT CONNECT ON DATABASE database name TO username;
```

- DB 인스턴스에 할당된 보안 그룹에 모든 연결이 통과할 수 있는 방화벽을 통한 액세스를 허용하는 데 필요한 규칙이 있는지 확인하십시오. 기본값 포트 5432를 사용해 DB 인스턴스를 생성했는데, 기업 방화벽 규칙이 외부 회사 디바이스의 해당 포트 연결을 차단하는 경우를 예로 들 수 있습니다.

  이 오류를 수정하려면 다른 포트를 사용하도록 DB 인스턴스를 수정해야 합니다. 또 DB 인스턴스에 적용된 보안 그룹의 새로운 포트에 대한 연결이 허용되어야 합니다. 데이터베이스 포트(Database port) 설정을 수정하려면 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.
- 사용하려는 포트가 PostgreSQL의 로컬 인스턴스 또는 컴퓨터에서 실행되는 다른 서비스에 의해 이미 사용 중인지 확인합니다. 예를 들어 동일한 포트(기본값은 5432)에서 실행 중인 로컬 PostgreSQL 데이터베이스가 있는 경우 RDS for PostgreSQL DB 인스턴스에 성공적으로 연결되지 않을 수 있습니다. 포트가 사용 가능한지 확인하거나 가능하면 다른 포트 번호로 연결을 시도합니다.
- 또한 보안 그룹 액세스 규칙 오류 단원도 참조하십시오.

보안 그룹 액세스 규칙 오류

지금까지 가장 많이 발생하고 있는 연결 문제는 DB 인스턴스에 할당된 보안 그룹의 액세스 규칙과 관련이 있습니다. DB 인스턴스를 만들 때 기본 보안 그룹을 사용한 경우 인스턴스에 액세스할 수 있도록 허용하는 규칙이 보안 그룹에 없을 확률이 큽니다.

연결이 되도록 만들려면, 생성 때 DB 인스턴스에 할당한 보안 그룹이 DB 인스턴스 액세스를 허용해야 합니다. 예를 들어, DB 인스턴스가 VPC 내부에서 생성된 경우 이 인스턴스의 VPC 보안 그룹에서 연결 권한을 부여해야 합니다. 애플리케이션이 실행되고 있는 디바이스 또는 Amazon EC2 인스턴스의 연결을 승인하지 않는 보안 그룹을 사용하여 DB 인스턴스를 생성했는지 여부를 확인합니다.

보안 그룹에서 인바운드 규칙을 추가하거나 편집할 수 있습니다. 소스로 내 IP를 선택하면 브라우저에서 감지된 IP 주소에서 DB 인스턴스에 액세스할 수 있습니다. 자세한 내용은 보안 그룹을 생성하여 VPC 내부의 DB 인스턴스에 대한 액세스를 제공 섹션을 참조하세요.

또한, DB 인스턴스가 VPC 외부에서 생성된 경우에는 이 인스턴스의 데이터베이스 보안 그룹에서 이 연결을 승인해야 합니다.

Amazon RDS 보안 그룹에 대한 자세한 내용은 보안 그룹을 통한 액세스 제어 단원을 참조하십시오.

SSL/TLS를 사용한 RDS for PostgreSQL 연결 보안

RDS for PostgreSQL은 PostgreSQL DB 인스턴스를 위한 SSL 암호화를 지원합니다. SSL을 사용하여 애플리케이션과 PostgreSQL DB 인스턴스 사이의 PostgreSQL 연결을 암호화할 수 있습니다. 또한 PostgreSQL DB 인스턴스에 대한 모든 연결에서 SSL을 사용하도록 지정할 수도 있습니다. 또한 RDS for PostgreSQL은 SSL에 대한 후속 프로토콜인 전송 계층 보안(TLS)을 지원합니다.

SSL/TLS를 사용하는 연결 암호화 등 Amazon RDS 및 데이터 보호에 대한 자세한 내용은 Amazon RDS의 데이터 보호 섹션을 참조하세요.

주제

- PostgreSQL DB 인스턴스와 함께 SSL 사용
- 새 SSL/TLS 인증서를 사용해 PostgreSQL DB 인스턴스에 연결할 애플리케이션 업데이트

PostgreSQL DB 인스턴스와 함께 SSL 사용

Amazon RDS는 PostgreSQL DB 인스턴스를 위한 SSL 암호화를 지원합니다. SSL을 사용하여 애플리케이션과 PostgreSQL DB 인스턴스 사이의 PostgreSQL 연결을 암호화할 수 있습니다. 기본적으로 RDS for PostgreSQL는 모든 클라이언트가 SSL/TLS를 사용하여 연결하기를 기대하지만 사용자가 이를 요구할 수도 있습니다. RDS for PostgreSQL은 전송 계층 보안(TLS) 버전 1.1, 1.2 및 1.3을 지원합니다.

SSL 지원 및 PostgreSQL 데이터베이스에 대한 일반적인 정보는 PostgreSQL 설명서의 SSL 지원을 참조하세요. JDBC를 통한 SSL 연결 사용에 대한 자세한 내용은 PostgreSQL 설명서에서 클라이언트 구성을 참조하세요.

모든 AWS 리전에서 PostgreSQL에 대한 SSL 지원 기능을 사용할 수 있습니다. Amazon RDS는 PostgreSQL DB 인스턴스가 생성될 때 해당 인스턴스의 SSL 인증서를 만듭니다. SSL 인증서 확인을 활성화하는 경우에는 SSL 인증서에 스푸핑 공격으로부터 보호해주는 SSL 인증서를 위한 일반 이름(CN)으로 DB 인스턴스 엔드포인트가 포함됩니다.

주제

- SSL을 통해 PostgreSQL DB 인스턴스에 연결
- PostgreSQL DB 인스턴스에 SSL 연결 요구
- SSL 연결 상태 확인
- RDS for PostgreSQL의 SSL 암호 제품군

SSL을 통해 PostgreSQL DB 인스턴스에 연결

SSL을 통해 PostgreSQL DB 인스턴스에 연결하려면

1. 인증서를 다운로드합니다.

   인증서 다운로드에 대한 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 단원을 참조하세요.
2. SSL을 통해 PostgreSQL DB 인스턴스에 연결합니다.

   SSL을 사용하여 연결하면 클라이언트가 인증서 체인을 확인할지 선택할 수 있습니다. 연결 파라미터가 `sslmode=verify-ca` 또는 `sslmode=verify-full`을 지정하면, 클라이언트는 RDS CA 인증서를 트러스트 스토어에 있거나 연결 URL에서 참조되게 할 것을 요구합니다. 이 요구 사항은 데이터베이스 인증서에 서명하는 인증서 체인을 확인하는 것입니다.

   psql 또는 JDBC와 같은 클라이언트가 SSL을 지원하도록 구성되어 있는 경우 클라이언트는 먼저 SSL을 이용해 데이터베이스에 연결을 시도하도록 기본 설정되어 있습니다. SSL을 이용해 연결할 수 없는 경우 클라이언트는 SSL 없이 연결하는 방식으로 전환됩니다. libpq 기반 클라이언트(예: psql)와 JDBC에서 사용되는 기본 `sslmode` 모드는 서로 다릅니다. libpq 기반 클라이언트와 JDBC 클라이언트는 `prefer`로 기본 설정되어 있습니다.

   `sslrootcert` 파라미터를 사용하여 인증서를 참조합니다(예: `sslrootcert=rds-ssl-ca-cert.pem`).

다음은 `psql`을 사용하여 인증서 확인과 함께 SSL을 사용하여 PostgreSQL DB 인스턴스에 연결하는 예입니다.

```
$ psql "host=db-name.555555555555.ap-southeast-1.rds.amazonaws.com
    port=5432 dbname=testDB user=testuser sslrootcert=rds-ca-rsa2048-g1.pem
 sslmode=verify-full"
```

PostgreSQL DB 인스턴스에 SSL 연결 요구

`rds.force_ssl` 파라미터를 사용하여 PostgreSQL DB 인스턴스에 대한 연결이 SSL을 사용하도록 요구할 수 있습니다. RDS for PostgreSQL 버전 15 이상의 `rds.force_ssl` 파라미터 기본값은 1(켜짐)입니다. 다른 모든 RDS for PostgreSQL 메이저 버전 14 이상에는 이 파라미터의 기본값이 0(꺼짐)입니다. `rds.force_ssl` 파라미터를 1(설정)로 설정하면 해당 DB 클러스터에 대한 연결 시 SSL/TLS를 요구합니다. `rds.force_ssl` 파라미터를 1(설정)로 설정하면 해당 DB 인스턴스에 대한 연결에 대해 SSL을 요구합니다.

이 파라미터의 값을 변경하려면 사용자 지정 DB 파라미터 그룹을 생성해야 합니다. 그런 다음 사용자 지정 DB 파라미터 그룹의 rds.force_ssl 값을 1로 변경하여 이 기능을 켭니다. RDS for PostgreSQL DB 인스턴스를 생성하기 전에 사용자 지정 DB 파라미터 그룹을 준비하는 경우 생성 과정 중에 기본 파라미터 그룹 대신 선택할 수 있습니다. RDS for PostgreSQL DB 인스턴스가 이미 실행된 후에 이 작업을 수행하는 경우 인스턴스가 사용자 지정 파라미터 그룹을 사용하도록 인스턴스를 재부팅해야 합니다. 자세한 정보는 Amazon RDS의 파라미터 그룹의 내용을 참조하세요.

DB 인스턴스에서 rds.force_ssl 기능이 활성화된 경우 SSL을 사용하지 않는 연결 시도는 다음 메시지와 함께 거부됩니다.

```
$ psql -h db-name.555555555555.ap-southeast-1.rds.amazonaws.com port=5432 dbname=testDB
 user=testuser
psql: error: FATAL: no pg_hba.conf entry for host "w.x.y.z", user "testuser", database
 "testDB", SSL off
```

SSL 연결 상태 확인

DB 인스턴스에 연결할 때 로그온 배너에 연결의 암호화된 상태가 표시됩니다.

```
Password for user master:
psql (10.3)
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.
postgres=>
```

또한, sslinfo 확장을 로드한 다음 ssl_is_used() 함수를 호출하여 SSL이 사용 중인지 확인할 수 있습니다. 이 함수는 연결이 SSL을 사용할 경우 t을 반환하고, 그렇지 않으면 f를 반환합니다.

```
postgres=> CREATE EXTENSION sslinfo;
CREATE EXTENSION
postgres=> SELECT ssl_is_used();
ssl_is_used
---------
t
(1 row)
```

자세한 정보를 보려면 다음 쿼리를 사용하여 pg_settings에서 정보를 얻을 수 있습니다.

```
SELECT name as "Parameter name", setting as value, short_desc FROM pg_settings WHERE
 name LIKE '%ssl%';
```

```
              Parameter name             |                  value                  |
                   short_desc
-----------------------------------------+-----------------------------------------
+-------------------------------------------------------
 ssl                                     | on                                      |
 Enables SSL connections.
 ssl_ca_file                             | /rdsdbdata/rds-metadata/ca-cert.pem     |
 Location of the SSL certificate authority file.
 ssl_cert_file                           | /rdsdbdata/rds-metadata/server-cert.pem |
 Location of the SSL server certificate file.
 ssl_ciphers                             | HIGH:!aNULL:!3DES                       |
 Sets the list of allowed SSL ciphers.
 ssl_crl_file                            |                                         |
 Location of the SSL certificate revocation list file.
 ssl_dh_params_file                      |                                         |
 Location of the SSL DH parameters file.
 ssl_ecdh_curve                          | prime256v1                              |
 Sets the curve to use for ECDH.
 ssl_key_file                            | /rdsdbdata/rds-metadata/server-key.pem  |
 Location of the SSL server private key file.
 ssl_library                             | OpenSSL                                 |
 Name of the SSL library.
 ssl_max_protocol_version                |                                         |
 Sets the maximum SSL/TLS protocol version to use.
 ssl_min_protocol_version                | TLSv1.2                                 |
 Sets the minimum SSL/TLS protocol version to use.
 ssl_passphrase_command                  |                                         |
 Command to obtain passphrases for SSL.
 ssl_passphrase_command_supports_reload | off                                      |
 Also use ssl_passphrase_command during server reload.
 ssl_prefer_server_ciphers               | on                                      |
 Give priority to server ciphersuite order.
(14 rows)
```

또한 다음 쿼리를 사용하여 프로세스, 클라이언트 및 애플리케이션별로 RDS for PostgreSQL DB 인스턴스의 SSL 사용량에 대한 모든 정보를 수집할 수도 있습니다.

```
SELECT datname as "Database name", usename as "User name", ssl, client_addr,
 application_name, backend_type
   FROM pg_stat_ssl
   JOIN pg_stat_activity
   ON pg_stat_ssl.pid = pg_stat_activity.pid
   ORDER BY ssl;
```

```
 Database name | User name | ssl |  client_addr   |    application_name    |
 backend_type
---------------+-----------+-----+----------------+------------------------
+-----------------------------
               |           | f   |                |                        | autovacuum
 launcher
               | rdsadmin  | f   |                |                        | logical
 replication launcher
               |           | f   |                |                        | background
 writer
               |           | f   |                |                        |
 checkpointer
               |           | f   |                |                        | walwriter
 rdsadmin      | rdsadmin  | t   | 127.0.0.1      |                        | client
 backend
 rdsadmin      | rdsadmin  | t   | 127.0.0.1      | PostgreSQL JDBC Driver | client
 backend
 postgres      | postgres  | t   | 204.246.162.36 | psql                   | client
 backend
(8 rows)
```

SSL 연결에 사용되는 암호를 식별하기 위해 다음과 같이 쿼리할 수 있습니다.

```
postgres=> SELECT ssl_cipher();
ssl_cipher
--------------------
DHE-RSA-AES256-SHA
(1 row)
```

`sslmode` 옵션에 대한 자세한 내용은 PostgreSQL 설명서의 데이터베이스 연결 제어 기능을 참조하세요.

RDS for PostgreSQL의 SSL 암호 제품군

PostgreSQL 구성 파라미터 ssl_ciphers는 TLS 1.2 이하를 사용할 때 데이터베이스에 대한 SSL 연결에 허용되는 암호 모음의 범주를 지정합니다.

RDS for PostgreSQL 16 이상에서는 허용 목록에 있는 암호 모음의 특정 값을 사용하도록 `ssl_ciphers` 파라미터를 수정할 수 있습니다. 이는 데이터베이스 인스턴스 재부팅이 필요하지 않은 동적 파라미터입니다. 허용 목록에 있는 암호 모음을 보려면 Amazon RDS 콘솔 또는 다음 AWS CLI 명령을 사용합니다.

```
aws rds describe-db-parameters --db-parameter-group-name <your-parameter-group> --
region <region> --endpoint-url <endpoint-url> --output json | jq '.Parameters[] |
 select(.ParameterName == "ssl_ciphers")'
```

다음 표에는 사용자 지정 구성을 지원하는 버전의 기본 암호 모음과 허용되는 암호 모음이 모두 나열되어 있습니다.

PostgreSQL 엔진 버전 번호	기본 ssl_cipher 모음 값	허용 목록에 있는 사용자 지정 ssl_cipher 모음 값
17	HIGH:!aNULL:!3DES	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
16	HIGH:!aNULL:!3DES	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

PostgreSQL 엔진 버전 번호	기본 ssl_cipher 모음 값	허용 목록에 있는 사용자 지정 ssl_cipher 모음 값
		TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
15	HIGH:!aNULL:!3DES	사용자 지정 ssl_ciphers는 지원되지 않음
14	HIGH:!aNULL:!3DES	사용자 지정 ssl_ciphers는 지원되지 않음
13	HIGH:!aNULL:!3DES	사용자 지정 ssl_ciphers는 지원되지 않음
12	HIGH:!aNULL:!3DES	사용자 지정 ssl_ciphers는 지원되지 않음
11.4 이상의 마이너 버전	HIGH:MEDIUM:+3DES:!aNULL:!RC4	사용자 지정 ssl_ciphers는 지원되지 않음
11.1, 11.2	HIGH:MEDIUM:+3DES:!aNULL	사용자 지정 ssl_ciphers는 지원되지 않음
10.9 이상의 마이너 버전	HIGH:MEDIUM:+3DES:!aNULL:!RC4	사용자 지정 ssl_ciphers는 지원되지 않음
10.7 이하 마이너 버전	HIGH:MEDIUM:+3DES:!aNULL	사용자 지정 ssl_ciphers는 지원되지 않음

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 암호 모음을 사용하도록 모든 인스턴스 연결을 구성하려면 다음 예제와 같이 파라미터 그룹을 수정합니다.

```
aws rds modify-db-parameter-group --db-parameter-group-name <your-parameter-group> --
parameters
 "ParameterName='ssl_ciphers',ParameterValue='TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384',ApplyMet
```

이 예제에서는 인스턴스가 타원 곡선 암호(ECC)를 갖춘 인증 기관을 사용하여 연결을 설정해야 하는 ECDSA 암호를 사용합니다. Amazon RDS에서 제공하는 인증 기관에 대한 자세한 내용은 Certificate authorities를 참조하세요.

Determining the SSL connection status에 설명된 방법을 통해 사용 중인 암호를 확인할 수 있습니다.

컨텍스트에 따라 암호의 이름이 다를 수 있습니다.

- 파라미터 그룹에서 구성할 수 있는 허용 목록에 있는 암호는 IANA 이름과 함께 참조됩니다.
- `sslinfo` 및 `psql` 로그온 배너는 OpenSSL 이름을 사용하는 암호를 가리킵니다.

RDS for PostgreSQL 16 이상에서 `ssl_max_protocol_version`의 값은 기본적으로 TLS v1.3입니다. TLS v1.3은 `ssl_ciphers` 파라미터에 지정된 암호 구성을 사용하지 않으므로 이 파라미터의 값을 TLS v1.2로 설정해야 합니다. 값을 TLS v1.2로 설정하면 연결은 `ssl_ciphers`에서 정의한 암호만 사용합니다.

```
aws rds modify-db-parameter-group --db-parameter-group-name <your-parameter-group> --
parameters
 "ParameterName='ssl_max_protocol_version',ParameterValue='TLSv1.2',ApplyMethod=immediate"
```

데이터베이스 연결이 SSL을 사용하도록 하려면 파라미터 그룹에서를 `rds.force_ssl parameter`를 1로 설정합니다. 파라미터 및 파라미터 그룹에 대한 자세한 내용은 Parameter groups for Amazon RDS를 참조하세요.

새 SSL/TLS 인증서를 사용해 PostgreSQL DB 인스턴스에 연결할 애플리케이션 업데이트

보안 소켓 계층 또는 전송 계층(SSL/TLS)에 사용되는 인증서는 일반적으로 설정된 수명을 가집니다. 서비스 공급자가 인증 기관(CA) 인증서를 업데이트할 때 클라이언트는 새 인증서를 사용하도록 애플리케이션을 업데이트해야 합니다. 다음은 클라이언트 애플리케이션에서 SSL/TLS를 사용하여 Amazon RDS for PostgreSQL DB 인스턴스에 연결하는지 판단하는 방법에 대한 정보를 확인할 수 있습니다. 또한 이러한 애플리케이션이 연결할 때 서버 인증서를 확인하는지 확인하는 방법에 대한 정보도 확인할 수 있습니다.

> **Note**
>
> SSL/TLS 연결 전에 서버 인증서를 확인하도록 구성된 클라이언트 애플리케이션은 클라이언트의 트러스트 스토어에 유효한 CA 인증서가 있어야 합니다. 새 인증서에 필요한 경우 클라이언트 트러스트 스토어를 업데이트합니다.

클라이언트 애플리케이션 트러스트 스토어에서 CA 인증서를 업데이트한 후에는 DB 인스턴스에서 인증서를 교환할 수 있습니다. 이 절차를 프로덕션 환경에서 구현하기 전에 비 프로덕션 환경에서 테스트해볼 것을 적극 권장합니다.

인증서 교환에 대한 자세한 내용은 SSL/TLS 인증서 교체 단원을 참조하십시오. 인증서 다운로드에 대한 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 단원을 참조하십시오. PostgreSQL DB 인스턴스에서 SSL/TLS를 사용하는 방법에 관한 자세한 내용은 PostgreSQL DB 인스턴스와 함께 SSL 사용 단원을 참조하십시오.

주제

- 애플리케이션에서 SSL을 사용해 PostgreSQL DB 인스턴스에 연결하는지 여부 확인
- 클라이언트에서 연결을 위해 인증서 확인이 필요한지 여부 확인
- 애플리케이션 트러스트 스토어 업데이트
- 다양한 유형의 애플리케이션에 대해 SSL/TLS 연결 사용

애플리케이션에서 SSL을 사용해 PostgreSQL DB 인스턴스에 연결하는지 여부 확인

`rds.force_ssl` 파라미터의 값에 대한 DB 인스턴스 구성을 확인하십시오. 기본적으로 이 `rds.force_ssl` 파라미터는 버전 15 이하의 PostgreSQL 버전을 사용하는 DB 인스턴스의 경우 `0`(꺼짐)으로 설정되어 있습니다. 기본적으로 이 `rds.force_ssl` 파라미터는 PostgreSQL 버전 15 이상의 메이저 버전을 사용하는 DB 인스턴스의 경우 1(켜짐)으로 설정되어 있습니다. `rds.force_ssl` 파라미터가 1(켜짐)로 설정된 경우 클라이언트는 연결 시 SSL/TLS를 사용해야 합니다. 파라미터 그룹에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.

RDS PostgreSQL 버전 9.5 이상 메이저 버전을 사용 중이고 `rds.force_ssl`이 1(켜짐)로 설정되어 있지 않으면 SSL을 사용하여 연결을 확인하도록 `pg_stat_ssl` 보기를 쿼리하십시오. 예를 들어 다음 쿼리에서는 SSL 연결과 SSL을 사용하는 클라이언트에 관한 정보만 반환합니다.

```
SELECT datname, usename, ssl, client_addr
  FROM pg_stat_ssl INNER JOIN pg_stat_activity ON pg_stat_ssl.pid =
 pg_stat_activity.pid
```

```
  WHERE ssl is true and usename<>'rdsadmin';
```

SSL/TLS 연결을 사용하는 행만 연결에 관한 정보와 함께 표시됩니다. 다음은 출력 샘플입니다.

```
 datname  | usename | ssl | client_addr
----------+---------+-----+-------------
 benchdb  | pgadmin | t   | 53.95.6.13
 postgres | pgadmin | t   | 53.95.6.13
(2 rows)
```

이 쿼리에서는 쿼리 시점의 현재 연결만 표시합니다. 결과가 표시되지 않는다 해도 SSL 연결을 사용하는 애플리케이션이 없는 것은 아닙니다. 다른 SSL 연결이 다른 시점에 설정될 수 있습니다.

클라이언트에서 연결을 위해 인증서 확인이 필요한지 여부 확인

psql 또는 JDBC와 같은 클라이언트가 SSL을 지원하도록 구성되어 있는 경우 클라이언트는 먼저 SSL을 이용해 데이터베이스에 연결을 시도하도록 기본 설정되어 있습니다. SSL을 이용해 연결할 수 없는 경우 클라이언트는 SSL 없이 연결하는 방식으로 전환됩니다. libpq 기반 클라이언트(예: psql)와 JDBC 모두에 사용되는 기본 `sslmode` 모드는 `prefer`로 설정됩니다. 서버의 인증서는 `sslrootcert`에서 `sslmode`가 `verify-ca` 또는 `verify-full`로 설정된 경우에만 확인됩니다. 인증서가 잘못된 경우 오류가 발생합니다.

`PGSSLROOTCERT`를 사용하여 `PGSSLMODE`가 `verify-ca` 또는 `verify-full`로 설정된 `PGSSLMODE` 환경 변수로 인증서를 확인하세요.

```
PGSSLMODE=verify-full PGSSLROOTCERT=/fullpath/ssl-cert.pem psql -h
 pgdbidentifier.cxxxxxxxx.us-east-2.rds.amazonaws.com -U masteruser -d postgres
```

`sslrootcert` 인수를 사용해 `sslmode`, `verify-ca` 또는 `verify-full`로 설정된 연결 문자열 형식의 `sslmode`로 인증서를 확인하세요.

```
psql "host=pgdbidentifier.cxxxxxxxx.us-east-2.rds.amazonaws.com sslmode=verify-full
 sslrootcert=/full/path/ssl-cert.pem user=masteruser dbname=postgres"
```

예를 들어 앞의 사례에서 잘못된 루트 인증서를 사용하는 경우 클라이언트에서 다음과 비슷한 오류가 발생합니다.

```
psql: SSL error: certificate verify failed
```

애플리케이션 트러스트 스토어 업데이트

PostgreSQL 애플리케이션에 대한 트러스트 스토어 업데이트에 대한 자세한 내용은 PostgreSQL 문서의 SSL을 이용한 TCP/IP 연결의 보안 단원을 참조하십시오.

루트 인증서 다운로드에 대한 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 단원을 참조하십시오.

인증서를 가져오는 샘플 스크립트는 트러스트 스토어로 인증서를 가져오기 위한 샘플 스크립트 섹션을 참조하세요.

> **Note**
>
> 트러스트 스토어를 업데이트할 때 새 인증서를 추가할 뿐 아니라 이전 인증서를 유지할 수도 있습니다.

다양한 유형의 애플리케이션에 대해 SSL/TLS 연결 사용

아래에서는 다양한 유형의 애플리케이션에 대해 SSL/TLS 연결을 사용하는 방법에 대한 정보를 제공합니다.

- psql

  클라이언트는 명령줄에서 옵션을 연결 문자열 또는 환경 변수로 지정하여 호출합니다. SSL/TLS 연결의 경우 관련 옵션은 `sslmode`(환경 변수 `PGSSLMODE`), `sslrootcert`(환경 변수 `PGSSLROOTCERT`)입니다.

  옵션 전체 목록은 PostgreSQL 문서의 파라미터 키 단어 단원을 참조하십시오. 환경 변수 전체 목록은 PostgreSQL 문서의 환경 변수 단원을 참조하십시오.

- pgAdmin

  이 브라우저 기반 클라이언트는 PostgreSQL 데이터베이스 연결 시 사용할 수 있는 더 사용자 친화적인 인터페이스입니다.

  연결 구성에 대한 자세한 내용은 pgAdmin 설명서를 참조하십시오.

- JDBC

  JDBC를 통해 Java 애플리케이션의 데이터베이스 연결을 활성화할 수 있습니다.

JDBC를 이용한 PostgreSQL 데이터베이스 연결에 대한 자세한 내용은 PostgreSQL JDBC 드라이버 설명서의 데이터베이스에 연결 단원을 참조하십시오. SSL/TLS을 이용한 연결에 대한 자세한 내용은 PostgreSQL JDBC 드라이버 설명서의 클라이언트 구성 단원을 참조하십시오.

- Python

  PostgreSQL 데이터베이스에 연결하기 위해 많이 사용되는 인기 있는 Python 라이브러리는 `psycopg2`입니다.

  `psycopg2` 사용에 대한 자세한 내용은 psycopg2 설명서를 참조하십시오. PostgreSQL 데이터베이스에 연결하는 방법에 대한 짧은 자습서는 Psycopg2 자습서를 참조하십시오. psycopg2 모듈 콘텐츠에서 연결 명령이 수락하는 옵션에 대한 정보를 얻을 수 있습니다.

> **Important**
>
> 데이터베이스 연결에서 SSL/TLS를 사용함을 확인하고 애플리케이션 트러스트 스토어를 업데이트한 후에는 데이터베이스에서 rds-ca-rsa2048-g1 인증서를 사용하도록 업데이트할 수 있습니다. 지침은 DB 인스턴스 또는 클러스터를 수정하여 CA 인증서 업데이트의 3단계를 참조하십시오.

Amazon RDS for PostgreSQL과 함께 Kerberos 인증 사용

사용자가 PostgreSQL이 실행되는 DB 인스턴스에 연결할 때 Kerberos를 사용하여 사용자를 인증할 수 있습니다. 이를 위해서는 Kerberos 인증을 위한 AWS Directory Service for Microsoft Active Directory를 사용하도록 인스턴스를 구성해야 합니다. AWS Directory Service for Microsoft Active Directory는 AWS Managed Microsoft AD라고도 불립니다. AWS Directory Service에서 사용할 수 있는 기능입니다. 자세한 내용은 AWS Directory Service 관리 가이드의 AWS Directory Service란 무엇입니까?를 참조하세요.

시작하려면 사용자 자격 증명을 저장할 AWS Managed Microsoft AD 디렉터리를 만듭니다. 그런 다음 PostgreSQL DB 인스턴스에 Active Directory의 도메인 및 기타 정보를 제공합니다. PostgreSQL DB 인스턴스에 대해 사용자가 인증될 때 AWS Managed Microsoft AD 디렉터리에 인증 요청이 전달됩니다.

모든 자격 증명을 동일한 디렉터리에 보관하면 시간과 노력을 절약할 수 있습니다. 여러 DB 인스턴스에 대한 자격 증명을 보관하고 관리할 수 있는 중앙 집중식 공간이 있습니다. 디렉터리를 사용하면 전체 보안 프로필을 향상할 수도 있습니다.

자체 온프레미스 Microsoft Active Directory에서 자격 증명에 액세스할 수도 있습니다. 이렇게 하려면 AWS Managed Microsoft AD 디렉터리가 온프레미스 Microsoft Active Directory를 신뢰하도록 신뢰 도메인 관계를 만듭니다. 그러면 사용자가 온프레미스 네트워크에서 워크로드에 액세스할 때와 동일한 Windows SSO(Single Sign-On) 환경을 사용하여 PostgreSQL 인스턴스에 액세스할 수 있습니다.

데이터베이스는 Kerberos 또는 AWS Identity and Access Management(IAM) 인증을 통한 암호 인증을 사용할 수 있습니다. IAM 인증에 관한 자세한 내용은 MariaDB, MySQL 및 PostgreSQL IAM 데이터베이스 인증 단원을 참조하십시오.

주제

- 리전 및 버전 사용 가능 여부
- PostgreSQL DB 인스턴스에 대한 Kerberos 인증 개요
- PostgreSQL DB 인스턴스에 대해 Kerberos 인증 설정
- Active Directory 도메인에서 RDS for PostgreSQL DB 인스턴스 관리
- Kerberos 인증을 사용하여 PostgreSQL 연결

리전 및 버전 사용 가능 여부

기능 가용성 및 해당 지원은 각 데이터베이스 엔진의 특정 버전 및 AWS 리전에 따라 다릅니다. Kerberos 인증을 사용하는 PostgreSQL용 RDS의 버전 및 리전 가용성에 대한 자세한 내용은 Amazon RDS에서 Kerberos 인증을 지원하는 리전 및 DB 엔진 단원을 참조하십시오.

PostgreSQL DB 인스턴스에 대한 Kerberos 인증 개요

PostgreSQL DB 인스턴스에 대해 Kerberos 인증을 설정하려면 다음 단계(나중에 자세히 설명함)를 수행하십시오.

1. AWS Managed Microsoft AD를 사용하여 AWS Managed Microsoft AD 디렉터리를 생성합니다. AWS Management Console, AWS CLI 또는 AWS Directory Service API를 사용하여 디렉터리를 생성할 수 있습니다. 디렉터리가 인스턴스와 통신할 수 있도록 디렉터리 보안 그룹에서 관련 아웃바운드 포트를 열어야 합니다.
2. AWS Managed Microsoft AD 디렉터리에 호출할 수 있는 Amazon RDS 액세스를 제공하는 역할을 생성합니다. 이를 위해 관리형 IAM 정책 `AmazonRDSDirectoryServiceAccess`를 사용하는 AWS Identity and Access Management(IAM) 역할을 생성합니다.

   IAM 역할이 액세스를 허용하게 하려면 올바른 AWS Security Token Service 리전에서 AWS STS 계정의 AWS(AWS) 엔드포인트를 활성화해야 합니다. AWS STS 엔드포인트는 기본적으로 모든 AWS 리전에서 활성화되어 있으므로 추가 작업 없이 사용할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS STS 리전에서 AWS 활성화 및 비활성화를 참조하세요.
3. Microsoft Active Directory 도구를 사용하여 AWS Managed Microsoft AD 디렉터리에서 사용자를 만들고 구성합니다. Active Directory에서 사용자를 생성하는 방법에 대한 자세한 내용은 AWS 관리 안내서의 AWS Directory Service 관리형 Microsoft AD에서 사용자 및 그룹 관리를 참조하세요.
4. 디렉터리와 DB 인스턴스를 다른 AWS 계정 또는 Virtual Private Cloud(VPC)에 배치하려면 VPC 피어링을 구성합니다. 자세한 내용은 Amazon VPC Peering Guide의 VPC 피어링이란?을 참조하십시오.
5. 콘솔, CLI 또는 RDS API에서 다음 메서드 중 하나를 사용하여 PostgreSQL DB 인스턴스를 생성하거나 수정합니다.
   - Amazon RDS DB 인스턴스 생성
   - Amazon RDS DB 인스턴스 수정
   - DB 인스턴스 복원
   - Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원

디렉터리와 동일한 Amazon Virtual Private Cloud(VPC) 또는 다른 AWS 계정이나 VPC에 인스턴스를 배치할 수 있습니다. PostgreSQL DB 인스턴스를 생성하거나 수정할 때는 다음을 수행합니다.

- 디렉터리를 만들 때 생성된 도메인 식별자(d-* 식별자)를 제공합니다.
- 생성한 IAM 역할의 이름을 제공합니다.
- DB 인스턴스의 보안 그룹이 디렉터리의 보안 그룹에서 인바운드 트래픽을 수신할 수 있는지 확인합니다.

6. RDS 마스터 사용자 자격 증명을 사용하여 PostgreSQL DB 인스턴스에 연결합니다. 외부에서 식별할 사용자를 PostgreSQL에서 생성합니다. 외부에서 식별되는 사용자는 Kerberos 인증을 사용하여 PostgreSQL DB 인스턴스에 로그인할 수 있습니다.

PostgreSQL DB 인스턴스에 대해 Kerberos 인증 설정

AWS Directory Service for Microsoft Active Directory(AWS Managed Microsoft AD)를 사용하여 PostgreSQL DB 인스턴스에 대해 Kerberos 인증을 설정합니다. Kerberos 인증을 설정하려면 다음 단계를 수행하십시오.

주제

- 1단계: AWS Managed Microsoft AD를 사용하여 디렉터리 생성
- 2단계: (선택 사항) 온프레미스 Active Directory와 AWS Directory Service 간에 신뢰 관계 생성
- 3단계: Amazon RDS가 AWS Directory Service에 액세스할 수 있는 IAM 역할 생성
- 4단계: 사용자 생성 및 구성
- 5단계: 디렉터리와 DB 인스턴스 사이에 VPC 간 트래픽 활성화
- 6단계: PostgreSQL DB 인스턴스 생성 또는 수정
- 7단계: Kerberos 보안 주체를 위한 PostgreSQL 사용자 생성
- 8단계: PostgreSQL 클라이언트 구성

1단계: AWS Managed Microsoft AD를 사용하여 디렉터리 생성

AWS Directory Service는 AWS 클라우드에서 완전 관리형 Microsoft Active Directory를 생성합니다. AWS Managed Microsoft AD 디렉터리를 생성할 때 AWS Directory Service에서 두 개의 도메인 컨트롤러와 DNS 서버가 자동으로 생성됩니다. 디렉터리 서버는 VPC 내 다른 서브넷에서 생성됩니다. 이러한 중복으로 인해 장애가 발생해도 디렉터리에 액세스할 수 있습니다.

AWS Managed Microsoft AD 디렉터리를 생성할 때 AWS Directory Service에서 다음 작업을 자동으로 수행합니다.

- VPC 내에서 Active Directory를 설정합니다.
- 사용자 이름 `Admin` 과 지정된 암호를 사용하여 디렉터리 관리자 계정을 생성합니다. 이 계정을 사용하여 디렉터리를 관리할 수 있습니다.

  > **Important**
  >
  > 반드시 이 암호를 저장해야 합니다. AWS Directory Service에서는 이 암호를 저장하지 않으므로 암호를 검색하거나 다시 설정할 수 없습니다.

- 디렉터리 컨트롤러에 대한 보안 그룹을 만듭니다. 보안 그룹이 PostgreSQL DB 인스턴스와의 통신을 허용해야 합니다.

AWS Directory Service for Microsoft Active Directory를 시작하면 AWS가 모든 디렉터리의 객체를 포함하는 OU(조직 단위)를 생성합니다. 디렉터리를 만들 때 입력한 NetBIOS 이름이 있는 이 OU는 도메인 루트에 있습니다. 도메인 루트는 AWS에서 소유하고 관리합니다.

`Admin` 디렉터리를 사용하여 생성한 AWS Managed Microsoft AD 계정은 OU의 가장 일반적인 관리 활동에 대한 권한이 있습니다.

- 사용자 생성, 업데이트 또는 삭제
- 도메인(예: 파일 또는 인쇄 서버)에 리소스를 추가한 다음 OU 내의 사용자에 해당 리소스에 대한 권한 할당
- 추가 OU 및 컨테이너 생성
- 권한 위임
- Active Directory 휴지통에서 삭제된 객체 복원
- Active Directory 웹 서비스에서 Active Directory 및 Windows PowerShell에 대한 DNS(Domain Name Service) 모듈 실행

또한 `Admin` 계정은 다음 도메인 차원 활동을 수행할 권한이 있습니다.

- DNS 구성 관리(레코드, 영역 및 전달자 추가, 제거 또는 업데이트)
- DNS 이벤트 로그 보기
- 보안 이벤트 로그 보기

AWS Managed Microsoft AD으로 디렉터리를 생성하려면

1. AWS Directory Service 콘솔 탐색 창에서 디렉터리를 선택한 후 디렉터리 설정을 선택합니다.
2. AWS Managed Microsoft AD를 선택합니다. AWS Managed Microsoft AD는 현재 Amazon RDS에서 사용하도록 지원되는 유일한 옵션입니다.
3. 다음을 선택합니다.
4. 디렉터리 정보 입력 페이지에서 다음 정보를 제공합니다.

   Edition

   요구 사항에 맞는 에디션을 선택합니다.

   디렉터리 DNS 이름

   디렉터리를 위한 정규화된 이름(예: **`corp.example.com`**)입니다.

   디렉터리 NetBIOS 이름

   디렉터리의 선택적 짧은 이름(예: `CORP`)입니다.

   디렉터리 설명

   디렉터리에 대한 선택적 설명을 입력합니다.

   관리자 암호

   디렉터리 관리자의 암호입니다. 디렉터리 생성 프로세스에서는 사용자 이름 `Admin`와 이 암호를 사용하여 관리자 계정을 생성합니다.

   디렉터리 관리자 암호는 "admin"이라는 단어를 포함할 수 없습니다. 암호는 대소문자를 구분하며 길이가 8~64자 사이여야 합니다. 또한 다음 네 범주 중 세 개에 해당하는 문자를 1자 이상 포함해야 합니다.

   - 소문자(a–z)
   - 대문자(A–Z)
   - 숫자(0–9)
   - 영숫자 외의 특수 문자(~!@#$%^&*_-+=`|\(){}[]:;'"<>,.?/)

   [Confirm Password]

   관리자 암호를 다시 입력합니다.

> **Important**
>
> 반드시 이 암호를 저장해야 합니다. AWS Directory Service에서는 이 암호를 저장하지 않으며 암호를 검색하거나 재설정할 수 없습니다.

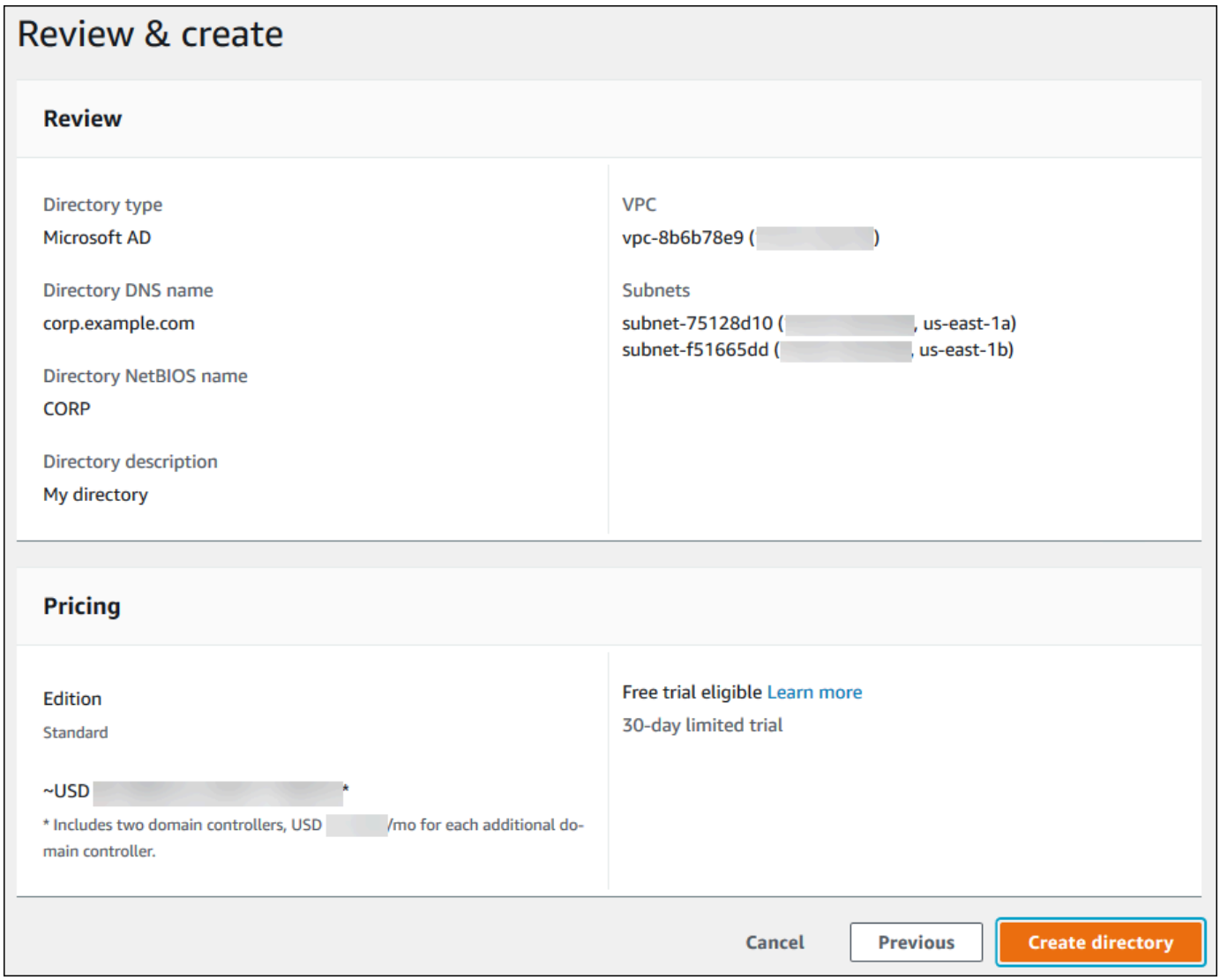
5. Next(다음)를 선택합니다.
6. Choose VPC and subnets(VPC 및 서브넷 선택) 페이지에 다음 정보를 입력합니다.

   VPC

   디렉터리에 대한 VPC를 선택합니다. PostgreSQL DB 인스턴스는 이와 동일한 VPC 또는 다른 VPC에서 생성할 수 있습니다.

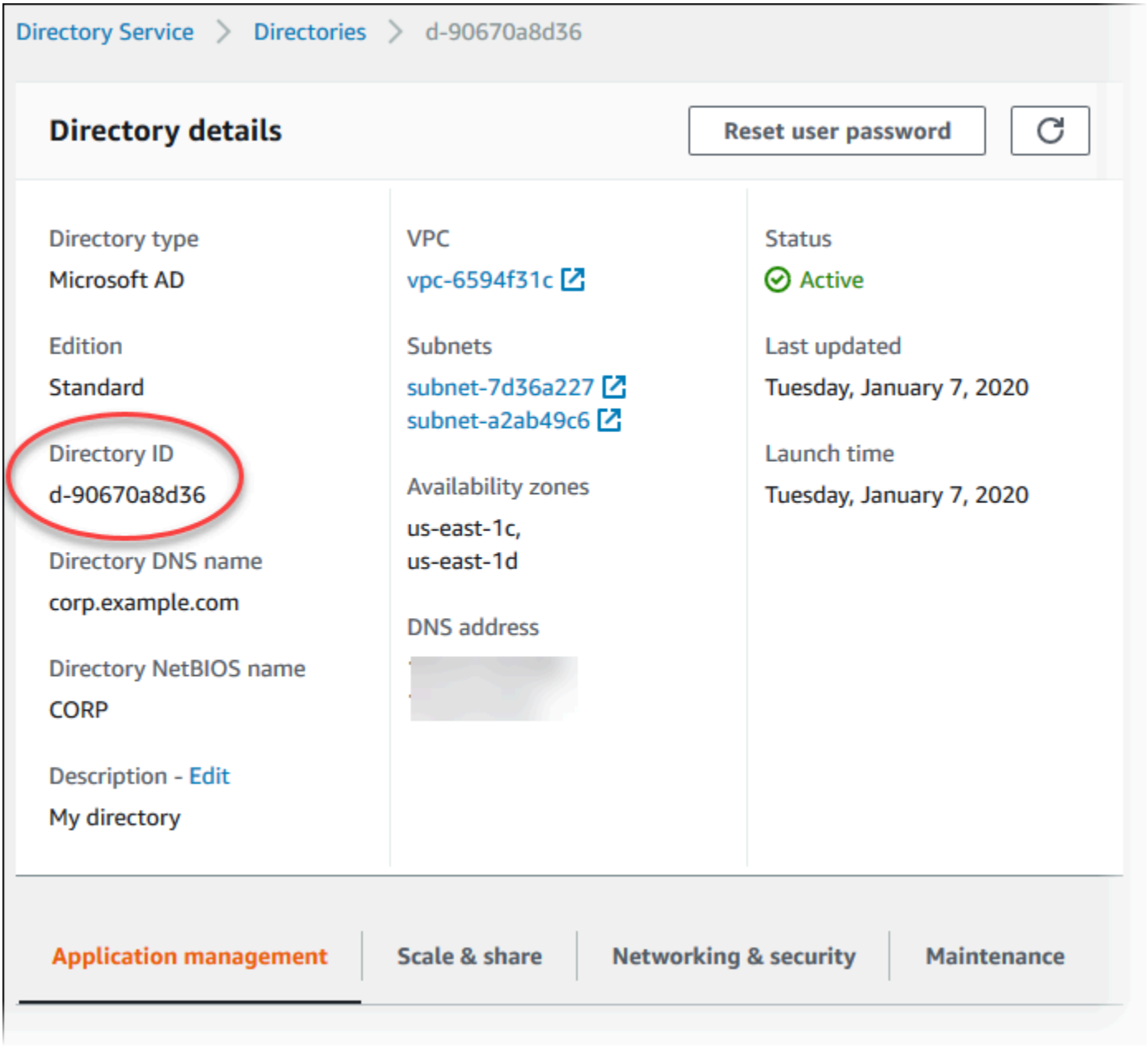
   서브넷

   디렉터리 서버에 대한 서브넷을 선택합니다. 두 서브넷이 서로 다른 가용 영역에 있어야 합니다.
7. Next(다음)를 선택합니다.
8. 디렉터리 정보를 검토합니다. 변경이 필요하면 이전을 선택하여 변경합니다. 정보가 올바르면 Create directory(디렉터리 생성)을 선택합니다.



디렉터리를 생성하는 데 몇 분 정도 걸립니다. 디렉터리가 성공적으로 생성되면 상태 값이 활성으로 변경됩니다.

디렉터리에 대한 정보를 보려면 디렉터리 목록에서 해당 디렉터리 ID를 선택합니다. 디렉터리 ID 값을 적어 두십시오. PostgreSQL DB 인스턴스를 생성하거나 수정할 때 이 값이 필요합니다.



2단계: (선택 사항) 온프레미스 Active Directory와 AWS Directory Service 간에 신뢰 관계 생성

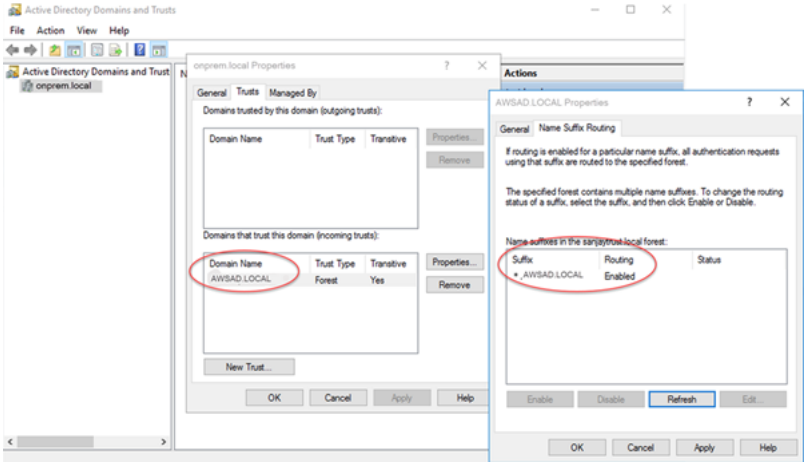
자체 온프레미스 Microsoft Active Directory를 사용하지 않으려는 경우 3단계: Amazon RDS가 AWS Directory Service에 액세스할 수 있는 IAM 역할 생성로 건너뜁니다.

온프레미스 Active Directory를 사용하여 Kerberos 인증을 받으려면 온프레미스 Microsoft Active Directory와 AWS Managed Microsoft AD에서 만든 1단계: AWS Managed Microsoft AD를 사용하여 디렉터리 생성 디렉터리 간에 forest trust를 사용하여 신뢰 도메인 관계를 생성해야 합니다. 신뢰는 AWS Managed Microsoft AD 디렉터리가 온프레미스 Microsoft Active Directory를 신뢰하는 단방향일 수도

있고, 두 Active Directory가 서로를 신뢰하는 양방향일 수도 있습니다. AWS Directory Service를 사용하여 신뢰를 설정하는 방법에 대한 자세한 내용은 AWS Directory Service 관리 안내서의 신뢰 관계를 생성해야 하는 경우를 참조하세요.

> **Note**
>
> 온프레미스 Microsoft Active Directory를 사용하는 경우 Windows 클라이언트에서 rds.amazonaws.com 대신 엔드포인트에 있는 AWS Directory Service의 도메인 이름을 사용하여 연결합니다. 자세한 내용은 Kerberos 인증을 사용하여 PostgreSQL 연결를 참조하세요.

온프레미스 Microsoft Active Directory 도메인 이름에 새로 만든 신뢰 관계에 해당하는 DNS 접미사 라우팅이 포함되어 있는지 확인합니다. 다음 스크린샷은 예를 보여줍니다.



3단계: Amazon RDS가 AWS Directory Service에 액세스할 수 있는 IAM 역할 생성

AWS Directory Service를 호출하는 Amazon RDS의 경우 AWS 계정에 관리형 IAM 정책 `AmazonRDSDirectoryServiceAccess`를 사용하는 IAM 역할이 필요합니다. 이 역할을 사용하여 Amazon RDS에서 AWS Directory Service를 호출할 수 있습니다.

AWS Management Console을 사용하여 DB 인스턴스를 생성할 때 콘솔 사용자에게 `iam:CreateRole` 권한이 있으면 콘솔에서 필요한 IAM 역할을 자동으로 생성합니다. 이 경우 역할 이름은 `rds-directoryservice-kerberos-access-role`입니다. 그렇지 않으면 IAM 역할을 수동으로 생성해야 합니다. 이 IAM 역할을 생성할 때 `Directory Service`를 선택하고 여기에 AWS 관리형 정책인 `AmazonRDSDirectoryServiceAccess`를 연결합니다.

서비스에 대한 IAM 역할 생성에 대한 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.

> **Note**
>
> RDS for Microsoft SQL Server에 대한 Windows 인증에 사용되는 IAM 역할은 Amazon RDS for PostgreSQL에 사용할 수 없습니다.

`AmazonRDSDirectoryServiceAccess` 관리형 정책인을 사용하는 대신 필요한 권한이 포함된 정책을 생성할 수 있습니다. 이 경우 IAM 역할에 다음과 같은 IAM 신뢰 정책이 있어야 합니다.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.amazonaws.com",
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

또한 역할에는 다음과 같은 IAM 역할 정책도 있어야 합니다.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ds:DescribeDirectories",
        "ds:AuthorizeApplication",
```

```
        "ds:UnauthorizeApplication",
        "ds:GetAuthorizedApplicationDetails"
      ],
    "Effect": "Allow",
    "Resource": "*"
    }
  ]
}
```

옵트인 AWS 리전의 경우 IAM 역할 신뢰 정책에서 리전별 서비스 위탁자를 사용합니다. 이러한 리전의 서비스에 대한 신뢰 정책을 만들 때 서비스 위탁자에 리전 코드를 지정합니다.

다음 예는 리전별 서비스 위탁자가 포함된 신뢰 정책을 보여 줍니다.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.REGION-CODE.amazonaws.com",
          "rds.REGION-CODE.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

REGION-CODE를 사용자의 특정 리전 코드로 바꾸세요. 예를 들어 아시아 태평양(멜버른) 리전에는 다음 서비스 위탁자를 사용합니다.

```
"Service": [
  "directoryservice.rds.ap-southeast-4.amazonaws.com",
```

```
  "rds.ap-southeast-4.amazonaws.com"
]
```

4단계: 사용자 생성 및 구성

Active Directory Users and Computers 도구를 사용하여 사용자를 생성할 수 있습니다. 이 도구는 Active Directory 도메인 서비스 및 Active Directory Lightweight Directory Services 도구 중 하나입니다. 자세한 내용은 Microsoft 설명서의 Active Directory 도메인에 사용자 및 컴퓨터 추가를 참조하세요. 이 경우 사용자는 도메인에 속하며 디렉터리에서 ID가 유지되는 개인 또는 기타 엔터티(예: 사용자의 컴퓨터)입니다.

AWS Directory Service 디렉터리에서 사용자를 생성하려면 AWS Directory Service 디렉터리의 멤버인 Windows 기반 Amazon EC2 인스턴스에 연결되어 있어야 합니다. 이와 동시에 사용자를 생성할 권한이 있는 사용자로 로그인한 상태이어야 합니다. 자세한 내용은 AWS Directory Service관리 안내서의 사용자 생성을 참조하십시오.

5단계: 디렉터리와 DB 인스턴스 사이에 VPC 간 트래픽 활성화

디렉터리와 DB 인스턴스를 동일한 VPC에 배치하려면 이 단계를 건너뛰고 6단계: PostgreSQL DB 인스턴스 생성 또는 수정 단원으로 이동하십시오.

디렉터리와 DB 인스턴스를 서로 다른 VPC에 배치하려면 VPC 피어링 또는 AWS Transit Gateway를 사용하여 VPC 간 트래픽을 구성하세요.

다음 절차는 VPC 피어링을 사용하여 VPC 간 트래픽을 활성화합니다. Amazon Virtual Private Cloud 피어링 안내서의 VPC 피어링이란? 지침을 따르십시오.

VPC 피어링을 사용하여 VPC 간 트래픽을 활성화하려면

1. 네트워크 트래픽이 양방향으로 흐를 수 있도록 적절한 VPC 라우팅 규칙을 설정합니다.
2. DB 인스턴스의 보안 그룹이 디렉터리의 보안 그룹에서 인바운드 트래픽을 수신할 수 있는지 확인합니다.
3. 트래픽을 차단하는 네트워크 ACL(액세스 제어 목록) 규칙이 없어야 합니다.

다른 AWS 계정이 디렉터리를 소유하는 경우 디렉터리를 공유해야 합니다.

AWS 계정 간에 디렉터리를 공유하려면

1. AWS 관리 안내서의 자습서: 원활한 EC2 도메인 조인을 위해 AWS Managed Microsoft AD 디렉터리 공유에 있는 지침에 따라 DB 인스턴스가 생성될 AWS Directory Service 계정과 디렉터리를 공유하는 작업을 시작합니다.
2. DB 인스턴스용 계정을 사용하여 AWS Directory Service 콘솔에 로그인하고 계속하기 전에 도메인이 SHARED 상태가 되었는지 확인합니다.
3. DB 인스턴스용 계정을 사용하여 AWS Directory Service 콘솔에 로그인하는 동안 디렉터리 ID 값을 기록해 둡니다. 이 디렉터리 ID를 사용하여 DB 인스턴스를 도메인에 조인합니다.

6단계: PostgreSQL DB 인스턴스 생성 또는 수정

디렉터리에서 사용할 PostgreSQL DB 인스턴스를 생성하거나 수정합니다. 콘솔, CLI 또는 RDS API를 사용하여 DB 인스턴스를 디렉터리에 연결할 수 있습니다. 이 작업을 다음 중 한 가지 방법으로 수행할 수 있습니다.

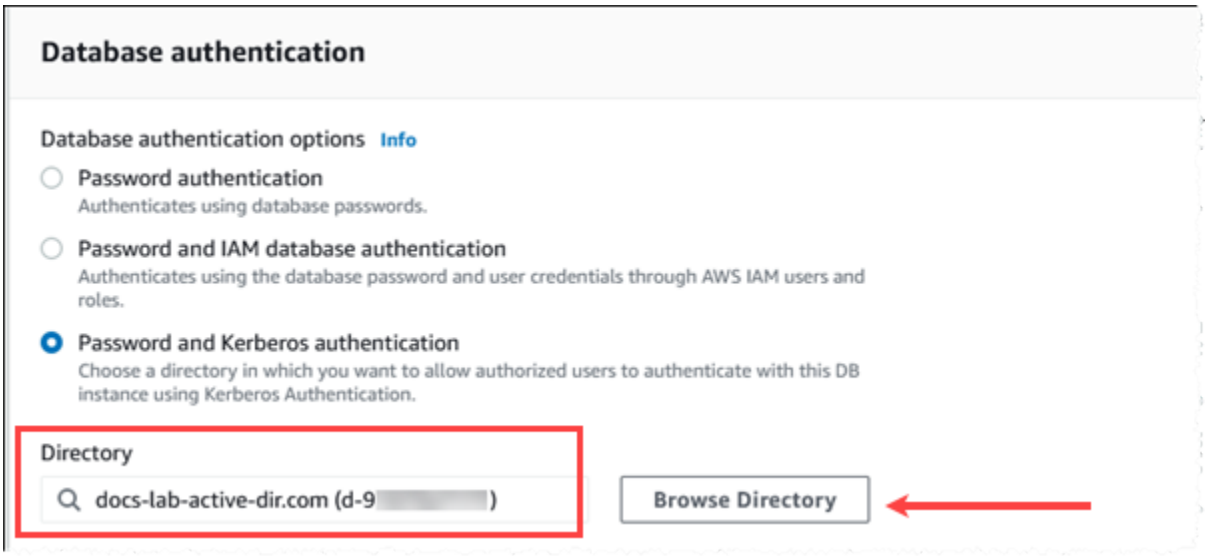
- 콘솔, create-db-instance CLI 명령 또는 CreateDBInstance RDS API 작업을 사용하여 새 PostgreSQL DB 인스턴스를 생성합니다. 지침은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요.
- 콘솔, modify-db-instance CLI 명령 또는 ModifyDBInstance RDS API 작업을 사용하여 기존 PostgreSQL DB 인스턴스를 수정합니다. 지침은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.
- 콘솔, restore-db-instance-from-db-snapshot CLI 명령 또는 RestoreDBInstanceFromDBSnapshot RDS API 작업을 사용하여 DB 스냅샷에서 PostgreSQL DB 인스턴스를 복원합니다. 지침은 DB 인스턴스 복원 섹션을 참조하세요.
- 콘솔, restore-db-instance-to-point-in-time CLI 명령 또는 RestoreDBInstanceToPointInTime RDS API 작업을 사용하여 PostgreSQL DB 인스턴스를 특정 시점으로 복원합니다. 지침은 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 섹션을 참조하세요.

Kerberos 인증은 VPC의 PostgreSQL DB 인스턴스에 대해서만 지원됩니다. DB 인스턴스는 디렉터리와 동일한 VPC 또는 다른 VPC에 있을 수 있습니다. DB 인스턴스가 디렉터리와 통신할 수 있도록 DB 인스턴스는 디렉터리의 VPC 내 수신 및 송신을 허용하는 보안 그룹을 사용해야 합니다.

콘솔

콘솔을 사용하여 DB 인스턴스를 생성, 수정 또는 복원하는 경우 [데이터베이스 인증(Database authentication)] 섹션에서 [암호 및 Kerberos 인증(Password and Kerberos authentication)]을 선택합니

다. 그런 다음 [디렉터리 찾아보기(Browse Directory)]를 선택합니다. 디렉터리를 선택하거나 [새 디렉토리 생성(Create a new directory)]을 클릭하여 Directory Service를 사용합니다.



AWS CLI

AWS CLI를 사용하는 경우 생성한 디렉터리를 DB 인스턴스에서 사용하려면 다음과 같은 파라미터가 필요합니다.

- `--domain` 파라미터의 경우 디렉터리를 만들 때 생성된 도메인 식별자("d-*" 식별자)를 사용하십시오.
- `--domain-iam-role-name` 파라미터의 경우 귀하가 생성한, 관리형 IAM 정책 `AmazonRDSDirectoryServiceAccess`를 사용하는 역할을 사용하십시오.

예를 들어, 다음 CLI 명령은 디렉터리를 사용하도록 DB 인스턴스를 수정합니다.

```
aws rds modify-db-instance --db-instance-identifier mydbinstance --domain d-Directory-ID --domain-iam-role-name role-name
```

> **Important**
>
> DB 인스턴스를 수정하여 Kerberos 인증을 사용 설정하는 경우 변경 후 DB 인스턴스를 재부팅합니다.

7단계: Kerberos 보안 주체를 위한 PostgreSQL 사용자 생성

이때 RDS for PostgreSQL DB 인스턴스가 AWS Managed Microsoft AD 도메인에 조인됩니다. 4단계: 사용자 생성 및 구성 에서 디렉터리에서 생성한 사용자는 PostgreSQL 데이터베이스 사용자로 설정하고 데이터베이스에 로그인할 수 있는 권한을 부여해야 합니다. `rds_superuser` 권한이 있는 데이터베이스 사용자로 로그인하면 됩니다. 예를 들어 , RDS for PostgreSQL DB 인스턴스를 생성할 때 기본값을 수락한 경우 다음 단계에 표시된 대로 `postgres`를 사용합니다.

Kerberos 보안 주체를 위한 PostgreSQL 데이터베이스 사용자를 생성하는 방법

1. `psql`을 사용하여 의 RDS for PostgreSQL DB 인스턴스 엔드포인트에 연결합니다. 다음 예에서는 `rds_superuser` 역할에 기본 `postgres` 계정을 사용합니다.

   ```
   psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password
   ```

2. 데이터베이스에 액세스하도록 할 각 Kerberos 보안 주체(Active Directory 사용자 이름)에 대한 데이터베이스 사용자 이름을 생성합니다. Active Directory 인스턴스에 정의된 표준 사용자 이름(ID), 즉 해당 사용자 이름에 대한 Active Directory 도메인의 소문자 `alias`(Active Directory에서는 사용자 이름)와 대문자 이름을 사용합니다. Active Directory 사용자 이름은 외부에서 인증된 사용자이므로 다음과 같이 이름을 따옴표로 묶습니다.

   ```
   postgres=> CREATE USER "username@CORP.EXAMPLE.COM" WITH LOGIN;
   CREATE ROLE
   ```

3. 데이터베이스 사용자에게 `rds_ad` 역할을 부여합니다.

   ```
   postgres=> GRANT rds_ad TO "username@CORP.EXAMPLE.COM";
   GRANT ROLE
   ```

Active Directory 사용자 ID에 대한 모든 PostgreSQL 사용자 생성을 완료하면 사용자는 Kerberos 보안 인증 정보를 사용하여 RDS for PostgreSQL DB 인스턴스에 액세스할 수 있습니다.

Kerberos를 사용하여 인증하는 데이터베이스 사용자는 Active Directory 도메인의 멤버인 클라이언트 컴퓨터에서 인증을 수행하도록 요구됩니다.

`rds_ad` 역할이 부여된 데이터베이스 사용자는 `rds_iam` 역할까지 가질 수는 없습니다. 이는 중첩된 멤버십에도 적용됩니다. 자세한 내용은 MariaDB, MySQL 및 PostgreSQL IAM 데이터베이스 인증 섹션을 참조하세요.

8단계: PostgreSQL 클라이언트 구성

PostgreSQL 클라이언트를 구성하려면 다음 단계를 수행하십시오.

- 도메인을 가리키도록 krb5.conf 파일(또는 동등한 파일)을 생성합니다.
- 클라이언트 호스트와 AWS Directory Service 간에 트래픽이 흐를 수 있는지 확인합니다. Netcat과 같은 네트워크 유틸리티를 사용하여 다음을 수행하십시오.
  - 포트 53의 DNS를 통한 트래픽을 확인합니다.
  - 포트 53 및 AWS Directory Service용 포트 88 및 464를 포함하는 Kerberos의 TCP/UDP를 통한 트래픽을 확인합니다.
- 데이터베이스 포트를 통해 클라이언트 호스트와 DB 인스턴스 간에 트래픽이 흐를 수 있는지 확인합니다. 예를 들어, psql을 사용하여 데이터베이스에 연결하고 액세스합니다.

다음은 AWS Managed Microsoft AD의 샘플 krb5.conf 콘텐츠입니다.

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
```

다음은 온프레미스 Microsoft Active Directory의 샘플 krb5.conf 콘텐츠입니다.

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
 ONPREM.COM = {
  kdc = onprem.com
  admin_server = onprem.com
 }
[domain_realm]
```

```
.example.com = EXAMPLE.COM
example.com = EXAMPLE.COM
.onprem.com = ONPREM.COM
onprem.com = ONPREM.COM
.rds.amazonaws.com = EXAMPLE.COM
.amazonaws.com.cn = EXAMPLE.COM
.amazon.com = EXAMPLE.COM
```

Active Directory 도메인에서 RDS for PostgreSQL DB 인스턴스 관리

콘솔, CLI 또는 RDS API를 사용하여 DB 인스턴스 및 Microsoft Active Directory와의 관계를 관리할 수 있습니다. 예를 들어, Active Directory를 연결하여 Kerberos 인증을 활성화할 수 있습니다. 또한 Active Directory 연결을 제거하여 Kerberos 인증을 비활성화할 수 있습니다. 또한 DB 인스턴스를 이동하여 한 Microsoft Active Directory에서 다른 Microsoft Active Directory로 외부적으로 인증해 줄 수 있습니다.

예를 들어 CLI를 사용하여 다음 작업을 수행할 수 있습니다.

- 실패한 멤버십에 대한 Kerberos 인증 활성화를 다시 시도하려면 modify-db-instance CLI 명령을 사용합니다. `--domain` 옵션에 대해 현재 멤버십의 디렉터리 ID를 지정합니다.
- DB 인스턴스에서 Kerberos 인증을 비활성화하려면 modify-db-instance CLI 명령을 사용합니다. `none` 옵션의 경우 `--domain`을 지정합니다.
- 한 도메인에서 다른 도메인으로 DB 인스턴스를 이동하려면 modify-db-instance CLI 명령을 사용합니다. `--domain` 옵션에 대해 새 도메인의 도메인 식별자를 지정합니다.

도메인 멤버십 이해

DB 인스턴스를 생성하거나 수정하면 멤버가 됩니다. 콘솔에서 또는 describe-db-instances CLI 명령을 실행하여 도메인 멤버십의 상태를 확인할 수 있습니다. DB 인스턴스의 상태는 다음 중 한 가지가 될 수 있습니다.

- `kerberos-enabled` - DB 인스턴스에 Kerberos 인증이 활성화되어 있습니다.
- `enabling-kerberos` - AWS에서 이 DB 인스턴스에 대한 Kerberos 인증 활성화를 진행 중입니다.
- `pending-enable-kerberos` - 이 DB 인스턴스에 대한 Kerberos 인증 활성화가 보류 중입니다.
- `pending-maintenance-enable-kerberos` - AWS에서 예약된 다음 유지 관리 기간에 DB 인스턴스에 대한 Kerberos 인증을 활성화하려 합니다.
- `pending-disable-kerberos` - 이 DB 인스턴스에 대한 Kerberos 인증 비활성화가 보류 중입니다.

- `pending-maintenance-disable-kerberos` - AWS에서 예약된 다음 유지 관리 기간에 DB 인스턴스에 대한 Kerberos 인증을 비활성화하려 합니다.
- `enable-kerberos-failed` - 구성 문제로 인해 AWS가 DB 인스턴스에 대해 Kerberos 인증을 활성화하지 못했습니다. DB 인스턴스 수정 명령을 다시 실행하기 전에 구성 문제를 해결하십시오.
- `disabling-kerberos` - AWS에서 이 DB 인스턴스에 대한 Kerberos 인증 비활성화를 진행 중입니다.

네트워크 연결 문제 또는 잘못된 IAM 역할로 인해 Kerberos 인증 활성화 요청이 실패할 수 있습니다. 경우에 따라 DB 인스턴스를 생성하거나 수정할 때 Kerberos 인증을 사용하려고 하면 실패할 수 있습니다. 이런 경우 올바른 IAM 역할을 사용하고 있는지 확인한 다음 도메인에 조인하도록 DB 인스턴스를 수정합니다.

> **Note**
>
> RDS for PostgreSQL을 사용한 Kerberos 인증에서만 트래픽을 도메인의 DNS 서버로 전송합니다. 다른 모든 DNS 요청은 PostgreSQL을 실행 중인 DB 인스턴스에서 아웃바운드 네트워크 액세스로 취급됩니다. RDS for PostgreSQL을 사용한 아웃바운드 네트워크 액세스에 대한 자세한 내용은 아웃바운드 네트워크 액세스에 사용자 지정 DNS 서버 사용 단원을 참조하십시오.

Kerberos 인증을 사용하여 PostgreSQL 연결

pgAdmin 인터페이스 또는 psql과 같은 명령줄 인터페이스를 사용하여 Kerberos 인증으로 PostgreSQL에 연결할 수 있습니다. 연결에 대한 자세한 내용은 PostgreSQL 데이터베이스 엔진을 실행하는 DB 인스턴스에 연결 섹션을 참조하세요. 연결에 필요한 엔드포인트, 포트 번호 및 기타 세부 정보를 얻는 방법에 대한 자세한 내용은 PostgreSQL DB 인스턴스에 연결 단원을 참조하십시오.

> **Note**
>
> PostgreSQL의 GSSAPI 인증 및 암호화는 Kerberos 라이브러리 `libkrb5.so`에 의해 구현됩니다. `postgres_fdw` 및 `dblink`와 같은 기능은 Kerberos 인증 또는 암호화를 사용한 아웃바운드 연결에도 동일한 라이브러리를 사용합니다.

pgAdmin

pgAdmin을 사용하여 Kerberos 인증으로 PostgreSQL에 연결하려면 다음 단계를 수행하십시오.

1. 클라이언트 컴퓨터에서 pgAdmin 애플리케이션을 실행합니다.
2. [Dashboard] 탭에서 [Add New Server]를 선택합니다.
3. 생성 - 서버 대화 상자에서 pgAdmin의 서버를 식별하기 위해 일반 탭에 이름을 입력합니다.
4. 연결 탭에서 RDS for PostgreSQL 데이터베이스에 있는 다음 정보를 입력합니다.
   - 호스트의 경우 에 대한 엔드포인트를 입력합니다. RDS for PostgreSQL DB 인스턴스 엔드포인트는 다음과 유사하게 표시됩니다.

     ```
     RDS-DB-instance.111122223333.aws-region.rds.amazonaws.com
     ```

     Windows 클라이언트에서 온-프레미스 Microsoft Active Directory에 연결하려면 호스트 엔드포인트의 `rds.amazonaws.com` 대신 AWS Managed Active Directory의 도메인 이름을 사용합니다. 예를 들어 AWS Managed Active Directory의 도메인 이름이 `corp.example.com`일 경우 그런 다음호스트에서는 엔드포인트가 다음과 같이 지정됩니다.

     ```
     RDS-DB-instance.111122223333.aws-region.corp.example.com
     ```
   - 포트에 할당된 포트를 입력합니다.
   - Maintenance database(유지 관리 데이터베이스)에 클라이언트가 연결될 초기 데이터베이스의 이름을 입력합니다.
   - Username(사용자 이름)에 7단계: Kerberos 보안 주체를 위한 PostgreSQL 사용자 생성 의 Kerberos 인증을 위해 입력했던 사용자 이름을 입력합니다.
5. 저장을 선택합니다.

psql

psql을 사용하여 Kerberos 인증으로 PostgreSQL에 연결하려면 다음 단계를 수행하십시오.

1. 명령 프롬프트에서 다음 명령을 실행합니다.

   ```
   kinit username
   ```

   *username*을 사용자 이름으로 대체합니다. 프롬프트에서 Microsoft Active Directory에 저장된 사용자 암호를 입력합니다.
2. PostgreSQL DB 인스턴스가 공개적으로 액세스 가능한 VPC를 사용하는 경우 DB 인스턴스 엔드포인트의 IP 주소를 EC2 클라이언트의 `/etc/hosts` 파일에 넣습니다. 예를 들어 다음 명령은 IP 주소를 얻은 다음 `/etc/hosts` 파일에 넣습니다.

```
% dig +short PostgreSQL-endpoint.AWS-Region.rds.amazonaws.com
;; Truncated, retrying in TCP mode.
ec2-34-210-197-118.AWS-Region.compute.amazonaws.com.
34.210.197.118

% echo " 34.210.197.118  PostgreSQL-endpoint.AWS-Region.rds.amazonaws.com" >> /etc/
hosts
```

Windows 클라이언트에서 온프레미스 Microsoft Active Directory를 사용하는 경우 특수 엔드포인트를 사용하여 연결해야 합니다. 호스트 엔드포인트에서 Amazon 도메인 `rds.amazonaws.com`을 사용하는 대신 AWS Managed Active Directory의 도메인 이름을 사용합니다.

예를 들어 AWS Managed Active Directory의 도메인 이름이 `corp.example.com`일 경우 엔드포인트에 *PostgreSQL-endpoint*.*AWS-Region*.*corp.example.com* 형식을 사용하고 `/etc/hosts` 파일에 넣습니다.

```
% echo " 34.210.197.118  PostgreSQL-endpoint.AWS-Region.corp.example.com" >> /etc/
hosts
```

3. 다음 psql 명령을 사용하여 Active Directory와 통합된 PostgreSQL DB 인스턴스에 로그인합니다.

```
psql -U username@CORP.EXAMPLE.COM -p 5432 -h PostgreSQL-endpoint.AWS-
Region.rds.amazonaws.com postgres
```

온프레미스 Active Directory를 사용하여 Windows 클러스터에서 PostgreSQL DB 클러스터에 로그인하려면 이전 단계의 도메인 이름(`corp.example.com`)과 함께 다음 psql 명령을 사용합니다.

```
psql -U username@CORP.EXAMPLE.COM -p 5432 -h PostgreSQL-endpoint.AWS-
Region.corp.example.com postgres
```

아웃바운드 네트워크 액세스에 사용자 지정 DNS 서버 사용

RDS for PostgreSQL은 DB 인스턴스에서 아웃바운드 네트워크 액세스를 지원하고, 고객이 소유한 사용자 정의 DNS 서버에서의 DNS(Domain Name Service) 확인을 허용합니다. 사용자 지정 DNS 서버를 통해 RDS for PostgreSQL DB 인스턴스에서 전체 주소 도메인 이름만을 확인할 수 있습니다.

주제

- 사용자 지정 DNS 해결 활성화
- 사용자 지정 DNS 해결 비활성화
- 사용자 지정 DNS 서버 설정

사용자 지정 DNS 해결 활성화

고객 VPC에서 DNS 해결을 활성화하려면, 먼저 사용자 지정 DB 파라미터 그룹을 RDS for PostgreSQL 인스턴스에 연결합니다. 그런 다음 `rds.custom_dns_resolution` 파라미터를 1로 설정하여 변경 사항이 적용되도록 DB 인스턴스를 다시 시작합니다.

사용자 지정 DNS 해결 비활성화

고객 VPC에서 DNS 해결을 비활성화하려면, 먼저 사용자 지정 DB 파라미터 그룹을 0으로 설정하여 `rds.custom_dns_resolution`을 비활성화합니다. 변경 사항이 적용되도록 DB 인스턴스를 다시 시작합니다.

사용자 지정 DNS 서버 설정

사용자 지정 DNS 이름 서버를 설정한 후 변경 사항이 DB 인스턴스에 전파되는 데 최대 30분이 걸립니다. 변경 사항이 DB 인스턴스에 전파된 후 DNS 조회를 필요로 하는 모든 아웃바운드 네트워크 트래픽은 포트 53을 통해 DNS 서버를 쿼리합니다.

> **Note**
>
> 사용자 지정 DNS 서버를 설정하지 않고 `rds.custom_dns_resolution`이 1로 설정된 경우 Amazon Route 53 프라이빗 영역을 사용하여 호스트가 확인됩니다. 자세한 내용은 프라이빗 호스팅 영역 작업을 참조하세요.

RDS for PostgreSQL DB 인스턴스의 사용자 지정 DNS 서버 설정

1. VPC에 연결된 동적 호스트 구성 프로토콜(DHCP) 옵션 세트에서 DNS 이름 서버의 IP 주소에 대해 `domain-name-servers` 옵션을 설정합니다. 자세한 내용은 DHCP 옵션 세트 단원을 참조하세요.

   > **Note**
   >
   > `domain-name-servers` 옵션은 최대 4개의 값을 받아들이지만 Amazon RDS DB 인스턴스는 첫 번째 값만을 사용합니다.

2. DNS 서버가 DNS 이름, Amazon EC2 프라이빗 DNS 이름, 고객별 DNS 이름을 비롯한 모든 조회 쿼리를 확인할 수 있는지 확인합니다. 아웃바운드 네트워크 트래픽에 DNS 서버가 처리할 수 없는 DNS 조회가 포함된 경우, DNS 서버에 적절한 업스트림 DNS 공급자가 구성되어 있어야 합니다.
3. 512바이트 이하의 UDP(User Datagram Protocol) 응답을 생성하도록 DNS 서버를 구성하세요.
4. 1,024바이트 이하의 TCP(Transmission Control Protocol) 응답을 생성하도록 DNS 서버를 구성하세요.
5. 포트 53을 통한 Amazon RDS DB 인스턴스로부터의 인바운드 트래픽을 허용하도록 DNS 서버를 구성하세요. DNS 서버가 Amazon VPC에 있는 경우, VPC에는 포트 53에서 UDP 및 TCP 트래픽을 허용하는 인바운드 규칙이 포함된 보안 그룹이 있어야 합니다. DNS 서버가 Amazon VPC에 없는 경우, 포트 53에서 UDP 및 TCP 인바운드 트래픽을 허용하는 적절한 방화벽 설정이 있어야 합니다.

   자세한 내용은 VPC의 보안 그룹 및 규칙 추가 및 제거 단원을 참조하세요.
6. 포트 53을 통한 아웃바운드 트래픽을 허용하도록 Amazon RDS DB 인스턴스의 VPC를 구성하세요. VPC에는 포트 53에서 UDP 및 TCP 트래픽을 허용하는 아웃바운드 규칙이 포함된 보안 그룹이 있어야 합니다.

   자세한 내용은 Amazon VPC 사용 설명서의 VPC의 보안 그룹 및 규칙 추가 및 제거를 참조하세요.
7. Amazon RDS DB 인스턴스와 DNS 서버 간 라우팅 경로가 DNS 트래픽을 허용하도록 올바로 구성되는지 확인하세요.

   또한 Amazon RDS DB 인스턴스와 DNS 서버가 같은 VPC에 있지 않은 경우, 그 사이에 피어링 연결을 구축해야 합니다. 자세한 내용은 Amazon VPC Peering Guide의 VPC 피어링이란?을 참조하세요.

RDS for PostgreSQL DB 엔진 업그레이드

PostgreSQL 데이터베이스에서 관리할 수 있는 업그레이드 유형은 다음과 같이 2가지입니다.

- 운영 체제 업데이트 – 경우에 따라 보안 수정 사항이나 OS 변경 사항을 적용하기 위해 Amazon RDS에서 데이터베이스의 기본 운영 체제를 업데이트해야 할 수 있습니다. RDS 콘솔, AWS Command Line Interface(AWS CLI) 또는 RDS API를 사용하여 Amazon RDS에서 OS 업데이트를 적용하는 시기를 결정할 수 있습니다. OS 업데이트에 대한 자세한 내용은 다음단원을 참조하십시오. DB 인스턴스에 업데이트 적용
- 데이터베이스 엔진 업그레이드 – Amazon RDS에서 새 버전의 데이터베이스 엔진이 지원되면 데이터베이스를 새 버전으로 업그레이드할 수 있습니다.

컨텍스트에 포함된 데이터베이스는 RDS for PostgreSQL DB 인스턴스 또는 다중 AZ DB 클러스터입니다.

PostgreSQL 데이터베이스의 엔진 업그레이드에는 메이저 버전 업그레이드와 마이너 버전 업그레이드라는 2가지 종류가 있습니다.

메이저 버전 업그레이드

메이저 버전 업그레이드에는 기존 애플리케이션과 호환되지 않는 데이터베이스 변경 사항이 포함될 수 있습니다. 따라서 데이터베이스의 메이저 버전 업그레이드를 수동으로 수행해야 합니다. DB 인스턴스 또는 다중 AZ DB 클러스터를 수정하여 메이저 버전 업그레이드를 시작할 수 있습니다. 메이저 버전 업그레이드를 수행하기 전에 RDS for PostgreSQL 업그레이드에 대한 메이저 버전 선택에 설명된 단계를 수행하는 것이 좋습니다.

Amazon RDS는 다음과 같은 방법으로 다중 AZ 메이저 버전 업그레이드를 처리합니다.

- 다중 AZ DB 인스턴스 배포 - Amazon RDS가 기본 인스턴스와 대기 인스턴스를 동시에 업그레이드합니다. 업그레이드가 완료되는 동안 데이터베이스를 몇 분간 사용하지 못할 수 있습니다.
- 다중 AZ DB 클러스터 배포 - Amazon RDS가 리더 인스턴스와 라이터 인스턴스를 동시에 업그레이드합니다. 업그레이드가 완료되는 동안 데이터베이스를 몇 분간 사용하지 못할 수 있습니다.

리전 내 읽기 전용 복제본이 있는 DB 인스턴스를 업그레이드하는 경우 Amazon RDS는 기본 DB 인스턴스와 함께 복제본을 업그레이드합니다.

Amazon RDS는 다중 AZ DB 클러스터 읽기 전용 복제본은 업그레이드하지 않습니다. 다중 AZ DB 클러스터의 메이저 버전 업그레이드를 수행하면 읽기 복제본의 복제 상태가 종료로 변경됩니다. 업그레이드가 완료된 후 읽기 전용 복제본은 수동으로 삭제하고 다시 생성해야 합니다.

> **Tip**
>
> 블루/그린 배포를 사용하면 메이저 버전 업그레이드에 필요한 다운타임을 최소화할 수 있습니다. 자세한 내용은 데이터베이스 업데이트에 블루/그린 배포 사용 섹션을 참조하세요.

마이너 버전 업그레이드

반대로 마이너 버전 업그레이드에는 기존 애플리케이션과 호환되는 변경 사항만 포함됩니다. 데이터베이스를 수정하여 마이너 버전 업그레이드를 수동으로 시작할 수 있습니다. 또는 데이터베이스를 생성하거나 수정할 때 마이너 버전 자동 업그레이드 옵션을 활성화할 수 있습니다. 이렇게 하면 Amazon RDS에서 새 버전을 테스트 및 승인한 후 인스턴스가 자동으로 업그레이드됩니다.

Amazon RDS는 다음과 같은 방법으로 다중 AZ 마이너 버전 업그레이드를 처리합니다.

- 다중 AZ DB 인스턴스 배포 - Amazon RDS가 기본 인스턴스와 대기 인스턴스를 동시에 업그레이드합니다. 업그레이드가 완료되는 동안 데이터베이스를 몇 분간 사용하지 못할 수 있습니다.
- 다중 AZ DB 클러스터 배포 - Amazon RDS가 리더 DB 인스턴스를 한 번에 하나씩 업그레이드합니다. 그러면 리더 DB 인스턴스가 새 라이터 DB 인스턴스로 전환됩니다. 그러면 Amazon RDS가 이전 라이터 인스턴스(현재는 리더 인스턴스)를 업그레이드합니다. 다중 AZ DB 클러스터는 일반적으로 마이너 버전 업그레이드의 가동 중지 시간을 약 35초로 줄입니다. RDS 프록시와 함께 사용하면 가동 중지 시간을 1초 이하로 더 줄일 수 있습니다. 자세한 내용은 Amazon RDS Proxy 섹션을 참조하세요. 또는 ProxySQL, PgBouncer 또는 AWS 고급 JDBC 래퍼 드라이버와 같은 오픈 소스 데이터베이스 프록시를 사용할 수 있습니다.

데이터베이스에 읽기 전용 복제본이 있는 경우 소스 인스턴스 또는 클러스터를 업그레이드하기 전에 읽기 전용 복제본을 모두 업그레이드해야 합니다.

자세한 내용은 RDS for PostgreSQL 마이너 버전 자동 업그레이드 섹션을 참조하세요. 마이너 버전 업그레이드를 수동으로 수행하는 방법에 대한 자세한 내용은 엔진 버전 수동 업그레이드 단원을 참조하세요.

데이터베이스 엔진 버전에 대한 자세한 내용 및 낙후된 데이터베이스 엔진 버전 정책에 대한 자세한 내용은 Amazon RDS FAQ의 데이터베이스 엔진 버전을 참조하세요.

주제

- PostgreSQL 업그레이드 고려 사항
- 유효한 업그레이드 대상 찾기

- PostgreSQL 버전 번호
- RDS for PostgreSQL의 RDS 버전 번호
- RDS for PostgreSQL 업그레이드에 대한 메이저 버전 선택
- RDS for PostgreSQL에 대한 메이저 버전 업그레이드를 수행하는 방법
- RDS for PostgreSQL 마이너 버전 자동 업그레이드
- RDS for PostgreSQL 데이터베이스에서 PostgreSQL 확장 업그레이드

PostgreSQL 업그레이드 고려 사항

데이터베이스를 안전하게 업그레이드하기 위해 Amazon RDS는 PostgreSQL 설명서에 설명된 `pg_upgrade` 유틸리티를 사용합니다.

백업 보존 기간이 0보다 큰 경우 업그레이드 프로세스 중에 Amazon RDS가 두 개의 DB 스냅샷을 생성합니다. 첫 번째 DB 스냅샷은 업그레이드 변경 이전 데이터베이스의 스냅샷입니다. 데이터베이스의 업그레이드가 완료되지 않으면 이 스냅샷을 복구하여 기존 버전을 실행하는 데이터베이스를 생성할 수 있습니다. 두 번째 DB 스냅샷은 업그레이드 완료 이후에 캡처됩니다. 이러한 DB 스냅샷은 백업 보존 기간이 만료되면 자동으로 삭제됩니다.

> **Note**
>
> 데이터베이스에 대한 백업 보존 기간을 0보다 큰 수로 설정하는 경우에만 Amazon RDS가 업그레이드 프로세스 중에 DB 스냅샷을 캡처합니다. DB 인스턴스의 백업 보존 기간을 수정하려면 the section called “DB 인스턴스 수정” 단원을 참조하세요. 다중 AZ DB 클러스터의 경우 사용자 지정 백업 보존 기간을 구성할 수 없습니다.

DB 인스턴스의 메이저 버전 업그레이드를 수행하면 리전 내 읽기 전용 복제본도 모두 자동으로 업그레이드됩니다. 업그레이드 워크플로가 시작된 후 읽기 전용 복제본은 기본 DB 인스턴스에서 `pg_upgrade`가 성공적으로 완료될 때까지 기다립니다. 그런 다음 프라이머리 DB 인스턴스 업그레이드는 읽기 전용 복제본 업그레이드가 완료될 때까지 기다립니다. 업그레이드가 완료될 때까지 중단이 발생합니다. 다중 AZ DB 클러스터의 메이저 버전 업그레이드를 수행하면 읽기 전용 복제본의 복제 상태가 종료로 변경됩니다.

업그레이드가 완료된 후에는 이전 버전의 DB 엔진으로 되돌릴 수 없습니다. 이때 이전 버전으로 되돌리려면 업그레이드 전에 캡처한 DB 스냅샷을 복원하여 새로운 데이터베이스를 생성해야 합니다.

유효한 업그레이드 대상 찾기

AWS Management Console을 사용하여 데이터베이스를 업그레이드할 때 데이터베이스의 유효한 업그레이드 대상이 표시됩니다. 또한 다음 AWS CLI 명령을 사용하여 데이터베이스의 유효한 업그레이드 대상을 식별할 수 있습니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-engine-versions \
  --engine postgres \
  --engine-version version-number \
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

Windows의 경우:

```
aws rds describe-db-engine-versions ^
  --engine postgres ^
  --engine-version version-number ^
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

예를 들어 PostgreSQL 버전 16.1 데이터베이스의 유효한 업그레이드 대상을 식별하려면 다음 AWS CLI 명령을 실행합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-engine-versions \
  --engine postgres \
  --engine-version 16.1 \
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

Windows의 경우:

```
aws rds describe-db-engine-versions ^
  --engine postgres ^
  --engine-version 16.1 ^
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

PostgreSQL 버전 번호

PostgreSQL 데이터베이스 엔진의 버전 번호 매기기 순서는 다음과 같습니다.

- PostgreSQL 버전 10 이상의 경우 엔진 버전 번호는 major.minor 형식입니다. 메이저 버전 번호는 버전 번호의 정수 부분입니다. 마이너 버전 번호는 버전 번호의 소수 부분입니다.

  메이저 버전 업그레이드는 10.minor에서 11.minor로의 업그레이드와 같이 버전 번호의 정수 부분이 증가합니다.
- PostgreSQL 버전 10 미만의 경우 엔진 버전 번호는 major.minor 형식입니다. 버전 번호의 정수 부분과 첫 번째 소수 부분 모두가 메이저 엔진 버전 번호입니다. 예를 들어, 9.6이 메이저 버전입니다. 버전 번호의 세 번째 부분이 마이너 버전 번호입니다. 예를 들어, 버전 9.6.12의 경우 12가 마이너 버전 번호입니다.

  메이저 버전 업그레이드는 버전 번호의 메이저 부분이 증가합니다. 예를 들어 9.6.12에서 11.14로 업그레이드하는 것은 메이저 버전 업그레이드입니다. 여기서 9.6과 11이 메이저 버전 번호입니다.

RDS 추가 지원 버전 번호 지정에 대한 자세한 내용은 Amazon RDS 추가 지원 버전 명명 규칙 섹션을 참조하세요.

RDS for PostgreSQL의 RDS 버전 번호

RDS 버전 번호는 *major*.*minor*.*patch* 명명 체계를 사용합니다. RDS 패치 버전에는 릴리스 후 마이너 버전에 추가된 중요한 버그 수정이 포함되어 있습니다. RDS 추가 지원 버전 번호 지정에 대한 자세한 내용은 Amazon RDS 추가 지원 버전 명명 규칙 섹션을 참조하세요.

데이터베이스의 Amazon RDS 버전 번호를 식별하려면 먼저 다음 명령을 사용하여 `rds_tools` 확장을 생성해야 합니다.

```
CREATE EXTENSION rds_tools;
```

PostgreSQL 버전 15.2-R2 릴리스부터 다음 SQL 쿼리를 사용하여 RDS for PostgreSQL 데이터베이스의 RDS 버전 번호를 확인할 수 있습니다.

```
postgres=> SELECT rds_tools.rds_version();
```

예를 들어 RDS PostgreSQL 15.2 데이터베이스를 쿼리하면 다음이 반환됩니다.

```
rds_version
----------------
 15.2.R2
(1 row)
```

RDS for PostgreSQL 업그레이드에 대한 메이저 버전 선택

메이저 버전 업그레이드에는 이전 버전의 데이터베이스와 호환되지 않는 변경 사항이 포함될 수 있습니다. 새 기능을 사용하면 기존 애플리케이션이 올바르게 작동하지 않을 수 있습니다. 따라서 Amazon RDS는 자동으로 메이저 버전 업그레이드를 적용하지 않습니다. 메이저 버전 업그레이드를 수행하려면 데이터베이스를 수동으로 수정합니다. 프로덕션 데이터베이스에 업그레이드를 적용하기 전에 철저하게 테스트하여 애플리케이션이 올바르게 작동하는지 확인해야 합니다. PostgreSQL 메이저 버전 업그레이드를 수행할 때 RDS for PostgreSQL에 대한 메이저 버전 업그레이드를 수행하는 방법에 설명된 단계를 따르는 것이 좋습니다.

PostgreSQL 단일 AZ DB 인스턴스 또는 다중 AZ DB 인스턴스 배포를 다음 메이저 버전으로 업그레이드하면 데이터베이스에 연결된 읽기 전용 복제본도 다음 메이저 버전으로 업그레이드됩니다. 경우에 따라 업그레이드할 때 상위 메이저 버전으로 건너뛸 수 있습니다. 업그레이드에서 메이저 버전을 건너뛰면 읽기 전용 복제본도 해당 대상 메이저 버전으로 업그레이드됩니다. 다른 메이저 버전을 건너뛰어 버전 11로 업그레이드하면 특정 제한 사항이 있습니다. 자세한 내용은 RDS for PostgreSQL에 대한 메이저 버전 업그레이드를 수행하는 방법에 설명된 단계에서 찾을 수 있습니다.

PostgreSQL 엔진 업그레이드가 진행될 때 대부분의 PostgreSQL 확장은 업그레이드되지 않습니다. 확장은 별도로 업그레이드해야 합니다. 자세한 내용은 RDS for PostgreSQL 데이터베이스에서 PostgreSQL 확장 업그레이드 섹션을 참조하세요.

다음 AWS CLI 쿼리를 실행하여 RDS for PostgreSQL 데이터베이스에 사용할 수 있는 메이저 버전을 확인할 수 있습니다.

```
aws rds describe-db-engine-versions --engine postgres  --engine-version your-version
 --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

다음 표에는 사용 가능한 모든 버전에 대해 이 쿼리 결과가 요약되어 있습니다. 버전 번호의 별표(*)는 해당 버전이 더 이상 지원되지 않음을 의미합니다. 현재 버전이 더 이상 지원되지 않는 경우 최신 마이너 버전 업그레이드 대상 또는 해당 버전에 대해 사용 가능한 다른 업그레이드 대상 중 하나로 업그레이드하는 것이 좋습니다.

현재 소스 버전	업그레이드 대상
17.5	없음
17.4	17.5
17.3	17.5, 17.4
17.2	17.5, 17.4, 17.3
17.1	17.5, 17.4, 17.3 , 17.2
16.9	17.5
16.8	17.5, 17.4
16.7	17.5, 17.4, 17.3 16.8
16.6	17.5, 17.4, 17.3 , 17.2 16.8, 16.7
16.5	17.5, 17.4, 17.3, 17.2, 17.1 16.8, 16.7, 16.6
16.4	17.5, 17.4, 17.3, 17.2, 17.1 16.8, 16.7, 16.6, 16.5
16.3	17.5, 17.4, 17.3, 17.2, 17.1 16.8, 16.7, 16.6, 16.5, 16.4
16.2	17.5, 17.4, 17.3, 17.2, 17.1 16.8, 16.7, 16.6, 16.5, 16.4, 16.3
16.1	17.5, 17.4, 17.3, 17.2, 17.1

현재 소스 버전	업그레이드 대상
	16.8, 16.7, 16.6, 16.5, 16.4, 16.3, 16.2
15.13	17.5 16.9
15.12	17.4 16.9, 16.8 15.13
15.11	17.3 16.9, 16.8, 16.7 15.13, 15.12
15.10	17.2 16.9, 16.8, 16.7, 16.6 15.13, 15.12, 15.11
15.9	17.1 16.9, 16.8, 16.7, 16.6, 16.5 15.13, 15.12, 15.11, 15.10
15.8	16.9, 16.8, 16.7, 16.6, 16.5, 16.4 15.13, 15.12, 15.11, 15.10, 15.9
15.7	16.9, 16.8, 16.7, 16.6, 16.5, 16.4, 16.3 15.13, 15.12, 15.11, 15.10, 15.9, 15.8
15.6	16.9, 16.8, 16.7, 16.6, 16.5, 16.4, 16.3, 16.2 15.13, 15.12, 15.11, 15.10, 15.9, 15.8, 15.7

현재 소스 버전	업그레이드 대상
15.5	16.9, 16.8, 16.7, 16.6, 16.5, 16.4, 16.3, 16.2, 16.1 15.13, 15.12, 15.11, 15.10, 15.9, 15.8, 15.7, 15.6
15.4	16.9, 16.8, 16.7, 16.6, 16.5, 16.4, 16.3, 16.2, 16.1 15.13, 15.12, 15.11, 15.10, 15.9, 15.8, 15.7, 15.6, 15.5
15.3*	16.9, 16.8, 16.7, 16.6, 16.5, 16.4, 16.3, 16.2, 16.1 15.13, 15.12, 15.11, 15.10, 15.9, 15.8, 15.7, 15.6, 15.5, 15.4
15.2*	16.9, 16.8, 16.7, 16.6, 16.5, 16.4, 16.3, 16.2, 16.1 15.13, 15.12, 15.11, 15.10, 15.9, 15.8, 15.7, 15.6, 15.5, 15.4
14.18	17.5 16.9 15.13
14.17	17.4 16.8 15.13, 15.12 14.18

현재 소스 버전	업그레이드 대상
14.16	17.3 16.7 15.13, 15.12, 15.11 14.18, 14.17
14.15	17.2 16.6 15.13, 15.12, 15.11, 15.10 14.18, 14.17, 14.16
14.14	17.1 16.5 15.13, 15.12, 15.11, 15.10, 15.9 14.18, 14.17, 14.16, 14.15
14.13	16.4 15.13, 15.12, 15.11, 15.10, 15.9, 15.8 14.18, 14.17, 14.16, 14.15, 14.14
14.12	16.3 15.13, 15.12, 15.11, 15.10, 15.9, 15.8, 15.7 14.18, 14.17, 14.16, 14.15, 14.14, 14.13

현재 소스 버전	업그레이드 대상
14.11	16.2 15.13, 15.12, 15.11, 15.10, 15.9, 15.8, 15.7, 15.6 14.18, 14.17, 14.16, 14.15, 14.14, 14.13, 14.12
14.10	16.1 15.13, 15.12, 15.11, 15.10, 15.9, 15.8, 15.7, 15.6, 15.5 14.18, 14.17, 14.16, 14.15, 14.14, 14.13, 14.12, 14.11
14.9	15.13, 15.12, 15.11, 15.10, 15.9, 15.8, 15.7, 15.6, 15.5, 15.4 14.18, 14.17, 14.16, 14.15, 14.14, 14.13, 14.12, 14.11, 14.10
14.8*	15.13, 15.12, 15.11, 15.10, 15.9, 15.8, 15.7, 15.6, 15.5, 15.4 14.18, 14.17, 14.16, 14.15, 14.14, 14.13, 14.12, 14.11, 14.10, 14.9
14.7*	15.13, 15.12, 15.11, 15.10, 15.9, 15.8, 15.7, 15.6, 15.5, 15.4 14.18, 14.17, 14.16, 14.15, 14.14, 14.13, 14.12, 14.11, 14.10, 14.9
14.6*	15.13, 15.12, 15.11, 15.10, 15.9, 15.8, 15.7, 15.6, 15.5, 15.4 14.18, 14.17, 14.16, 14.15, 14.14, 14.13, 14.12, 14.11, 14.10, 14.9

현재 소스 버전	업그레이드 대상
14.5*	15.13, 15.12, 15.11, 15.10, 15.9, 15.8, 15.7, 15.6, 15.5, 15.4 14.18, 14.17, 14.16, 14.15, 14.14, 14.13, 14.12, 14.11, 14.10, 14.9
14.4*	15.13, 15.12, 15.11, 15.10, 15.9, 15.8, 15.7, 15.6, 15.5, 15.4 14.18, 14.17, 14.16, 14.15, 14.14, 14.13, 14.12, 14.11, 14.10, 14.9
14.3*	15.13, 15.12, 15.11, 15.10, 15.9, 15.8, 15.7, 15.6, 15.5, 15.4 14.18, 14.17, 14.16, 14.15, 14.14, 14.13, 14.12, 14.11, 14.10, 14.9
14.2*	15.13, 15.12, 15.11, 15.10, 15.9, 15.8, 15.7, 15.6, 15.5, 15.4 14.18, 14.17, 14.16, 14.15, 14.14, 14.13, 14.12, 14.11, 14.10, 14.9
14.1*	15.13, 15.12, 15.11, 15.10, 15.9, 15.8, 15.7, 15.6, 15.5, 15.4 14.18, 14.17, 14.16, 14.15, 14.14, 14.13, 14.12, 14.11, 14.10, 14.9
13.21	17.5 16.9 15.13 14.17

현재 소스 버전	업그레이드 대상
13.20	17.4 16.8 15.12 14.18, 14.17 13.21
13.19	17.3 16.7 15.11 14.18, 14.17, 14.16 13.21, 13.20
13.18	16.6 15.10 14.18, 14.17, 14.15 13.21, 13.20, 13.19
13.17	16.5 15.9 14.18, 14.17, 14.15, 14.14 13.21, 13.20, 13.19, 13.18

현재 소스 버전	업그레이드 대상
13.16	16.4 15.8 14.18, 14.17, 14.15, 14.14, 14.13 13.21, 13.20, 13.19, 13.18, 13.17
13.15	16.3 15.8, 15.7 14.18, 14.17, 14.15, 14.14, 14.13, 14.12 13.21, 13.20, 13.19, 13.18, 13.17, 13.16
13.14	16.2 15.6 14.18, 14.17, 14.15, 14.14, 14.13, 14.12, 14.11 13.21, 13.20, 13.19, 13.18, 13.17, 13.16, 13.15
13.13	16.1 15.5 14.18, 14.17, 14.15, 14.14, 14.13, 14.12, 14.11, 14.10 13.21, 13.20, 13.19, 13.18, 13.17, 13.16, 13.15, 13.14

현재 소스 버전	업그레이드 대상
13.12	15.4 14.18, 14.17, 14.15, 14.14, 14.13, 14.12, 14.11, 14.10, 14.9 13.21, 13.20, 13.19, 13.18, 13.17, 13.16, 13.15, 13.14, 13.13
13.11	15.3 14.18, 14.17, 14.15, 14.14, 14.13, 14.12, 14.11, 14.10, 14.9 13.21, 13.20, 13.19, 13.18, 13.17, 13.16, 13.15, 13.14, 13.13, 13.12
13.10*	15.2 14.18, 14.17, 14.15, 14.14, 14.13, 14.12, 14.11, 14.10, 14.9 13.21, 13.20, 13.19, 13.18, 13.17, 13.16, 13.14, 13.13, 13.12, 13.11
13.9*	14.18, 14.17, 14.15, 14.14, 14.13, 14.12, 14.11, 14.10, 14.9 13.21, 13.20, 13.19, 13.18, 13.17, 13.16, 13.14, 13.13, 13.12, 13.11
13.8*	14.18, 14.17, 14.15, 14.14, 14.13, 14.12, 14.11, 14.10, 14.9 13.21, 13.20, 13.19, 13.18, 13.17, 13.16, 13.14, 13.13, 13.12, 13.11

현재 소스 버전	업그레이드 대상
13.7*	14.18, 14.17, 14.15, 14.14, 14.13, 14.12, 14.11, 14.10, 14.9 13.21, 13.20, 13.19, 13.18, 13.17, 13.16, 13.14, 13.13, 13.11
13.6*	14.18, 14.17, 14.15, 14.14, 14.13, 14.12, 14.11, 14.10, 14.9 13.21, 13.20, 13.19, 13.18, 13.17, 13.16, 13.14, 13.13, 13.11
13.5*	14.18, 14.17, 14.15, 14.14, 14.13, 14.12, 14.11, 14.10, 14.9 13.21, 13.20, 13.19, 13.18, 13.17, 13.16, 13.14, 13.13, 13.11
13.4*	14.18, 14.17, 14.15, 14.14, 14.13, 14.12, 14.11, 14.10, 14.9 13.21, 13.20, 13.19, 13.18, 13.17, 13.16, 13.14, 13.13, 13.11
13.3*	14.18, 14.17, 14.15, 14.14, 14.13, 14.12, 14.11, 14.10, 14.9 13.21, 13.20, 13.19, 13.18, 13.17, 13.16, 13.14, 13.13, 13.11
13.2*, 13.1*	14.18, 14.17, 14.15, 14.14, 14.13, 14.12, 14.11, 14.10, 14.9 13.21, 13.20, 13.19, 13.18, 13.17, 13.16, 13.14, 13.13, 13.11

현재 소스 버전	업그레이드 대상
12.22	17.2 16.6 15.10 14.15 13.18
12.21	17.1 16.5 15.9 14.14 13.18, 13.17 12.22
12.20	16.4 15.8 14.13 13.18, 13.17, 13.16 12.22, 12.21
12.19	16.3 15.7 14.12 13.18, 13.17, 13.16, 13.15 12.22, 12.21, 12.20

현재 소스 버전	업그레이드 대상
12.18	16.2 15.6 14.11 13.18, 13.17, 13.16, 13.15, 13.14 12.22, 12.21, 12.20, 12.19
12.17	16.1 15.5 14.10 13.18, 13.17, 13.16, 13.15, 13.14, 13.13 12.22, 12.21, 12.20, 12.19, 12.18
12.16	15.4 14.9 13.18, 13.17, 13.16, 13.15, 13.14, 13.13, 13.12 12.22, 12.21, 12.20, 12.19, 12.18, 12.17
12.15	15.3 14.8 13.18, 13.17, 13.16, 13.15, 13.14, 13.13, 13.12, 13.11 12.22, 12.21, 12.2012.19, 12.18, 12.17, 12.16

현재 소스 버전	업그레이드 대상
12.14*	15.2 14.7 13.18, 13.17, 13.16, 13.15, 13.14, 13.13, 13.12, 13.11 12.22, 12.21, 12.20, 12.19, 12.18, 12.17, 12.16, 12.15
12.13*	14.6 13.18, 13.17, 13.16, 13.15, 13.14, 13.13, 13.12, 13.11 12.22, 12.21, 12.20, 12.19, 12.18, 12.17, 12.16, 12.15
12.12*	14.5 13.18, 13.17, 13.16, 13.15, 13.14, 13.13, 13.12, 13.11 12.22, 12.21, 12.20, 12.19, 12.18, 12.17, 12.16, 12.15
12.11*	14.4 13.18, 13.17, 13.16, 13.15, 13.14, 13.13, 13.12, 13.11 12.22, 12.21, 12.20, 12.19, 12.18, 12.17, 12.16, 12.15

현재 소스 버전	업그레이드 대상
12.10*	14.2 13.18, 13.17, 13.16, 13.15, 13.14, 13.13, 13.12, 13.11 12.22, 12.21, 12.20, 12.19, 12.18, 12.17, 12.16, 12.15
12.9*	14.1 13.18, 13.17, 13.16, 13.15, 13.14, 13.13, 13.12, 13.11 12.22, 12.21, 12.20, 12.19, 12.18, 12.17, 12.16, 12.15
12.8*	13.18, 13.17, 13.16, 13.15, 13.14, 13.13, 13.12, 13.11 12.22, 12.21, 12.20, 12.19, 12.18, 12.17, 12.16, 12.15
12.7*	13.18, 13.17, 13.16, 13.15, 13.14, 13.13, 13.12, 13.11 12.22, 12.21, 12.20, 12.19, 12.18, 12.17, 12.16, 12.15
12.6*, 12.5*, 12.4*, 12.3*, 12.2*	13.18, 13.17, 13.16, 13.15, 13.14, 13.13, 13.12, 13.11 12.22, 12.21, 12.20, 12.19, 12.18, 12.17, 12.16, 12.15

현재 소스 버전	업그레이드 대상
11.22	16.1 15.5 14.10 13.13 12.17 11.22-RDS.20240418

* 이 버전은 더 이상 지원되지 않습니다.

RDS for PostgreSQL에 대한 메이저 버전 업그레이드를 수행하는 방법

Amazon RDS for PostgreSQL 데이터베이스에서 메이저 버전 업그레이드를 수행할 때 다음 프로세스를 따르는 것이 좋습니다.

1. 버전 호환 가능 파라미터 그룹 준비 – 사용자 지정 파라미터 그룹을 사용하는 경우 두 가지 옵션이 있습니다. 새 DB 엔진 버전에 대한 기본 파라미터 그룹을 지정할 수 있습니다. 또는 새 DB 엔진 버전에 대한 사용자 지정 파라미터 그룹을 직접 만들 수도 있습니다. 자세한 내용은 the section called “파라미터 그룹” 및 the section called “DB 클러스터 파라미터 그룹” 단원을 참조하세요.
2. 지원되지 않는 데이터베이스 클래스 확인 – 데이터베이스의 인스턴스 클래스가 업그레이드하려는 PostgreSQL 버전과 호환되는지 확인합니다. 자세한 내용은 DB 인스턴스 클래스에 지원되는 DB 엔진 단원을 참조하십시오.
3. 지원되지 않는 사용 확인:
   - 준비된 트랜잭션 – 업그레이드하기 전에 열려 있는 준비된 트랜잭션을 모두 커밋하거나 롤백합니다.

     다음 쿼리를 사용하여 데이터베이스에 열려 있는 준비된 트랜잭션이 없음을 확인할 수 있습니다.

     ```
     SELECT count(*) FROM pg_catalog.pg_prepared_xacts;
     ```

   - Reg* 데이터 형식 – 업그레이드를 시도하기 전에 reg* 데이터 형식의 사용을 모두 제거하십시오. `regtype` 및 `regclass` 이외에는 reg* 데이터 형식을 업그레이드할 수 없습니다. pg_upgrade

유틸리티는 Amazon RDS에서 업그레이드를 수행하는 데 사용하는 이 데이터 형식을 유지할 수 없습니다.

지원되지 않는 reg* 데이터 형식이 사용되지 않음을 확인하려면 각 데이터베이스에 다음 쿼리를 사용합니다.

```
SELECT count(*) FROM pg_catalog.pg_class c, pg_catalog.pg_namespace n,
 pg_catalog.pg_attribute a
  WHERE c.oid = a.attrelid
      AND NOT a.attisdropped
      AND a.atttypid IN ('pg_catalog.regproc'::pg_catalog.regtype,
                         'pg_catalog.regprocedure'::pg_catalog.regtype,
                         'pg_catalog.regoper'::pg_catalog.regtype,
                         'pg_catalog.regoperator'::pg_catalog.regtype,
                         'pg_catalog.regconfig'::pg_catalog.regtype,
                         'pg_catalog.regdictionary'::pg_catalog.regtype)
      AND c.relnamespace = n.oid
      AND n.nspname NOT IN ('pg_catalog', 'information_schema');
```

4. 잘못된 데이터베이스 확인:

   - 잘못된 데이터베이스가 없는지 확인합니다. pg_database 카탈로그의 datconnlimit 열에는 DROP DATABASE 작업 중에 중단된 데이터베이스를 유효하지 않은 것으로 표시하는 -2 값이 포함되어 있습니다.

     다음 쿼리를 사용하여 잘못된 데이터베이스를 확인합니다.

     ```
     SELECT datname FROM pg_database WHERE datconnlimit = - 2;
     ```

   - 이전 쿼리는 잘못된 데이터베이스 이름을 반환합니다. DROP DATABASE *invalid_db_name*;를 사용하여 잘못된 데이터베이스를 삭제할 수 있습니다. 다음 명령을 사용하여 잘못된 데이터베이스를 삭제할 수도 있습니다.

     ```
     SELECT 'DROP DATABASE ' || quote_ident(datname) || ';' FROM pg_database WHERE
      datconnlimit = -2 \gexec
     ```

   잘못된 데이터베이스에 대한 자세한 내용은 잘못된 데이터베이스가 있는 autovacuum의 동작 이해를 참조하세요.

5. 논리적 복제 슬롯 처리 - 데이터베이스에 논리적 복제 슬롯이 있으면 업그레이드할 수 없습니다. 논리적 복제 슬롯은 일반적으로 AWS DMS 마이그레이션 및 데이터베이스에서 데이터 레이크, BI 도구 및 기타 대상으로 테이블을 복제하는 데 사용됩니다. 업그레이드하기 전에 사용 중인 논리적 복제 슬롯의 용도를 알고 삭제해도 되는지 확인합니다. 논리적 복제 슬롯이 계속 사용 중인 경우 삭제하면 안 되며 업그레이드를 진행할 수 없습니다.

   논리적 복제 슬롯이 필요하지 않은 경우 다음 SQL을 사용하여 삭제할 수 있습니다.

   ```
   SELECT * FROM pg_replication_slots WHERE slot_type NOT LIKE 'physical';
   SELECT pg_drop_replication_slot(slot_name);
   ```

   `pglogical` 확장을 사용하는 논리적 복제 설정에서도 성공적인 메이저 버전 업그레이드를 위해서는 슬롯을 삭제해야 합니다. `pglogical` 확장을 사용하여 생성한 슬롯을 식별하고 삭제하는 방법에 대한 자세한 내용은 RDS for PostgreSQL용 논리적 복제 슬롯 관리 단원을 참조하십시오.

6. 읽기 전용 복제본 처리 – 단일 AZ DB 인스턴스 또는 다중 AZ DB 인스턴스 배포 업그레이드에서는 기본 DB 인스턴스와 함께 리전 내 읽기 전용 복제본도 업그레이드됩니다. Amazon RDS는 다중 AZ DB 클러스터 읽기 전용 복제본은 업그레이드하지 않습니다.

   읽기 전용 복제본은 별도로 업그레이드할 수 없습니다. 가능하다 하더라도 기본 및 복제본 데이터베이스가 서로 다른 PostgreSQL 메이저 버전을 갖는 상황이 발생할 수 있습니다. 그런데 읽기 전용 복제본 업그레이드는 프라이머리 DB 인스턴스의 가동 중지 시간을 증가시킬 수 있습니다. 읽기 전용 복제본 업그레이드를 방지하려면 업그레이드 프로세스를 시작하기 전에 복제본을 독립형 인스턴스로 승격하거나 삭제합니다.

   업그레이드 과정에서 읽기 전용 복제본 인스턴스의 현재 파라미터 그룹을 기반으로 읽기 전용 복제본의 파라미터 그룹이 다시 생성됩니다. 읽기 전용 복제본을 수정하여 업그레이드가 완료된 후에만 사용자 지정 파라미터 그룹을 읽기 전용 복제본에 적용할 수 있습니다. 읽기 전용 복제본에 대한 자세한 내용은 Amazon RDS for PostgreSQL의 읽기 전용 복제본 작업을(를) 참조하세요.

7. 백업 수행 – 메이저 버전 업그레이드를 하기 전에 백업을 수행하여 데이터베이스에 대해 알려진 복원 지점을 생성하는 것이 좋습니다. 백업 보존 기간이 0보다 큰 경우 업그레이드 프로세스는 업그레이드 전후에 데이터베이스의 DB 스냅샷을 생성합니다. 백업 보존 기간을 변경하려면 Amazon RDS DB 인스턴스 수정 및 the section called “다중 AZ DB 클러스터 수정” 단원을 참조하세요.

   백업을 수동으로 수행하려면 the section called “단일 AZ DB 인스턴스용 DB 스냅샷 생성” 및 the section called “다중 AZ DB 클러스터의 스냅샷 생성” 단원을 참조하세요.

8. 메이저 버전 업그레이드 전에 특정 확장 업그레이드 – 업그레이드 시 메이저 버전을 건너뛰려면 메이저 버전 업그레이드를 수행하기 전에 특정 확장을 업데이트해야 합니다. 예를 들어 버전 9.5.x 또

는 9.6에서 버전 11.x로 업그레이드하면 메이저 버전을 건너뜁니다. 업데이트할 확장에는 공간 데이터 처리를 위한 PostGIS 및 관련 확장이 포함됩니다.

- `address_standardizer`
- `address_standardizer_data_us`
- `postgis_raster`
- `postgis_tiger_geocoder`
- `postgis_topology`

`rdkit` 버전 4.6.0 이하 및 PostgreSQL 버전 16 이하를 사용하는 경우 `rdkit` 비호환성으로 인해 PostgreSQL 버전 17로 곧바로 업그레이드할 수 없습니다. 다음은 업그레이드 옵션입니다.

- PostgreSQL 버전 13 이하를 사용하는 경우 먼저 버전 14.14 이상 14 버전, 15.9 이상 15 버전 또는 16.5 이상 16 버전으로 메이저 버전 업그레이드를 수행한 다음 PostgreSQL 17로 버전 업그레이드를 수행해야 합니다.
- PostgreSQL 버전 14, 15 또는 16을 사용하는 경우 먼저 버전 14.14 이상 14 버전, 15.9 이상 15 버전 또는 16.5 이상 16 버전으로 마이너 버전 업그레이드를 수행한 다음 PostgreSQL 버전 17로 업그레이드해야 합니다.

사용 중인 각 확장에 대해 다음 명령을 실행합니다.

```
ALTER EXTENSION PostgreSQL-extension UPDATE TO 'new-version';
```

자세한 내용은 RDS for PostgreSQL 데이터베이스에서 PostgreSQL 확장 업그레이드을 참조하세요. PostGIS 업그레이드에 대한 자세한 내용은 6단계: PostGIS 확장 업그레이드 섹션을 참조하세요.

9. 메이저 버전 업그레이드 전에 특정 확장 삭제 – 메이저 버전을 버전 11.x로 건너뛰는 업그레이드는 `pgRouting` 확장 업데이트를 지원하지 않습니다. 버전 9.4.x, 9.5.x 또는 9.6.x에서 버전 11.x로 업그레이드하면 메이저 버전을 건너뜁니다. `pgRouting` 확장을 삭제한 다음 업그레이드 후에 호환 가능한 버전으로 다시 설치하는 것이 안전합니다. 업데이트할 수 있는 확장 버전에 대한 자세한 내용은 지원되는 PostgreSQL 확장 버전 단원을 참조하십시오.

   PostgreSQL 버전 11 이상에서는 `tsearch2` 및 `chkpass` 확장이 더 이상 지원되지 않습니다. 버전 11.x로 업그레이드하는 경우 업그레이드 전에 `tsearch2` 및 `chkpass` 확장을 삭제합니다.

10. 알 수 없는 데이터 형식 삭제 – 대상 버전에 따라 `unknown` 데이터 형식을 삭제합니다.

PostgreSQL 버전 10은 unknown 데이터 형식에 대한 지원이 중지되었습니다. 버전 9.6 데이터베이스가 unknown 데이터 형식을 사용하는 경우 버전 10으로 업그레이드하면 다음과 같은 오류 메시지가 표시됩니다.

```
Database instance is in a state that cannot be upgraded: PreUpgrade checks failed:
The instance could not be upgraded because the 'unknown' data type is used in user
 tables.
Please remove all usages of the 'unknown' data type and try again."
```

문제가 되는 열을 제거하거나 지원되는 데이터 형식으로 변경할 수 있도록 데이터베이스에서 unknown 데이터 형식을 찾으려면 다음 SQL을 사용합니다.

```
SELECT DISTINCT data_type FROM information_schema.columns WHERE data_type ILIKE
 'unknown';
```

11. 업그레이드 모의 실습 수행 – 프로덕션 데이터베이스에서 업그레이드를 수행하기 전에 프로덕션 데이터베이스의 복제본에서 메이저 버전 업그레이드를 테스트하는 것이 좋습니다. 중복 테스트 데이터베이스의 실행 계획을 모니터링하여 실행 계획 회귀가 발생할 수 있는지 확인하고 성능을 평가할 수 있습니다. 중복 테스트 인스턴스를 만들려면 최근 스냅샷에서 데이터베이스를 복원하거나 데이터베이스의 특정 시점 복원을 수행하여 최근 복원 가능 시간으로 복원할 수 있습니다.

    자세한 내용은 the section called “스냅샷에서 복원” 또는 the section called “시점 복구” 섹션을 참조하세요. 다중 AZ DB 클러스터의 경우 the section called “스냅샷에서 다중 AZ DB 클러스터로 복원” 및 the section called “다중 AZ DB 클러스터를 특정 시점으로 복원” 단원을 참조하세요.

    업그레이드 수행에 대한 자세한 내용은 the section called “엔진 버전 수동 업그레이드” 단원을 참조하세요.

    버전 9.6 데이터베이스를 버전 10으로 업그레이드할 때 PostgreSQL 10은 기본적으로 병렬 쿼리를 활성화한다는 점에 유의하세요. 테스트 데이터베이스의 `max_parallel_workers_per_gather` 파라미터를 2로 변경하여 업그레이드 전에 병렬 처리의 영향을 테스트할 수 있습니다.

    > **Note**
    >
    > `default.postgresql10` DB 파라미터 그룹의 `max_parallel_workers_per_gather` 파라미터 기본값은 2입니다.

자세한 내용은 PostgreSQL 설명서의 Parallel Query(병렬 쿼리)를 참조하세요. 버전 10에서 병렬 처리를 사용 중지하려면 `max_parallel_workers_per_gather` 파라미터를 0으로 설정합니다.

메이저 버전 업그레이드 중에 `public` 및 `template1` 데이터베이스와 각 데이터베이스에 있는 `public` 스키마의 이름이 일시적으로 변경됩니다. 이러한 객체는 원래 이름 뒤에 임의의 문자열이 추가된 상태로 로그에 표시됩니다. 그러면 `locale` 및 `owner`와 같은 사용자 지정 설정이 메이저 버전 업그레이드 중에 유지되도록 문자열이 추가됩니다. 업그레이드가 완료되면 객체의 이름이 원래 이름으로 다시 변경됩니다.

> **Note**
>
> 메이저 버전 업그레이드 프로세스 중에는 DB 인스턴스 또는 다중 AZ DB 클러스터의 특정 시점으로 복원을 수행할 수 없습니다. Amazon RDS에서 업그레이드가 수행된 후 데이터베이스 자동 백업이 수행됩니다. 업그레이드를 시작하기 이전의 시간 및 데이터베이스 자동 백업을 완료한 이후의 시간으로 특정 시점 복원을 수행할 수 있습니다.

12사전 확인 절차 오류로 업그레이드가 실패하는 경우 문제 해결 – 메이저 버전 업그레이드 과정에서 Amazon RDS for PostgreSQL은 먼저 사전 확인 절차를 실행하여 업그레이드 실패를 일으킬 수 있는 문제를 식별합니다. 사전 확인 절차는 인스턴스의 모든 데이터베이스에서 호환되지 않는 모든 잠재적 조건을 확인합니다.

사전 확인 시 문제가 발생하면 업그레이드 사전 확인이 실패했음을 나타내는 로그 이벤트가 생성됩니다. 사전 확인 프로세스 세부 정보는 데이터베이스의 모든 데이터베이스에 대해 `pg_upgrade_precheck.log`라는 업그레이드 로그에 있습니다. Amazon RDS는 파일 이름에 타임스탬프를 추가합니다. 로그 보기에 대한 자세한 내용은 Amazon RDS 로그 파일 모니터링 단원을 참조하세요.

사전 확인 시 복제본 업그레이드가 실패하면 실패한 복제본에 대한 복제가 중단되고 복제본이 종료 상태가 됩니다. 해당 읽기 전용 복제본을 삭제하고 업그레이드된 프라이머리 DB 인스턴스를 기반으로 새 읽기 전용 복제본을 다시 생성합니다.

사전 확인 로그에서 식별된 모든 문제점을 해결한 후 메이저 버전 업그레이드를 다시 시도하세요. 다음은 사전 확인 로그의 예제입니다.

```
-----------------------------------------------------------------------
Upgrade could not be run on Wed Apr 4 18:30:52 2018
-----------------------------------------------------------------------
The instance could not be upgraded from 9.6.11 to 10.6 for the following reasons.
```

```
Please take appropriate action on databases that have usage incompatible with the
 requested major engine version upgrade and try the upgrade again.

* There are uncommitted prepared transactions. Please commit or rollback all prepared
 transactions.* One or more role names start with 'pg_'. Rename all role names that
 start with 'pg_'.

* The following issues in the database 'my"million$"db' need to be corrected before
 upgrading:** The ["line","reg*"] data types are used in user tables. Remove all
 usage of these data types.
** The database name contains characters that are not supported by RDS for
 PostgreSQL. Rename the database.
** The database has extensions installed that are not supported on the target
 database version. Drop the following extensions from your database: ["tsearch2"].

* The following issues in the database 'mydb' need to be corrected before
 upgrading:** The database has views or materialized views that depend on
 'pg_stat_activity'. Drop the views.
```

13. 데이터베이스를 업그레이드하는 동안 읽기 전용 복제본 업그레이드가 실패할 경우 문제 해결 – 실패한 읽기 전용 복제본이 `incompatible-restore` 상태가 되고 데이터베이스에서 복제가 종료됩니다. 해당 읽기 전용 복제본을 삭제하고 업그레이드된 프라이머리 DB 인스턴스를 기반으로 새 읽기 전용 복제본을 다시 생성합니다.

> **Note**
>
> Amazon RDS는 다중 AZ DB 클러스터의 읽기 전용 복제본을 업그레이드하지 않습니다. 다중 AZ DB 클러스터의 메이저 버전 업그레이드를 수행하면, 읽기 전용 복제본의 복제 상태가 종료로 변경됩니다.

다음과 같은 이유로 읽기 전용 복제본 업그레이드가 실패할 수 있습니다.

- 대기 시간이 지난 후에도 프라이머리 DB 인스턴스를 따라잡지 못했습니다.
- storage-full, incompatible-restore 등과 같이 종료 또는 호환되지 않는 수명 주기 상태에 있었습니다.
- 프라이머리 DB 인스턴스 업그레이드가 시작되었을 때 읽기 전용 복제본에서 별도의 마이너 버전 업그레이드가 실행되었습니다.
- 읽기 전용 복제본에서 호환되지 않는 파라미터를 사용했습니다.
- 읽기 전용 복제본이 프라이머리 DB 인스턴스와 통신하여 데이터 폴더를 동기화하지 못했습니다.

14프로덕션 데이터베이스 업그레이드 – 메이저 버전 업그레이드의 모의 실습이 성공한 경우 안심하고 프로덕션 데이터베이스를 업그레이드해도 됩니다. 자세한 내용은 엔진 버전 수동 업그레이드 단원을 참조하십시오.

15ANALYZE 작업을 실행하여 pg_statistic 테이블을 새로 고칩니다. 모든 PostgreSQL 데이터베이스의 각 데이터베이스에 대해 이 작업을 수행해야 합니다. 옵티마이저 통계는 메이저 버전 업그레이드 중에 전송되지 않으므로 성능 문제를 방지하려면 모든 통계를 다시 생성해야 합니다. 파라미터 없이 명령을 실행하여 다음과 같이 현재 데이터베이스의 모든 일반 테이블에 대한 통계를 생성합니다.

```
ANALYZE VERBOSE;
```

VERBOSE 플래그는 선택 사항이지만 이 플래그를 사용하면 진행 상황이 표시됩니다. 자세한 내용은 PostgreSQL 설명서의 ANALYZE를 참조하세요.

> **Note**
>
> 성능 문제를 방지하려면 업그레이드 후 시스템에서 ANALYZE를 실행합니다.

메이저 버전 업그레이드가 완료된 후에는 다음 작업을 수행하는 것이 좋습니다.

- PostgreSQL 업그레이드는 PostgreSQL 확장 버전을 업그레이드하지 않습니다. 확장을 업그레이드하려면 RDS for PostgreSQL 데이터베이스에서 PostgreSQL 확장 업그레이드 단원을 참조하십시오.
- 필요할 경우 Amazon RDS를 사용하여 pg_upgrade 유틸리티에서 생성되는 2개의 로그를 표시합니다. 이러한 로그는 pg_upgrade_internal.log 및 pg_upgrade_server.log입니다. Amazon RDS는 이러한 로그의 파일 이름에 타임스탬프를 추가합니다. 다른 로그와 마찬가지로 이러한 로그를 볼 수 있습니다. 자세한 내용은 Amazon RDS 로그 파일 모니터링 섹션을 참조하세요.

  업그레이드 로그를 Amazon CloudWatch Logs에 업로드할 수도 있습니다. 자세한 내용은 Amazon CloudWatch Logs에 PostgreSQL 로그 게시 섹션을 참조하세요.
- 모든 사항이 예상대로 작동하는지 확인하려면 비슷한 워크로드로 업그레이드된 데이터베이스에서 애플리케이션을 테스트합니다. 업그레이드를 확인한 후 이 테스트 인스턴스를 삭제할 수 있습니다.

RDS for PostgreSQL 마이너 버전 자동 업그레이드

DB 인스턴스 또는 다중 AZ DB 클러스터를 생성하거나 수정할 때 마이너 버전 자동 업그레이드 옵션을 활성화하면 자동으로 데이터베이스가 업그레이드될 수 있습니다.

RDS는 각 RDS for PostgreSQL 메이저 버전마다 하나의 마이너 버전을 자동 업그레이드 버전으로 지정합니다. Amazon RDS가 마이너 버전을 테스트하고 승인하면 유지 관리 기간 중에 자동으로 마이너 버전 업그레이드가 실행됩니다. RDS는 자동으로 새로 릴리스된 마이너 버전을 자동 업그레이드 버전으로 설정하지 않습니다. RDS가 더 새로운 자동 업그레이드 버전을 지정하기 전에 다음과 같은 여러 기준이 고려됩니다.

- 알려진 보안 문제
- PostgreSQL 커뮤니티 버전의 버그
- 마이너 버전 릴리스 이후 전반적인 플릿 안정성

다음 AWS CLI 명령을 사용하여 특정 AWS 리전의 지정된 PostgreSQL 마이너 버전에 대한 현재의 자동 마이너 업그레이드 대상 버전을 확인할 수 있습니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-engine-versions \
--engine postgres \
--engine-version minor-version \
--region region \
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" \
--output text
```

Windows의 경우:

```
aws rds describe-db-engine-versions ^
--engine postgres ^
--engine-version minor-version ^
--region region ^
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" ^
--output text
```

예를 들어 다음 AWS CLI 명령은 미국 동부(오하이오) AWS 리전(us-east-2)의 PostgreSQL 마이너 버전 16.1에 대한 자동 마이너 업그레이드 대상을 안내합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds describe-db-engine-versions \
--engine postgres \
```

```
--engine-version 16.1 \
--region us-east-2 \
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" \
--output table
```

Windows의 경우:

```
aws rds describe-db-engine-versions ^
--engine postgres ^
--engine-version 16.1 ^
--region us-east-2 ^
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" ^
--output table
```

다음과 같은 출력이 표시됩니다.

```
---------------------------------
|    DescribeDBEngineVersions    |
+--------------+-----------------+
|  AutoUpgrade |  EngineVersion  |
+--------------+-----------------+
|  False       |  16.2           |
|  True        |  16.3           |
|  False       |  16.4           |
|  False       |  16.5           |
|  False       |  16.6           |
|  False       |  17.1           |
|  False       |  17.2           |
+--------------+-----------------+
```

이 예시에서 PostgreSQL 버전 16.3의 경우 `AutoUpgrade` 값은 `True`입니다. 따라서 자동 마이너 업그레이드 대상은 출력에서 강조 표시된 PostgreSQL 버전 16.3입니다.

PostgreSQL 데이터베이스는 다음 기준이 충족되면 유지 관리 기간 중에 자동으로 업그레이드됩니다.

- 데이터베이스에 마이너 버전 자동 업그레이드 옵션이 활성화되어 있습니다.
- 데이터베이스가 현재 자동 업그레이드 마이너 버전보다 낮은 DB 엔진 버전을 실행 중입니다.

자세한 내용은 마이너 엔진 버전 자동 업그레이드 단원을 참조하십시오.

> **Note**
>
> PostgreSQL 업그레이드는 PostgreSQL 확장을 업그레이드하지 않습니다. 확장을 업그레이드하려면 RDS for PostgreSQL 데이터베이스에서 PostgreSQL 확장 업그레이드 단원을 참조하십시오.

RDS for PostgreSQL 데이터베이스에서 PostgreSQL 확장 업그레이드

PostgreSQL 엔진 업그레이드는 PostgreSQL 확장을 업그레이드하지 않습니다. 버전 업그레이드 후 확장을 업데이트하려면 `ALTER EXTENSION UPDATE` 명령을 사용합니다.

> **Note**
>
> PostGIS 확장 프로그램 업데이트에 대한 자세한 내용은 PostGIS 확장을 사용하여 공간 데이터 관리(6단계: PostGIS 확장 업그레이드) 섹션을 참조하세요.
> `pg_repack` 확장을 업데이트하려면 확장을 중지한 후 업그레이드된 데이터베이스에 새 버전을 생성합니다. 자세한 내용은 `pg_repack` 설명서의 pg_repack 설치를 참조하세요.

확장 버전을 업그레이드하려면 다음 명령을 사용하세요.

```
ALTER EXTENSION extension_name UPDATE TO 'new_version';
```

지원되는 PostgreSQL 확장 버전 목록은 지원되는 PostgreSQL 확장 버전 단원을 참조하십시오.

현재 설치된 확장을 나열하려면 다음 명령에서 PostgreSQL pg_extension 카탈로그를 사용합니다.

```
SELECT * FROM pg_extension;
```

설치에 사용할 수 있는 특정 확장 버전의 목록을 보려면 다음 명령에서 PostgreSQL pg_available_extension_versions 보기를 사용하십시오.

```
SELECT * FROM pg_available_extension_versions;
```

PostgreSQL DB 스냅샷 엔진 버전 업그레이드

Amazon RDS를 사용하여 PostgreSQL DB 인스턴스의 스토리지 볼륨 DB 스냅샷을 생성할 수 있습니다. 생성되는 DB 스냅샷은 Amazon RDS 인스턴스에서 사용하는 엔진 버전에 기반합니다. DB 스냅샷의 엔진 버전을 업그레이드할 수 있습니다.

새 엔진 버전으로 업그레이드된 DB 스냅샷을 복원한 후에는 업그레이드가 성공적이었는지 테스트해야 합니다. 메이저 버전 업그레이드에 대한 자세한 내용은 RDS for PostgreSQL DB 엔진 업그레이드 단원을 참조하십시오. DB 스냅샷을 복원하는 방법은 DB 인스턴스 복원 단원을 참조하십시오.

암호화되었거나 암호화되지 않은 수동 DB 스냅샷을 업그레이드할 수 있습니다.

RDS for PostgreSQL DB 스냅샷에 사용 가능한 엔진 버전을 보려면 다음 AWS CLI 예제를 사용합니다.

```
aws rds describe-db-engine-versions --engine postgres  --engine-version example-
engine-version --query "DBEngineVersions[*].ValidUpgradeTarget[*].
{EngineVersion:EngineVersion}" --output text --include-all
```

RDS for PostgreSQL DB 스냅샷에 사용할 수 있는 엔진 버전에 대한 자세한 내용은 RDS for PostgreSQL 업그레이드에 대한 메이저 버전 선택 섹션을 참조하세요.

> **Note**
>
> 자동 백업 과정에서 생성되는 자동 DB 스냅샷은 업그레이드할 수 없습니다.

콘솔

DB 스냅샷을 업그레이드하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Snapshots]를 선택합니다.
3. 업그레이드할 스냅샷을 선택합니다.
4. 작업에서 Upgrade snapshot(스냅샷 업그레이드)을 선택합니다. Upgrade snapshot(스냅샷 업그레이드) 페이지가 표시됩니다.

5. 업그레이드할 New engine version(새 엔진 버전)을 선택합니다.
6. 스냅샷을 업그레이드하려면 변경 내용 저장을 선택합니다.

   업그레이드 중에는 이 DB 스냅샷의 모든 스냅샷 작업이 비활성화됩니다. 또한 DB 스냅샷 상태가 사용 가능에서 업그레이드 중으로 바뀐 다음 완료되면 활성으로 바뀝니다 스냅샷 손상 문제로 인해 DB 스냅샷을 업그레이드할 수 없는 경우, 상태가 사용할 수 없음으로 바뀝니다. 이 상태로부터 스냅샷을 복구할 수는 없습니다.

   > **Note**
   >
   > DB 스냅샷 업그레이드에 실패하면 스냅샷이 원래 버전의 원래 상태로 롤백됩니다.

AWS CLI

DB 스냅샷을 새 데이터베이스 엔진 버전으로 업그레이드하려면 AWS CLI modify-db-snapshot 명령을 사용합니다.

파라미터

- `--db-snapshot-identifier` – 업그레이드할 DB 스냅샷의 식별자입니다. 식별자는 고유의 Amazon 리소스 이름(ARN)이어야 합니다. 자세한 내용은 Amazon RDS의 Amazon 리소스 이름(ARN) 섹션을 참조하세요.
- `--engine-version` – DB 스냅샷을 업그레이드할 엔진 버전입니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-snapshot \
    --db-snapshot-identifier my_db_snapshot \
    --engine-version new_version
```

Windows의 경우:

```
aws rds modify-db-snapshot ^
    --db-snapshot-identifier my_db_snapshot ^
    --engine-version new_version
```

RDS API

DB 스냅샷을 새 데이터베이스 엔진 버전으로 업그레이드하려면 Amazon RDS API ModifyDBSnapshot 작업을 호출하십시오.

- `DBSnapshotIdentifier` – 업그레이드할 DB 스냅샷의 식별자입니다. 식별자는 고유의 Amazon 리소스 이름(ARN)이어야 합니다. 자세한 내용은 Amazon RDS의 Amazon 리소스 이름(ARN) 섹션을 참조하세요.
- `EngineVersion` – DB 스냅샷을 업그레이드할 엔진 버전입니다.

Amazon RDS for PostgreSQL의 읽기 전용 복제본 작업

인스턴스에 읽기 복제본을 추가하여 Amazon RDS for PostgreSQL DB 인스턴스에 대한 읽기를 조정할 수 있습니다. 다른 Amazon RDS 데이터베이스 엔진과 마찬가지로 RDS for PostgreSQL은 PostgreSQL의 기본 복제 메커니즘을 사용하여 소스 DB에 대한 변경 사항이 반영되도록 읽기 복제본을 최신 상태로 유지합니다. 읽기 전용 복제본 및 Amazon RDS에 대한 일반적인 정보는 DB 인스턴스 읽기 전용 복제본 작업 섹션을 참조하세요.

다음으로, RDS for PostgreSQL에서의 읽기 전용 복제본 작업 관련 정보를 찾을 수 있습니다.

읽기 복제본의 논리적 디코딩

PostgreSQL용 RDS는 PostgreSQL 16.1을 사용하여 스탠바이 상태에서의 논리적 복제를 지원합니다. 이를 통해 읽기 전용 예비 복제본에서 논리적 디코딩을 생성하여 기본 DB 인스턴스의 부하를 줄일 수 있습니다. 여러 시스템에서 데이터를 동기화해야 하는 애플리케이션의 가용성을 높일 수 있습니다. 이 기능은 데이터 웨어하우스 및 데이터 분석의 성능을 향상합니다.

또한 지정된 예비 복제본의 복제 슬롯은 해당 예비 복제본을 기본 스탠바이로 계속 승격시킵니다. 즉, 기본 DB 인스턴스가 장애 조치 처리되거나 예비 복제본을 새 기본 인스턴스로 승격하는 경우 복제 슬롯은 유지되며 이전 예비 복제본 구독자는 영향을 받지 않습니다.

읽기 전용 복제본에 논리적 디코딩을 생성하려면

1. 논리적 복제 활성화 - 예비 복제본에서 논리적 디코딩을 생성하려면 소스 DB 인스턴스와 물리적 복제본에서 논리적 복제를 활성화해야 합니다. 자세한 내용은 PostgreSQL을 사용한 읽기 전용 복제본 구성 섹션을 참조하세요.

   - 새로 만든 RDS for PostgreSQL DB 인스턴스의 논리적 복제를 활성화하려면 새 DB 사용자 지정 파라미터 그룹을 만들고 정적 파라미터 `rds.logical_replication`을 1로 설정합니다. 그런 다음 이 DB 파라미터 그룹을 소스 DB 인스턴스 및 물리적 읽기 복제본과 연결합니다. 자세한 내용은 Amazon RDS의 DB 인스턴스에 DB 파라미터 그룹 연결 섹션을 참조하세요.
   - PostgreSQL DB 인스턴스용 기존 RDS의 논리적 복제를 활성화하려면 소스 DB 인스턴스의 DB 사용자 지정 파라미터 그룹과 물리적 읽기 복제본을 수정하여 정적 파라미터 `rds.logical_replication`을 1로 설정합니다. 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요.

> **Note**
>
> 이러한 파라미터 변경 사항을 적용하려면 DB 인스턴스를 재부팅해야만 합니다.

다음 쿼리를 사용하여 소스 DB 인스턴스의 `rds.logical_replication`와 물리적 읽기 전용 복제본의 값과 `wal_level` 값을 확인할 수 있습니다.

```
Postgres=>SELECT name,setting FROM pg_settings WHERE name IN
 ('wal_level','rds.logical_replication');

 name                    | setting
-------------------------+---------
 rds.logical_replication | on
 wal_level               | logical
(2 rows)
```

2. 소스 데이터베이스에 테이블 생성 - 소스 DB 인스턴스의 데이터베이스에 연결합니다. 자세한 내용은 PostgreSQL 데이터베이스 엔진을 실행하는 DB 인스턴스에 연결 섹션을 참조하세요.

   다음 쿼리를 사용하여 소스 데이터베이스에 테이블을 만들고 값을 삽입할 수 있습니다.

   ```
   Postgres=>CREATE TABLE LR_test (a int PRIMARY KEY);
   CREATE TABLE
   ```

   ```
   Postgres=>INSERT INTO LR_test VALUES (generate_series(1,10000));
   INSERT 0 10000
   ```

3. 소스 테이블에 대한 발행물 생성 - 다음 쿼리를 사용하여 소스 DB 인스턴스의 테이블에 대한 발행물을 만들 수 있습니다.

   ```
   Postgres=>CREATE PUBLICATION testpub FOR TABLE LR_test;
   CREATE PUBLICATION
   ```

   SELECT 쿼리를 사용하여 소스 DB 인스턴스와 물리적 읽기 복제본 인스턴스 모두에서 생성된 발행의 세부 정보를 확인할 수 있습니다.

```
Postgres=>SELECT * from pg_publication;

oid    | pubname | pubowner | puballtables | pubinsert | pubupdate | pubdelete |
 pubtruncate | pubviaroot
-------+---------+----------+--------------+-----------+-----------+-----------
+-------------+------------
 16429 | testpub |    16413 | f            | t         | t         | t         | t
          | f
(1 row)
```

4. 논리적 복제 인스턴스에서 구독 생성 - RDS for PostgreSQL DB 인스턴스를 논리적 복제 인스턴스로 하나 더 생성합니다. 이 논리적 복제본 인스턴스가 물리적 읽기 복제본 인스턴스에 액세스할 수 있도록 VPC가 올바르게 설정되어 있는지 확인하세요. 자세한 내용은 Amazon VPC 및 Amazon RDS 섹션을 참조하세요. 소스 DB 인스턴스가 유휴 상태인 경우 연결 문제가 발생할 수 있으며 기본 인스턴스는 데이터를 스탠바이 인스턴스로 전송하지 않습니다.

```
Postgres=>CREATE SUBSCRIPTION testsub CONNECTION 'host=Physical replica host name
 port=port
                dbname=source_db_name user=user password=password'
 PUBLICATION testpub;
NOTICE:  created replication slot "testsub" on publisher
CREATE SUBSCRIPTION
```

```
Postgres=>CREATE TABLE LR_test (a int PRIMARY KEY);
CREATE TABLE
```

SELECT 쿼리를 사용하여 논리적 복제본 인스턴스의 구독 세부 정보를 확인할 수 있습니다.

```
Postgres=>SELECT oid,subname,subenabled,subslotname,subpublications FROM
 pg_subscription;

oid    | subname | subenabled | subslotname | subpublications
-------+---------+------------+-------------+-----------------
 16429 | testsub | t          | testsub     | {testpub}
(1 row)
postgres=> select count(*) from LR_test;
 count
-------
 10000
(1 row)
```

5. 논리적 복제 슬롯 상태 검사 - 소스 DB 인스턴스의 물리적 복제 슬롯만 볼 수 있습니다.

```
Postgres=>select slot_name, slot_type, confirmed_flush_lsn from
 pg_replication_slots;

slot_name                                    | slot_type | confirmed_flush_lsn
---------------------------------------------+-----------+---------------------
 rds_us_west_2_db_dhqfsmo5wbbjqrn3m6b6ivdhu4 | physical  |
(1 row)
```

하지만 읽기 복제본 인스턴스에서는 애플리케이션이 논리적 변경을 적극적으로 소비함에 따라 논리적 복제 슬롯과 `confirmed_flush_lsn` 값이 변경되는 것을 확인할 수 있습니다.

```
Postgres=>select slot_name, slot_type, confirmed_flush_lsn from
 pg_replication_slots;

slot_name | slot_type | confirmed_flush_lsn
-----------+-----------+---------------------
 testsub   | logical   | 0/500002F0
(1 row)
```

```
Postgres=>select slot_name, slot_type, confirmed_flush_lsn from
 pg_replication_slots;

slot_name | slot_type | confirmed_flush_lsn
-----------+-----------+---------------------
 testsub   | logical   | 0/5413F5C0
(1 row)
```

PostgreSQL을 사용한 읽기 전용 복제본 제한

PostgreSQL 읽기 전용 복제본에 대한 제한 사항은 다음과 같습니다.

- PostgreSQL 읽기 전용 복제본은 읽기 전용입니다. 읽기 전용 복제본은 쓰기 가능한 DB 인스턴스가 아니지만, 독립 실행형 RDS for PostgreSQL DB 인스턴스로 승격할 수 있습니다. 그러나 이 프로세스는 되돌릴 수 없습니다.

- RDS for PostgreSQL DB 인스턴스에서 14.1 이전 버전의 PostgreSQL을 실행하는 경우에는 다른 읽기 전용 복제본에서 읽기 전용 복제본을 생성할 수 없습니다. RDS for PostgreSQL은 RDS for PostgreSQL 버전 14.1 이상 릴리스에서만 계단식 읽기 전용 복제를 지원합니다. 자세한 내용은 RDS for PostgreSQL에서의 계단식 읽기 전용 복제본 사용 섹션을 참조하세요.
- PostgreSQL 읽기 전용 복제본을 승격하면 읽기 전용 복제본이 쓰기 가능한 DB 인스턴스가 됩니다. 소스 DB 인스턴스에서 미리 쓰기 로그(WAL) 파일 수신이 중단되며, 더 이상 읽기 전용 인스턴스가 아니게 됩니다. RDS for PostgreSQL DB 인스턴스와 마찬가지로 승격된 DB 인스턴스에서 새 읽기 전용 복제본을 생성할 수 있습니다. 자세한 내용은 읽기 전용 복제본을 독립 DB 인스턴스로 승격 섹션을 참조하세요.
- 복제 체인(일련의 계단식 읽기 전용 복제본) 내에서 PostgreSQL 읽기 전용 복제본을 승격하면 기존의 다운스트림 읽기 전용 복제본은 승격된 인스턴스에서 자동으로 WAL 파일을 계속 수신합니다. 자세한 내용은 RDS for PostgreSQL에서의 계단식 읽기 전용 복제본 사용 섹션을 참조하세요.
- 원본 DB 인스턴스에서 사용자 트랜잭션이 발생하지 않는 경우 PostgreSQL 읽기 전용 복제본은 최대 5분까지 복제 지연을 보고합니다. 복제 지연은 `currentTime - lastCommitedTransactionTimestamp`로 계산되며, 이는 처리 중인 트랜잭션이 없을 때 Write-Ahead Log(WAL) 세그먼트가 발생할 때까지 일정 기간 동안 복제 지연 값이 증가함을 의미합니다. 기본적으로 RDS for PostgreSQL는 WAL 세그먼트를 5분마다 전환하므로 트랜잭션 레코드가 생성되고 보고된 지연이 감소합니다.
- RDS for PostgreSQL 14.1 이전 버전의 PostgreSQL 읽기 전용 복제본에 대해서는 자동 백업을 설정할 수 없습니다. 읽기 전용 복제본에 대한 자동 백업은 RDS for PostgreSQL 14.1 이상 버전에서만 지원됩니다. RDS for PostgreSQL 13 이전 버전의 경우 백업하려면 읽기 전용 복제본에서 스냅샷을 생성하면 됩니다.
- 시점 복구(PITR)는 읽기 전용 복제본에 지원되지 않습니다. PITR은 읽기 전용 복제본을 제외한 프라이머리(라이터) 인스턴스에서만 사용할 수 있습니다. 자세한 내용은 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원를 참조하세요.
- PostgreSQL 버전 12 이하의 읽기 전용 복제본은 60~90일의 유지 관리 기간 동안 암호 교체를 적용하기 위해 자동으로 재부팅됩니다. 예약된 재부팅 전에 복제본과 소스 간의 연결이 끊어져도 복제를 재개하기 위해 여전히 재부팅됩니다. PostgreSQL 버전 13 이상의 경우 암호 교체 프로세스 중에 읽기 전용 복제본에서 잠시 복제 연결이 끊어지고 다시 연결되는 현상이 발생할 수 있습니다.

PostgreSQL을 사용한 읽기 전용 복제본 구성

RDS for PostgreSQL은 PostgreSQL의 기본 스트리밍 복제 기능을 사용하여 소스 DB 인스턴스의 읽기 전용 복제본을 생성합니다. 이 읽기 전용 복제본 DB 인스턴스는 비동기식으로 생성된 소스 DB 인스턴스의 물리적 복제본입니다. 이는 소스 DB 인스턴스와 읽기 전용 복제본 간에 미리 쓰기 로그(WAL)

데이터를 전송하는 특수 연결에 의해 생성됩니다. 자세한 내용은 PostgreSQL 설명서에서 스트리밍 복제를 참조하세요.

PostgreSQL은 소스 DB 인스턴스에서 이루어지는 대로 이 보안 연결에 대한 데이터베이스 변경 사항을 비동기식으로 스트리밍합니다. `ssl` 파라미터를 1로 설정하여 클라이언트 애플리케이션에서 소스 DB 인스턴스 또는 읽기 전용 복제본으로의 통신을 암호화할 수 있습니다. 자세한 내용은 PostgreSQL DB 인스턴스와 함께 SSL 사용을 참조하세요.

PostgreSQL은 복제 역할을 사용하여 스트리밍 복제를 실행합니다. 이 역할에는 권한이 부여되지만, 데이터까지 수정하지는 못합니다. PostgreSQL은 단일 프로세스를 통해 복제를 처리합니다.

소스 DB 인스턴스의 작업 또는 사용자에게 영향을 주지 않고 PostgreSQL 읽기 전용 복제본을 생성할 수 있습니다. Amazon RDS는 서비스에 영향을 미치지 않고 소스 DB 인스턴스와 읽기 전용 복제본에 필요한 파라미터와 권한을 설정합니다. 소스 DB 인스턴스의 스냅샷이 생성되고, 이 스냅샷이 읽기 전용 복제본을 생성하는 데 사용됩니다. 향후 언제든지 읽기 전용 복제본을 삭제해도 중단이 발생하지 않습니다.

동일 리전 내의 소스 DB 인스턴스 하나에서 최대 15개까지 읽기 전용 복제본을 생성할 수 있습니다. RDS for PostgreSQL 14.1부터는 소스 DB 인스턴스에서 체인(계단식)에 최대 3가지 수준의 읽기 전용 복제본을 생성할 수도 있습니다. 자세한 내용은 RDS for PostgreSQL에서의 계단식 읽기 전용 복제본 사용 섹션을 참조하세요. 모든 경우에 소스 DB 인스턴스에는 자동 백업이 구성되어야 합니다. 이 작업을 수행하려면 DB 인스턴스의 백업 보존 기간을 0이 아닌 값으로 설정하면 됩니다. 자세한 내용은 읽기 전용 복제본 생성을 참조하세요.

소스 DB 인스턴스와 동일한 AWS 리전에 RDS for PostgreSQL DB 인스턴스의 읽기 전용 복제본을 생성할 수 있습니다. 이를 리전 내 복제라고도 합니다. 소스 DB 인스턴스와 다른 AWS 리전에서 읽기 전용 복제본을 생성할 수도 있습니다. 이를 교차 리전 복제라고도 합니다. 교차 리전 읽기 전용 복제본 설정에 대한 자세한 내용은 다른 AWS 리전에서 읽기 전용 복제본 생성 섹션을 참조하세요. 리전 내 및 교차 리전에 대한 복제 프로세스를 지원하는 다양한 메커니즘은 서로 다른 RDS for PostgreSQL 버전에서 스트리밍 복제가 작동하는 방식에 설명된 대로 RDS for PostgreSQL 버전에 따라 약간 다릅니다.

효과적인 복제를 위해서는 읽기 전용 복제본도 각각 원본 DB 인스턴스와 동일한 양의 컴퓨팅 및 스토리지 리소스를 가져야 합니다. 소스 DB 인스턴스 크기를 조정하는 경우 읽기 전용 복제본 크기도 조정됩니다.

Amazon RDS는 파라미터가 읽기 전용 복제본의 시작을 방해하는 경우 읽기 전용 복제본의 호환되지 않는 파라미터를 재정의합니다. 예를 들어 `max_connections` 파라미터 값이 읽기 전용 복제본보다 원본 DB 인스턴스에서 더 크다고 가정하겠습니다. 이 경우 Amazon RDS가 원본 DB 인스턴스의 파라미터 값이 동일하도록 읽기 전용 복제본의 파라미터를 업데이트합니다.

RDS for PostgreSQL 읽기 전용 복제본은 소스 DB 인스턴스의 외부 데이터 래퍼(FDW)를 통해 사용 가능한 외부 데이터베이스에 액세스할 수 있습니다. 예를 들어 RDS for PostgreSQL DB 인스턴스에서 `mysql_fdw` 래퍼를 사용하여 RDS for MySQL의 데이터에 액세스한다고 가정해봅니다. 이 경우 읽기 전용 복제본도 해당 데이터에 액세스할 수 있습니다. 기타 지원되는 FDW는 `oracle_fdw`, `postgres_fdw`, `tds_fdw`입니다. 자세한 내용은 Amazon RDS for PostgreSQL용 지원되는 외부 데이터 래퍼 작업을 참조하세요.

다중 AZ 구성을 통한 RDS for PostgreSQL 읽기 전용 복제본 사용

단일 AZ 또는 다중 AZ DB 인스턴스를 통해 읽기 전용 복제본을 생성할 수 있습니다. 다중 AZ 배포는 대기 복제본을 통해 중요 데이터의 내구성과 가용성을 개선하는 데 사용할 수 있습니다. 대기 본제본은 소스 DB가 장애 조치될 경우 워크로드를 가정할 수 있는 전담 읽기 전용 복제본입니다. 대기 복제본을 사용하여 읽기 트래픽을 처리할 수 없습니다. 단, 트래픽이 많은 다중 AZ DB 인스턴스에서 읽기 전용 쿼리를 오프로드할 목적으로 읽기 전용 복제본을 생성할 수는 있습니다. 다중 AZ 배포에 대한 자세한 내용은 Amazon RDS에 대한 다중 AZ 인스턴스 배포 섹션을 참조하세요.

다중 AZ 배포의 소스 DB 인스턴스가 대기로 장애 조치되는 경우 연결된 읽기 전용 복제본이 전환되어 대기(이제는 프라이머리)를 복제 소스로 사용합니다. RDS for PostgreSQL 버전에 따라 다음과 같이 읽기 전용 복제본을 다시 시작해야 할 수 있습니다.

- PostgreSQL 13 이상 버전 - 재시작이 필요하지 않습니다. 읽기 전용 복제본이 새 프라이머리 복제본과 자동으로 동기화됩니다. 그러나 경우에 따라 클라이언트 애플리케이션이 읽기 전용 복제본에 대한 도메인 이름 서비스(DNS) 세부 정보를 캐시할 수 있습니다. 이 경우에는 Time-to-Live(TTL) 값을 30초 미만으로 설정합니다. 이렇게 하면 읽기 전용 복제본이 오래된 IP 주소를 보유하지 못하게 되므로 새 프라이머리 복제본과 동기화되지 않습니다. 이 방법과 기타 모범 사례에 대해 자세히 알아보려면 Amazon RDS 기본 운영 지침 섹션을 참조하세요.
- PostgreSQL 12 및 모든 이전 버전 - 대기 복제본(현재 프라이머리)의 IP 주소와 인스턴스 이름이 다르기 때문에 대기 복제본으로 장애 조치 후 읽기 전용 복제본이 자동으로 다시 시작됩니다. 다시 시작하면 읽기 전용 복제본이 새 프라이머리 복제본과 동기화됩니다.

장애 조치에 대해 자세히 알아보려면 Amazon RDS에 대한 다중 AZ DB 인스턴스 장애 조치 섹션을 참조하세요. 다중 AZ 배포에서 읽기 전용 복제본이 작동하는 방식에 대한 자세한 내용은 DB 인스턴스 읽기 전용 복제본 작업 섹션을 참조하세요.

읽기 전용 복제본이 장애 조치를 지원하도록 하려면 Amazon RDS가 다른 가용 영역에 복제본의 대기 복제본을 만들 수 있게 읽기 전용 복제본을 다중 AZ DB 인스턴스로 생성하면 됩니다. 읽기 전용 복제본을 다중 AZ DB 인스턴스로 생성하는 작업은 원본 데이터베이스가 다중 AZ DB 인스턴스인지 여부와는 무관합니다.

RDS for PostgreSQL에서의 계단식 읽기 전용 복제본 사용

버전 14.1부터 RDS for PostgreSQL에서 계단식 읽기 전용 복제본을 지원합니다. 계단식 읽기 전용 복제본을 사용하면 소스 RDS for PostgreSQL DB 인스턴스에 오버헤드를 추가하지 않고도 읽기 전용 복제본 크기를 조정할 수 있습니다. 소스 DB 인스턴스에서는 WAL 로그에 업데이트된 내용을 각 읽기 전용 복제본으로 전송하지 않습니다. 대신 각 계단식 읽기 전용 복제본에서 WAL 로그 업데이트를 함께 구성된 다음 읽기 전용 복제본으로 보냅니다. 이렇게 하면 소스 DB 인스턴스에 가해지는 부담이 줄어듭니다.

계단식 읽기 전용 복제본을 사용하면 RDS for PostgreSQL DB 인스턴스가 WAL 데이터를 체인의 첫 번째 읽기 전용 복제본으로 전송합니다. 뒤이어 해당 읽기 전용 복제본이 WAL 데이터를 체인의 두 번째 복제본으로 전송하는 식으로 이루어집니다. 결과적으로 체인의 모든 읽기 전용 복제본이 소스 DB 인스턴스에만 오버헤드가 발생하는 일 없이 RDS for PostgreSQL DB 인스턴스에서 변경됩니다.

소스 RDS for PostgreSQL DB 인스턴스에서 체인에 최대 3개의 읽기 전용 복제본을 생성할 수 있습니다. 예를 들어 RDS for PostgreSQL 14.1 DB 인스턴스, `rpg-db-main`이 있다고 가정해봅니다. 다음을 수행할 수 있습니다.

- `rpg-db-main`부터 시작해서 체인에 첫 번째 읽기 전용 복제본 `read-replica-1`을 생성합니다.
- 다음으로 `read-replica-1`에서 체인에 다음 읽기 전용 복제본 `read-replica-2`를 생성합니다.
- 마지막으로 `read-replica-2`에서 체인에 세 번째 읽기 전용 복제본 `read-replica-3`을 생성합니다.

체인에서 `rpg-db-main`에 대한 세 번째 계단식 읽기 전용 복제본 다음으로 또 다른 읽기 전용 복제본을 생성할 수 없습니다. RDS for PostgreSQL 소스 DB 인스턴스부터 계단식 읽기 전용 복제본 체인의 마지막에 이르는 전체 인스턴스는 최대 4개의 DB 인스턴스로 구성될 수 있습니다.

읽기 전용 복제본을 계단식으로 실행하려면 RDS for PostgreSQL에서 자동 백업을 설정합니다. 먼저 읽기 전용 복제본을 생성한 다음 RDS for PostgreSQL DB 인스턴스에서 자동 백업을 켜면 됩니다. 이 프로세스는 다른 Amazon RDS DB 엔진에서와 동일합니다. 자세한 내용은 읽기 전용 복제본 생성을 참조하세요.

모든 읽기 전용 복제본과 마찬가지로 계단식 구성에 포함된 읽기 전용 복제본을 승격할 수 있습니다. 읽기 전용 복제본 체인의 한 읽기 전용 복제본을 승격하면 체인에서 해당 복제본이 제거됩니다. 예를 들어 `rpg-db-main` DB 인스턴스의 일부 워크로드를 회계 부서에서만 사용할 수 있도록 새 인스턴스로 옮기려고 합니다. 이 예제에서 3개의 읽기 전용 복제본 체인이 있다고 가정하고 `read-replica-2`를 승격하기로 결정합니다. 체인은 다음과 같이 변화합니다.

- `read-replica-2`를 승격하면 복제 체인에서 제거됩니다.
  - 이제 전체 읽기/쓰기 DB 인스턴스가 됩니다.
  - 승격 전과 마찬가지로 `read-replica-3`으로 계속 복제합니다.
- `rpg-db-main`은 `read-replica-1`로 계속 복제를 진행합니다.

읽기 전용 복제본 승격에 대한 자세한 내용은 읽기 전용 복제본을 독립 DB 인스턴스로 승격 섹션을 참조하세요.

> **Note**
>
> - RDS for PostgreSQL은 계단식 복제본에 대한 메이저 버전 업그레이드를 지원하지 않습니다. 메이저 버전 업그레이드를 수행하기 전에 계단식 복제본을 제거해야 합니다. 소스 DB 인스턴스 및 1단계 복제본에서 업그레이드를 완료한 후 다시 만들 수 있습니다.
> - 계단식 읽기 전용 복제본의 경우 RDS for PostgreSQL은 첫 번째 복제 수준에서 각 소스 DB 인스턴스에 대해 15개의 읽기 전용 복제본을 지원하고, 두 번째 및 세 번째 복제 수준에서 각 소스 DB 인스턴스에 대해 5개의 읽기 전용 복제본을 지원합니다.

RDS for PostgreSQL로 교차 리전 계단식 읽기 복제본 생성

RDS for PostgreSQL은 교차 리전 계단식 읽기 복제본을 지원합니다. 소스 DB 인스턴스에서 교차 리전 복제본을 생성한 다음, 이 복제본에서 동일 리전 복제본을 생성할 수 있습니다. 소스 DB 인스턴스에서 동일한 리전 복제본을 생성한 다음, 이 복제본에서 교차 리전 복제본을 생성할 수 있습니다.

교차 리전 복제본을 생성한 다음 동일 리전 복제본 생성

버전 14.1 이상의 RDS for PostgreSQL DB 인스턴스, `rpg-db-main` 코드를 사용하여 다음을 수행할 수 있습니다.

1. `rpg-db-main`(US-EAST-1)로 시작하여 체인에서 첫 번째 교차 리전 읽기 복제본인 `read-replica-1`(US-WEST-2)를 생성합니다.
2. 첫 번째 교차 리전 `read-replica-1`(US-WEST-2)을 사용하여 체인에 두 번째 읽기 복제본 `read-replica-2`(US-WEST-2)를 생성합니다.
3. `read-replica-2`를 사용하여 체인에서 세 번째 읽기 복제본 `read-replica-3`(US-WEST-2)을 생성합니다.

동일 리전 복제본을 생성한 다음 교차 리전 복제본 생성

버전 14.1 이상의 RDS for PostgreSQL DB 인스턴스, `rpg-db-main` 코드를 사용하여 다음을 수행할 수 있습니다.

1. `rpg-db-main`(US-EAST-1)로 시작하여 체인에서 첫 번째 읽기 복제본인 `read-replica-1`(US-EAST-1)을 생성합니다.
2. `read-replica-1`(US-EAST-1)을 사용하여 체인에서 첫 번째 교차 리전 읽기 복제본인 `read-replica-2`(US-WEST-2)를 생성합니다.
3. `read-replica-2`(US-WEST-2)를 사용하여 체인에서 세 번째 읽기 복제본인 `read-replica-3`(US-WEST-2)을 생성합니다.

교차 리전 읽기 복제본 생성의 제한 사항

- 데이터베이스 복제본의 교차 리전 캐스케이딩 체인은 최대 2개의 리전에 걸쳐 최대 4개의 레벨로 구성할 수 있습니다. 4가지 수준에는 데이터베이스 소스와 3개의 읽기 복제본이 포함됩니다.

계단식 읽기 복제본 사용의 장점

- 읽기 확장성 향상 - 읽기 쿼리를 여러 복제본에 분산하여 캐스케이딩 복제를 통해 부하를 분산할 수 있습니다. 이렇게 하면 쓰기 데이터베이스의 부담을 줄여 특히 읽기가 많은 애플리케이션의 성능이 향상됩니다.
- 지리적 분포 - 계단식 복제본은 서로 다른 지리적 위치에 위치할 수 있습니다. 이렇게 하면 기본 데이터베이스에서 멀리 떨어져 있는 사용자의 대기 시간이 줄어들고 로컬 읽기 복제본을 제공하여 성능과 사용자 경험이 향상됩니다.
- 고가용성 및 재해 복구 - 기본 서버에 장애가 발생하면 복제본을 기본 서버로 승격하여 연속성을 보장할 수 있으며, 계단식 복제는 여러 계층의 장애 복구 옵션을 제공하여 시스템의 전반적인 복원력을 향상시킴으로써 이를 더욱 강화합니다.
- 유연성 및 모듈식 성장 - 시스템이 성장함에 따라 기본 데이터베이스를 크게 재구성하지 않고도 다양한 수준에서 새로운 복제본을 추가할 수 있습니다. 이러한 모듈식 접근 방식을 통해 복제 설정을 확장하고 관리할 수 있습니다.

교차 리전 읽기 복제본 사용 모범 사례

- 복제본을 승격하기 전에 추가 복제본을 생성하세요. 이렇게 하면 시간을 절약하고 워크로드를 효율적으로 처리할 수 있습니다.

서로 다른 RDS for PostgreSQL 버전에서 스트리밍 복제가 작동하는 방식

PostgreSQL을 사용한 읽기 전용 복제본 구성에서 설명한 바와 같이, RDS for PostgreSQL은 PostgreSQL의 기본 스트리밍 복제 프로토콜을 사용하여 소스 DB 인스턴스에서 WAL 데이터를 전송합니다. 소스 WAL 데이터를 리전 내 읽기 전용 복제본과 교차 리전 읽기 전용 복제본으로 전송합니다. 버전 9.4에서는 PostgreSQL에 복제 프로세스를 위한 지원 메커니즘으로 물리적 복제 슬롯이 도입되었습니다.

물리적 복제 슬롯은 소스 DB 인스턴스가 모든 읽기 전용 복제본에서 WAL 데이터를 사용하기 전에 WAL 데이터를 제거하지 못하도록 합니다. 각 읽기 전용 복제본은 소스 DB 인스턴스에 자체 물리적 슬롯이 있습니다. 슬롯은 복제본에 필요할 수 있는 가장 오래된 WAL(논리 시퀀스 번호(LSN)별)을 추적합니다. 모든 슬롯과 DB 연결이 지정된 WAL(LSN) 이상으로 진행된 후에는 해당 LSN이 다음 체크포인트에서 제거할 수 있는 후보가 됩니다.

Amazon RDS는 Amazon S3를 사용하여 WAL 데이터를 아카이브합니다. 리전 내 읽기 전용 복제본의 경우 필요하다면 아카이브된 데이터를 사용하여 읽기 전용 복제본을 복구할 수 있습니다. 소스 DB와 읽기 전용 복제본 간의 연결이 어떤 이유로든 중단되는 경우에 복구가 필요할 수 있습니다.

다음 테이블에서는 PostgreSQL 버전과 RDS for PostgreSQL에서 사용하는 리전 내 및 교차 리전에 대한 지원 메커니즘 간의 차이점을 간략히 확인할 수 있습니다.

버전	리전 내	교차 리전
PostgreSQL 14.1 이상 버전	• 복제 슬롯 • Amazon S3 아카이브	• 복제 슬롯
PostgreSQL 13 이하 버전	• Amazon S3 아카이브	• 복제 슬롯

자세한 내용은 복제 프로세스 모니터링 및 튜닝을 참조하세요.

PostgreSQL 복제를 제어하는 파라미터 이해

다음 파라미터는 복제 프로세스에 영향을 미치며, 읽기 전용 복제본이 소스 DB 인스턴스의 최신 상태를 유지하는 정도를 결정합니다.

max_wal_senders

`max_wal_senders` 파라미터는 스트리밍 복제 프로토콜을 통해 소스 DB 인스턴스가 동시에 지원할 수 있는 최대 연결 수를 지정합니다.

기본값은 다음과 같이 RDS for PostgreSQL 버전에 따라 다릅니다.

- 버전 13, 14 및 15의 경우 기본값은 20입니다.
- 버전 16 이상의 경우 기본값은 35입니다.

이 파라미터는 실제 읽기 전용 복제본 수보다 약간 더 높게 설정되어야 합니다. 이 파라미터를 읽기 전용 복제본의 수에 비해 너무 낮게 설정하면 복제가 중지됩니다.

자세한 내용은 PostgreSQL 설명서에서 max_wal_senders 섹션을 참조하세요.

> **Note**
>
> `max_wal_senders`는 변경 사항을 적용하려면 DB 인스턴스 재부팅이 필요한 정적 파라미터입니다.

wal_keep_segments

`wal_keep_segments` 파라미터는 소스 DB 인스턴스에서 `pg_wal` 디렉터리에 보관하는 미리 쓰기 로그(WAL) 파일의 수를 지정합니다. 기본 설정은 32입니다.

`wal_keep_segments`가 배포에 대해 충분히 큰 값으로 설정되지 않으면 읽기 전용 복제본이 따라잡지 못하면서 스트리밍 복제가 중지될 수 있습니다. 이 경우에는 Amazon RDS가 복제 오류를 생성하고 읽기 전용 복제본의 복구를 시작합니다. 이렇게 하려면 Amazon S3에서 소스 DB 인스턴스의 아카이브된 WAL 데이터를 재실행하면 됩니다. 이 복구 프로세스는 읽기 전용 복제본이 스트리밍 복제를 이어갈 만큼 충분히 따라잡을 때까지 계속됩니다. 예: 복제 중단에서 읽기 전용 복제본을 복구하는 방법에서 이 프로세스가 PostgreSQL 로그에서 캡처된 대로 실행 중임을 확인할 수 있습니다.

> **Note**
>
> PostgreSQL 버전 13에서 `wal_keep_segments` 파라미터 이름은 `wal_keep_size`입니다. `wal_keep_segments`와 동일한 목적을 수행하지만, 기본값은 파일 수가 아닌 메가바이트(MB)(2,048MB)입니다. 자세한 내용은 PostgreSQL 설명서에서 wal_keep_segments 및 wal_keep_size를 참조하세요.

max_slot_wal_keep_size

`max_slot_wal_keep_size` 파라미터는 RDS for PostgreSQL DB 인스턴스가 슬롯을 제공하기 위해 `pg_wal` 디렉터리에 유지하는 WAL 데이터의 양을 제어합니다. 이 파라미터는 복제 슬롯

을 사용하는 구성에 사용됩니다. 이 파라미터의 기본값은 -1입니다. 즉, 소스 DB 인스턴스에 보관되는 WAL 데이터의 양에는 제한이 없습니다. 복제 슬롯 모니터링에 대한 자세한 내용은 RDS for PostgreSQL DB 인스턴스에 대한 복제 슬롯 모니터링 섹션을 참조하세요.

이 파라미터에 대한 자세한 내용은 PostgreSQL 설명서에서 max_slot_wal_keep_size를 참조하세요.

읽기 전용 복제본에 WAL 데이터를 제공하는 스트림이 중단될 때마다 PostgreSQL은 복구 모드로 전환됩니다. Amazon S3의 아카이브된 WAL 데이터를 사용하거나 복제 슬롯과 연결된 WAL 데이터를 사용하여 읽기 전용 복제본을 복원합니다. 프로세스가 완료되면 PostgreSQL이 스트리밍 복제를 다시 구성합니다.

예: 복제 중단에서 읽기 전용 복제본을 복구하는 방법

다음 예제에서는 읽기 전용 복제본에 대한 복구 프로세스를 보여 주는 로그 세부 정보를 찾아볼 수 있습니다. 이 예제는 소스 DB와 같은 AWS 리전에서 PostgreSQL 버전 12.9를 실행하는 RDS for PostgreSQL DB 인스턴스에서 비롯되었으므로, 복제 슬롯이 사용되지 않습니다. 복구 프로세스는 리전 내 읽기 전용 복제본이 있는 14.1 이전 버전의 PostgreSQL을 실행하는 다른 RDS for PostgreSQL DB 인스턴스와 동일합니다.

읽기 전용 복제본과 소스 DB 인스턴스의 연결이 끊어졌을 때 Amazon RDS는 `ERROR: requested WAL segment ... has already been removed`와 함께 `FATAL: could not receive data from WAL stream` 메시지로 로그에 문제를 기록합니다. 굵은 줄에 나와 있는 것처럼 Amazon RDS는 아카이브된 WAL 파일을 재생하여 복제본을 복구합니다.

```
2014-11-07 19:01:10 UTC::@:[23180]:DEBUG:  switched WAL source from archive to stream
 after failure
2014-11-07 19:01:10 UTC::@:[11575]:LOG: started streaming WAL from primary at 1A/
D3000000 on timeline 1
2014-11-07 19:01:10 UTC::@:[11575]:FATAL: could not receive data from WAL stream:
ERROR:  requested WAL segment 000000010000001A000000D3 has already been removed
2014-11-07 19:01:10 UTC::@:[23180]:DEBUG: could not restore file "00000002.history"
 from archive: return code 0
2014-11-07 19:01:15 UTC::@:[23180]:DEBUG: switched WAL source from stream to archive
 after failure recovering 000000010000001A000000D3
2014-11-07 19:01:16 UTC::@:[23180]:LOG:  restored log file "000000010000001A000000D3"
 from archive
```

Amazon RDS가 복제본에서 아카이빙된 WAL 데이터를 따라잡을 수 있을 만큼 충분히 재생하면 읽기 전용 복제본으로의 스트리밍이 다시 시작됩니다. Amazon RDS는 스트리밍이 재개되면 다음과 유사한 로그 파일에 항목을 씁니다.

```
2014-11-07 19:41:36 UTC::@:[24714]:LOG:started streaming WAL from primary at 1B/
B6000000 on timeline 1
```

공유 메모리를 제어하는 파라미터 설정

설정한 파라미터에 따라 트랜잭션 ID, 잠금 및 준비된 트랜잭션을 추적하기 위한 공유 메모리 크기가 결정됩니다. 대기 인스턴스의 공유 메모리 구조는 기본 인스턴스의 공유 메모리 구조와 같거나 커야 합니다. 이렇게 하면 복구 중에 전자의 공유 메모리가 부족해지는 경우를 방지할 수 있습니다. 복제본의 파라미터 값이 기본 복제본의 파라미터 값보다 작으면 Amazon RDS에서는 자동으로 복제본 파라미터를 조정하고 엔진을 다시 시작합니다.

영향을 받는 파라미터는 다음과 같습니다.

- max_connections
- max_worker_processes
- max_wal_senders
- max_prepared_transactions
- max_locks_per_transaction

메모리 부족으로 인한 복제본의 RDS 재부팅을 방지하려면 각 복제본에 파라미터 변경 사항을 롤링 재부팅으로 적용하는 것이 좋습니다. 파라미터를 설정할 때 다음 규칙을 적용해야 합니다.

- 파라미터 값 늘리기:
  - 항상 먼저 모든 읽기 전용 복제본의 파라미터 값을 늘리고 모든 복제본을 롤링 재부팅해야 합니다. 그런 다음 기본 인스턴스에 파라미터 변경 사항을 적용하고 재부팅합니다.
- 파라미터 값 줄이기:
  - 먼저 기본 인스턴스의 파라미터 값을 줄이고 재부팅을 수행해야 합니다. 그런 다음 모든 관련 읽기 전용 복제본에 파라미터 변경 사항을 적용하고 롤링 재부팅해야 합니다.

복제 프로세스 모니터링 및 튜닝

RDS for PostgreSQL DB 인스턴스와 읽기 전용 복제본을 정기적으로 모니터링하는 것이 좋습니다. 읽기 전용 복제본이 소스 DB 인스턴스의 변경 사항을 따르도록 해야 합니다. Amazon RDS는 복제 프로세스가 중단될 때 읽기 전용 복제본을 명료하게 복구합니다. 그러나 복구할 필요가 아예 없도록 하는 것이 가장 좋습니다. 복제 슬롯을 사용하여 복구하는 것이 Amazon S3 아카이브를 사용하는 것보다 빠르지만, 복구 프로세스는 읽기 성능에 영향을 줄 수 있습니다.

다음을 수행하여 읽기 전용 복제본이 소스 DB 인스턴스를 얼마나 잘 따라잡는지 확인할 수 있습니다.

- 소스 DB 인스턴스와 복제본 간 **ReplicaLag**의 양을 확인합니다 복제본 지연은 읽기 전용 복제본이 소스 DB 인스턴스보다 지연되는 시간(초)입니다. 이 지표는 다음 쿼리 결과를 보고합니다.

  ```
  SELECT extract(epoch from now() - pg_last_xact_replay_timestamp()) AS "ReplicaLag";
  ```

  복제본 지연은 읽기 전용 복제본이 소스 DB 인스턴스를 얼마나 잘 따라잡는지 나타냅니다. 이는 소스 DB 인스턴스와 특정 읽기 전용 인스턴스 간의 지연 시간입니다. 복제본 지연 값이 높으면 소스 DB 인스턴스와 해당 읽기 전용 복제본이 사용하는 DB 인스턴스 클래스나 스토리지 유형(또는 둘 다) 간에 불일치가 있다는 의미일 수 있습니다. DB 소스 인스턴스와 모든 읽기 전용 복제본의 DB 인스턴스 클래스 및 스토리지 유형은 동일해야 합니다.

  복제본 지연은 간헐적인 연결 문제의 결과로 나타날 수도 있습니다. Amazon CloudWatch에서 Amazon RDS `ReplicaLag` 지표를 보고 복제 지연을 모니터링할 수 있습니다. Amazon RDS의 `ReplicaLag` 및 기타 지표에 대한 자세한 내용은 Amazon RDS에 대한 Amazon CloudWatch 지표 섹션을 참조하세요.
- 설정을 조절하는 데 사용할 수 있는 정보는 PostgreSQL 로그를 확인하세요. 모든 체크포인트에서 PostgreSQL 로그는 다음 예제와 같이 재사용된 트랜잭션 로그 파일 수를 캡처합니다.

  ```
  2014-11-07 19:59:35 UTC::@:[26820]:LOG:  checkpoint complete: wrote 376 buffers
   (0.2%);
  0 transaction log file(s) added, 0 removed, 1 recycled; write=35.681 s, sync=0.013 s,
   total=35.703 s;
  sync files=10, longest=0.013 s, average=0.001 s
  ```

  이 정보를 사용하여 지정된 기간 동안 얼마나 많은 트랜잭션 파일이 재사용되는지 파악할 수 있습니다. 필요할 경우 `wal_keep_segments` 설정을 변경할 수 있습니다. 예를 들어 `checkpoint complete` 시 PostgreSQL 로그에 5분 간격으로 `35 recycled`가 표시된다고 가정합니다. 이 경우

wal_keep_segments 기본값인 32는 스트리밍 활동에 보조를 맞추기에 충분하지 않으므로 이 파라미터의 값을 높여야 합니다.

- Amazon CloudWatch를 사용하여 복제 문제를 예측할 수 있는 지표를 모니터링합니다. PostgreSQL 로그를 직접 분석하는 대신 Amazon CloudWatch를 사용하여 수집된 지표를 확인할 수 있습니다. 예를 들어 TransactionLogsGeneration 지표의 값을 확인하여 소스 DB 인스턴스에서 얼마나 많은 WAL 데이터를 생성 중인지 확인 가능합니다. DB 인스턴스의 워크로드로 인해 많은 양의 WAL 데이터가 생성되는 경우가 있습니다. 그렇다면 소스 DB 인스턴스와 읽기 전용 복제본의 DB 인스턴스 클래스를 변경해야 할 수 있습니다. 네트워크 성능이 높은(10Gbps) 인스턴스 클래스를 사용하면 복제본 지연이 줄어들 수 있습니다.

RDS for PostgreSQL DB 인스턴스에 대한 복제 슬롯 모니터링

모든 RDS for PostgreSQL 버전은 교차 리전 읽기 전용 복제본의 복제 슬롯을 사용합니다. RDS for PostgreSQL 14.1 이상 버전은 리전 내 읽기 전용 복제본의 복제 슬롯을 사용합니다. 또한 리전 내 읽기 전용 복제본은 Amazon S3를 사용하여 WAL 데이터를 아카이브합니다. 즉, DB 인스턴스와 읽기 전용 복제본이 PostgreSQL 14.1 이상을 실행하는 경우 복제 슬롯과 Amazon S3 아카이브를 모두 사용하여 읽기 전용 복제본을 복구할 수 있습니다. 복제 슬롯을 사용하여 읽기 전용 복제본을 복구하는 것이 Amazon S3 아카이브에서 복구하는 것보다 빠릅니다. 따라서 복제 슬롯 및 관련 지표를 모니터링하는 것이 좋습니다.

다음과 같이 pg_replication_slots 보기를 쿼리하여 RDS for PostgreSQL DB 인스턴스의 복제 슬롯을 볼 수 있습니다.

```
postgres=> SELECT * FROM pg_replication_slots;
slot_name                  | plugin | slot_type | datoid | database | temporary |
 active | active_pid | xmin | catalog_xmin | restart_lsn | confirmed_flush_lsn |
 wal_status | safe_wal_size | two_phase
---------------------------+--------+-----------+--------+----------+-----------
+--------+------------+------+--------------+-------------+---------------------
+------------+---------------+-----------
rds_us_west_1_db_555555555 |        | physical  |        |          | f         | t
  |      13194 |      |              | 23/D8000060 |                     | reserved   |
               | f
(1 row)
```

reserved 값의 wal_status는 슬롯이 보유한 WAL 데이터의 양이 max_wal_size 파라미터의 범위 안에 있음을 나타냅니다. 즉, 복제 슬롯의 크기가 적절하다는 뜻입니다. 지원되는 다른 상태 값은 다음과 같습니다.

- `extended` - 슬롯이 `max_wal_size` 설정을 초과하지만, WAL 데이터는 유지됩니다.
- `unreserved` - 슬롯에 더 이상 어떤 필수 WAL 데이터도 없습니다. 이중 일부는 다음 체크포인트에서 제거됩니다.
- `lost` - 일부 필수 WAL 데이터가 제거되었습니다. 슬롯을 더 이상 사용할 수 없습니다.

`wal_status`의 `unreserved` 및 `lost` 상태는 `max_slot_wal_keep_size`가 음수가 아닌 경우에만 표시됩니다.

`pg_replication_slots` 보기에는 복제 슬롯의 현재 상태가 표시됩니다. Amazon CloudWatch를 통해 다음 지표를 모니터링하여 복제 슬롯의 성능을 평가할 수 있습니다.

- **`OldestReplicationSlotLag`** - 가장 지연된 복제본에서 사용하지 않은 소스의 Write-Ahead Log(WAL) 데이터의 양을 표시합니다.
- **`TransactionLogsDiskUsage`** - WAL 데이터에 얼마나 많은 스토리지가 사용되고 있는지 보여줍니다. 읽기 전용 복제본이 크게 지연되면 이 지표의 값이 크게 증가할 수 있습니다.

Amazon CloudWatch와 RDS for PostgreSQL에 대한 해당 지표 사용에 대해 자세히 알아보려면 Amazon CloudWatch로 Amazon RDS 지표 모니터링 섹션을 참조하세요. RDS for PostgreSQL DB 인스턴스의 스트리밍 복제 모니터링에 대한 자세한 내용은 AWS 데이터베이스 블로그에서 Amazon RDS PostgreSQL 복제의 모범 사례를 참조하세요.

RDS for PostgreSQL의 읽기 전용 복제본 문제 해결

다음에서 몇 가지 일반적인 RDS for PostgreSQL의 읽기 전용 복제본 문제에 대한 문제 해결 아이디어를 찾아볼 수 있습니다.

읽기 전용 복제본 지연을 유발하는 쿼리 종료

데이터베이스에서 오랫동안 실행 중인 활성 또는 유휴 상태의 트랜잭션은 WAL 복제 프로세스를 방해하여 복제 지연을 증가시킬 수 있습니다. 따라서 PostgreSQL `pg_stat_activity` 보기를 사용하여 이러한 트랜잭션의 런타임을 모니터링해야 합니다.

다음과 비슷한 방식으로 기본 인스턴스에서 쿼리를 실행하여 오랫동안 실행 중인 쿼리의 프로세스 ID(PID)를 찾으세요.

```
SELECT datname, pid,usename, client_addr, backend_start,
xact_start, current_timestamp - xact_start AS xact_runtime, state,
```

```
backend_xmin FROM pg_stat_activity WHERE state='active';
```

```
SELECT now() - state_change as idle_in_transaction_duration, now() - xact_start as
 xact_duration,*
FROM  pg_stat_activity
WHERE state  = 'idle in transaction'
AND   xact_start is not null
ORDER BY 1 DESC;
```

쿼리의 PID를 식별한 후 쿼리를 종료하도록 선택할 수 있습니다.

다음과 비슷한 방식으로 기본 인스턴스에서 쿼리를 실행하여 오랫동안 실행 중인 쿼리를 종료하세요.

```
SELECT pg_terminate_backend(PID);
```

Amazon RDS Optimized Reads로 RDS for PostgreSQL 쿼리 성능 개선

Amazon RDS Optimized Reads로 RDS for PostgreSQL의 쿼리 처리 속도를 높일 수 있습니다. RDS Optimized Reads를 사용하는 RDS for PostgreSQL DB 인스턴스 또는 다중 AZ DB 클러스터는 이를 사용하지 않는 것에 비해 쿼리 처리 속도가 최대 50% 더 빠릅니다.

주제

- PostgreSQL의 RDS Optimized Reads 개요
- RDS Optimized Reads 사용 사례
- RDS Optimized Reads 모범 사례
- RDS Optimized Reads 사용
- RDS Optimized Reads를 사용하는 DB 인스턴스 모니터링
- PostgreSQL의 RDS Optimized Reads 제한 사항

PostgreSQL의 RDS Optimized Reads 개요

Optimized Reads는 NVMe 기반 DB 인스턴스 클래스를 사용하는 경우 RDS for PostgreSQL 버전 15.2 이상, 14.7 이상, 13.10 이상에서 기본적으로 사용할 수 있습니다. NVMe를 사용하는 인스턴스를 나타내는 하드웨어 사양은 에 대한 DB 인스턴스 클래스의 하드웨어 사양 섹션을 참조하세요.

RDS Optimized Reads가 켜져 있는 RDS for PostgreSQL DB 인스턴스 또는 다중 AZ DB 클러스터를 사용할 경우, Solid State Drive(SSD) 블록 수준 스토리지에 기반한 Non-Volatile Memory Express(NVMe)를 사용하여 최대 50% 더 빠른 쿼리 성능을 실현할 수 있습니다. PostgreSQL에서 생성된 임시 테이블을 로컬 스토리지에 배치하면 네트워크를 통해 EBS(Elastic Block Storage)로 향하는 트래픽을 줄여 쿼리 처리 속도를 높일 수 있습니다.

PostgreSQL에서 임시 객체는 세션 종료 시 자동으로 삭제되는 임시 네임스페이스에 할당됩니다. 삭제하는 동안 임시 네임스페이스는 테이블, 함수, 연산자 또는 확장 같은 스키마로 한정된 개체를 포함하여 세션에 종속된 모든 객체를 제거합니다.

RDS for PostgreSQL에서는 임시 객체가 저장되는 이 임시 작업 영역에 대해`temp_tablespaces` 파라미터가 구성됩니다.

다음 쿼리는 테이블스페이스의 이름과 위치를 반환합니다.

```
postgres=> show temp_tablespaces;
```

```
temp_tablespaces
---------------------
rds_temp_tablespace
(1 row)
```

rds_temp_tablespace는 NVMe 로컬 스토리지를 가리키는 RDS에서 구성한 테이블스페이스입니다. AWS Management Console을 사용하여 rds_temp_tablespace 이외의 테이블스페이스를 가리키도록 Parameter group에 있는 이 파라미터를 수정하면 Amazon EBS 스토리지로 언제든지 다시 전환할 수 있습니다. 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 단원을 참조하십시오. SET 명령을 사용하여 세션 수준에서 temp_tablespaces 파라미터의 값을 pg_default로 수정할 수도 있습니다. 파라미터를 수정하면 임시 작업 영역이 Amazon EBS로 리디렉션됩니다. Amazon EBS로 다시 전환하면 RDS 인스턴스 또는 클러스터의 로컬 스토리지가 특정 SQL 작업을 수행하기에 충분한 용량이 아닐 때 도움이 됩니다.

```
postgres=> SET temp_tablespaces TO 'pg_default';
SET
```

```
postgres=> show temp_tablespaces;

 temp_tablespaces
------------------
 pg_default
```

RDS Optimized Reads 사용 사례

다음은 Optimized Reads가 도움이 될 수 있는 몇 가지 사용 사례입니다.

- CTE(Common Table Expression), 파생된 테이블, 그룹화 작업을 포함하는 분석 쿼리.
- 애플리케이션의 최적화되지 않은 쿼리를 처리하는 읽기 전용 복제본.
- 항상 적절한 인덱스를 사용할 수 없는 GROUP BY 및 ORDER BY 같은 복잡한 작업이 포함된 온디맨드 또는 동적 보고 쿼리.
- 내부 임시 테이블을 사용하는 기타 워크로드.
- 정렬을 위한 CREATE INDEX 또는 REINDEX 작업입니다.

RDS Optimized Reads 모범 사례

RDS Optimized Reads에 대한 다음 모범 사례를 따릅니다.

- 실행 중 인스턴스 스토어가 가득 차서 쿼리가 실패하는 경우에 대비하여 읽기 전용 쿼리의 재시도 로직을 추가합니다.
- CloudWatch 지표 `FreeLocalStorage`를 사용하여 인스턴스 스토어에서 사용 가능한 스토리지 공간을 모니터링합니다. DB 인스턴스 또는 다중 AZ DB 클러스터의 워크로드로 인해 인스턴스 스토어가 한도에 도달하면 더 큰 DB 인스턴스 클래스를 사용하도록 수정하세요.

RDS Optimized Reads 사용

단일 AZ DB 인스턴스 배포, 다중 AZ DB 인스턴스 배포 또는 다중 AZ DB 클러스터 배포에서 NVMe 기반 DB 인스턴스 클래스 중 하나를 사용하여 RDS for PostgreSQL DB 인스턴스를 프로비저닝하면 DB 인스턴스는 자동으로 RDS Optimized Reads를 사용합니다.

다중 AZ 배포에 대한 자세한 정보는 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 섹션을 참조하세요.

RDS Optimized Read를 켜려면 다음 중 하나를 수행합니다.

- NVMe 기반 DB 인스턴스 클래스 중 하나를 사용하여 RDS for PostgreSQL DB 인스턴스 또는 다중 AZ DB 클러스터를 생성합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 단원을 참조하십시오.
- NVMe 기반 DB 인스턴스 클래스 중 하나를 사용하여 기존 RDS for PostgreSQL DB 인스턴스 또는 다중 AZ DB 클러스터를 수정합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

RDS Optimized Reads는 로컬 NVMe SSD 스토리지가 있는 DB 인스턴스 클래스 중 하나 이상이 지원되는 모든 AWS 리전에서 사용 가능합니다. 자세한 내용은 DB 인스턴스 클래스 단원을 참조하십시오.

최적화되지 않은 읽기 RDS 인스턴스로 다시 전환하려면 RDS 인스턴스 또는 클러스터의 DB 인스턴스 클래스를 데이터베이스 워크로드에 대해 EBS 스토리지만 지원하는 유사한 인스턴스 클래스로 수정하세요. 예를 들어, 현재 DB 인스턴스 클래스가 db.r6gd.4xlarge인 경우 db.r6g.4xlarge를 선택하여 다시 전환합니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

RDS Optimized Reads를 사용하는 DB 인스턴스 모니터링

다음 CloudWatch 지표를 사용하여 RDS Optimized Reads를 사용하는 DB 인스턴스를 모니터링할 수 있습니다.

- `FreeLocalStorage`

- `ReadIOPSLocalStorage`
- `ReadLatencyLocalStorage`
- `ReadThroughputLocalStorage`
- `WriteIOPSLocalStorage`
- `WriteLatencyLocalStorage`
- `WriteThroughputLocalStorage`

이러한 지표는 사용 가능한 인스턴스 스토어 스토리지, IOPS 및 처리량(throughput)에 대한 데이터를 제공합니다. 이러한 지표에 대한 자세한 내용은 Amazon RDS에 대한 Amazon CloudWatch 지표 섹션을 참조하세요.

로컬 스토리지의 현재 사용량을 모니터링하려면 데이터베이스에 로그인하고 다음 쿼리를 실행합니다.

```
SELECT
    spcname AS "Name",
    pg_catalog.pg_size_pretty(pg_catalog.pg_tablespace_size(oid)) AS "size"
FROM
    pg_catalog.pg_tablespace
WHERE
    spcname IN ('rds_temp_tablespace');
```

임시 파일 및 사용량에 대한 자세한 내용은 PostgreSQL을 사용한 임시 파일 관리 섹션을 참조하세요.

PostgreSQL의 RDS Optimized Reads 제한 사항

PostgreSQL의 RDS Optimized Reads에는 다음과 같은 제한 사항이 적용됩니다.

- 인스턴스 스토어가 가득 차면 트랜잭션이 실패할 수 있습니다.

Amazon RDS에서 PostgreSQL로 데이터 가져오기

Amazon RDS로 이동하려는 기존 PostgreSQL 배포가 있다고 가정하세요. 작업의 복잡성은 데이터베이스의 크기와 전송하는 데이터베이스 객체 유형에 따라 다릅니다. 예를 들어 프로시저 및 트리거와 함께 기가바이트 규모의 데이터세트가 데이터베이스에 저장되어 있다고 가정하겠습니다. 이러한 데이터베이스는 트리거나 프로시저 없이 근소한 메가바이트 규모의 테스트 데이터가 저장된 데이터베이스보다 더욱 복잡할 가능성이 높습니다.

다음과 같은 경우에는 기본 PostgreSQL 데이터베이스 마이그레이션 도구를 사용하는 것이 좋습니다.

- 같은 유형의 마이그레이션을 수행합니다. 이 경우 대상 데이터베이스와 동일한 데이터베이스 엔진으로 데이터베이스에서 마이그레이션합니다.
- 전체 데이터베이스를 마이그레이션합니다.
- 기본 도구를 사용하여 최소한의 가동 중지로 시스템을 마이그레이션할 수 있습니다.

대부분의 다른 경우에는 AWS Database Migration Service(AWS DMS)를 사용하여 데이터베이스 마이그레이션을 수행하는 것이 최선의 방식입니다. AWS DMS는 가동 중지 없이 데이터베이스를 마이그레이션할 수 있으며 대부분의 데이터베이스 엔진에서는 대상 데이터베이스로 전환할 준비가 될 때까지 지속적으로 복제를 계속할 수 있습니다. AWS DMS를 사용하여 동일하거나 다른 데이터베이스 엔진으로 마이그레이션할 수 있습니다. 원본 데이터베이스와 다른 데이터베이스 엔진으로 마이그레이션하는 경우 AWS Schema Conversion Tool(AWS SCT)를 사용하여 마이그레이션할 수 있습니다. AWS SCT를 사용하여 AWS DMS로 마이그레이션되지 않은 스키마 객체를 마이그레이션합니다. AWS DMS에 대한 자세한 내용은 AWS Database Migration Service란 무엇입니까?를 참조하세요.

가져오기에 대해서만 다음 설정을 포함하도록 DB 파라미터 그룹을 수정합니다. 파라미터 설정을 테스트하여 DB 인스턴스 크기에 가장 효율적인 설정을 찾아야 합니다. 또한 가져오기가 완료된 후 이들 파라미터를 프로덕션 값으로 되돌려야 합니다.

DB 인스턴스 설정을 다음과 같이 수정합니다.

- DB 인스턴스 백업 비활성화(backup_retention을 0으로 설정).
- Multi-AZ 비활성화.

다음 설정을 포함하도록 DB 파라미터 그룹을 수정합니다. 이들 설정은 데이터를 가져올 때만 사용해야 합니다. 파라미터 설정을 테스트하여 DB 인스턴스 크기에 가장 효율적인 설정을 찾아야 합니다. 또한 가져오기가 완료된 후 이들 파라미터를 프로덕션 값으로 되돌려야 합니다.

파라미터	가져오기 시 권장 값	설명
maintenance_work_mem	524288, 1048576, 2097152 또는 4194304(KB). 이들 설정은 512MB, 1GB, 2GB 및 4GB에 해당합니다.	이 설정의 값은 호스트 크기에 따라 달라집니다. 이 파라미터는 CREATE INDEX 문에서 사용되며 각 병렬 명령은 이 크기의 메모리를 사용할 수 있습니다. 이 값을 너무 높게 설정하여 메모리가 부족해지지 않도록 최선의 값을 계산합니다.
max_wal_size	256(버전 9.6), 4096(버전 10 이상)	자동 체크포인트 중에 WAL이 증가할 수 있는 최대 크기입니다. 이 파라미터를 늘리면 충돌 복구에 소요되는 시간이 늘어날 수 있습니다. 이 파라미터는 PostgreSQL 9.6 이상에서 checkpoint_segments 를 대체합니다. PostgreSQL 버전 9.6의 경우 이 값은 16MB 단위입니다. 이후 버전의 경우 이 값은 1MB 단위입니다. 예를 들어 버전 9.6에서 128은 각각 16MB 크기인 128개의 청크를 의미합니다. 버전 12.4에서 2048는 각각 1MB 크기인 2048개의 청크를 의미합니다.
checkpoint_timeout	1800	이 설정의 값은 WAL 교체 빈도를 줄일 수 있습니다.
synchronous_commit	Off	쓰기 속도를 높이려면 이 설정을 비활성화합니다. 이 파라미터를 끄면 서버 충돌 시 데이터 손실 위험이 증가합니다(FSYNC를 끄지 않음).
wal_buffers	8192	이 값은 8KB 단위입니다. 이는 다시 WAL 생성 속도에 도움이 됩니다.
autovacuum	0	리소스를 사용하지 않으므로 데이터를 로드하는 동안에는 PostgreSQL auto-vacuum 파라미터를 비활성화합니다.

이런 설정과 함께 pg_dump -Fc(압축) 또는 pg_restore -j(병렬) 명령을 사용합니다.

> **Note**
>
> PostgreSQL의 pg_dumpall 명령을 사용하려면 DB 인스턴스를 생성할 때 부여되지 않은 super_user 권한이 필요하며, 따라서 이 명령은 데이터 가져오기에 사용할 수 없습니다.

주제

- Amazon EC2 인스턴스에서 PostgreSQL 데이터베이스 가져오기
- \copy 명령을 사용하여 PostgreSQL DB 인스턴스의 테이블로 데이터 가져오기
- PostgreSQL DB 인스턴스용 RDS로 Amazon S3 데이터 가져오기
- DB 인스턴스 간에 PostgreSQL 데이터베이스 전송

Amazon EC2 인스턴스에서 PostgreSQL 데이터베이스 가져오기

Amazon EC2 인스턴스상의 PostgreSQL 서버에 데이터가 있는데 이를 PostgreSQL DB 인스턴스로 이동하려는 경우 다음 프로세스를 사용하여 데이터를 마이그레이션할 수 있습니다.

1. 로드할 데이터를 포함한 pg_dump를 사용하여 파일 만들기
2. 대상 DB 인스턴스 만들기
3. psql을 사용하여 DB 인스턴스에서 데이터베이스를 만들고 데이터 로드
4. DB 인스턴스의 DB 스냅샷 만들기

다음 섹션에서는 위에 나열된 각 단계에 대한 자세한 내용을 제공합니다.

1단계: 로드할 데이터를 포함하여 pg_dump로 파일 생성

pg_dump 유틸리티는 COPY 명령을 사용하여 PostgreSQL 데이터베이스의 스키마와 데이터 덤프를 만듭니다. pg_dump에 의해 생성되는 덤프 스크립트는 같은 이름을 가진 데이터베이스로 데이터를 로드하고 테이블, 인덱스 및 외래 키를 다시 만듭니다. pg_restore 명령과 -d 파라미터를 사용하여 데이터를 다른 이름의 데이터베이스로 복원할 수 있습니다.

데이터 덤프를 만들기 전, 대상 DB 인스턴스에서 개수를 확인할 수 있도록 행 개수를 확인하기 위해 덤프할 테이블을 쿼리해야 합니다.

다음 명령을 실행하면 mydb2라는 데이터베이스에 대해 mydb2dump.sql이라는 덤프 파일이 생성됩니다.

```
prompt>pg_dump dbname=mydb2 -f mydb2dump.sql
```

2단계: 대상 DB 인스턴스 만들기

Amazon RDS 콘솔, AWS CLI 또는 API를 사용하여 대상 PostgreSQL DB 인스턴스를 만듭니다. 백업 보존 설정을 0으로 지정하여 인스턴스를 만들고 다중 AZ를 비활성화합니다. 그러면 데이터를 더 빠르게 가져올 수 있습니다. 데이터를 덤프하려면 먼저 인스턴스에 데이터베이스를 생성해야 합니다. 데이터베이스 이름은 덤프 데이터가 저장된 데이터베이스와 동일하게 지정할 수 있습니다. 그렇지 않으면 다른 이름으로 지정하는 것도 가능합니다. 이 경우에는 pg_restore 명령과 -d 파라미터를 사용하여 데이터를 새로운 이름의 데이터베이스로 복원할 수 있습니다.

예를 들어 다음 명령을 사용하여 데이터베이스를 덤프 및 복원하고 이름을 바꿀 수 있습니다.

```
pg_dump -Fc -v -h [endpoint of instance] -U [master username] [database]
 > [database].dump
createdb [new database name]
pg_restore -v -h [endpoint of instance] -U [master username] -d [new database
 name] [database].dump
```

3단계: psql을 사용하여 DB 인스턴스에서 데이터베이스를 만들고 데이터 로드

pg_dump 명령을 실행할 때 사용한 것과 같은 연결을 사용하여 대상 DB 인스턴스에 연결하고 데이터베이스를 다시 만들 수 있습니다. psql을 사용할 때는 마스터 사용자 이름과 마스터 암호를 사용하여 DB 인스턴스에 데이터베이스를 만들 수 있습니다.

다음 예제에서는 psql과 mydb2dump.sql이라는 덤프 파일을 사용하여 mypginstance라는 PostgreSQL DB 인스턴스에 mydb2라는 데이터베이스를 만듭니다.

대상 LinuxmacOS, 또는Unix:

```
psql \
   -f mydb2dump.sql \
   --host mypginstance.555555555555.aws-region.rds.amazonaws.com \
   --port 8199 \
   --username myawsuser \
   --password password \
   --dbname mydb2
```

Windows의 경우:

```
psql ^
```

```
    -f mydb2dump.sql ^
    --host mypginstance.555555555555.aws-region.rds.amazonaws.com ^
    --port 8199 ^
    --username myawsuser ^
    --password password ^
    --dbname mydb2
```

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외의 암호를 지정하는 것이 좋습니다.

4단계: DB 인스턴스의 DB 스냅샷 만들기

데이터가 DB 인스턴스로 로드된 것을 확인했으면, 대상 PostgreSQL DB 인스턴스의 DB 스냅샷을 만드는 것이 좋습니다. DB 스냅샷은 DB 인스턴스를 알려진 상태로 복원하는 데 사용할 수 있는 DB 인스턴스의 완전한 백업입니다. 로드 직후에 생성된 DB 스냅샷은 사고 발생 시 데이터를 다시 로드하지 않아도 됩니다. 또한 이러한 스냅샷을 사용하여 새 DB 인스턴스를 시드할 수 있습니다. DB 스냅샷 생성에 대한 정보는 Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성 단원을 참조하세요.

\copy 명령을 사용하여 PostgreSQL DB 인스턴스의 테이블로 데이터 가져오기

PostgreSQL \copy 명령은 psql 대화형 클라이언트 도구에서 사용할 수 있는 메타 명령입니다. \copy를 사용하여 RDS for PostgreSQL DB 인스턴스의 테이블로 데이터를 가져올 수 있습니다. \copy 명령을 사용하려면 먼저 \copy가 대상 DB 인스턴스에 복사할 데이터의 대상을 갖도록 테이블 구조를 생성해야 합니다.

\copy를 사용하여 클라이언트 워크스테이션으로 내보내고 저장된 파일과 같은 쉼표로 구분된 값(CSV) 파일에서 데이터를 로드할 수 있습니다.

CSV 데이터를 대상 RDS for PostgreSQL DB 인스턴스로 가져오려면 먼저 psql을 사용하여 대상 DB 인스턴스에 연결합니다.

```
psql --host=db-instance.111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=target-db
```

그리고 다음 파라미터와 함께 \copy 명령을 실행하여 데이터 대상과 해당 형식을 식별합니다.

- target_table - CSV 파일에서 복사되는 데이터를 수신해야 하는 테이블의 이름입니다.

- `column_list` - 테이블의 열 사양입니다.
- `'filename'`- 로컬 워크스테이션에 있는 CSV 파일의 전체 경로입니다.

```
\copy target_table from '/path/to/local/filename.csv' WITH DELIMITER ',' CSV;
```

CSV 파일에 열 머리글 정보가 있는 경우 이 버전의 명령 및 파라미터를 사용할 수 있습니다.

```
\copy target_table (column-1, column-2, column-3, ...)
    from '/path/to/local/filename.csv' WITH DELIMITER ',' CSV HEADER;
```

`\copy` 명령이 실패하면 PostgreSQL이 오류 메시지를 출력합니다.

다음 예와 같이 `\copy` 메타 명령과 함께 `psql` 명령을 사용하여 데이터베이스 미리 보기 환경에서 새 DB 인스턴스를 생성합니다. 이 예제에서는 source-table을 원본 테이블 이름으로, source-table.csv를 .csv 파일로, target-db를 대상 데이터베이스로 사용합니다.

대상 LinuxmacOS, 또는Unix:

```
$psql target-db \
    -U <admin user> \
    -p <port> \
    -h <DB instance name> \
    -c "\copy source-table from 'source-table.csv' with DELIMITER ','"
```

Windows의 경우:

```
$psql target-db ^
    -U <admin user> ^
    -p <port> ^
    -h <DB instance name> ^
    -c "\copy source-table from 'source-table.csv' with DELIMITER ','"
```

`\copy` 명령에 대한 자세한 내용은 PostgreSQL 설명서의 메타 명령(Meta-Commands) 섹션에서 psql 페이지를 참조하세요.

PostgreSQL DB 인스턴스용 RDS로 Amazon S3 데이터 가져오기

Amazon Simple Storage Service를 사용하여 저장된 데이터를 Aurora PostgreSQL DB 클러스터 인스턴스의 테이블로 가져올 수 있습니다. 이 작업을 수행하려면 먼저 RDS for PostgreSQL `aws_s3` 확장

을 설치해야 합니다. 이 확장은 Amazon S3 버킷에서 데이터를 가져오는 데 사용하는 기능을 제공합니다. 버킷은 객체 또는 파일에 대한 Amazon S3 컨테이너입니다. 데이터는 쉼표로 구분된 값 (CSV) 파일, 텍스트 파일 또는 압축 (gzip) 파일에 있을 수 있습니다. 다음에서는 확장 프로그램을 설치하는 방법과 Amazon S3에서 테이블로 데이터를 가져오는 방법을 알아볼 수 있습니다.

Amazon S3를 RDS for PostgreSQL로 가져오려면 데이터베이스에서 PostgreSQL 버전 10.7 이상을 실행 중이어야 합니다.

Amazon S3 데이터가 저장되지 않은 경우 먼저 버킷을 생성하고 데이터를 저장해야 합니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 다음 주제를 참조하세요.

- 버킷 생성
- 버킷에 객체 추가

Amazon S3에서 계정 간 가져오기가 지원됩니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 계정 간 권한 부여를 참조하세요.

S3에서 데이터를 가져오는 동안 고객 관리형 키를 암호화에 사용할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 AWS KMS에 저장된 KMS 키를 참조하세요.

주제

- aws_s3 확장 설치
- Amazon S3 데이터에서 데이터 가져오기 개요
- Amazon S3 버킷에 대한 액세스 권한 설정
- Amazon S3에서 RDS for PostgreSQL DB 인스턴스로 데이터 가져오기
- 함수 참조

aws_s3 확장 설치

Amazon S3를 RDS for PostgreSQL DB 인스턴스와 함께 사용하려면 먼저 aws_s3 확장을 설치해야 합니다. 이 확장은 Amazon S3에서 데이터를 가져오기 위한 함수도 제공합니다. 또한 RDS for PostgreSQL DB 인스턴스에서 Amazon S3 버킷으로 데이터를 내보내는 기능도 제공합니다. 자세한 내용은 RDS for PostgreSQL DB 인스턴스에서 Amazon S3로 데이터 내보내기 단원을 참조하세요. aws_s3 확장은 필요할 때 자동으로 설치되는 aws_commons 확장의 일부 도우미 기능에 따라 다릅니다.

aws_s3 확장을 설치하려면

1. psql(또는 PGAdmin)을 사용하여 RDS for PostgreSQL DB 인스턴스에 rds_superuser 권한을 가진 사용자로 연결합니다. 설정 과정에서 기본 이름을 계속 사용했다면 postgres로 연결합니다.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password
```

2. 다음 명령을 실행하여 확장을 생성합니다.

```
postgres=> CREATE EXTENSION aws_s3 CASCADE;
NOTICE: installing required extension "aws_commons"
CREATE EXTENSION
```

3. 확장 프로그램이 설치되었는지 확인하려면 psql \dx 메타 명령을 사용하면 됩니다.

```
postgres=> \dx
       List of installed extensions
    Name     | Version |   Schema   |                 Description
-------------+---------+------------+---------------------------------------------
 aws_commons | 1.2     | public     | Common data types across AWS services
 aws_s3      | 1.1     | public     | AWS S3 extension for importing data from S3
 plpgsql     | 1.0     | pg_catalog | PL/pgSQL procedural language
(3 rows)
```

이제 Amazon S3에서 데이터를 가져오고 Amazon S3로 데이터를 내보내는 기능을 사용할 수 있습니다.

Amazon S3 데이터에서 데이터 가져오기 개요

Amazon RDS로 S3 데이터 가져오기

먼저 함수에 제공해야 하는 세부 정보를 수집합니다. 여기에는 RDS for PostgreSQL DB 인스턴스의 테이블 이름과 버킷 이름, 파일 경로, 파일 유형 및 Amazon S3 데이터가 저장된 AWS 리전이 포함됩니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 객체 보기를 참조하세요.

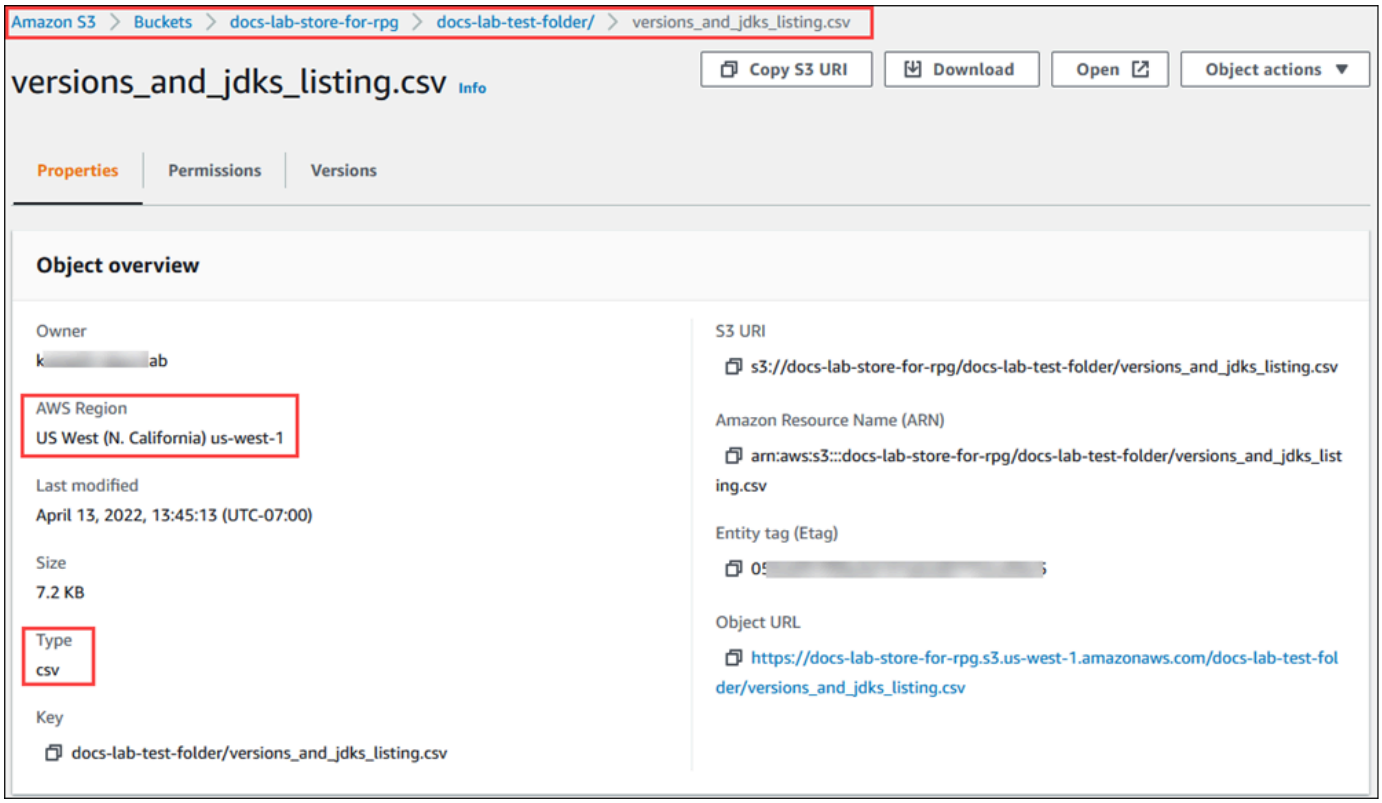
> **Note**
>
> Amazon S3에서 멀티 파트 데이터 가져오기는 현재 지원되지 않습니다.

1. `aws_s3.table_import_from_s3` 함수가 데이터를 가져올 테이블의 이름을 가져옵니다. 예를 들어 다음 명령은 이후 단계에서 사용할 수 있는 테이블 `t1`을 만듭니다.

```
postgres=> CREATE TABLE t1
    (col1 varchar(80),
    col2 varchar(80),
    col3 varchar(80));
```

2. Amazon S3 버킷 및 가져올 데이터에 대한 세부 정보를 가져옵니다. 이 작업을 수행하려면 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 열고 버킷을 선택합니다. 목록에서 데이터가 포함된 버킷을 찾습니다. 버킷을 선택하고 객체 개요 페이지를 연 다음 속성을 선택합니다.

   버킷 이름, 경로, AWS 리전, 파일 형식을 적어 둡니다. IAM 역할을 통해 Amazon S3에 대한 액세스를 설정하는 Amazon 리소스 이름(ARN) 이 나중에 필요합니다. 자세한 내용은 Amazon S3 버킷에 대한 액세스 권한 설정 단원을 참조하세요. 다음 그림에 예가 나와 있습니다.



3. AWS CLI 명령 `aws s3 cp`를 사용하여 Amazon S3 버킷의 데이터 경로를 확인할 수 있습니다. 정보가 정확하면 이 명령이 Amazon S3 파일의 복사본을 다운로드합니다.

```
aws s3 cp s3://amzn-s3-demo-bucket/sample_file_path ./
```

4. Amazon S3 버킷의 파일에 대한 액세스를 허용하도록 RDS for PostgreSQL DB 인스턴스에 대한 권한을 설정합니다. 이렇게 하려면 AWS Identity and Access Management(IAM) 역할 또는 보안 자격 증명을 사용합니다. 자세한 내용은 Amazon S3 버킷에 대한 액세스 권한 설정 단원을 참조하세요.
5. 경로 및 수집된 기타 Amazon S3 객체 세부 정보(2단계 참조)를 `create_s3_uri` 함수에 제공하여 Amazon S3 URI 객체를 생성합니다. 이 함수에 대한 자세한 내용은 aws_commons.create_s3_uri 단원을 참조하세요. 다음은 psql 세션 중에 이 객체를 구성하는 예입니다.

```
postgres=> SELECT aws_commons.create_s3_uri(
   'docs-lab-store-for-rpg',
   'versions_and_jdks_listing.csv',
   'us-west-1'
) AS s3_uri \gset
```

   다음 단계에서는 이 객체(`aws_commons._s3_uri_1`)를 `aws_s3.table_import_from_s3` 함수에 전달하여 데이터를 테이블로 가져옵니다.
6. `aws_s3.table_import_from_s3` 함수를 호출하여 Amazon S3에서 테이블로 데이터를 가져옵니다. 참조 정보는 aws_s3.table_import_from_s3 단원을 참조하세요. 예시는 Amazon S3에서 RDS for PostgreSQL DB 인스턴스로 데이터 가져오기 섹션을 참조하세요.

Amazon S3 버킷에 대한 액세스 권한 설정

Amazon S3 파일에서 데이터를 가져오려면 RDS for PostgreSQL DB 인스턴스에 파일이 저장된 Amazon S3 버킷에 액세스할 수 있는 권한을 부여합니다. 다음 항목에서 설명하는 두 방법 중 하나를 사용하여 Amazon S3 버킷에 대한 액세스 권한을 부여합니다.

주제

- IAM 역할을 사용해 Amazon S3 버킷에 액세스
- 보안 자격 증명을 사용해 Amazon S3 버킷에 액세스
- Amazon S3 액세스 문제 해결

IAM 역할을 사용해 Amazon S3 버킷에 액세스

Amazon S3 파일에서 데이터를 로드하기 전에 RDS for PostgreSQL DB 인스턴스에 파일이 저장된 Amazon S3 버킷에 액세스할 수 있는 권한을 부여합니다. 그러면 추가 자격 증명 정보를 관리하거나 aws_s3.table_import_from_s3 함수 호출에서 제공할 필요가 없습니다.

이렇게 하려면 Amazon S3 버킷에 대한 액세스 권한을 부여하는 IAM 정책을 생성합니다. IAM 역할을 생성하여 정책을 역할에 연결합니다. 그런 다음 IAM 역할을 DB 인스턴스에 할당합니다.

IAM 역할을 통해 RDS for PostgreSQL DB 인스턴스에 Amazon S3 액세스 권한 부여

1. IAM 정책을 생성합니다.

   이 정책은 RDS for PostgreSQL DB 인스턴스가 Amazon S3에 액세스할 수 있도록 허용하는 버킷 및 객체 권한을 부여합니다.

   정책에 다음과 같은 필수 작업을 포함하여 Amazon S3 버킷에서 Amazon RDS로의 파일 전송을 허용합니다.

   - `s3:GetObject`
   - `s3:ListBucket`

   정책에 다음 리소스를 포함하여 Amazon S3 버킷 및 그 안의 객체를 식별합니다. 다음은 Amazon S3에 액세스하기 위한 Amazon 리소스 이름(ARN) 형식입니다.

   - arn:aws:s3:::*amzn-s3-demo-bucket*
   - arn:aws:s3:::*amzn-s3-demo-bucket*/*

   RDS for PostgreSQL용 IAM 정책 생성에 대한 자세한 내용은 IAM 데이터베이스 액세스를 위한 IAM 정책 생성 및 사용 단원을 참조하세요. IAM 사용 설명서의 자습서: 첫 번째 고객 관리형 정책 생성 및 연결도 참조하세요.

   다음 AWS CLI 명령은 이 옵션으로 `rds-s3-import-policy`라는 IAM 정책을 만듭니다. *amzn-s3-demo-bucket*이라는 버킷에 대한 액세스 권한을 부여합니다.

   > **Note**
   >
   > 이 명령에서 반환되는 정책 Amazon 리소스 이름(ARN)을 적어 둡니다. IAM 역할에 정책을 연결할 때 이후 단계에 ARN이 필요합니다.

   Example

   대상 LinuxmacOS, 또는Unix:

```
aws iam create-policy \
   --policy-name rds-s3-import-policy \
   --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3import",
         "Action": [
           "s3:GetObject",
           "s3:ListBucket"
         ],
         "Effect": "Allow",
         "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket",
           "arn:aws:s3:::amzn-s3-demo-bucket/*"
         ]
       }
     ]
   }'
```

Windows의 경우:

```
aws iam create-policy ^
   --policy-name rds-s3-import-policy ^
   --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3import",
         "Action": [
           "s3:GetObject",
           "s3:ListBucket"
         ],
         "Effect": "Allow",
         "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket",
           "arn:aws:s3:::amzn-s3-demo-bucket/*"
         ]
       }
     ]
   }'
```

2. IAM 역할을 생성합니다.

   이렇게 하면, Amazon RDS에서 IAM 역할을 수임하여 사용자 대신 Amazon S3 버킷에 액세스할 수 있도록 역할을 생성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자에게 권한을 위임하기 위한 역할 생성을 참조하세요.

   서비스 권한을 특정 리소스로 제한하는 리소스 기반 정책의 `aws:SourceArn` 및 `aws:SourceAccount` 전역 조건 컨텍스트 키를 사용하는 것이 좋습니다. 이는 혼동된 대리자 문제를 방지하는 가장 효과적인 방법입니다.

   두 전역 조건 컨텍스트 키와 계정 ID를 포함한 `aws:SourceArn` 값을 모두 사용하는 경우, `aws:SourceAccount` 값 및 `aws:SourceArn` 값의 계정은 동일한 정책 문에서 사용될 경우 반드시 같은 계정 ID를 사용해야 합니다.

   - 단일 리소스에 대한 교차 서비스 액세스를 원하는 경우 `aws:SourceArn`을 사용하세요.
   - 해당 계정의 모든 리소스가 교차 서비스 사용과 연결되도록 허용하려는 경우 `aws:SourceAccount`를 사용하세요.

   정책에서는 리소스의 전체 ARN이 포함된 `aws:SourceArn` 전역 조건 컨텍스트 키를 사용해야 합니다. 다음 예제에서는 AWS CLI 명령을 사용하여 `rds-s3-import-role`이라는 역할을 생성하는 방법을 보여줍니다.

   Example

   대상 LinuxmacOS, 또는Unix:

```
aws iam create-role \
   --role-name rds-s3-import-role \
   --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
```

```
                "aws:SourceArn": "arn:aws:rds:us-east-1:111122223333:db:dbname"
                }
            }
        }
    ]
}'
```

Windows의 경우:

```
aws iam create-role ^
   --role-name rds-s3-import-role ^
   --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
                "aws:SourceArn": "arn:aws:rds:us-east-1:111122223333:db:dbname"
                }
            }
       }
     ]
   }'
```

3. 생성한 IAM 역할에 생성한 IAM 정책을 연결합니다.

   다음 AWS CLI 명령은 앞서 생성한 정책을 `rds-s3-import-role`이라는 역할에 연결합니다. *your-policy-arn*을 이전 단계에서 기록한 정책 ARN으로 바꿉니다.

   Example

   대상 LinuxmacOS, 또는Unix:

```
aws iam attach-role-policy \
   --policy-arn your-policy-arn \
   --role-name rds-s3-import-role
```

Windows의 경우:

```
aws iam attach-role-policy ^
   --policy-arn your-policy-arn ^
   --role-name rds-s3-import-role
```

4. IAM 역할을 DB 인스턴스에 추가합니다.

   이렇게 하려면 다음에 설명한 대로 AWS Management Console 또는 AWS CLI를 사용합니다.

콘솔

콘솔을 사용하여 PostgreSQL DB 인스턴스에 대해 IAM 역할을 추가하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 세부 정보를 표시하고자 하는 PostgreSQL DB 인스턴스 이름을 선택합니다.
3. 연결성 및 보안(Connectivity & security) 탭에 있는 IAM 역할 관리(Manage IAM roles) 섹션의 이 인스턴스에 IAM 역할 추가(Add IAM roles to this cluster/instance)에서 추가할 역할을 선택합니다.
4. 기능에서 s3Import를 선택합니다.
5. [Add role]을 선택합니다.

AWS CLI

CLI를 사용하여 PostgreSQL DB 인스턴스에 대해 IAM 역할을 추가하려면

- 다음 명령을 사용해 `my-db-instance`라는 PostgreSQL DB 인스턴스에 역할을 추가합니다. *`your-role-arn`*을 이전 단계에서 기록한 정책 ARN으로 교체합니다. `s3Import` 옵션의 값에 대해 `--feature-name`를 사용합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds add-role-to-db-instance \
   --db-instance-identifier my-db-instance \
   --feature-name s3Import \
   --role-arn your-role-arn   \
```

```
    --region your-region
```

Windows의 경우:

```
aws rds add-role-to-db-instance ^
   --db-instance-identifier my-db-instance ^
   --feature-name s3Import ^
   --role-arn your-role-arn ^
   --region your-region
```

RDS API

Amazon RDS API를 사용하여 PostgreSQL DB 인스턴스에 대한 IAM 역할을 추가하려면 AddRoleToDBInstance 작업을 호출합니다.

보안 자격 증명을 사용해 Amazon S3 버킷에 액세스

원할 경우 IAM 역할을 사용하는 대신 보안 자격 증명을 사용해 Amazon S3 버킷에 대한 액세스 권한을 부여여할 수 있습니다. aws_s3.table_import_from_s3 함수 호출에서 `credentials` 파라미터를 지정하면 됩니다.

`credentials` 파라미터는 `aws_commons._aws_credentials_1` 자격 증명을 포함하는 AWS 유형의 구조입니다. 다음과 같이 aws_commons.create_aws_credentials 함수를 사용해 `aws_commons._aws_credentials_1` 구조에서 액세스 키와 비밀 키를 설정하십시오.

```
postgres=> SELECT aws_commons.create_aws_credentials(
   'sample_access_key', 'sample_secret_key', '')
AS creds \gset
```

다음과 같이 `aws_commons._aws_credentials_1` 구조를 생성한 후 aws_s3.table_import_from_s3 함수를 `credentials` 파라미터와 함께 사용해 데이터를 가져옵니다.

```
postgres=> SELECT aws_s3.table_import_from_s3(
   't', '', '(format csv)',
   :'s3_uri',
   :'creds'
);
```

또는 aws_commons.create_aws_credentials 함수 호출 인라인을 `aws_s3.table_import_from_s3` 함수 호출에 포함할 수 있습니다.

```
postgres=> SELECT aws_s3.table_import_from_s3(
   't', '', '(format csv)',
   :'s3_uri',
   aws_commons.create_aws_credentials('sample_access_key', 'sample_secret_key', '')
);
```

Amazon S3 액세스 문제 해결

Amazon S3 파일 데이터를 가져오려 할 때 연결 문제가 발생할 경우 다음 권장 사항을 참조하세요.

- Amazon RDS 자격 증명 및 액세스 문제 해결
- Amazon Simple Storage Service 사용 설명서의 Amazon S3 문제 해결
- IAM 사용 설명서의 Amazon S3 문제 해결 및 IAM

Amazon S3에서 RDS for PostgreSQL DB 인스턴스로 데이터 가져오기

aws_s3 확장의 `table_import_from_s3` 함수를 사용하여 Amazon S3 버킷에서 데이터를 가져옵니다. 참조 정보는 aws_s3.table_import_from_s3 단원을 참조하세요.

> **Note**
>
> 다음 예에서는 Amazon S3 버킷에 대한 액세스 권한을 허용하기 위해 IAM 역할 메서드를 사용합니다. 따라서 `aws_s3.table_import_from_s3` 함수 호출에는 자격 증명 파라미터가 포함되지 않습니다.

다음은 대표적인 예를 보여줍니다.

```
postgres=> SELECT aws_s3.table_import_from_s3(
   't1',
   '',
   '(format csv)',
   :'s3_uri'
);
```

파라미터는 다음과 같습니다.

- `t1` – 데이터를 복사할 PostgreSQL DB 인스턴스의 테이블에 지정된 이름입니다.

- `''` – 데이터베이스 테이블의 열 목록입니다(선택 사항). 이 파라미터를 사용해 S3 데이터 중 어떤 열이 어떤 테이블 열에 들어가는지 표시할 수 있습니다. 열을 지정하지 않으면 모든 열이 테이블에 복사됩니다. 열 목록 사용에 대한 예시는 사용자 지정 구분 기호를 사용하는 Amazon S3 파일 가져오기 단원을 참조하십시오.
- `(format csv)` – PostgreSQL COPY 인수입니다. 복사 프로세스에서는 PostgreSQL COPY 명령의 인수와 형식을 사용하여 데이터를 가져옵니다. 형식에 대한 선택 사항에는 이 예에 표시된 대로 쉼표로 구분된 값(CSV), 텍스트 및 이진법이 있습니다. 기본값은 텍스트입니다.
- `s3_uri` – Amazon S3 파일을 식별하는 정보가 포함된 구조입니다. aws_commons.create_s3_uri 함수를 사용하여 `s3_uri` 구조를 생성하는 예제는 Amazon S3 데이터에서 데이터 가져오기 개요 단원을 참조하세요.

이 함수에 대한 자세한 내용은 aws_s3.table_import_from_s3 단원을 참조하세요.

이 `aws_s3.table_import_from_s3` 함수는 문자를 반환합니다. Amazon S3 버킷에서 가져올 다른 파일 종류를 지정하려면 다음 예 중 하나를 참조하세요.

> **Note**
>
> 0바이트 파일을 가져오면 오류가 발생합니다.

주제

- 사용자 지정 구분 기호를 사용하는 Amazon S3 파일 가져오기
- Amazon S3 압축(gzip) 파일 가져오기
- 인코딩된 Amazon S3 파일 가져오기

사용자 지정 구분 기호를 사용하는 Amazon S3 파일 가져오기

다음 예제에서는 사용자 지정 구분 기호를 사용하는 파일을 가져오는 방법을 보여줍니다. 또한 `column_list` 함수의 aws_s3.table_import_from_s3 파라미터를 사용해 데이터베이스 테이블에서 데이터를 배치할 곳을 제어하는 방법을 보여줍니다.

이 예에서는 다음 정보가 Amazon S3 파일의 파이프로 구분된 열에 정리되어 있다고 가정합니다.

```
1|foo1|bar1|elephant1
2|foo2|bar2|elephant2
3|foo3|bar3|elephant3
```

```
4|foo4|bar4|elephant4
...
```

사용자 지정 구분 기호를 사용하는 파일을 가져오려면

1. 가져온 데이터에 대해 데이터베이스에서 테이블을 생성합니다.

   ```
   postgres=> CREATE TABLE test (a text, b text, c text, d text, e text);
   ```

2. aws_s3.table_import_from_s3 함수의 다음과 같은 형식을 사용해 Amazon S3 파일에서 데이터를 가져옵니다.

   aws_commons.create_s3_uri 함수 호출 인라인을 `aws_s3.table_import_from_s3` 함수 호출에 포함하여 파일을 지정할 수 있습니다.

   ```
   postgres=> SELECT aws_s3.table_import_from_s3(
      'test',
      'a,b,d,e',
      'DELIMITER ''|''',
      aws_commons.create_s3_uri('amzn-s3-demo-bucket', 'pipeDelimitedSampleFile', 'us-
   east-2')
   );
   ```

이제 데이터는 다음 열의 테이블에 있습니다.

```
postgres=> SELECT * FROM test;
a | b | c | d | e
---+------+---+---+------+-----------
1 | foo1 | | bar1 | elephant1
2 | foo2 | | bar2 | elephant2
3 | foo3 | | bar3 | elephant3
4 | foo4 | | bar4 | elephant4
```

Amazon S3 압축(gzip) 파일 가져오기

다음 예에서는 gzip으로 압축된 Amazon S3에서 파일을 가져오는 방법을 보여줍니다. 가져오는 파일에 다음과 같은 Amazon S3 메타데이터가 있어야 합니다.

- 키: `Content-Encoding`
- 값: `gzip`

AWS Management Console을 사용하여 파일을 업로드하는 경우 일반적으로 시스템에서 메타데이터를 적용합니다. AWS Management Console, AWS CLI 또는 API를 사용하여 Amazon S3에 파일을 업로드하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 객체 업로드를 참조하세요.

Amazon S3 메타데이터에 대한 자세한 내용과 시스템 제공 메타데이터에 대한 세부 정보는 Amazon Simple Storage Service 사용 설명서의 Amazon S3 콘솔에서 객체 메타데이터 편집을 참조하세요.

아래와 같이 gzip 파일을 PostgreSQL DB 인스턴스용 RDS로 가져옵니다.

```
postgres=> CREATE TABLE test_gzip(id int, a text, b text, c text, d text);
postgres=> SELECT aws_s3.table_import_from_s3(
 'test_gzip', '', '(format csv)',
 'amzn-s3-demo-bucket', 'test-data.gz', 'us-east-2'
);
```

인코딩된 Amazon S3 파일 가져오기

다음 예에서는 Windows-1252 인코딩이 있는 Amazon S3에서 파일을 가져오는 방법을 보여줍니다.

```
postgres=> SELECT aws_s3.table_import_from_s3(
 'test_table', '', 'encoding ''WIN1252''',
 aws_commons.create_s3_uri('amzn-s3-demo-bucket', 'SampleFile', 'us-east-2')
);
```

함수 참조

Functions

- aws_s3.table_import_from_s3
- aws_commons.create_s3_uri
- aws_commons.create_aws_credentials

aws_s3.table_import_from_s3

Amazon S3 데이터를 Amazon RDS 테이블로 가져옵니다. `aws_s3` 확장은 `aws_s3.table_import_from_s3` 함수를 제공합니다. 반환 값은 텍스트입니다.

구문

필수 파라미터는 table_name, column_list 및 options입니다. 이 파라미터에서는 데이터베이스 테이블을 식별하고 데이터가 테이블로 복사되는 방식을 지정합니다.

다음 파라미터를 사용할 수도 있습니다.

- s3_info 파라미터는 가져올 Amazon S3 파일을 지정합니다. 이 파라미터를 사용하는 경우 IAM 역할에서 PostgreSQL DB 인스턴스에 대해 Amazon S3에 액세스할 수 있는 권한을 제공합니다.

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   s3_info aws_commons._s3_uri_1
)
```

- credentials 파라미터에서는 Amazon S3에 액세스할 수 있는 자격 증명을 지정합니다. 이 파라미터를 사용할 때는 IAM 역할을 사용하지 마십시오.

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   s3_info aws_commons._s3_uri_1,
   credentials aws_commons._aws_credentials_1
)
```

파라미터

table_name

데이터를 가져올 필수 텍스트 문자열로서, PostgreSQL 데이터베이스 테이블의 이름을 포함합니다.

column_list

데이터를 복사할 PostgreSQL 데이터베이스 테이블 열의 목록(선택 사항)을 포함하는 필수 텍스트 문자열입니다. 문자열이 비어 있는 경우 테이블의 모든 열이 사용됩니다. 관련 예시는 사용자 지정 구분 기호를 사용하는 Amazon S3 파일 가져오기 단원을 참조하십시오.

options

PostgreSQL COPY 명령에 대한 인수를 포함하는 필수 텍스트 스트링입니다. 이 인수에서는 데이터가 PostgreSQL 테이블에 복사되는 방식을 지정합니다. 자세한 내용은 PostgreSQL COPY 설명서를 참조하십시오.

s3_info

S3 객체에 대한 다음 정보를 포함하는 aws_commons._s3_uri_1 복합 키입니다.

- bucket – 파일이 포함된 Amazon S3 버킷의 이름입니다.
- file_path – 파일 경로를 포함한 Amazon S3 파일 이름입니다.
- region - 파일이 위치한 AWS 리전입니다. AWS 리전 이름 및 연결된 값의 목록은 리전, 가용 영역 및 로컬 영역 섹션을 참조하십시오.

credentials

가져오기 작업에 사용할 다음 자격 증명을 포함하는 aws_commons._aws_credentials_1 복합 유형입니다.

- 액세스 키
- 비밀 키
- 세션 토큰

aws_commons._aws_credentials_1 복합 구조 생성에 대한 자세한 내용은 aws_commons.create_aws_credentials 단원을 참조하십시오.

대체 구문

s3_info 및 credentials 파라미터 대신에 확장 파라미터 집합을 사용하면 테스트에 도움이 됩니다. 다음은 aws_s3.table_import_from_s3 함수에 대한 추가 구문 변형입니다.

- s3_info 파라미터를 사용해 Amazon S3 파일을 식별하는 대신 bucket, file_path 및 region 파라미터의 조합을 사용하십시오. IAM 역할은 이러한 형식의 함수를 통해 PostgreSQL DB 인스턴스에 대해 Amazon S3에 액세스할 수 있는 권한을 제공합니다.

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   bucket text,
   file_path text,
```

```
   region text
)
```

- `credentials` 파라미터를 사용해 Amazon S3 액세스를 지정하는 대신 `access_key`, `session_key` 및 `session_token` 파라미터의 조합을 사용하십시오.

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   bucket text,
   file_path text,
   region text,
   access_key text,
   secret_key text,
   session_token text
)
```

대체 파라미터

bucket

파일이 들어 있는 Amazon S3 버킷의 이름이 포함된 텍스트 문자열입니다.

file_path

파일 경로를 포함한 Amazon S3 파일 이름이 포함된 텍스트 문자열입니다.

region

파일의 AWS 리전 위치를 식별하는 텍스트 문자열입니다. AWS 리전 이름 및 연결된 값의 목록은 리전, 가용 영역 및 로컬 영역 섹션을 참조하세요.

access_key

가져오기 작업에 사용할 액세스 키가 포함된 텍스트 문자열입니다. 기본값은 NULL입니다.

secret_key

가져오기 작업에 사용할 비밀 키가 포함된 텍스트 문자열입니다. 기본값은 NULL입니다.

session_token

(선택 사항) 가져오기 작업에 사용할 세션 키가 포함된 텍스트 문자열입니다. 기본값은 NULL입니다.

aws_commons.create_s3_uri

Amazon S3 파일 정보를 저장할 `aws_commons._s3_uri_1` 구조를 생성합니다. `aws_commons.create_s3_uri` 함수의 `s3_info` 파라미터에서 aws_s3.table_import_from_s3 함수의 결과를 사용합니다.

구문

```
aws_commons.create_s3_uri(
   bucket text,
   file_path text,
   region text
)
```

파라미터

bucket

파일의 Amazon S3 버킷 이름이 포함된 필수 텍스트 문자열입니다.

file_path

파일 경로를 포함한 Amazon S3 파일 이름이 포함된 필수 텍스트 문자열입니다.

region

파일이 위치한 AWS 리전이 포함된 필수 텍스트 문자열입니다. AWS 리전 이름 및 연결된 값의 목록은 리전, 가용 영역 및 로컬 영역 섹션을 참조하세요.

aws_commons.create_aws_credentials

`aws_commons._aws_credentials_1` 구조에서 액세스 키와 비밀 키를 설정합니다. `aws_commons.create_aws_credentials` 함수의 `credentials` 파라미터에서 aws_s3.table_import_from_s3 함수의 결과를 사용합니다.

구문

```
aws_commons.create_aws_credentials(
   access_key text,
   secret_key text,
   session_token text
)
```

파라미터

access_key

Amazon S3 파일 가져오기에 사용할 액세스 키가 포함된 필수 텍스트 문자열입니다. 기본값은 NULL입니다.

secret_key

Amazon S3 파일 가져오기에 사용할 비밀 키가 포함된 필수 텍스트 문자열입니다. 기본값은 NULL입니다.

session_token

Amazon S3 파일 가져오기에 사용할 세션 토큰이 포함된 텍스트 문자열(선택 사항)입니다. 기본값은 NULL입니다. 선택 사항인 `session_token`을 제공하는 경우 임시 자격 증명을 사용할 수 있습니다.

DB 인스턴스 간에 PostgreSQL 데이터베이스 전송

Amazon RDS에 대해 PostgreSQL 전송 가능 데이터베이스를 사용하여 두 DB 인스턴스 간에 PostgreSQL 데이터베이스를 전송할 수 있습니다. 이 방법은 서로 다른 DB 인스턴스 간에 대규모 데이터베이스를 마이그레이션하는 매우 빠른 방법입니다. 이 방법을 사용하여 DB 인스턴스가 모두 동일한 메이저 버전의 PostgreSQL을 실행해야 합니다.

이 기능을 사용하려면 소스와 대상 DB 인스턴스 모두에 `pg_transport` 확장을 설치해야 합니다. `pg_transport` 확장은 최소한의 처리로 데이터베이스 파일을 이동하는 물리적 전송 메커니즘을 제공합니다. 이 메커니즘은 기존의 덤프 및 로드 프로세스보다 적은 가동 중지로 훨씬 빠르게 데이터를 이동합니다.

> **Note**
>
> PostgreSQL 전송 가능 데이터베이스는 RDS for PostgreSQL 11.5 이상 및 RDS for PostgreSQL 버전 10.10 이상에서 사용할 수 있습니다.

RDS for PostgreSQL DB 인스턴스에서 다른 RDS로 PostgreSQL DB 인스턴스를 전송하려면 먼저 전송을 위해 DB 인스턴스 설정에 설명된 대로 소스 및 대상 인스턴스를 설정합니다. 그런 다음 PostgreSQL 데이터베이스 전송에 설명된 함수를 사용하여 데이터베이스를 전송할 수 있습니다.

주제

- 데이터베이스 전송 중 발생하는 사항
- PostgreSQL 전송 가능 데이터베이스 사용에 대한 제한 사항
- PostgreSQL 데이터베이스를 전송하도록 설정
- PostgreSQL 데이터베이스를 소스에서 대상으로 전송
- 전송 가능한 데이터베이스 함수 참조
- 전송 가능한 데이터베이스 파라미터 참조

데이터베이스 전송 중 발생하는 사항

PostgreSQL 전송 가능 데이터베이스 기능은 소스 DB 인스턴스에서 대상으로 데이터베이스를 가져오는 폴링 모델을 사용합니다. `transport.import_from_server` 함수는 대상 DB 인스턴스에서 전송 중 데이터베이스를 생성합니다. 전송 기간 동안에는 대상 DB 인스턴스에서 전송 중 데이터베이스에 액세스할 수 없습니다.

전송이 시작되면 소스 데이터베이스의 모든 현재 세션이 종료됩니다. 소스 DB 인스턴스의 소스 데이터베이스 이외의 모든 데이터베이스는 전송의 영향을 받지 않습니다.

소스 데이터베이스는 특수한 읽기 전용 모드로 설정됩니다. 이 모드에 있는 동안 소스 데이터베이스에 연결하고 읽기 전용 쿼리를 실행할 수 있습니다. 그러나 쓰기 가능 쿼리 및 일부 다른 유형의 명령은 차단됩니다. 전송되는 특정 소스 데이터베이스만 이러한 제한의 영향을 받습니다.

전송 중에는 대상 DB 인스턴스를 특정 시점으로 복원할 수 없습니다. 전송은 트랜잭션이 아니며 PostgreSQL 미리 쓰기 로그를 사용하여 변경 사항을 기록하지 않기 때문입니다. 대상 DB 인스턴스에 자동 백업이 활성화되어 있으면 전송이 완료된 후 백업이 자동으로 수행됩니다. 특정 시점으로 복원은 백업이 완료된 후 일정 시간 동안 사용할 수 있습니다.

전송이 실패하면 `pg_transport` 확장은 소스 및 대상 DB 인스턴스에 대한 모든 변경을 취소하려고 시도합니다. 여기에는 대상에 부분적으로 전송된 데이터베이스 제거가 포함됩니다. 실패 유형에 따라 소스 데이터베이스는 쓰기 가능 쿼리를 계속 거부할 수 있습니다. 이 경우 다음 명령을 사용하여 쓰기 가능 쿼리를 허용하십시오.

```
ALTER DATABASE db-name SET default_transaction_read_only = false;
```

PostgreSQL 전송 가능 데이터베이스 사용에 대한 제한 사항

전송 가능 데이터베이스에는 다음과 같은 제한 사항이 있습니다.

- 읽기 전용 복제본 – 읽기 전용 복제본 또는 읽기 전용 복제본의 상위 인스턴스에서 전송 가능 데이터베이스를 사용할 수 없습니다.
- 지원되지 않는 열 유형 – 이 방법으로 전송하려는 데이터베이스 테이블에서는 `reg` 데이터 유형을 사용할 수 없습니다. 이러한 유형은 전송 중에 자주 변경되는 시스템 카탈로그 객체 ID(OID)에 따라 다릅니다.
- 테이블스페이스 – 모든 소스 데이터베이스 객체는 기본 `pg_default` 테이블스페이스에 있어야 합니다.
- 호환성 – 소스 및 대상 DB 인스턴스는 모두 동일한 메이저 버전의 PostgreSQL을 실행해야 합니다.
- 확장 - 소스 DB 인스턴스에는 `pg_transport`만 설치할 수 있습니다.
- 역할 및 ACL – 소스 데이터베이스의 액세스 권한 및 소유권 정보는 대상 데이터베이스로 전달되지 않습니다. 모든 데이터베이스 객체는 전송의 로컬 대상 사용자가 생성하고 소유합니다.
- 동시 전송 - 작업자 프로세스가 올바르게 구성된 경우 단일 DB 인스턴스에서 가져오기와 내보내기를 포함하여 최대 32개의 동시 전송을 지원할 수 있습니다.
- RDS for PostgreSQL DB 인스턴스 전용 - RDS for PostgreSQL DB 인스턴스에 대해 PostgreSQL 전송 가능 데이터베이스만 지원할 수 있습니다. Amazon EC2에서 실행되는 온프레미스 데이터베이스 또는 데이터베이스에는 사용할 수 없습니다.

PostgreSQL 데이터베이스를 전송하도록 설정

시작하기 전에 RDS for PostgreSQL DB 인스턴스가 다음 요구 사항을 충족하는지 확인하세요.

- 소스 및 대상 RDS for PostgreSQL DB 인스턴스는 동일한 버전의 PostgreSQL을 실행해야 합니다.
- 대상 DB에는 전송할 소스 DB와 같은 이름의 데이터베이스를 가질 수 없습니다.
- 전송을 실행하는 데 사용하는 계정은 소스 DB와 대상 DB 모두에 대한 `rds_superuser` 권한이 필요합니다.
- 소스 DB 인스턴스의 보안 그룹은 대상 DB 인스턴스의 인바운드 액세스를 허용해야 합니다. 소스 및 대상 DB 인스턴스가 VPC에 있는 경우 이미 해당 경우일 수 있습니다. 보안 그룹에 대한 자세한 내용은 보안 그룹을 통한 액세스 제어 섹션을 참조하세요.

소스 DB 인스턴스에서 대상 DB 인스턴스로 데이터베이스를 전송하려면 각 인스턴스와 연결된 DB 파라미터 그룹을 몇 번 변경해야 합니다. 즉, 소스 DB 인스턴스에 대한 사용자 지정 DB 파라미터 그룹을 생성하고 대상 DB 인스턴스에 대한 사용자 지정 DB 파라미터 그룹을 생성해야 합니다.

> **Note**
>
> DB 인스턴스가 이미 사용자 지정 DB 파라미터 그룹을 사용하여 구성된 경우 다음 절차의 2단계부터 시작할 수 있습니다.

데이터베이스 전송을 위한 사용자 지정 DB 그룹 파라미터 구성

다음 단계에서는 `rds_superuser` 권한을 가진 계정을 사용하세요.

1. 소스 및 대상 DB 인스턴스가 기본 DB 파라미터 그룹을 사용하는 경우 인스턴스에 적합한 버전을 사용하여 사용자 지정 DB 파라미터 그룹을 생성해야 합니다. 이렇게 하면 여러 파라미터의 값을 변경할 수 있습니다. 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.
2. 사용자 지정 DB 파라미터 그룹에서 다음 파라미터 값을 변경합니다.

   - `shared_preload_libraries` - 라이브러리 목록에 `pg_transport`를 추가합니다.
   - `pg_transport.num_workers` - 기본값은 3입니다. 데이터베이스에 필요한 경우 이 값을 늘리거나 줄입니다. 200GB 데이터베이스의 경우 8이하의 값이 좋습니다. 이 파라미터의 기본값을 늘리면 `max_worker_processes` 값도 늘려야 합니다.
   - `pg_transport.work_mem` - 기본값은 PostgreSQL 버전에 따라 128MB 또는 256MB입니다. 기본 설정은 일반적으로 변경되지 않을 수 있습니다.
   - `max_worker_processes` - 이 파라미터의 값은 다음 계산을 사용하여 설정해야 합니다.

     ```
     (3 * pg_transport.num_workers) + 9
     ```

     전송과 관련된 다양한 백그라운드 작업자 프로세스를 처리하기 위해 대상에서 필요한 값입니다. `max_worker_processes`,에 관한 자세한 내용은 PostgreSQL 문서에서 리소스 소비를 참조하세요.

   `pg_transport` 파라미터에 대한 자세한 내용은 전송 가능한 데이터베이스 파라미터 참조 단원을 참조하십시오.
3. 소스 RDS for PostgreSQL DB 인스턴스 및 대상 인스턴스를 재부팅하면 파라미터 설정이 적용됩니다.
4. RDS for PostgreSQL 소스 DB 인스턴스에 연결합니다.

```
psql --host=source-instance.111122223333.aws-region.rds.amazonaws.com --port=5432
 --username=postgres --password
```

5. DB 인스턴스의 퍼블릭 스키마에서 불필요한 확장을 제거합니다. 실제 전송 작업 중에만 pg_transport 확장이 허용됩니다.
6. 다음과 같이 pg_transport 확장을 설치합니다.

```
postgres=> CREATE EXTENSION pg_transport;
CREATE EXTENSION
```

7. RDS for PostgreSQL 대상 DB 인스턴스에 연결합니다. 불필요한 확장을 제거한 다음 pg_transport 확장을 설치합니다.

```
postgres=> CREATE EXTENSION pg_transport;
CREATE EXTENSION
```

PostgreSQL 데이터베이스를 소스에서 대상으로 전송

PostgreSQL 데이터베이스를 전송하도록 설정에 설명된 프로세스를 완료한 후에는 전송을 시작할 수 있습니다. 이렇게 하려면 대상 DB 인스턴스에서 transport.import_from_server 함수를 실행합니다. 다음 구문에서는 함수 파라미터를 찾을 수 있습니다.

```
SELECT transport.import_from_server(
   'source-db-instance-endpoint',
    source-db-instance-port,
   'source-db-instance-user',
   'source-user-password',
   'source-database-name',
   'destination-user-password',
   false);
```

예에 표시된 false 값은 함수에 이것이 모의 실습이 아님을 알려줍니다. 전송 설정을 테스트하려면 다음에서 처럼 함수를 호출할 때 dry_run 옵션에서 true를 지정할 수 있습니다.

```
postgres=> SELECT transport.import_from_server(
    'docs-lab-source-db.666666666666aws-region.rds.amazonaws.com', 5432,
    'postgres', '********', 'labdb', '******', true);
INFO:  Starting dry-run of import of database "labdb".
```

```
INFO:  Created connections to remote database        (took 0.03 seconds).
INFO:  Checked remote cluster compatibility          (took 0.05 seconds).
INFO:  Dry-run complete                           (took 0.08 seconds total).
 import_from_server
--------------------

(1 row)
```

INFO 행은 pg_transport.timing 파라미터가 기본값인 true로 설정하므로 출력됩니다. 다음과 같이 명령을 실행하고 소스 데이터베이스를 대상으로 가져올 때 dry_run을 false로 설정합니다.

```
INFO:  Starting import of database "labdb".
INFO:  Created connections to remote database        (took 0.02 seconds).
INFO:  Marked remote database as read only           (took 0.13 seconds).
INFO:  Checked remote cluster compatibility          (took 0.03 seconds).
INFO:  Signaled creation of PITR blackout window     (took 2.01 seconds).
INFO:  Applied remote database schema pre-data       (took 0.50 seconds).
INFO:  Created connections to local cluster          (took 0.01 seconds).
INFO:  Locked down destination database              (took 0.00 seconds).
INFO:  Completed transfer of database files          (took 0.24 seconds).
INFO:  Completed clean up                            (took 1.02 seconds).
INFO:  Physical transport complete                (took 3.97 seconds total).
import_from_server
--------------------
(1 row)
```

이 함수를 사용하려면 데이터베이스 사용자 암호를 제공해야 합니다. 따라서 전송이 완료된 후 사용한 사용자 역할의 암호를 변경하는 것이 좋습니다. 또는 SQL 바인드 변수를 사용하여 임시 사용자 역할을 생성할 수 있습니다. 전송에 이러한 임시 역할을 사용한 후 나중에 해당 역할을 삭제하십시오.

전송에 성공하지 못하면 다음과 같은 오류 메시지가 표시될 수 있습니다.

```
pg_transport.num_workers=8 25% of files transported failed to download file data
```

"파일 데이터를 다운로드 실패" 오류 메시지는 작업자 프로세스 수가 데이터베이스 크기에 맞게 올바르게 설정되지 않았음을 나타냅니다. pg_transport.num_workers에 대한 값 세트를 늘리거나 줄여야 할 수 있습니다. 각 실패는 완료 비율을 보고하므로 변경 사항이 미치는 영향을 확인할 수 있습니다. 예를 들어 한 경우에 8에서 4로 설정을 변경하면 다음과 같은 결과가 발생합니다.

```
pg_transport.num_workers=4 75% of files transported failed to download file data
```

max_worker_processes 파라미터는 전송 프로세스 중에도 고려됩니다. 다시 말해 데이터베이스를 성공적으로 전송하려면 pg_transport.num_workers와 max_worker_processes 두 가지를 모두 수정해야 할 수 있습니다. 표시된 예제는 pg_transport.num_workers가 2로 설정되었을 때 마침내 작동했습니다.

```
pg_transport.num_workers=2 100% of files transported
```

transport.import_from_server 함수와 해당 파라미터에 대한 자세한 내용은 전송 가능한 데이터베이스 함수 참조 섹션을 참조하세요.

전송 가능한 데이터베이스 함수 참조

transport.import_from_server 함수는 PostgreSQL 데이터베이스를 소스 DB 인스턴스에서 대상 DB 인스턴스로 가져와서 전송합니다. 물리적 데이터베이스 연결 전송 메커니즘을 사용하여 이를 수행합니다.

전송을 시작하기 전에 이 함수는 소스 및 대상 DB 인스턴스가 동일한 버전이며 마이그레이션을 위해 호환되는지 확인합니다. 또한 대상 DB 인스턴스 소스에 충분한 공간이 있는지 확인합니다.

구문

```
transport.import_from_server(
   host text,
   port int,
   username text,
   password text,
   database text,
   local_password text,
   dry_run bool
)
```

반환 값

없음.

Parameters

다음 표에서 transport.import_from_server 함수 파라미터에 대한 설명을 확인할 수 있습니다.

파라미터	설명
host	소스 DB 인스턴스의 엔드포인트입니다.
port	소스 DB 인스턴스의 포트를 나타내는 정수입니다. PostgreSQL DB 인스턴스는 종종 포트 5432를 사용합니다.
username	소스 DB 인스턴스의 사용자입니다. 이 사용자는 rds_superuser 역할의 멤버여야 합니다.
password	소스 DB 인스턴스의 사용자 암호입니다.
database	전송할 소스 DB 인스턴스의 데이터베이스 이름입니다.
local_password	대상 DB 인스턴스에 대한 현재 사용자의 로컬 암호입니다. 이 사용자는 rds_superuser 역할의 멤버여야 합니다.
dry_run	모의 실습 수행 여부를 지정하는 선택적 부울 값입니다. 기본값은 false이며 이는 전송이 진행됨을 의미합니다. 실제 전송을 수행하지 않고 소스와 대상 DB 인스턴스 간의 호환성을 확인하려면 dry_run을 true로 설정하십시오.

예

관련 예제는 PostgreSQL 데이터베이스를 소스에서 대상으로 전송 섹션을 참조하세요

전송 가능한 데이터베이스 파라미터 참조

pg_transport 확장 동작을 제어하는 몇 가지 파라미터입니다. 다음에서 이러한 파라미터에 대한 설명을 확인할 수 있습니다.

pg_transport.num_workers

전송 프로세스에 사용할 작업자 수입니다. 기본값은 3입니다. 유효한 값은 1 – 32입니다. 대용량 데이터베이스 전송에서도 일반적으로 8명 미만 작업자가 필요합니다. 전송 중에 대상 DB 인스턴스의 이 설정 값은 대상 및 소스 모두에서 사용됩니다.

pg_transport.timing

전송 중 타이밍 정보를 보고할지 여부를 지정합니다. 기본값은 `true`로, 타이밍 정보가 보고됩니다. 이 파라미터는 설정은 `true`로 두어 진행 상황을 모니터링할 수 있을 것을 권장합니다. 예제 출력은 PostgreSQL 데이터베이스를 소스에서 대상으로 전송을 참조하세요.

pg_transport.work_mem

각 작업자에게 할당할 최대 메모리 양입니다. 기본값은 PostgreSQL 버전에 따라 131072킬로바이트(KB) 또는 262144KB(256MB)입니다. 최소값은 64MB(65,536KB)입니다. 유효한 값은 이진 base-2 단위로 나타낸 킬로바이트(KB)입니다(1KB = 1024바이트).

전송 시 이 파라미터에 지정된 것보다 적은 메모리를 사용할 수 있습니다. 대용량 데이터베이스 전송에서도 일반적으로 작업자당 256MB(262144KB) 미만의 메모리가 필요합니다.

RDS for PostgreSQL DB 인스턴스에서 Amazon S3로 데이터 내보내기

RDS for PostgreSQL DB 인스턴스에서 데이터를 쿼리하여 Amazon S3 버킷에 저장된 파일로 직접 내보낼 수 있습니다. 이 작업을 수행하려면 먼저 RDS for PostgreSQL aws_s3 확장을 설치해야 합니다. 이 확장은 쿼리 결과를 Amazon S3로 내보내는 데 사용하는 기능을 제공합니다. 다음에서는 확장 프로그램을 설치하는 방법과 Amazon S3로 데이터를 내보내는 방법을 확인할 수 있습니다.

> **Note**
>
> Amazon S3로 계정 간 내보내기는 지원되지 않습니다.

현재 사용 가능한 모든 RDS PostgreSQL 버전은 Amazon Simple Storage Service로 데이터 내보내기를 지원합니다. 자세한 버전 정보는 Amazon RDS for PostgreSQL 릴리스 정보에서 Amazon RDS for PostgreSQL 업데이트를 참조하세요.

내보내기용으로 버킷을 설정하지 않은 경우 Amazon Simple Storage Service 사용 설명서의 다음 주제를 참조하세요.

- Amazon S3 설정
- 버킷 생성

기본적으로 RDS for PostgreSQL에서 Amazon S3로 내보낸 데이터는 AWS 관리형 키를 통한 서버 측 암호화를 사용합니다. 버킷 암호화를 사용하는 경우 Amazon S3 버킷은 AWS Key Management Service(AWS KMS) 키(SSE-KMS)로 암호화되어야 합니다. 현재 Amazon S3 관리형 키(SSE-S3)로 암호화된 버킷은 지원되지 않습니다.

> **Note**
>
> AWS Management Console, AWS CLI 또는 Amazon RDS API를 사용하여 DB 스냅샷 데이터를 Amazon S3에 저장할 수 있습니다. 자세한 내용은 Amazon S3 for Amazon RDS로 DB 스냅샷 데이터 내보내기 단원을 참조하세요.

주제

- aws_s3 확장 설치

- Amazon S3으로 데이터 내보내기 개요
- 내보낼 Amazon S3 파일 경로 지정
- Amazon S3 버킷에 대한 액세스 권한 설정
- aws_s3.query_export_to_s3 함수를 사용하여 쿼리 데이터 내보내기
- 함수 참조
- Amazon S3 액세스 문제 해결

aws_s3 확장 설치

Amazon Simple Storage Service를 RDS for PostgreSQL DB 인스턴스와 함께 사용하려면 먼저 aws_s3 확장을 설치해야 합니다. 이 확장은 RDS for PostgreSQL DB 인스턴스의 라이터 인스턴스에서 Amazon S3 버킷으로 데이터를 내보내는 기능을 제공합니다. Amazon S3에서 데이터를 가져오기 위한 함수도 제공합니다. 자세한 내용은 PostgreSQL DB 인스턴스용 RDS로 Amazon S3 데이터 가져오기 단원을 참조하세요. aws_s3 확장은 필요할 때 자동으로 설치되는 aws_commons 확장의 일부 도우미 기능에 따라 다릅니다.

aws_s3 확장을 설치하려면

1. psql(또는 PGAdmin)을 사용하여 RDS for PostgreSQL DB 인스턴스에 rds_superuser 권한을 가진 사용자로 연결합니다. 설정 과정에서 기본 이름을 계속 사용했다면 postgres로 연결합니다.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password
```

2. 다음 명령을 실행하여 확장을 생성합니다.

```
postgres=> CREATE EXTENSION aws_s3 CASCADE;
NOTICE: installing required extension "aws_commons"
CREATE EXTENSION
```

3. 확장 프로그램이 설치되었는지 확인하려면 psql \dx 메타 명령을 사용하면 됩니다.

```
postgres=> \dx
       List of installed extensions
    Name     | Version |   Schema   |                Description
-------------+---------+------------+-------------------------------------------
 aws_commons | 1.2     | public     | Common data types across AWS services
```

```
 aws_s3      | 1.1     | public     | AWS S3 extension for importing data from S3
 plpgsql     | 1.0     | pg_catalog | PL/pgSQL procedural language
(3 rows)
```

이제 Amazon S3에서 데이터를 가져오고 Amazon S3로 데이터를 내보내는 기능을 사용할 수 있습니다.

RDS for PostgreSQL 버전에서 Amazon S3로 내보내기를 지원하는지 확인

`describe-db-engine-versions` 명령을 사용하여 RDS for PostgreSQL 버전이 Amazon S3로 내보내기를 지원하는지 확인할 수 있습니다. 다음 예에서는 버전 10.14에 대한 지원을 확인합니다.

```
aws rds describe-db-engine-versions --region us-east-1
--engine postgres --engine-version 10.14 | grep s3Export
```

출력에 `"s3Export"` 문자열이 포함된 경우 엔진은 Amazon S3 내보내기를 지원합니다. 그렇지 않으면 엔진이 이 내보내기를 지원하지 않습니다.

Amazon S3으로 데이터 내보내기 개요

RDS for PostgreSQL 데이터베이스에 저장된 데이터를 Amazon S3 버킷으로 내보내려면 다음 절차를 따르세요.

데이터를 S3로 내보내려면

1. 데이터를 내보내는 데 사용할 Amazon S3 파일 경로를 식별합니다. 이 프로세스에 대한 자세한 내용은 내보낼 Amazon S3 파일 경로 지정 단원을 참조하십시오.
2. Amazon S3 버킷에 액세스할 수 있는 권한을 부여합니다.

   Amazon S3 파일로 데이터를 내보내려면 RDS for PostgreSQL DB 인스턴스에 내보내기 시 스토리지에 사용할 Amazon S3 버킷에 액세스할 수 있는 권한을 부여합니다. 여기에는 다음 단계가 포함됩니다.

   1. 내보낼 Amazon S3 버킷에 대한 액세스를 제공하는 IAM 정책을 생성합니다.
   2. IAM 역할 생성.
   3. 생성한 정책을 생성한 역할에 연결하십시오.
   4. 이 IAM 역할을 DB 인스턴스에 추가합니다.

이 프로세스에 대한 자세한 내용은 Amazon S3 버킷에 대한 액세스 권한 설정 단원을 참조하십시오.

3. 데이터를 가져올 데이터베이스 쿼리를 식별합니다. `aws_s3.query_export_to_s3` 함수를 호출하여 쿼리 데이터를 내보냅니다.

   앞의 준비 작업을 완료한 후 aws_s3.query_export_to_s3 함수를 사용하여 쿼리 결과를 Amazon S3으로 내보냅니다. 이 프로세스에 대한 자세한 내용은 aws_s3.query_export_to_s3 함수를 사용하여 쿼리 데이터 내보내기 단원을 참조하세요.

내보낼 Amazon S3 파일 경로 지정

다음 정보를 지정하여 데이터를 내보낼 Amazon S3 위치를 식별합니다.

- 버킷 이름 – 버킷은 Amazon S3 객체 또는 파일을 위한 컨테이너입니다.

  Amazon S3을 이용한 데이터 저장에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 Creating a bucket 및 Working with objects를 참조하세요.
- 파일 경로 – 파일 경로는 내보낸 데이터가 Amazon S3 버킷에서 저장되는 위치를 식별합니다. 파일 경로는 다음과 같이 구성됩니다.
  - 가상 폴더 경로를 식별하는 선택적 경로 접두사입니다.
  - 저장할 하나 이상의 파일을 식별하는 파일 접두사입니다. 내보내는 데이터가 클 경우 각각 최대 크기가 약 6GB인 여러 파일에 저장됩니다. 추가 파일 이름의 파일 접두사도 동일하지만 `_part`*XX*가 추가됩니다. *XX*는 2, 3 등을 나타냅니다.

  예를 들어 `exports` 폴더와 `query-1-export` 파일 접두사가 있는 파일 경로는 `/exports/query-1-export`입니다.
- AWS 리전(선택 사항) – Amazon S3 버킷이 위치한 AWS 리전입니다. AWS 리전 값을 지정하지 않으면 Amazon RDS는 DB 인스턴스 내보내기와 동일한 AWS 리전의 Amazon S3에 파일을 저장합니다.

  > **Note**
  >
  > 현재 AWS 리전은 내보내는 DB 인스턴스의 리전과 동일해야 합니다.

  AWS 리전 이름 및 연결된 값의 목록은 리전, 가용 영역 및 로컬 영역 섹션을 참조하십시오.

내보내는 데이터를 저장할 위치에 대한 Amazon S3 파일 정보를 보관하려면 aws_commons.create_s3_uri 함수를 사용하여 다음과 같이 aws_commons._s3_uri_1 복합 구조를 생성할 수 있습니다.

```
psql=> SELECT aws_commons.create_s3_uri(
   'amzn-s3-demo-bucket',
   'sample-filepath',
   'us-west-2'
) AS s3_uri_1 \gset
```

나중에 이 s3_uri_1 값을 aws_s3.query_export_to_s3 함수에 대한 호출의 파라미터로 제공합니다. 예제는 aws_s3.query_export_to_s3 함수를 사용하여 쿼리 데이터 내보내기을 참조하세요.

Amazon S3 버킷에 대한 액세스 권한 설정

데이터를 Amazon S3으로 내보내려면 PostgreSQL DB 인스턴스에 파일이 저장될 Amazon S3 버킷에 액세스할 수 있는 권한을 부여합니다.

이렇게 하려면 다음 절차를 따르십시오.

IAM 역할을 통해 PostgreSQL DB 인스턴스에 액세스할 수 있는 권한을 Amazon S3에 부여하려면

1. IAM 정책을 생성합니다.

   이 정책은 PostgreSQL DB 인스턴스가 Amazon S3에 액세스할 수 있도록 허용하는 권한을 버킷 및 객체에 부여합니다.

   이 정책을 생성하는 과정에서 다음 단계를 수행하십시오.

   a. 다음과 같은 필수 작업을 정책에 포함하여 PostgreSQL DB 인스턴스에서 Amazon S3 버킷으로 파일 전송을 허용합니다.

      - s3:PutObject
      - s3:AbortMultipartUpload

   b. Amazon S3 버킷과 버킷의 객체를 식별하는 Amazon 리소스 이름(ARN)을 포함합니다. Amazon S3에 액세스하기 위한 ARN 형식은 arn:aws:s3:::*amzn-s3-demo-bucket*/*입니다.

PostgreSQL용 Amazon RDS에 대한 IAM 정책 생성에 대한 자세한 내용은 IAM 데이터베이스 액세스를 위한 IAM 정책 생성 및 사용 단원을 참조하십시오. IAM 사용 설명서의 자습서: 첫 번째 고객 관리형 정책 생성 및 연결도 참조하십시오.

다음 AWS CLI 명령은 이 옵션으로 `rds-s3-export-policy`라는 IAM 정책을 만듭니다. *`amzn-s3-demo-bucket`*이라는 버킷에 대한 액세스 권한을 부여합니다.

> **Warning**
>
> 특정 버킷에 액세스하도록 구성된 엔드포인트 정책이 있는 프라이빗 VPC 내에 데이터베이스를 설정하는 것이 좋습니다. 자세한 내용은 Amazon VPC 사용 설명서의 Amazon S3용 엔드포인트 정책 사용을 참조하십시오.
> 모든 리소스에 액세스할 수 있는 정책은 생성하지 않는 것이 좋습니다. 이러한 액세스 권한은 데이터 보안에 위협이 될 수 있습니다. `S3:PutObject`를 사용하여 모든 리소스에 액세스할 수 있는 권한을 `"Resource":"*"`에 부여하는 정책을 생성하면 내보내기 권한이 있는 사용자가 계정의 모든 버킷으로 데이터를 내보낼 수 있습니다. 또한 사용자는 AWS 리전 내의 공개적으로 쓰기 가능한 버킷으로 데이터를 내보낼 수 있습니다.

정책을 만든 후에 정책의 Amazon 리소스 이름(ARN)을 기록하십시오. IAM 역할에 정책을 연결할 때 이후 단계에 ARN이 필요합니다.

```
aws iam create-policy  --policy-name rds-s3-export-policy  --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3export",
         "Action": [
           "s3:PutObject*",
           "s3:ListBucket",
           "s3:GetObject*",
           "s3:DeleteObject*",
           "s3:GetBucketLocation",
           "s3:AbortMultipartUpload"
         ],
         "Effect": "Allow",
         "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket/*"
         ]
```

```
      }
    ]
  }'
```

2. IAM 역할 생성.

   Amazon RDS이 이 IAM 역할을 수임하여 사용자 대신 Amazon S3 버킷에 액세스할 수 있도록 역할을 생성합니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자에게 권한을 위임하기 위한 역할 생성을 참조하세요.

   서비스 권한을 특정 리소스로 제한하는 리소스 기반 정책의 `aws:SourceArn` 및 `aws:SourceAccount` 전역 조건 컨텍스트 키를 사용하는 것이 좋습니다. 이는 혼동된 대리자 문제를 방지하는 가장 효과적인 방법입니다.

   두 전역 조건 컨텍스트 키와 계정 ID를 포함한 `aws:SourceArn` 값을 모두 사용하는 경우, `aws:SourceAccount` 값 및 `aws:SourceArn` 값의 계정은 동일한 정책 문에서 사용될 경우 반드시 같은 계정 ID를 사용해야 합니다.

   - 단일 리소스에 대한 교차 서비스 액세스를 원하는 경우 `aws:SourceArn`을 사용하세요.
   - 해당 계정의 모든 리소스가 교차 서비스 사용과 연결되도록 허용하려는 경우 `aws:SourceAccount`를 사용하세요.

   정책에서는 리소스의 전체 ARN이 포함된 `aws:SourceArn` 전역 조건 컨텍스트 키를 사용해야 합니다. 다음 예제에서는 AWS CLI 명령을 사용하여 `rds-s3-export-role`이라는 역할을 생성하는 방법을 보여줍니다.

   Example

   대상 LinuxmacOS, 또는Unix:

```
aws iam create-role  \
    --role-name rds-s3-export-role  \
    --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
```

```
        "Condition": {
            "StringEquals": {
               "aws:SourceAccount": "111122223333",
               "aws:SourceArn": "arn:aws:rds:us-east-1:111122223333:db:dbname"
               }
            }
      }
    ]
  }'
```

Windows의 경우:

```
aws iam create-role  ^
    --role-name rds-s3-export-role  ^
    --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
                "aws:SourceArn": "arn:aws:rds:us-east-1:111122223333:db:dbname"
                }
             }
       }
     ]
   }'
```

3. 생성한 IAM 역할에 생성한 IAM 정책을 연결합니다.

   다음 AWS CLI 명령은 앞서 생성한 정책을 `rds-s3-export-role.`라는 역할에 연결합니다. *your-policy-arn*을 이전 단계에서 기록한 정책 ARN으로 바꿉니다.

```
aws iam attach-role-policy  --policy-arn your-policy-arn  --role-name rds-s3-
export-role
```

4. IAM 역할을 DB 인스턴스에 추가합니다. 이렇게 하려면 다음에 설명한 대로 AWS Management Console 또는 AWS CLI를 사용합니다.

콘솔

콘솔을 사용하여 PostgreSQL DB 인스턴스에 대해 IAM 역할을 추가하려면

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 세부 정보를 표시하고자 하는 PostgreSQL DB 인스턴스 이름을 선택합니다.
3. Connectivity & security(연결성 및 보안) 탭에 있는 Manage IAM roles(IAM 역할 관리) 섹션의 이 인스턴스에 IAM 역할 추가에서 추가할 역할을 선택합니다.
4. 기능에서 s3Export를 선택합니다.
5. [Add role]을 선택합니다.

AWS CLI

CLI를 사용하여 PostgreSQL DB 인스턴스에 대해 IAM 역할을 추가하려면

- 다음 명령을 사용해 `my-db-instance`라는 PostgreSQL DB 인스턴스에 역할을 추가합니다. *`your-role-arn`*을 이전 단계에서 기록한 정책 ARN으로 교체합니다. `s3Export` 옵션의 값에 대해 `--feature-name`를 사용합니다.

  Example

  대상 LinuxmacOS, 또는Unix:

```
aws rds add-role-to-db-instance \
   --db-instance-identifier my-db-instance \
   --feature-name s3Export \
   --role-arn your-role-arn   \
   --region your-region
```

  Windows의 경우:

```
aws rds add-role-to-db-instance ^
   --db-instance-identifier my-db-instance ^
   --feature-name s3Export ^
```

```
    --role-arn your-role-arn ^
    --region your-region
```

aws_s3.query_export_to_s3 함수를 사용하여 쿼리 데이터 내보내기

aws_s3.query_export_to_s3 함수를 호출하여 PostgreSQL 데이터를 Amazon S3으로 내보냅니다.

주제

- 사전 조건
- aws_s3.query_export_to_s3 호출
- 사용자 지정 구분 기호를 사용하는 CSV 파일로 내보내기
- 인코딩을 사용하여 이진 파일로 내보내기

사전 조건

aws_s3.query_export_to_s3 함수를 사용하기 전에 다음 사전 조건을 충족해야 합니다.

- Amazon S3으로 데이터 내보내기 개요에 설명된 대로 필요한 PostgreSQL 확장을 설치합니다.
- 내보낼 Amazon S3 파일 경로 지정에 설명된 대로 데이터를 내보낼 Amazon S3 위치를 결정합니다.
- Amazon S3 버킷에 대한 액세스 권한 설정에 설명된 대로 Amazon S3에 대한 내보내기 액세스 권한이 DB 인스턴스에 있는지 확인합니다.

다음 예제에서는 sample_table이라는 데이터베이스 테이블을 사용합니다. 이 예제에서는 *amzn-s3-demo-bucket*이라는 버킷으로 데이터를 내보냅니다. 예제 테이블과 데이터는 psql에서 다음 SQL 문을 사용하여 생성됩니다.

```
psql=> CREATE TABLE sample_table (bid bigint PRIMARY KEY, name varchar(80));
psql=> INSERT INTO sample_table (bid,name) VALUES (1, 'Monday'), (2,'Tuesday'), (3,
 'Wednesday');
```

aws_s3.query_export_to_s3 호출

다음은 aws_s3.query_export_to_s3 함수를 호출하는 기본 방법을 보여줍니다.

이 예제에서는 s3_uri_1 변수를 사용하여 Amazon S3 파일을 식별하는 정보가 포함된 구조를 식별합니다. aws_commons.create_s3_uri 함수를 사용하여 구조를 생성합니다.

```
psql=> SELECT aws_commons.create_s3_uri(
   'amzn-s3-demo-bucket',
   'sample-filepath',
   'us-west-2'
) AS s3_uri_1 \gset
```

파라미터가 다음 두 aws_s3.query_export_to_s3 함수 호출에 따라 달라도 이러한 예제에 대한 결과는 동일합니다. sample_table 표의 모든 행은 *amzn-s3-demo-bucket*이라는 버킷으로 내보내집니다.

```
psql=> SELECT * FROM aws_s3.query_export_to_s3('SELECT * FROM
 sample_table', :'s3_uri_1');

psql=> SELECT * FROM aws_s3.query_export_to_s3('SELECT * FROM
 sample_table', :'s3_uri_1', options :='format text');
```

파라미터는 다음과 같이 설명됩니다.

- 'SELECT * FROM sample_table' – 첫 번째 파라미터는 SQL 쿼리를 포함하는 필수 텍스트 문자열입니다. PostgreSQL 엔진은 이 쿼리를 실행합니다. 쿼리 결과는 다른 파라미터에서 식별된 S3 버킷에 복사됩니다.
- :'s3_uri_1' – 이 파라미터는 Amazon S3 파일을 식별하는 구조입니다. 이 예제에서는 변수를 사용하여 이전에 생성된 구조를 식별합니다. 대신 다음과 같이 aws_commons.create_s3_uri 함수 호출 내에 aws_s3.query_export_to_s3 함수 호출을 인라인을 포함시켜 구조를 생성할 수 있습니다.

  ```
  SELECT * from aws_s3.query_export_to_s3('select * from sample_table',
     aws_commons.create_s3_uri('amzn-s3-demo-bucket', 'sample-filepath', 'us-west-2')
  );
  ```

- options :='format text' – options 파라미터는 PostgreSQL COPY 인수를 포함하는 선택적 텍스트 문자열입니다. 복사 프로세스에서는 PostgreSQL COPY 명령의 인수 및 형식을 사용합니다.

지정된 파일이 Amazon S3 버킷에 없으면 생성됩니다. 파일이 이미 있는 경우 파일을 덮어씁니다. Amazon S3에서 내보낸 데이터에 액세스하는 구문은 다음과 같습니다.

```
s3-region://bucket-name[/path-prefix]/file-prefix
```

내보내는 데이터가 클 경우 각각 최대 크기가 약 6GB인 여러 파일에 저장됩니다. 추가 파일 이름의 파일 접두사도 동일하지만 _part*XX*가 추가됩니다. *XX*는 2, 3 등을 나타냅니다. 예를 들어 데이터 파일을 저장하는 경로를 다음과 같이 지정한다고 가정합니다.

```
s3-us-west-2://amzn-s3-demo-bucket/my-prefix
```

내보내기 시 세 개의 데이터 파일을 만들어야 하는 경우 Amazon S3 버킷에 다음 데이터 파일이 포함됩니다.

```
s3-us-west-2://amzn-s3-demo-bucket/my-prefix
s3-us-west-2://amzn-s3-demo-bucket/my-prefix_part2
s3-us-west-2://amzn-s3-demo-bucket/my-prefix_part3
```

이 함수에 대한 전체 참조 및 이 함수를 호출하는 추가 방법은 aws_s3.query_export_to_s3 단원을 참조하세요. Amazon S3에서 파일에 액세스하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 객체 보기를 참조하세요.

사용자 지정 구분 기호를 사용하는 CSV 파일로 내보내기

다음 예제에서는 aws_s3.query_export_to_s3 함수를 호출하여 사용자 지정 구분 기호를 사용하는 파일로 데이터를 내보내는 방법을 보여줍니다. 이 예제에서는 PostgreSQL COPY 명령의 인수를 사용하여 쉼표로 구분된 값(CSV) 형식과 콜론(:) 구분 기호를 지정합니다.

```
SELECT * from aws_s3.query_export_to_s3('select * from basic_test', :'s3_uri_1',
 options :='format csv, delimiter $$:$$');
```

인코딩을 사용하여 이진 파일로 내보내기

다음 예제에서는 aws_s3.query_export_to_s3 함수를 호출하여 Windows-1253 인코딩이 있는 이진 파일로 데이터를 내보내는 방법을 보여줍니다.

```
SELECT * from aws_s3.query_export_to_s3('select * from basic_test', :'s3_uri_1',
 options :='format binary, encoding WIN1253');
```

함수 참조

함수

- aws_s3.query_export_to_s3

- aws_commons.create_s3_uri

aws_s3.query_export_to_s3

PostgreSQL 쿼리 결과를 Amazon S3 버킷으로 내보냅니다. aws_s3 확장은 aws_s3.query_export_to_s3 함수를 제공합니다.

두 가지 필수 파라미터는 query 및 s3_info입니다. 이러한 파라미터는 내보낼 쿼리를 정의하고 내보낼 Amazon S3 버킷을 식별합니다. 다양한 내보내기 파라미터를 정의하기 위해 options라는 선택적 파라미터가 제공됩니다. aws_s3.query_export_to_s3 함수 사용 예는 aws_s3.query_export_to_s3 함수를 사용하여 쿼리 데이터 내보내기 단원을 참조하십시오.

구문

```
aws_s3.query_export_to_s3(
    query text,
    s3_info aws_commons._s3_uri_1,
    options text,
    kms_key text
)
```

입력 파라미터

query

PostgreSQL 엔진이 실행하는 SQL 쿼리를 포함하는 필수 텍스트 문자열입니다. 이 쿼리의 결과는 s3_info 파라미터에서 식별된 S3 버킷에 복사됩니다.

s3_info

S3 객체에 대한 다음 정보를 포함하는 aws_commons._s3_uri_1 복합 키입니다.

- bucket – 파일을 포함할 Amazon S3 버킷의 이름입니다.
- file_path – Amazon S3 파일 이름 및 경로입니다.
- region - 버킷이 있는 AWS 리전입니다. AWS 리전 이름 및 연결된 값의 목록은 리전, 가용 영역 및 로컬 영역 섹션을 참조하십시오.

  현재, 이 값은 내보내는 DB 인스턴스의 AWS 리전과 동일해야 합니다. 기본값은 내보내는 DB 인스턴스의 AWS 리전입니다.

aws_commons._s3_uri_1 복합 구조를 생성하려면 aws_commons.create_s3_uri 함수를 참조하십시오.

options

PostgreSQL COPY 명령에 대한 인수를 포함하는 선택적 텍스트 문자열입니다. 이러한 인수는 내보낼 때 데이터를 복사하는 방법을 지정합니다. 자세한 내용은 PostgreSQL COPY 설명서를 참조하십시오.

kms_key text

데이터를 내보낼 S3 버킷의 고객 관리형 KMS 키가 포함된 선택적 텍스트 문자열입니다.

대체 입력 파라미터

`s3_info` 파라미터 대신에 확장 파라미터 세트를 사용하면 테스트에 도움이 됩니다. 다음은 `aws_s3.query_export_to_s3` 함수에 대한 추가 구문 변형입니다.

`s3_info` 파라미터를 사용해 Amazon S3 파일을 식별하는 대신 `bucket`, `file_path` 및 `region` 파라미터의 조합을 사용하십시오.

```
aws_s3.query_export_to_s3(
    query text,
    bucket text,
    file_path text,
    region text,
    options text,
    kms_key text
)
```

query

PostgreSQL 엔진이 실행하는 SQL 쿼리를 포함하는 필수 텍스트 문자열입니다. 이 쿼리의 결과는 `s3_info` 파라미터에서 식별된 S3 버킷에 복사됩니다.

bucket

파일이 들어 있는 Amazon S3 버킷의 이름이 포함된 필수 텍스트 문자열입니다.

file_path

파일 경로를 포함한 Amazon S3 파일 이름이 포함된 필수 텍스트 문자열입니다.

region

버킷이 있는 AWS 리전을 포함하는 선택적 텍스트 문자열입니다. AWS 리전 이름 및 연결된 값의 목록은 리전, 가용 영역 및 로컬 영역 섹션을 참조하십시오.

현재, 이 값은 내보내는 DB 인스턴스의 AWS 리전과 동일해야 합니다. 기본값은 내보내는 DB 인스턴스의 AWS 리전입니다.

options

PostgreSQL COPY 명령에 대한 인수를 포함하는 선택적 텍스트 문자열입니다. 이러한 인수는 내보낼 때 데이터를 복사하는 방법을 지정합니다. 자세한 내용은 PostgreSQL COPY 설명서를 참조하십시오.

kms_key text

데이터를 내보낼 S3 버킷의 고객 관리형 KMS 키가 포함된 선택적 텍스트 문자열입니다.

출력 파라미터

```
aws_s3.query_export_to_s3(
    OUT rows_uploaded bigint,
    OUT files_uploaded bigint,
    OUT bytes_uploaded bigint
)
```

rows_uploaded

지정된 쿼리에 대해 Amazon S3에 성공적으로 업로드된 테이블 행 수입니다.

files_uploaded

Amazon S3에 업로드된 파일 수입니다. 파일은 약 6GB 크기로 생성됩니다. 생성된 각 추가 파일 이름에 _part*XX*가 추가됩니다. *XX*는 2, 3 등을 나타냅니다.

bytes_uploaded

Amazon S3에 업로드된 총 바이트 수입니다.

예시

```
psql=> SELECT * from aws_s3.query_export_to_s3('select * from sample_table', 'amzn-s3-demo-bucket', 'sample-filepath');
psql=> SELECT * from aws_s3.query_export_to_s3('select * from sample_table', 'amzn-s3-demo-bucket', 'sample-filepath','us-west-2');
psql=> SELECT * from aws_s3.query_export_to_s3('select * from sample_table', 'amzn-s3-demo-bucket', 'sample-filepath','us-west-2','format text');
```

aws_commons.create_s3_uri

Amazon S3 파일 정보를 저장할 aws_commons._s3_uri_1 구조를 생성합니다. aws_commons.create_s3_uri 함수의 s3_info 파라미터에서 aws_s3.query_export_to_s3 함수의 결과를 사용합니다. aws_commons.create_s3_uri 함수 사용 예는 내보낼 Amazon S3 파일 경로 지정 단원을 참조하십시오.

구문

```
aws_commons.create_s3_uri(
   bucket text,
   file_path text,
   region text
)
```

입력 파라미터

bucket

파일의 Amazon S3 버킷 이름이 포함된 필수 텍스트 문자열입니다.

file_path

파일 경로를 포함한 Amazon S3 파일 이름이 포함된 필수 텍스트 문자열입니다.

region

파일이 위치한 AWS 리전이 포함된 필수 텍스트 문자열입니다. AWS 리전 이름 및 연결된 값의 목록은 리전, 가용 영역 및 로컬 영역 섹션을 참조하십시오.

Amazon S3 액세스 문제 해결

데이터를 Amazon S3로 내보내려고 할 때 연결 문제가 발생하면 먼저 DB 인스턴스에 연결된 VPC 보안 그룹의 아웃바운드 액세스 규칙이 네트워크 연결을 허용하는지 확인합니다. 특히 보안 그룹에는 DB 인스턴스가 TCP 트래픽을 포트 443 및 IPv4 주소(0.0.0.0/0)로 보낼 수 있도록 허용하는 규칙이 있어야 합니다. 자세한 내용은 보안 그룹을 생성하여 VPC 내부의 DB 인스턴스에 대한 액세스를 제공을 참조하세요.

권장 사항은 다음을 참조하세요.

- Amazon RDS 자격 증명 및 액세스 문제 해결

- Amazon Simple Storage Service 사용 설명서의 Amazon S3 문제 해결
- IAM 사용 설명서의 Amazon S3 문제 해결 및 IAM

RDS for PostgreSQL DB 인스턴스에서 AWS Lambda 함수 호출

AWS Lambda는 서버를 프로비저닝하거나 관리하지 않고도 코드를 실행할 수 있는 이벤트 기반 컴퓨팅 서비스입니다. RDS for PostgreSQL을 비롯한 많은 AWS 서비스에서 사용할 수 있습니다. 예를 들어 Lambda 함수를 사용하여 데이터베이스의 이벤트 알림을 처리하거나 새 파일이 Amazon S3에 업로드될 때마다 파일에서 데이터를 로드할 수 있습니다. Lambda에 대한 자세한 내용은 AWS Lambda 개발자 안내서에서 AWS Lambda란 무엇입니까?를 참조하세요.

> **Note**
>
> AWS Lambda 함수 호출은 다음 RDS for PostgreSQL 버전에서 지원됩니다.
>
> - 모든 PostgreSQL 17 버전
> - 모든 PostgreSQL 16 버전
> - 모든 PostgreSQL 15 버전
> - PostgreSQL 14.1 이상의 마이너 버전
> - PostgreSQL 13.2 이상의 마이너 버전
> - PostgreSQL 12.6 이상의 마이너 버전

Lambda 함수와 함께 작동하도록 RDS for PostgreSQL을 설정하는 것은 AWS Lambda, IAM, VPC, RDS for PostgreSQL DB 인스턴스를 포괄하는 다단계 프로세스입니다. 다음에서 필요한 단계에 대한 요약을 찾을 수 있습니다.

Lambda 함수에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 Lambda 시작하기와 AWS Lambda 기본을 참조하세요.

주제

- 1단계: AWS Lambda에 대한 아웃바운드 연결을 위해 RDS for PostgreSQL DB 인스턴스 구성
- 2단계: RDS for PostgreSQL DB 인스턴스 및 AWS Lambda에 대한 IAM 구성
- 3단계: RDS for PostgreSQL DB 인스턴스용 aws_lambda 확장 설치
- 4단계: RDS for PostgreSQL DB 인스턴스와 함께 Lambda 도우미 함수 사용(선택 사항)
- 5단계: RDS for PostgreSQL DB 인스턴스에서 Lambda 함수 호출
- 6단계: 다른 사용자에게 Lambda 함수를 호출할 수 있는 권한 부여
- 예제: RDS for PostgreSQL DB 인스턴스에서 Lambda 함수 호출
- Lambda 함수 오류 메시지

- AWS Lambda 함수 및 파라미터 참조

1단계: AWS Lambda에 대한 아웃바운드 연결을 위해 RDS for PostgreSQL DB 인스턴스 구성

Lambda 함수는 항상 AWS Lambda 서비스가 소유한 Amazon VPC 내에서 실행됩니다. Lambda는 이 VPC에 네트워크 액세스 및 보안 규칙을 적용하고 VPC를 자동으로 유지 관리 및 모니터링합니다. RDS for PostgreSQL DB 인스턴스는 네트워크 트래픽을 Lambda 서비스의 VPC로 전송합니다. 이를 구성하는 방법은 DB 인스턴스가 퍼블릭인지 프라이빗인지에 따라 다릅니다.

- 퍼블릭 RDS for PostgreSQL DB 인스턴스 – DB 인스턴스는 VPC의 퍼블릭 서브넷에 있고 인스턴스의 "PubliclyAccessible" 속성이 `true`인 경우 퍼블릭입니다. 이 속성의 값을 찾으려면 describe-db-instances AWS CLI 명령을 사용합니다. 또는 AWS Management Console을 사용하여 연결 및 보안 탭을 열고 퍼블릭 액세스 가능이 예인지 확인할 수 있습니다. 인스턴스가 VPC의 퍼블릭 서브넷에 있는지 확인하려면 AWS Management Console 또는 AWS CLI를 사용할 수 있습니다.

  Lambda에 대한 액세스를 설정하려면 AWS Management Console 또는AWS CLI를 사용하여 VPC의 보안 그룹에 대한 아웃바운드 규칙을 생성합니다. 아웃바운드 규칙은 TCP가 포트 443을 사용하여 IPv4 주소(0.0.0.0/0)로 패킷을 보낼 수 있도록 지정합니다.
- 프라이빗 RDS for PostgreSQL DB 인스턴스 – 이 경우 인스턴스의 "PubliclyAccessible" 속성은 `false`이거나 프라이빗 서브넷에 있습니다. 인스턴스가 Lambda와 함께 작동하도록 허용하려면 Network Address Translation)NAT 게이트웨이를 사용할 수 있습니다. 자세한 내용은 NAT 게이트웨이 단원을 참조하세요. 또는 Lambda용 VPC 엔드포인트로 VPC를 구성합니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 참조하세요. 엔드포인트는 RDS for PostgreSQL DB 인스턴스의 Lambda 함수 호출에 대한 응답을 반환합니다. VPC 엔드포인트는 자체 프라이빗 DNS 확인을 사용합니다. RDS for PostgreSQL은 `rds.custom_dns_resolution` 값을 기본값 0(사용 설정되지 않음)에서 1로 변경할 때까지 Lambda VPC 엔드포인트를 사용할 수 없습니다. 그렇게 하려면 다음을 수행하세요.
  - 사용자 지정 DB 파라미터 그룹을 생성합니다.
  - `rds.custom_dns_resolution` 파라미터 값을 기본값인 `0`에서 `1`로 변경합니다.
  - 사용자 정의 DB 파라미터 그룹을 사용하도록 DB 인스턴스를 수정합니다.
  - 수정된 파라미터가 적용되도록 인스턴스를 재부팅합니다.

이제 VPC가 네트워크 수준에서 AWS Lambda VPC와 상호 작용할 수 있습니다. 다음으로 IAM을 사용하여 권한을 구성합니다.

2단계: RDS for PostgreSQL DB 인스턴스 및 AWS Lambda에 대한 IAM 구성

RDS for PostgreSQL DB 인스턴스에서 Lambda 함수를 호출하려면 특정 권한이 필요합니다. 필요한 권한을 구성하려면 Lambda 함수 호출을 허용하는 IAM 정책을 생성하고 해당 정책을 역할에 할당한 다음 DB 인스턴스에 그 역할을 적용하는 것이 좋습니다. 이 접근 방식은 DB 인스턴스에 사용자를 대신하여 지정된 Lambda 함수를 호출할 수 있는 권한을 부여합니다. 다음 단계에서는 AWS CLI를 사용하여 이를 수행하는 방법을 보여줍니다.

Lambda와 함께 Amazon RDS 인스턴스를 사용하기 위한 IAM 권한 구성

1. create-policy AWS CLI 명령을 사용하여 RDS for PostgreSQL DB 인스턴스가 지정된 Lambda 함수를 호출하도록 허용하는 IAM 정책을 생성합니다. 문 ID(Sid)는 정책 문에 대한 선택적 설명이며 사용량에 영향을 미치지 않습니다. 이 정책은 DB 인스턴스에 지정된 Lambda 함수를 호출하는 데 필요한 최소 권한을 부여합니다.

   ```
   aws iam create-policy  --policy-name rds-lambda-policy --policy-document '{
       "Version": "2012-10-17",
       "Statement": [
           {
           "Sid": "AllowAccessToExampleFunction",
           "Effect": "Allow",
           "Action": "lambda:InvokeFunction",
           "Resource": "arn:aws:lambda:aws-region:444455556666:function:my-function"
           }
       ]
   }'
   ```

   또는 모든 Lambda 함수를 호출할 수 있도록 미리 정의된 AWSLambdaRole 정책을 사용할 수 있습니다. 자세한 내용은 Lambda에 대한 자격 증명 기반 IAM 정책을 참조하세요.

2. create-role AWS CLI 명령을 사용하여 정책이 런타임에 수임할 수 있는 IAM 역할을 생성합니다.

   ```
   aws iam create-role  --role-name rds-lambda-role --assume-role-policy-document '{
       "Version": "2012-10-17",
       "Statement": [
           {
           "Effect": "Allow",
           "Principal": {
               "Service": "rds.amazonaws.com"
           },
           "Action": "sts:AssumeRole"
   ```

```
        }
    ]
}'
```

3. attach-role-policy AWS CLI 명령을 사용하여 역할에 정책을 적용합니다.

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::444455556666:policy/rds-lambda-policy \
    --role-name rds-lambda-role --region aws-region
```

4. add-role-to-db-instance AWS CLI 명령을 사용하여 RDS for PostgreSQL DB 인스턴스에 역할을 적용합니다. 이 마지막 단계를 통해 DB 인스턴스의 데이터베이스 사용자가 Lambda 함수를 호출할 수 있습니다.

```
aws rds add-role-to-db-instance \
       --db-instance-identifier my-instance-name \
       --feature-name Lambda \
       --role-arn  arn:aws:iam::444455556666:role/rds-lambda-role   \
       --region aws-region
```

VPC 및 IAM 구성이 완료되면 이제 aws_lambda 확장을 설치할 수 있습니다. 확장은 언제든지 설치할 수 있지만 올바른 VPC 지원 및 IAM 권한을 설정할 때까지 aws_lambda 확장은 RDS for PostgreSQL DB 인스턴스의 기능에 아무 것도 추가하지 않습니다.

3단계: RDS for PostgreSQL DB 인스턴스용 **aws_lambda** 확장 설치

AWS Lambda을 RDS for PostgreSQL DB 인스턴스와 사용하려면 RDS for PostgreSQL DB 인스턴스에 aws_lambda PostgreSQL 확장을 추가합니다. 이 확장은 RDS for PostgreSQL DB 인스턴스에 PostgreSQL에서 Lambda 함수를 호출할 수 있는 기능을 제공합니다.

RDS for PostgreSQL DB 인스턴스에 **aws_lambda** 확장 설치

PostgreSQL psql 명령줄 또는 pgAdmin 도구를 사용하여 RDS for PostgreSQL DB 인스턴스에 연결합니다.

1. rds_superuser 권한이 있는 사용자로 RDS for PostgreSQL DB 인스턴스에 연결합니다. 기본 postgres 사용자가 예제에 표시됩니다.

```
psql -h instance.444455556666.aws-region.rds.amazonaws.com -U postgres -p 5432
```

2. aws_lambda 확장을 설치합니다. aws_commons 확장도 필요합니다. 이 확장은 aws_lambda 및 기타 여러 PostgreSQL용 Aurora 확장에 대한 도우미 함수를 제공합니다. RDS for PostgreSQLDB 인스턴스에 아직 없는 경우 다음과 같이 aws_lambda와 함께 설치됩니다.

```
CREATE EXTENSION IF NOT EXISTS aws_lambda CASCADE;
NOTICE:  installing required extension "aws_commons"
CREATE EXTENSION
```

aws_lambda 확장이 DB 인스턴스에 설치됩니다. 이제 Lambda 함수를 호출하기 위한 편의 구조를 생성할 수 있습니다.

4단계: RDS for PostgreSQL DB 인스턴스와 함께 Lambda 도우미 함수 사용 (선택 사항)

aws_commons 확장의 도우미 함수를 사용하여 PostgreSQL에서 보다 쉽게 호출할 수 있는 엔터티를 준비할 수 있습니다. 이렇게 하려면 Lambda 함수에 대한 다음 정보가 필요합니다.

- 함수 이름(Function name) – Lambda 함수의 이름, Amazon 리소스 이름(ARN), 버전 또는 별칭입니다. 2단계: 인스턴스 및 Lambda에 대한 IAM 구성에서 생성한 IAM 정책에는 ARN이 필요하므로 함수의 ARN을 사용하는 것이 좋습니다.
- AWS 리전 – (선택 사항) RDS for PostgreSQL DB 인스턴스와 동일한 리전에 있지 않은 경우 Lambda 함수가 있는 AWS 리전입니다.

Lambda 함수 이름 정보를 보관하려면 aws_commons.create_lambda_function_arn 함수를 사용합니다. 이 도우미 함수는 호출 함수에 필요한 세부 정보를 사용하여 aws_commons._lambda_function_arn_1 복합 구조를 생성합니다. 다음에서 이 복합 구조를 설정하는 세 가지 대안을 찾을 수 있습니다.

```
SELECT aws_commons.create_lambda_function_arn(
   'my-function',
   'aws-region'
) AS aws_lambda_arn_1 \gset
```

```
SELECT aws_commons.create_lambda_function_arn(
   '111122223333:function:my-function',
   'aws-region'
) AS lambda_partial_arn_1 \gset
```

```
SELECT aws_commons.create_lambda_function_arn(
   'arn:aws:lambda:aws-region:111122223333:function:my-function'
) AS lambda_arn_1 \gset
```

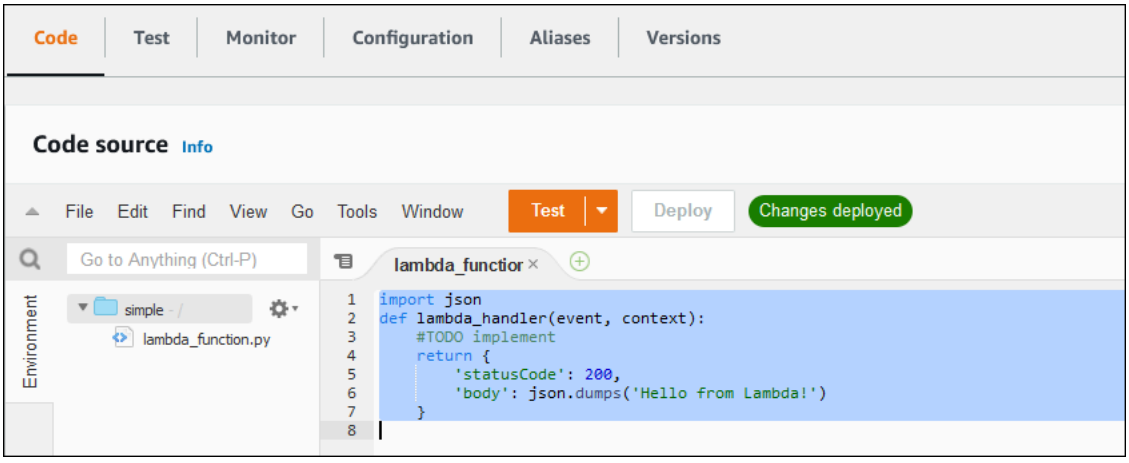
이러한 값은 aws_lambda.invoke 함수 호출에 사용할 수 있습니다. 예를 보려면 5단계: RDS for PostgreSQL DB 인스턴스에서 Lambda 함수 호출 섹션을 참조하세요.

5단계: RDS for PostgreSQL DB 인스턴스에서 Lambda 함수 호출

aws_lambda.invoke 함수는 invocation_type에 따라 동기식 또는 비동기식으로 작동합니다. 이 파라미터에 대한 두 가지 대안은 다음과 같이 RequestResponse(기본값) 및 Event입니다.

- **RequestResponse** - 이 호출 유형은 동기식이며, 호출 유형을 지정하지 않고 호출할 때의 기본 동작입니다. 응답 페이로드에는 aws_lambda.invoke 함수의 결과가 포함됩니다. 워크플로가 진행하기 전에 Lambda 함수에서 결과를 수신해야 하는 경우 이 호출 유형을 사용합니다.
- **Event** - 이 호출 유형은 비동기식이며, 응답에 결과가 포함된 페이로드가 포함되지 않습니다. 워크플로에서 처리를 계속하기 위해 Lambda 함수의 결과가 필요하지 않은 경우 이 호출 유형을 사용합니다.

설정에 대한 간단한 테스트로 psql을 사용하여 DB 인스턴스에 연결하고 명령줄에서 예제 함수를 호출할 수 있습니다. 다음 스크린샷에 표시된 간단한 Python 함수와 같이 Lambda 서비스에 기본 함수 중 하나가 설정되어 있다고 가정합니다.



예제 함수 호출

1. psql 또는 pgAdmin을 사용하여 DB 인스턴스에 연결합니다.

```
psql -h instance.444455556666.aws-region.rds.amazonaws.com -U postgres -p 5432
```

2. ARN을 사용하여 함수를 호출합니다.

```
SELECT * from
 aws_lambda.invoke(aws_commons.create_lambda_function_arn('arn:aws:lambda:aws-
region:444455556666:function:simple', 'us-west-1'), '{"body": "Hello from
 Postgres!"}'::json );
```

응답은 다음과 같습니다.

```
status_code |                          payload                          |
 executed_version | log_result
-------------+-----------------------------------------------------------
+------------------+------------
         200 | {"statusCode": 200, "body": "\"Hello from Lambda!\""} | $LATEST
     |
(1 row)
```

호출 시도가 성공하지 못한 경우 Lambda 함수 오류 메시지 섹션을 참조하세요.

6단계: 다른 사용자에게 Lambda 함수를 호출할 수 있는 권한 부여

이 시점에서는 자신만 `rds_superuser` 권한으로 Lambda 함수를 호출할 수 있습니다. 자신이 생성하는 함수를 다른 사용자가 호출할 수 있도록 허용하려면 권한을 부여해야 합니다.

Lambda 함수를 호출할 수 있는 권한을 부여하는 방법

1. `psql` 또는 pgAdmin을 사용하여 DB 인스턴스에 연결합니다.

```
psql -h instance.444455556666.aws-region.rds.amazonaws.com -U postgres -p 5432
```

2. SQL 명령을 실행합니다.

```
postgres=>  GRANT USAGE ON SCHEMA aws_lambda TO db_username;
GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA aws_lambda TO db_username;
```

예제: RDS for PostgreSQL DB 인스턴스에서 Lambda 함수 호출

다음에서 aws_lambda.invoke 함수를 호출하는 몇 가지 예를 찾을 수 있습니다. 예제 대부분은 4단계: RDS for PostgreSQL DB 인스턴스와 함께 Lambda 도우미 함수 사용(선택 사항)에서 생성한 복합 구조 aws_lambda_arn_1을 사용하여 함수 세부 정보 전달을 단순화합니다. 비동기 호출의 예는 예제: Lambda 함수의 비동기 (이벤트) 호출 섹션을 참조하세요. 나열된 다른 모든 예에서 동기 호출을 사용합니다.

Lambda 호출 유형에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 Lambda 함수 호출을 참조하세요. aws_lambda_arn_1에 대한 자세한 정보는 aws_commons.create_lambda_function_arn 섹션을 참조하십시오.

예제 목록

- 예제: Lambda 함수의 동기(RequestResponse) 호출
- 예제: Lambda 함수의 비동기 (이벤트) 호출
- 예: 함수 응답에서 Lambda 실행 로그 캡처
- 예제: Lambda 함수에 클라이언트 컨텍스트 포함
- 예제: 특정 버전의 Lambda 함수 호출

예제: Lambda 함수의 동기(RequestResponse) 호출

다음은 동기식 Lambda 함수 호출의 두 가지 예입니다. 이러한 aws_lambda.invoke 함수 호출의 결과는 동일합니다.

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json);
```

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json, 'RequestResponse');
```

파라미터는 다음과 같이 설명됩니다.

- :'aws_lambda_arn_1' - 이 파라미터는 4단계: RDS for PostgreSQL DB 인스턴스와 함께 Lambda 도우미 함수 사용(선택 사항)에서 생성된 복합 구조를 aws_commons.create_lambda_function_arn 도우미 함수와 함께 식별합니다. 다음과 같이 aws_lambda.invoke 호출 내에서 이 구조를 인라인으로 생성할 수도 있습니다.

```
SELECT * FROM aws_lambda.invoke(aws_commons.create_lambda_function_arn('my-function',
 'aws-region'),
'{"body": "Hello from Postgres!"}'::json
);
```

- `'{"body": "Hello from PostgreSQL!"}'::json` – Lambda 함수에 전달할 JSON 페이로드입니다.
- `'RequestResponse'` – Lambda 호출 유형.

예제: Lambda 함수의 비동기 (이벤트) 호출

다음은 비동기 Lambda 함수 호출의 일례입니다. `Event` 호출 유형은 지정된 입력 페이로드를 사용하여 Lambda 함수 호출을 예약하고 즉시 반환합니다. Lambda 함수 결과에 의존하지 않는 특정 워크플로에서 `Event` 호출 유형을 사용합니다.

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json, 'Event');
```

예: 함수 응답에서 Lambda 실행 로그 캡처

`aws_lambda.invoke` 함수 호출에서 `log_type` 파라미터를 사용하여 함수 응답에 실행 로그의 마지막 4KB를 포함할 수 있습니다. 기본적으로 이 파라미터는 `None`으로 설정되지만 다음과 같이 `Tail`을 지정하여 응답에서 Lambda 실행 로그의 결과를 캡처할 수 있습니다.

```
SELECT *, select convert_from(decode(log_result, 'base64'), 'utf-8') as log FROM
 aws_lambda.invoke(:'aws_lambda_arn_1', '{"body": "Hello from Postgres!"}'::json,
 'RequestResponse', 'Tail');
```

응답에 실행 로그를 포함하도록 aws_lambda.invoke 함수의 `log_type` 파라미터를 `Tail`로 설정합니다. `log_type` 파라미터의 기본값은 `None`입니다.

반환 문자열 `log_result` 은 `base64` 인코딩된 문자열입니다. `decode` 및 `convert_from` PostgreSQL 함수의 조합을 사용하여 내용을 디코딩할 수 있습니다.

`log_type`에 대한 자세한 정보는 aws_lambda.invoke 섹션을 참조하십시오.

예제: Lambda 함수에 클라이언트 컨텍스트 포함

`aws_lambda.invoke` 함수에는 다음과 같이 페이로드와 별도로 정보를 전달하는 데 사용할 수 있는 `context` 파라미터가 있습니다.

```
SELECT *, convert_from(decode(log_result, 'base64'), 'utf-8') as log FROM
 aws_lambda.invoke(:'aws_lambda_arn_1', '{"body": "Hello from Postgres!"}'::json,
 'RequestResponse', 'Tail');
```

클라이언트 컨텍스트를 포함하려면 aws_lambda.invoke 함수의 `context` 파라미터에 JSON 객체를 사용하세요.

`context` 파라미터에 대한 자세한 내용은 aws_lambda.invoke 레퍼런스를 참조하세요.

예제: 특정 버전의 Lambda 함수 호출

`aws_lambda.invoke` 호출에 `qualifier` 파라미터를 포함하여 Lambda 함수의 특정 버전을 지정할 수 있습니다. 다음에서 버전의 별칭으로 '*custom_version*'을 사용하여 이를 수행하는 예를 찾을 수 있습니다.

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json, 'RequestResponse', 'None', NULL, 'custom_version');
```

다음과 같이 대신 함수 이름 세부 정보와 함께 Lambda 함수 한정자를 제공할 수도 있습니다.

```
SELECT * FROM aws_lambda.invoke(aws_commons.create_lambda_function_arn('my-
function:custom_version', 'us-west-2'),
'{"body": "Hello from Postgres!"}'::json);
```

`qualifier` 및 기타 파라미터에 대한 자세한 내용은 aws_lambda.invoke 레퍼런스를 참조하세요.

Lambda 함수 오류 메시지

다음 목록에서 가능한 원인 및 해결 방법을 비롯하여 오류 메시지에 대한 정보를 찾을 수 있습니다.

- VPC 구성 문제

  VPC 구성 문제로 인해 연결을 시도할 때 다음과 같은 오류 메시지가 발생할 수 있습니다.

  ```
  ERROR:  invoke API failed
  DETAIL: AWS Lambda client returned 'Unable to connect to endpoint'.
  ```

```
CONTEXT:  SQL function "invoke" statement 1
```

이 오류의 일반적인 원인은 잘못 구성된 VPC 보안 그룹입니다. VPC가 Lambda VPC에 연결할 수 있도록 VPC 보안 그룹의 포트 443에서 TCP에 대한 아웃바운드 규칙이 열려 있는지 확인합니다.

DB 인스턴스가 프라이빗인 경우 VPC에 대한 프라이빗 DNS 설정을 확인합니다. 1단계: AWS Lambda에 대한 아웃바운드 연결을 위해 RDS for PostgreSQL DB 인스턴스 구성에 설명된 대로 `rds.custom_dns_resolution` 파라미터를 1로 설정하고 AWS PrivateLink를 설정했는지 확인합니다. 자세한 내용은 인터페이스 VPC 엔드포인트(AWS PrivateLink)를 참조하세요.

- Lambda 함수를 호출하는 데 필요한 권한 부족

  다음 오류 메시지 중 하나가 표시되면 함수를 호출하는 사용자(역할)에 적절한 권한이 없는 것입니다.

  ```
  ERROR:  permission denied for schema aws_lambda
  ```

  ```
  ERROR:  permission denied for function invoke
  ```

  Lambda 함수를 호출하려면 사용자(역할)에게 특정 권한을 부여해야 합니다. 자세한 내용은 6단계: 다른 사용자에게 Lambda 함수를 호출할 수 있는 권한 부여을 참조하세요.

- Lambda 함수의 잘못된 오류 처리

  요청 처리 중에 Lambda 함수가 예외를 발생키면, `aws_lambda.invoke` 는 다음과 같은 PostgreSQL 오류와 함께 실패합니다.

  ```
  SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
   Postgres!"}'::json);
  ERROR:  lambda invocation failed
  DETAIL:  "arn:aws:lambda:us-west-2:555555555555:function:my-function" returned error
   "Unhandled", details: "<Error details string>".
  ```

  Lambda 함수 또는 PostgreSQL 애플리케이션의 오류를 처리해야 합니다.

AWS Lambda 함수 및 파라미터 참조

다음은 RDS for PostgreSQL에서 Lambda 간접 호출에 사용할 함수 및 파라미터에 대한 참조입니다.

함수 및 파라미터

- aws_lambda.invoke
- aws_commons.create_lambda_function_arn
- aws_lambda 파라미터

aws_lambda.invoke

RDS for PostgreSQL DB 인스턴스에 대해 Lambda 함수를 실행합니다.

Lambda 함수 호출에 대한 자세한 내용은 AWS Lambda 개발자 안내서에서 호출을 참조하세요.

구문

JSON

```
aws_lambda.invoke(
IN function_name TEXT,
IN payload JSON,
IN region TEXT DEFAULT NULL,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSON DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSON,
OUT executed_version TEXT,
OUT log_result TEXT)
```

```
aws_lambda.invoke(
IN function_name aws_commons._lambda_function_arn_1,
IN payload JSON,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSON DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSON,
OUT executed_version TEXT,
OUT log_result TEXT)
```

JSONB

```
aws_lambda.invoke(
IN function_name TEXT,
IN payload JSONB,
IN region TEXT DEFAULT NULL,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSONB DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSONB,
OUT executed_version TEXT,
OUT log_result TEXT)
```

```
aws_lambda.invoke(
IN function_name aws_commons._lambda_function_arn_1,
IN payload JSONB,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSONB DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSONB,
OUT executed_version TEXT,
OUT log_result TEXT
)
```

입력 파라미터

function_name

Lambda 함수의 식별 이름입니다. 값은 함수 이름, ARN 또는 부분 ARN일 수 있습니다. 가능한 형식 목록은 AWS Lambda 개발자 안내서에서 Lambda 함수 이름 형식을 참조하세요.

payload

Lambda 함수에 대한 입력입니다. 형식은 JSON 또는 JSONB 일 수 있습니다. 자세한 내용은 PostgreSQL 설명서의 JSON 유형을 참조하십시오.

region

(선택 사항) 함수의 Lambda 리전입니다. 기본적으로 RDS는 `function_name`의 전체 ARN에서 AWS 리전을 확인하거나 RDS for PostgreSQL DB 인스턴스 리전을 사용합니다. 이 리전 값이 `function_name` ARN에 제공된 값과 충돌하면 오류가 발생합니다.

invocation_type

Lambda 함수의 호출 유형입니다. 값은 대소문자를 구분합니다. 가능한 값은 다음을 포함합니다.

- `RequestResponse` – 기본값입니다. Lambda 함수에 대한 이러한 유형의 호출은 동시에 발생하며 결과에 응답 페이로드를 돌려보냅니다. 워크플로가 Lambda 함수 결과 수신에 즉시 의존하는 경우 `RequestResponse` 호출 유형을 사용합니다.
- `Event` – Lambda 함수에 대한 이러한 유형의 호출은 비동기식이며 반환된 페이로드없이 즉시 반환됩니다. 워크플로를 이동하기 전에 Lambda 함수 결과가 필요하지 않은 경우 `Event` 호출 유형을 사용합니다.
- `DryRun` – 이 유형의 호출은 Lambda 함수를 실행하지 않고 액세스를 테스트합니다.

log_type

`log_result` 출력 파라미터에 반환할 Lambda 로그의 유형입니다. 값은 대소문자를 구분합니다. 가능한 값은 다음을 포함합니다.

- – 추적 반환된 `log_result` 출력 파라미터에는 실행 로그의 마지막 4KB가 포함됩니다.
- – 없음 Lambda 로그 정보가 반환되지 않았습니다.

context

JSON 또는 JSONB 형식의 클라이언트 컨텍스트 사용할 필드에는 보다 `custom` 및 `env` 가 포함됩니다.

한정자

호출할 Lambda 함수의 버전을 식별하는 한정자입니다. 이 값이 `function_name` ARN에 제공된 값과 충돌하면 오류가 발생합니다.

출력 파라미터

status_code

HTTP 상태 응답 코드입니다. 자세한 내용은 AWS Lambda 개발자 안내서에서 Lambda 호출 응답 요소를 참조하세요.

payload

실행된 Lambda 함수에서 반환된 정보입니다. 형식은 JSON 또는 JSONB입니다.

executed_version

실행된 Lambda 함수 버전입니다.

log_result

Lambda 함수가 호출 될 때 `log_type` 값이 `Tail` 인 경우 반환된 실행 로그 정보입니다. 결과에는 Base64로 인코딩된 실행 로그의 마지막 4KB가 포함됩니다.

aws_commons.create_lambda_function_arn

Lambda 파일 정보를 저장할 `aws_commons._lambda_function_arn_1` 구조를 생성합니다. aws_lambda.invoke `aws_commons.create_lambda_function_arn` 함수의 `function_name` 파라미터에 aws_lambda.invoke 함수의 결과를 사용할 수 있습니다.

구문

```
aws_commons.create_lambda_function_arn(
    function_name TEXT,
    region TEXT DEFAULT NULL
    )
    RETURNS aws_commons._lambda_function_arn_1
```

입력 파라미터

function_name

Lambda 함수 이름이 포함된 필수 텍스트 문자열입니다. 값은 함수 이름, 부분 ARN 또는 전체 ARN 일 수 있습니다.

region

Lambda 함수가 있는 AWS 리전을 포함하는 선택적 텍스트 문자열입니다. 리전 이름 및 연결된 값의 목록은 섹션을 참조하십시오리전, 가용 영역 및 로컬 영역

aws_lambda 파라미터

이 표에서는 `aws_lambda` 함수와 관련된 파라미터를 찾아볼 수 있습니다.

파라미터	설명
aws_lambda.connect_timeout_ms	이는 동적 파라미터이며 AWS Lambda에 연결하는 동안 최대 대기 시간을 설정합니다. 기본값은 1000입니다. 이 파라미터에 허용되는 값은 1~900,000입니다.
aws_lambda.request_timeout_ms	이는 동적 파라미터이며 AWS Lambda의 응답을 기다리는 동안 최대 대기 시간을 설정합니다. 기본값은 3000입니다. 이 파라미터에 허용되는 값은 1~900,000입니다.
aws_lambda.endpoint_override	AWS Lambda에 연결하는 데 사용할 수 있는 엔드포인트를 지정합니다. 빈 문자열은 해당 리전의 기본 AWS Lambda 엔드포인트를 선택합니다. 이 정적 파라미터 변경을 적용하려면 데이터베이스를 다시 시작해야 합니다.

Amazon RDS for PostgreSQL의 일반적인 DBA 태스크

데이터베이스 관리자(DBA)는 Amazon RDS for PostgreSQL DB 인스턴스를 관리할 때 다양한 작업을 수행합니다. PostgreSQL을 이미 익숙하게 사용하는 DBA라면 하드웨어에서 PostgreSQL을 실행하는 것과 RDS for PostgreSQL을 실행하는 것 사이의 중요한 차이점을 알고 있어야 합니다. 예를 들어 이는 관리형 서비스이므로, Amazon RDS에서 DB 인스턴스에 대한 shell 액세스를 허용하지 않습니다. 즉, `pg_hba.conf` 및 다른 구성 파일에 직접 액세스할 수 없습니다. RDS for PostgreSQL의 경우 일반적으로 온프레미스 인스턴스의 PostgreSQL 구성 파일을 변경하면 RDS for PostgreSQL DB 인스턴스와 연결된 사용자 지정 DB 파라미터 그룹도 변경됩니다. 자세한 내용은 Amazon RDS의 파라미터 그룹 섹션을 참조하세요.

또한 온프레미스 PostgreSQL 인스턴스와 동일한 방식으로 로그 파일에 액세스할 수 없습니다. 로깅에 대해 자세히 알아보려면 RDS for PostgreSQL 데이터베이스 로그 파일 섹션을 참조하세요.

다른 예로, PostgreSQL `superuser` 계정에 액세스할 권한이 없습니다. RDS for PostgreSQL에서 `rds_superuser` 역할은 가장 높은 권한을 가진 역할이며 설정 시 `postgres`에 부여됩니다. PostgreSQL 온프레미스 사용에 익숙하든, RDS for PostgreSQL을 처음 사용하든 관계없이 `rds_superuser` 역할 및 역할, 사용자, 그룹, 권한으로 작업하는 방식을 이해하는 것이 좋습니다. 자세한 내용은 PostgreSQL 역할 및 권한 이해 섹션을 참조하세요.

다음은 RDS for PostgreSQL의 일반적인 DBA 태스크 일부입니다.

주제

- RDS for PostgreSQL에서 지원되는 데이터 정렬
- PostgreSQL 역할 및 권한 이해
- PostgreSQL에서 연결 중단 처리
- Amazon RDS for PostgreSQL에서 PostgreSQL 자동 정리 사용
- RDS for PostgreSQL에서 지원되는 로깅 메커니즘 작업
- PostgreSQL을 사용한 임시 파일 관리
- pgBadger를 사용한 PostgreSQL의 로그 분석
- PGSnapper를 사용하여 PostgreSQL 모니터링
- RDS for PostgreSQL DB 인스턴스에 파라미터로 작업

RDS for PostgreSQL에서 지원되는 데이터 정렬

데이터 정렬은 데이터베이스에 저장된 문자열을 정렬하고 비교하는 방법을 결정하는 일련의 규칙입니다. 데이터 정렬은 컴퓨터 시스템에서 기본적인 역할을 하며 운영 체제의 일부로 포함됩니다. 새 문자가 언어에 추가되거나 순서 지정 규칙이 변경되면 시간이 지남에 따라 데이터 정렬이 변경됩니다.

데이터 정렬 라이브러리는 데이터 정렬에 대한 특정 규칙 및 알고리즘을 정의합니다. PostgreSQL에서 가장 많이 사용되는 데이터 정렬 라이브러리는 GNU C(glibc) 및 유니코드용 국제화 구성 요소(ICU)입니다. 기본적으로 RDS for PostgreSQL은 멀티바이트 문자 시퀀스에 대한 유니코드 문자 정렬 순서를 포함하는 glibc 데이터 정렬을 사용합니다.

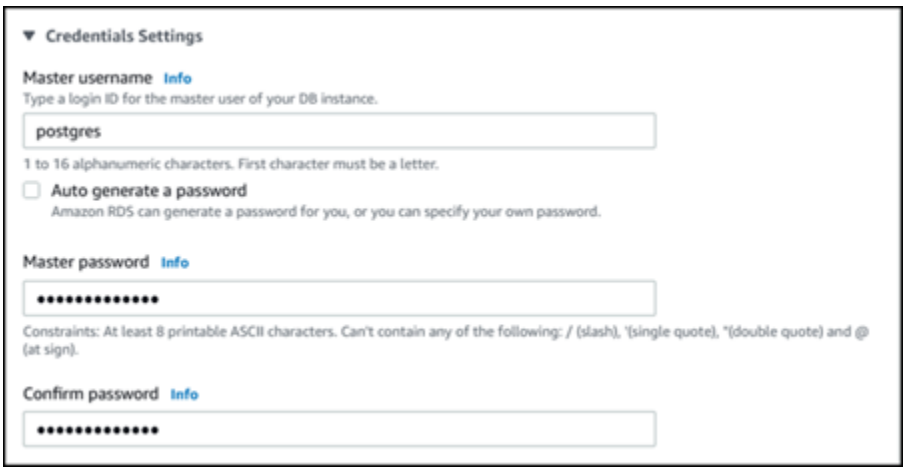
새로운 RDS for PostgreSQL의 DB 인스턴스를 만들면 운영 체제에서 사용 가능한 데이터 정렬을 확인합니다. `CREATE DATABASE` 명령의 PostgreSQL 파라미터인 `LC_COLLATE` 및 `LC_CTYPE`은 해당 데이터베이스의 기본 데이터 정렬로 사용되는 데이터 정렬을 지정하는 데 사용됩니다. 또는 `CREATE DATABASE`의 `LOCALE` 파라미터를 사용하여 이러한 파라미터를 설정할 수도 있습니다. 이는 데이터베이스의 문자열에 대한 기본 데이터 정렬과 문자를 문자, 숫자 또는 기호로 분류하는 규칙을 결정합니다. 열, 색인 또는 쿼리에 사용할 데이터 정렬을 선택할 수도 있습니다.

RDS for PostgreSQL은 데이터 정렬 지원을 위해 운영 체제의 glibc 라이브러리를 사용합니다. RDS for PostgreSQL 인스턴스는 최신 버전의 운영 체제로 정기적으로 업데이트됩니다. 이러한 업데이트에는 glibc 라이브러리의 최신 버전도 포함되는 경우가 있습니다. 드물게 최신 버전의 glibc에서는 일부 문자의 정렬 순서 또는 데이터 정렬이 변경되어 데이터가 다르게 정렬되거나 잘못된 색인 항목이 생성될 수 있습니다. 업데이트 중에 데이터 정렬의 정렬 순서 문제가 발견되면 색인을 다시 작성해야 할 수 있습니다.

glibc 업데이트의 영향을 줄이기 위해 RDS for PostgreSQL에는 이제 독립적인 기본 데이터 정렬 라이브러리가 포함되어 있습니다. 이 데이터 정렬 라이브러리는 RDS for PostgreSQL 14.6, 13.9, 12.13, 11.18, 10.23 및 최신 마이너 버전 릴리스에서 사용할 수 있습니다. glibc 2.26-59.amzn2와 호환되며 정렬 순서 안정성을 제공하여 잘못된 쿼리 결과가 나오지 않도록 방지합니다.

PostgreSQL 역할 및 권한 이해

AWS Management Console을 사용하여 RDS for PostgreSQL DB 인스턴스를 만들면 관리자 계정이 동시에 생성됩니다. 기본적으로 해당 이름은 다음 스크린샷에 나와 있는 것처럼 `postgres`입니다.



기본값(postgres)을 그대로 사용하지 않고 다른 이름을 선택할 수 있습니다. 이 경우 선택한 이름은 문자로 시작해야 하며 영숫자 1~16자 사이어야 합니다. 단순하게 하기 위해 이 안내서 전체에서 기본 사용자 계정을 기본값(postgres)으로 참조합니다.

AWS Management Console 대신 `create-db-instance` AWS CLI를 사용하는 경우 명령에서 `master-username` 파라미터로 이름을 전달하여 이름을 생성합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 단원을 참조하세요.

AWS Management Console, AWS CLI 또는 Amazon RDS API를 사용하든, 기본 `postgres` 이름을 사용하든, 다른 이름을 선택하든 관계없이 첫 번째 데이터베이스 사용자 계정은 `rds_superuser` 그룹의 멤버이며 `rds_superuser` 권한을 가집니다.

주제

- rds_superuser 역할 이해
- PostgreSQL 데이터베이스에 대한 사용자 액세스 제어
- 사용자 암호 관리 위임 및 제어
- SCRAM for PostgreSQL 암호 암호화 사용

rds_superuser 역할 이해

PostgreSQL에서는 역할로 데이터베이스의 다양한 객체에 대해 그룹 또는 사용자에게 부여된 사용자, 그룹 또는 특정 권한 집합을 정의할 수 있습니다. `CREATE USER` 및 `CREATE GROUP`에 대한 PostgreSQL 명령은 데이터베이스 사용자를 구분하기 위한 특정 속성을 가진 보다 일반적인 `CREATE ROLE`로 대체되었습니다. 데이터베이스 사용자는 LOGIN 권한을 가진 역할로 간주할 수 있습니다.

> **Note**
>
> CREATE USER 및 CREATE GROUP 명령을 계속 사용할 수 있습니다. 자세한 내용은 PostgreSQL 설명서에서 데이터베이스 역할을 참조하세요.

postgres 사용자는 RDS for PostgreSQL DB 인스턴스에서 가장 높은 권한을 지닌 데이터베이스 사용자입니다. 이는 다음 CREATE ROLE 문으로 정의되는 특성을 가지고 있습니다.

```
CREATE ROLE postgres WITH LOGIN NOSUPERUSER INHERIT CREATEDB CREATEROLE NOREPLICATION
 VALID UNTIL 'infinity'
```

속성 NOSUPERUSER, NOREPLICATION, INHERIT 및 VALID UNTIL 'infinity'는 별도로 지정하지 않는 한 CREATE ROLE의 기본 옵션입니다.

기본적으로 postgres에는 rds_superuser 역할에 부여된 권한과 역할 및 데이터베이스를 생성할 수 있는 권한이 있습니다. rds_superuser 역할이 있으면 postgres 사용자는 다음과 같은 작업을 할 수 있습니다.

- Amazon RDS에서 사용할 수 있는 확장 기능을 추가합니다. 자세한 내용은 Amazon RDS for PostgreSQL에서 지원되는 PostgreSQL 기능 작업
- 사용자에 대한 역할을 생성하고 사용자에게 권한을 부여합니다. 자세한 내용은 PostgreSQL 설명서에서 CREATE ROLE 및 GRANT를 참조하세요.
- 데이터베이스를 생성합니다. 자세한 내용은 PostgreSQL 설명서에서 CREATE DATABASE를 참조하세요.
- 이러한 권한이 없는 사용자 역할에 rds_superuser 권한을 부여하고 필요에 따라 권한을 회수합니다. 이 역할은 수퍼유저 태스크를 수행하는 사용자에게만 부여하는 것이 좋습니다. 즉, 데이터베이스 관리자(DBA) 또는 시스템 관리자에게 이 역할을 부여할 수 있습니다.
- rds_superuser 역할이 없는 데이터베이스 사용자에게 rds_replication 역할을 부여(회수)합니다.
- rds_superuser 역할이 없는 데이터베이스 사용자에게 rds_password 역할을 부여(회수)합니다.
- pg_stat_activity 보기를 사용하여 모든 데이터베이스 연결에 대한 상태 정보를 가져옵니다. 필요한 경우 rds_superuser는 pg_terminate_backend 또는 pg_cancel_backend를 사용하여 연결을 중지할 수 있습니다.

CREATE ROLE postgres... 문에서 postgres 사용자 역할이 특히 PostgreSQL superuser 권한을 허용하지 않음을 알 수 있습니다. RDS for PostgreSQL은 관리형 서비스이므로 호스트 OS에 액세스할 수 없으며, PostgreSQL superuser 계정을 사용하여 연결할 수 없습니다. 독립 실행형 PostgreSQL에 대한 superuser 액세스 권한이 필요한 대부분의 태스크는 Amazon RDS에서 자동으로 관리됩니다.

권한 부여에 대한 자세한 내용은 PostgreSQL 설명서에서 GRANT를 참조하세요.

rds_superuser 역할은 에서 미리 정의된 여러 역할 중 하나입니다. RDS for PostgreSQL DB 인스턴스

> **Note**
>
> PostgreSQL 13 및 이전 릴리스에서는 미리 정의된 역할을 기본 역할이라고 합니다.

다음 목록에서 새로운 에 대해 자동으로 생성되는 미리 정의된 다른 역할 중 일부를 확인할 수 있습니다. RDS for PostgreSQL DB 인스턴스 미리 정의된 역할 및 권한은 변경할 수 없습니다. 미리 정의된 역할에 대한 권한은 삭제하거나 이름을 바꾸거나 수정할 수 없습니다. 이를 시도할 시에는 오류가 발생합니다.

- rds_password - 데이터베이스 사용자를 대상으로 암호를 변경하고 암호 제약 조건을 설정할 수 있는 역할입니다. rds_superuser 역할에는 기본적으로 이 역할이 부여되며, 이 역할을 통해 데이터베이스 사용자에게 역할을 부여할 수 있습니다. 자세한 내용은 PostgreSQL 데이터베이스에 대한 사용자 액세스 제어 섹션을 참조하세요.
  - 14 이전의 RDS for PostgreSQL 버전에서 rds_password 역할은 암호를 변경하고 데이터베이스 사용자와 rds_superuser 역할이 있는 사용자에 대해 암호 제한을 설정할 수 있습니다. 14 이후의 RDS for PostgreSQL 버전에서 rds_password 역할은 데이터베이스 사용자에 대해서만 암호를 변경하고 암호 제약 조건을 설정할 수 있습니다. rds_superuser 역할이 있는 사용자만 rds_superuser 역할을 가진 다른 사용자에게 이러한 작업을 수행할 수 있습니다.
- rdsadmin - superuser 권한이 있는 관리자가 독립 실행형 PostgreSQL 데이터베이스에서 수행할 수 있는 많은 관리 태스크를 처리하기 위해 생성된 역할입니다. 이 역할은 RDS for PostgreSQL에서 다양한 관리 태스크에 내부적으로 사용됩니다.
- rdstopmgr - 다중 AZ 배포를 지원하기 위해 Amazon RDS에서 내부적으로 사용하는 역할입니다.
- rds_reserved – Amazon RDS에서 데이터베이스 연결을 예약하는 데 내부적으로 사용하는 역할입니다.

역할 및 해당 권한 보기

PostgreSQL 버전에 따라 서로 다른 명령을 사용하여 RDS for PostgreSQL DB 인스턴스에서 사전 정의된 역할과 해당 권한을 볼 수 있습니다.

PostgreSQL 15 이하 버전의 경우

RDS for PostgreSQL DB 인스턴스에 연결하고 psql에서 \du 명령을 사용합니다.

```
postgres=> \du
                                                    List of roles
    Role name    |                         Attributes                          |
                 Member of
-----------------+-------------------------------------------------------------
+-----------------------------------------------------
 postgres        | Create role, Create DB                                     +|
 {rds_superuser}
                 | Password valid until infinity                               |
 rds_ad          | Cannot login                                                | {}
 rds_iam         | Cannot login                                                | {}
 rds_password    | Cannot login                                                | {}
 rds_replication | Cannot login                                                | {}
 rds_superuser   | Cannot login                                                |
 {pg_monitor,pg_signal_backend,rds_password,rds_replication}
 rdsadmin        | Superuser, Create role, Create DB, Replication, Bypass RLS+| {}
                 | Password valid until infinity                               |
```

PostgreSQL 16 이상 버전의 경우

PostgreSQL 16부터 자세한 역할 멤버십 정보를 보려면 \drg+ 명령을 사용합니다.

```
postgres=> \drg+
                            List of role grants
   Role name   |          Member of          |       Options       | Grantor
---------------+-----------------------------+---------------------+----------
 postgres      | rds_superuser               | INHERIT, SET        | rdsadmin
 rds_superuser | pg_checkpoint               | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | pg_monitor                  | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | pg_signal_backend           | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | pg_use_reserved_connections | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | rds_password                | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | rds_replication             | ADMIN, INHERIT, SET | rdsadmin
```

PostgreSQL 버전에 관계없이 역할 멤버십을 확인하려면 다음 SQL 쿼리를 사용할 수 있습니다.

```
SELECT m.rolname AS "Role name", r.rolname AS "Member of"
FROM pg_catalog.pg_roles m
JOIN pg_catalog.pg_auth_members pam ON (pam.member = m.oid)
LEFT JOIN pg_catalog.pg_roles r ON (pam.roleid = r.oid)
LEFT JOIN pg_catalog.pg_roles g ON (pam.grantor = g.oid)
WHERE m.rolname !~ '^pg_'
ORDER BY 1, 2;
```

출력에서 rds_superuser는 데이터베이스 사용자 역할은 아니지만(로그인할 수 없음), 다른 많은 역할의 권한을 가집니다. 데이터베이스 사용자 postgres가 rds_superuser 역할의 멤버임을 알 수도 있습니다. 앞서 언급한 바와 같이 postgres는 Amazon RDS 콘솔 데이터베이스 생성(Create database) 페이지의 기본값입니다. 다른 이름을 선택한 경우 해당 이름이 대신 역할 목록에 표시됩니다.

PostgreSQL 데이터베이스에 대한 사용자 액세스 제어

PostgreSQL의 새 데이터베이스는 항상 모든 데이터베이스 사용자와 역할이 객체를 만들 수 있도록 허용하는 데이터베이스 public 스키마의 기본 권한 집합으로 생성됩니다. 이러한 권한을 통해 데이터베이스 사용자는 데이터베이스에 연결하고, 가령 연결된 동안 임시 테이블을 만들 수 있습니다.

RDS for PostgreSQL DB 인스턴스 에 생성한 데이터베이스 인스턴스에 액세스할 수 있는 사용자 권한을 보다 효과적으로 제어하려면 이러한 기본 public 권한을 회수하는 것이 좋습니다. 이렇게 하고 나서 다음 절차에 표시된 것처럼 데이터베이스 사용자에게 보다 세분화된 구체적인 권한을 부여하면 됩니다.

새 데이터베이스 인스턴스에 대한 역할 및 권한을 설정하는 방법

새로 생성한 RDS for PostgreSQL DB 인스턴스에 데이터베이스를 설정하여 여러 연구자가 사용할 수 있도록 하고, 이들 모두 데이터베이스에 대한 읽기-쓰기 액세스 권한이 필요하다고 가정합니다.

1. 다음과 같이 psql 또는 pgAdmin을 사용하여 RDS for PostgreSQL DB 인스턴스에 연결합니다.

   ```
   psql --host=your-db-instance.666666666666.aws-region.rds.amazonaws.com --port=5432
    --username=postgres --password
   ```

   메시지가 표시되면 암호를 입력합니다. psql 클라이언트는 기본 관리 연결 데이터베이스인 postgres=>를 연결하고 프롬프트로 표시합니다.

2. 데이터베이스 사용자가 public 스키마에서 객체를 생성하지 못하도록 하려면 다음을 수행합니다.

```
postgres=> REVOKE CREATE ON SCHEMA public FROM PUBLIC;
REVOKE
```

3. 다음으로 새 데이터베이스 인스턴스를 생성합니다.

```
postgres=> CREATE DATABASE lab_db;
CREATE DATABASE
```

4. 새 데이터베이스의 PUBLIC 스키마에서 모든 권한을 회수합니다.

```
postgres=> REVOKE ALL ON DATABASE lab_db FROM public;
REVOKE
```

5. 데이터베이스 사용자를 위한 역할을 생성합니다.

```
postgres=> CREATE ROLE lab_tech;
CREATE ROLE
```

6. 이 역할이 있는 데이터베이스 사용자에게 데이터베이스 연결 기능을 제공합니다.

```
postgres=> GRANT CONNECT ON DATABASE lab_db TO lab_tech;
GRANT
```

7. lab_tech 역할이 있는 모든 사용자에게 이 데이터베이스에 대한 모든 권한을 부여합니다.

```
postgres=> GRANT ALL PRIVILEGES ON DATABASE lab_db TO lab_tech;
GRANT
```

8. 다음과 같이 데이터베이스 사용자를 생성합니다.

```
postgres=> CREATE ROLE lab_user1 LOGIN PASSWORD 'change_me';
CREATE ROLE
postgres=> CREATE ROLE lab_user2 LOGIN PASSWORD 'change_me';
CREATE ROLE
```

9. 다음과 같이 두 사용자에게 lab_tech 역할과 관련된 권한을 부여합니다.

```
postgres=> GRANT lab_tech TO lab_user1;
GRANT ROLE
```

```
postgres=> GRANT lab_tech TO lab_user2;
GRANT ROLE
```

이 시점에서 lab_user1과 lab_user2는 lab_db 데이터베이스에 연결할 수 있습니다. 이 예제에서는 여러 데이터베이스 인스턴스와 다양한 스키마 생성, 제한된 권한 부여를 포함할 수 있는 엔터프라이즈 사용에 대한 모범 사례를 따르지 않습니다. 전체 정보 및 추가 시나리오를 알아보려면 PostgreSQL 사용자 및 역할 관리를 참조하세요.

PostgreSQL 데이터베이스의 권한에 대한 자세한 내용은 PostgreSQL 설명서에서 GRANT 명령을 참조하세요.

사용자 암호 관리 위임 및 제어

DBA로서 사용자 암호 관리를 위임할 수 있습니다. 또는 데이터베이스 사용자가 암호를 변경하거나 암호 사용 주기와 같은 암호 제약 조건을 재구성하지 못하도록 할 수 있습니다. 선택한 데이터베이스 사용자만 암호 설정을 변경할 수 있도록 하려면 제한된 암호 관리 기능을 켜면 됩니다. 이 기능을 활성화하면 rds_password 역할이 부여된 데이터베이스 사용자만 암호를 관리할 수 있습니다.

> **Note**
>
> 제한된 암호 관리를 사용하려면 RDS for PostgreSQL DB 인스턴스가 PostgreSQL 10.6 이상을 실행해야 합니다.

기본적으로 이 기능은 다음과 같이 off입니다.

```
postgres=> SHOW rds.restrict_password_commands;
  rds.restrict_password_commands
--------------------------------
 off
(1 row)
```

이 기능을 켜려면 사용자 정의 파라미터 그룹을 사용하고 rds.restrict_password_commands에 대한 설정을 1로 변경합니다. RDS for PostgreSQL DB 인스턴스를 재부팅해야 설정이 적용됩니다.

이 기능을 활성화하면 다음 SQL 명령에 대한 rds_password 권한이 필요합니다.

```
CREATE ROLE myrole WITH PASSWORD 'mypassword';
CREATE ROLE myrole WITH PASSWORD 'mypassword' VALID UNTIL '2023-01-01';
```

```
ALTER ROLE myrole WITH PASSWORD 'mypassword' VALID UNTIL '2023-01-01';
ALTER ROLE myrole WITH PASSWORD 'mypassword';
ALTER ROLE myrole VALID UNTIL '2023-01-01';
ALTER ROLE myrole RENAME TO myrole2;
```

암호에서 MD5 해시 알고리즘을 사용하는 경우에도 역할 이름 변경(`ALTER ROLE myrole RENAME TO newname`)이 제한됩니다.

이 기능이 활성화된 상태에서 `rds_password` 역할 권한 없이 이러한 SQL 명령을 시도하면 다음 오류가 발생합니다.

```
ERROR: must be a member of rds_password to alter passwords
```

암호 관리에만 사용하는 몇 가지 역할에만 `rds_password` 권한을 부여하는 것이 좋습니다. `rds_superuser` 권한이 없는 데이터베이스 사용자에게 `rds_password` 권한을 부여할 경우 `CREATEROLE` 속성도 부여해야 합니다.

만료 및 클라이언트 측에 필요한 복잡성 등의 암호 요구 사항을 확인해야 합니다. 암호 관련 변경 사항에 대해 자체 클라이언트 측 유틸리티를 사용하는 경우 유틸리티가 `rds_password`의 멤버여야 하며 `CREATE ROLE` 권한을 가져야 합니다.

SCRAM for PostgreSQL 암호 암호화 사용

Salted Challenge Response Authentication Mechanism(SCRAM)은 암호를 암호화하는 데 사용하는 PostgreSQL의 기본 메시지 다이제스트(MD5) 알고리즘을 대체합니다. SCRAM 인증 메커니즘은 MD5보다 더 안전한 것으로 간주됩니다. 이러한 2가지 암호 보호 방법에 대한 자세한 내용은 PostgreSQL 설명서의 암호 인증을 참조하세요.

MD5가 아닌 SCRAM을 의 암호 암호화 체계로 사용하는 것이 좋습니다. RDS for PostgreSQL DB 인스턴스 이는 암호 인증 및 암호화에 scram-sha-256 알고리즘을 사용하는 암호화 챌린지-응답 메커니즘입니다.

SCRAM을 지원하려면 클라이언트 애플리케이션의 라이브러리를 업데이트해야 할 수 있습니다. 예를 들어, 42.2.0 이전의 JDBC 버전은 SCRAM을 지원하지 않습니다. 자세한 내용은 PostgreSQL JDBC 드라이버 설명서의 PostgreSQL JDBC 드라이버를 참조하세요. 기타 PostgreSQL 드라이버 및 SCRAM 지원 목록은 PostgreSQL 설명서의 드라이버 목록을 참조하세요.

RDS for PostgreSQL 버전 13.1 이상은 scram-sha-256을 지원합니다. 또한 다음 절차에 설명된 대로 이러한 버전을 사용하여 DB 인스턴스를 SCRAM이 필요하도록 구성할 수 있습니다.

SCRAM이 필요하도록 RDS for PostgreSQL DB 인스턴스 설정

RDS for PostgreSQL DB 인스턴스가 scram-sha-256 알고리즘을 사용하는 암호만 수락하도록 요구할 수 있습니다.

> **Important**
>
> PostgreSQL 데이터베이스를 사용하는 기존 RDS 프록시의 경우, SCRAM만 사용하도록 데이터베이스 인증을 수정하면 최대 60초 동안 프록시를 사용할 수 없게 됩니다. 문제를 방지하려면 다음 중 하나를 수행합니다.
>
> - 데이터베이스에서 SCRAM 및 MD5 인증이 모두 허용되는지 확인합니다.
> - SCRAM 인증만 사용하려면 새 프록시를 생성하고, 애플리케이션 트래픽을 새 프록시로 마이그레이션한 다음 이전에 데이터베이스와 연결된 프록시를 삭제합니다.

시스템을 변경하기 전에 다음과 같이 전체 프로세스를 이해해야 합니다.

- 모든 데이터베이스 사용자에 대한 전체 역할 및 암호 암호화에 대한 정보를 가져옵니다.
- 암호 암호화를 제어하는 파라미터에 대해 RDS for PostgreSQL DB 인스턴스의 파라미터 설정을 다시 확인합니다.
- RDS for PostgreSQL DB 인스턴스에서 기본 파라미터 그룹을 사용한다면 필요한 경우 파라미터를 수정할 수 있도록 사용자 지정 DB 파라미터 그룹을 생성하고, 이를 RDS for PostgreSQL DB 인스턴스에 적용해야 합니다. RDS for PostgreSQL DB 인스턴스에서 사용자 지정 파라미터 그룹을 사용하는 경우 필요에 따라 과정 후반부에서 필요한 파라미터를 수정할 수 있습니다.
- password_encryption 파라미터를 scram-sha-256으로 변경합니다.
- 모든 데이터베이스 사용자에게 암호를 업데이트해야 함을 알립니다. postgres 계정에도 동일한 작업을 수행합니다. 새 암호는 scram-sha-256 알고리즘을 통해 암호화되어 저장됩니다.
- 모든 암호가 암호화 유형으로 암호화되었는지 확인합니다.
- 모든 암호가 scram-sha-256을 사용하는 경우 rds.accepted_password_auth_method 파라미터를 md5+scram에서 scram-sha-256으로 변경할 수 있습니다.

> **Warning**
>
> rds.accepted_password_auth_method를 scram-sha-256으로만 변경하면 md5로 암호화된 암호가 있는 사용자(역할)는 연결할 수 없습니다.

RDS for PostgreSQL DB 인스턴스에서 SCRAM을 사용하도록 준비

RDS for PostgreSQL DB 인스턴스를 변경하기에 모든 기존 데이터베이스 사용자 계정을 확인해야 합니다. 또한 암호에 사용되는 암호화 유형도 확인하세요. `rds_tools` 확장을 사용하여 이러한 작업을 수행할 수 있습니다. `rds_tools`를 지원하는 PostgreSQL 버전을 확인하려면 Amazon RDS for PostgreSQL의 확장 버전을 참조하세요.

데이터베이스 사용자(역할) 및 암호 암호화 방법 목록을 가져오려면

1. 다음과 같이 `psql`을 사용하여 RDS for PostgreSQL DB 인스턴스에 연결합니다.

```
psql --host=db-name.111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password
```

2. `rds_tools` 확장을 설치합니다.

```
postgres=> CREATE EXTENSION rds_tools;
CREATE EXTENSION
```

3. 역할 및 암호화 목록을 가져옵니다.

```
postgres=> SELECT * FROM
      rds_tools.role_password_encryption_type();
```

출력은 다음과 비슷합니다.

```
       rolname        | encryption_type
----------------------+-----------------
 pg_monitor           |
 pg_read_all_settings |
 pg_read_all_stats    |
 pg_stat_scan_tables  |
 pg_signal_backend    |
 lab_tester           | md5
 user_465             | md5
 postgres             | md5
(8 rows)
```

사용자 지정 DB 파라미터 그룹 생성

> **Note**
>
> RDS for PostgreSQL DB 인스턴스에서 이미 사용자 지정 파라미터 그룹을 사용하는 경우 새로 만들 필요가 없습니다.

Amazon RDS 파라미터 그룹에 대한 개요는 RDS for PostgreSQL DB 인스턴스에 파라미터로 작업 섹션을 참조하세요.

암호에 사용되는 암호 암호화 유형은 하나의 파라미터 `password_encryption`으로 설정됩니다. RDS for PostgreSQL DB 인스턴스에서 허용하는 암호화는 다른 파라미터인 `rds.accepted_password_auth_method`로 설정됩니다. 이러한 값 중 하나를 기본값에서 변경하려면 사용자 지정 DB 파라미터 그룹을 생성하여 인스턴스에 적용해야 합니다.

AWS Management Console 또는 RDS API를 사용하여 사용자 지정 DB 파라미터 그룹을 생성할 수도 있습니다. 자세한 내용은 단원을 참조하세요.

DB 파라미터 그룹을 DB 인스턴스에 연결합니다.

사용자 지정 DB 파라미터 그룹을 생성하려면

1. `create-db-parameter-group` CLI 명령을 사용하여 사용자 지정 DB 파라미터 그룹을 생성합니다. 이 예에서는 `postgres13`을 사용자 지정 파라미터 그룹의 소스로 사용합니다.

   대상 LinuxmacOS, 또는Unix:

   ```
   aws rds create-db-parameter-group --db-parameter-group-name 'docs-lab-scram-passwords' \
     --db-parameter-group-family postgres13  --description 'Custom parameter group for SCRAM'
   ```

   Windows의 경우:

   ```
   aws rds create-db-parameter-group --db-parameter-group-name "docs-lab-scram-passwords" ^
     --db-parameter-group-family postgres13  --description "Custom DB parameter group for SCRAM"
   ```

2. modify-db-instance CLI 명령을 사용하여 다음과 같이 사용자 지정 파라미터 그룹을 RDS for PostgreSQL DB 클러스터에 적용합니다.

   대상 LinuxmacOS, 또는Unix:

   ```
   aws rds modify-db-instance --db-instance-identifier 'your-instance-name' \
           --db-parameter-group-name "docs-lab-scram-passwords
   ```

   Windows의 경우:

   ```
   aws rds modify-db-instance --db-instance-identifier "your-instance-name" ^
           --db-parameter-group-name "docs-lab-scram-passwords
   ```

   사용자 지정 DB 파라미터 그룹이 포함된 RDS for PostgreSQL DB 인스턴스를 다시 동기화하려면 클러스터의 프라이머리 및 기타 모든 인스턴스를 재부팅합니다. 사용자에게 미치는 영향을 최소화하기 위해 정기 유지 관리 기간 동안 진행되도록 예약합니다.

SCRAM을 사용하도록 암호 암호화 구성

RDS for PostgreSQL DB 인스턴스에서 사용하는 암호 암호화 메커니즘은 DB 파라미터 그룹에 password_encryption 파라미터로 설정되어 있습니다. 허용되는 값은 설정하지 않거나 md5 또는 scram-sha-256입니다. 기본값은 다음과 같이 RDS for PostgreSQL 버전에 따라 달라집니다.

- RDS for PostgreSQL 14 이상 – 기본값은 scram-sha-256입니다.
- RDS for PostgreSQL 13 – 기본값은 md5입니다.

사용자 지정 DB 파라미터 그룹이 RDS for PostgreSQL DB 인스턴스에 연결되어 있으면 암호 암호화 파라미터 값을 변경할 수 있습니다.

Name	Values	Allowed values	Modifiable	Source	Apply type
password_encryption	md5	md5, scram-sha-256	true	system	dynamic
rds.accepted_password_auth_method	md5+scram	md5+scram, scram	true	system	dynamic

암호의 암호화 설정을 scram-sha-256으로 변경하려면

- 다음과 같이 암호 암호화 값을 scram-sha-256으로 변경합니다. 파라미터가 동적이기 때문에 변경 사항을 즉시 적용할 수 있으므로 변경 사항을 적용하려고 재시작하지 않아도 됩니다.

  대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group --db-parameter-group-name \
  'docs-lab-scram-passwords' --parameters
 'ParameterName=password_encryption,ParameterValue=scram-
sha-256,ApplyMethod=immediate'
```

  Windows의 경우:

```
aws rds modify-db-parameter-group --db-parameter-group-name ^
  "docs-lab-scram-passwords" --parameters
 "ParameterName=password_encryption,ParameterValue=scram-
sha-256,ApplyMethod=immediate"
```

사용자 역할의 암호를 SCRAM으로 마이그레이션

다음에 설명된 대로 사용자 역할에 대한 암호를 SCRAM으로 마이그레이션할 수 있습니다.

MD5에서 SCRAM으로 데이터베이스 사용자(역할) 암호를 마이그레이션하려면

1. 다음과 같이 관리자 사용자로 로그인(기본 사용자 이름 `postgres`)합니다.

```
psql --host=db-name.111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password
```

2. 다음 명령을 사용하여 RDS for PostgreSQL DB 인스턴스의 `password_encryption` 파라미터 설정을 확인합니다.

```
postgres=> SHOW password_encryption;
 password_encryption
---------------------
 md5
 (1 row)
```

3. 이 파라미터의 값을 scram-sha-256으로 변경합니다. 이 파라미터는 동적 파라미터이므로 변경 후에 인스턴스를 재부팅할 필요가 없습니다. 다음과 같이 scram-sha-256으로 설정되어 있는지 값을 다시 확인합니다.

```
postgres=> SHOW password_encryption;
 password_encryption
---------------------
 scram-sha-256
 (1 row)
```

4. 모든 데이터베이스 사용자에게 암호 변경을 알립니다. 계정 postgres(rds_superuser 권한을 지닌 데이터베이스 사용자)의 자체 암호도 변경해야 합니다.

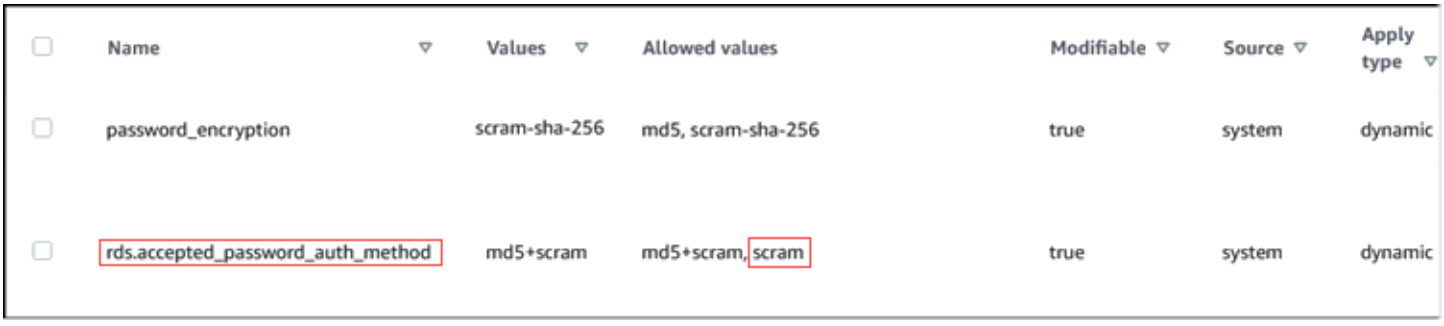
```
labdb=> ALTER ROLE postgres WITH LOGIN PASSWORD 'change_me';
ALTER ROLE
```

5. 에 모든 데이터베이스에 대해 프로세스를 반복합니다. RDS for PostgreSQL DB 인스턴스

SCRAM이 필요하도록 파라미터 변경

과정의 마지막 단계입니다. 다음 절차에서 변경한 후에도 암호에 md5 암호화를 계속 사용하는 모든 사용자 계정(역할)은 에 로그인할 수 없습니다. RDS for PostgreSQL DB 인스턴스

rds.accepted_password_auth_method는 로그인하는 동안 RDS for PostgreSQL DB 인스턴스가 사용자 암호에 대해 허용하는 암호화 방법을 지정합니다. 기본값은 md5+scram이므로, 두 방법 중 하나를 사용할 수 있습니다. 다음 이미지에서는 이 파라미터에 대한 기본 설정을 찾을 수 있습니다.



Name	Values	Allowed values	Modifiable	Source	Apply type
password_encryption	scram-sha-256	md5, scram-sha-256	true	system	dynamic
rds.accepted_password_auth_method	md5+scram	md5+scram, scram	true	system	dynamic

이 파라미터에 허용되는 값은 md5+scram 또는 scram입니다. 이 파라미터 값을 scram으로 변경하여 필수로 설정합니다.

암호에 SCRAM 인증을 요구하도록 파라미터 값을 변경하려면

1. RDS for PostgreSQL DB 인스턴스의 모든 데이터베이스에 대한 전체 데이터베이스 사용자 암호가 암호 암호화에 `scram-sha-256`을 사용하는지 확인합니다. 이렇게 하려면 다음과 같이 역할(사용자) 및 암호화 유형에 대해 `rds_tools`를 쿼리합니다.

```
postgres=> SELECT * FROM rds_tools.role_password_encryption_type();
  rolname        | encryption_type
  ---------------------+-----------------
  pg_monitor           |
  pg_read_all_settings |
  pg_read_all_stats    |
  pg_stat_scan_tables  |
  pg_signal_backend    |
  lab_tester           | scram-sha-256
  user_465             | scram-sha-256
  postgres             | scram-sha-256
  ( rows)
```

2. 의 모든 DB 인스턴스에서 쿼리를 반복합니다. RDS for PostgreSQL DB 인스턴스

   모든 암호가 scram-sha-256을 사용하는 경우 그대로 진행하면 됩니다.

3. 다음과 같이 허용된 암호 인증의 값을 scram-sha-256으로 변경합니다.

   대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group --db-parameter-group-name 'docs-lab-scram-
passwords' \
  --parameters
 'ParameterName=rds.accepted_password_auth_method,ParameterValue=scram,ApplyMethod=immediat
```

   Windows의 경우:

```
aws rds modify-db-parameter-group --db-parameter-group-name "docs-lab-scram-
passwords" ^
  --parameters
 "ParameterName=rds.accepted_password_auth_method,ParameterValue=scram,ApplyMethod=immediat
```

PostgreSQL에서 연결 중단 처리

클라이언트 애플리케이션이 중단되거나 비정상적으로 종료된 경우에도 서버에서 데이터베이스 세션이 활성 상태로 유지될 때 연결 중단이 발생합니다. 이 상황은 일반적으로 클라이언트가 데이터베이스 연결을 제대로 닫거나 진행 중인 요청을 취소하지 않고 충돌을 처리하거나 예기치 않게 종료할 때 발생합니다.

PostgreSQL은 서버 프로세스가 유휴 상태이거나 클라이언트로 데이터를 전송하려고 할 때 연결 중단을 효율적으로 식별하고 정리합니다. 그러나 유휴 상태이거나 클라이언트 입력을 기다리거나 쿼리를 적극적으로 실행하는 세션의 경우 감지가 어렵습니다. 이러한 시나리오를 처리하기 위해 PostgreSQL은 `tcp_keepalives_*`, `tcp_user_timeout` 및 `client_connection_check_interval` 파라미터를 제공합니다.

주제

- TCP 킵얼라이브 이해
- RDS for PostgreSQL의 주요 TCP 킵얼라이브 파라미터
- TCP 킵얼라이브 설정 사용 사례
- 모범 사례

TCP 킵얼라이브 이해

TCP Keepalive는 연결 무결성을 유지하고 확인하는 데 도움이 되는 프로토콜 수준 메커니즘입니다. 각 TCP 연결은 킵얼라이브 동작을 제어하는 커널 수준 설정을 유지 관리합니다. 킵얼라이브 타이머가 만료되면 시스템은 다음을 수행합니다.

- 데이터 및 ACK 플래그 세트가 없는 프로브 패킷을 전송합니다.
- TCP/IP 사양에 따라 원격 엔드포인트에서 응답을 예상합니다.
- 응답 또는 응답 없음을 기반으로 연결 상태를 관리합니다.

RDS for PostgreSQL의 주요 TCP 킵얼라이브 파라미터

파라미터	설명	기본값
`tcp_keepalives_idle`	Specifies number of seconds of inactivity before sending keepalive message.	300

파라미터	설명	기본값
tcp_keepalives_interval	Specifies number of seconds between retransmissions of unacknowledged keepalive messages.	30
tcp_keepalives_count	Maximum lost keepalive messages before declaring connection dead	2
tcp_user_timeout	Specifies how long (in Milliseconds) unacknowledged data can remain before forcibly closing the connection.	0
client_connection_check_interval	Sets the interval (in Milliseconds) for checking client connection status during long-running queries. This ensures quicker detection of closed connections.	0

TCP 킵얼라이브 설정 사용 사례

유휴 세션 연결 유지

활동이 없어 방화벽 또는 라우터에 의해 유휴 연결이 종료되지 않도록 하는 방법:

- 정기적으로 킵얼라이브 패킷을 전송하도록 tcp_keepalives_idle을 구성합니다.

연결 중단 감지

연결 중단을 즉시 감지하는 방법:

- tcp_keepalives_idle, tcp_keepalives_interval 및 tcp_keepalives_count를 조정합니다. 예를 들어 Aurora PostgreSQL 기본값을 사용하면 연결 중단을 감지하는 데 약 1분(프로브 2개 × 30초)이 걸립니다. 이러한 값을 낮추면 감지 속도가 빨라질 수 있습니다.

- `tcp_user_timeout`을 사용하여 확인의 최대 대기 시간을 지정합니다.

TCP 킵얼라이브 설정은 커널이 연결 중단을 감지하는 데 도움이 되지만 소켓을 사용할 때까지 PostgreSQL이 작동하지 않을 수 있습니다. 세션이 긴 쿼리를 실행하는 경우 쿼리가 완료된 후에야 연결 중단이 감지될 수 있습니다. PostgreSQL 14 이상 버전에서는 `client_connection_check_interval`이 쿼리 실행 중에 소켓을 주기적으로 폴링하여 연결 중단 감지를 가속화할 수 있습니다.

모범 사례

- 적절한 킵얼라이브 간격 설정: `tcp_user_timeout`, `tcp_keepalives_idle`, `tcp_keepalives_count` 및 `tcp_keepalives_interval`을 조정하여 감지 속도와 리소스 사용의 균형을 맞춥니다.
- 환경에 최적화: 네트워크 동작, 방화벽 정책 및 세션 요구 사항에 맞게 설정을 조정합니다.
- PostgreSQL 기능 활용: 효율적인 연결 확인을 위해 PostgreSQL 14 이상 버전에서 `client_connection_check_interval`을 사용합니다.

Amazon RDS for PostgreSQL에서 PostgreSQL 자동 정리 사용

자동 정리 기능을 사용하여 PostgreSQL DB 인스턴스의 상태를 유지 관리하는 것이 좋습니다. Autovacuum은 VACUUM 및 ANALYZE 명령의 시작을 자동화합니다. Autovacuum은 삽입되고 업데이트되거나 삭제된 튜플 수가 많은 테이블이 있는지 확인합니다. 확인이 끝나면 Autovacuum은 PostgreSQL 데이터베이스에서 폐기된 데이터 또는 튜플을 제거하여 스토리지를 회수합니다.

기본적으로 자동 정리는 기본 PostgreSQL DB 파라미터 그룹을 사용하여 만든 RDS for PostgreSQL DB 인스턴스에서 켜져 있습니다. 자동 정리 기능과 관련된 다른 구성 파라미터도 기본적으로 설정됩니다. 이러한 기본값은 다소 일반적이기 때문에 특정 워크로드에 대해 자동 정리 기능과 관련된 일부 파라미터를 조정하면 도움이 될 수 있습니다.

다음에서 자동 정리에 대한 자세한 정보와 RDS for PostgreSQL DB 인스턴스에 대한 파라미터를 조정하는 방법을 확인할 수 있습니다. 더욱 개괄적인 수준의 정보는 PostgreSQL로 작업하기 위한 모범 사례 섹션을 참조하세요.

주제

- Autovacuum에 메모리 할당
- 트랜잭션 ID 랩어라운드의 가능성 감소

- 데이터베이스의 테이블을 vacuum해야 하는지 여부를 결정
- 현재 Autovacuum을 수행할 수 있는 테이블 결정
- 현재 Autovacuum이 실행 중인지 여부 및 실행 기간 확인
- 수동 vacuum freeze 수행
- Autovacuum이 실행 중인 경우 테이블 인덱스 다시 지정
- 대용량 인덱스를 사용하여 autovacuum 관리
- Autovacuum에 영향을 주는 기타 파라미터
- 테이블 수준 Autovacuum 파라미터 설정
- autovacuum 및 vacuum 활동 로그
- 잘못된 데이터베이스를 사용한 autovacuum 동작 이해
- RDS for PostgreSQL에서 공격적인 vacuum 블로커 식별 및 해결

Autovacuum에 메모리 할당

autovacuum 성능에 영향을 미치는 가장 중요한 파라미터 중 하나는 `autovacuum_work_mem` 파라미터입니다. RDS for PostgreSQL 버전 14 이하에서는 `autovacuum_work_mem` 파라미터가 -1로 설정되어 `maintenance_work_mem`의 설정이 대신 사용됨을 나타냅니다. 다른 모든 버전의 경우 `autovacuum_work_mem`는 GREATEST({DBInstanceClassMemory/32768}, 65536)에 의해 결정됩니다.

수동 vacuum 작업은 항상 기본 `maintenance_work_mem` 설정인 GREATEST({DBInstanceClassMemory/63963136*1024}, 65536)을 사용하며, 보다 대상화된 수동 `VACUUM` 작업을 위해 `SET` 명령을 사용하여 세션 수준에서 조정할 수도 있습니다.

`autovacuum_work_mem`는 인덱스 vacuum용 데드 튜플(`pg_stat_all_tables.n_dead_tup`)의 ID를 저장하기 위한 autovacuum 메모리를 결정합니다.

`autovacuum_work_mem` 파라미터 값을 결정하기 위해 계산할 때는 다음 사항에 유의하세요.

- 파라미터를 너무 낮게 설정하면 vacuum 프로세스가 테이블을 여러 번 스캔해야 작업이 완료될 수 있습니다. 이러한 다중 스캔은 성능에 부정적인 영향을 줄 수 있습니다. 더 큰 인스턴스의 경우 `maintenance_work_mem` 또는 `autovacuum_work_mem`을 최소 1GB로 설정하면 데드 튜플 수가 많은 테이블을 vacuum하는 성능이 향상될 수 있습니다. 그러나 PostgreSQL 버전 16 이하에서는 vacuum 메모리 사용량이 1GB로 제한되므로 한 번의 전달로 약 1억 7,900만 개의 데드 튜플을 처리하기에 충분합니다. 테이블에 이보다 더 많은 데드 튜플이 있는 경우 vacuum은 테이블의 인덱스를

여러 번 전달해야 하므로 필요한 시간이 크게 늘어납니다. PostgreSQL 버전 17부터는 1GB의 제한이 없으며, autovacuum은 radix 트리를 사용하여 1억 7,900만 개 이상의 튜플을 처리할 수 있습니다.

튜플 식별자의 크기는 6바이트입니다. 테이블의 인덱스를 vacuum하는 데 필요한 메모리를 추정하려면 pg_stat_all_tables.n_dead_tup를 쿼리하여 데드 튜플 수를 찾은 다음 이 수에 6을 곱하여 인덱스를 단일 전달로 vacuum하는 데 필요한 메모리를 결정합니다. 다음 쿼리를 사용할 수 있습니다.

```
SELECT
    relname AS table_name,
    n_dead_tup,
    pg_size_pretty(n_dead_tup * 6) AS estimated_memory
FROM
    pg_stat_all_tables
WHERE
    relname = 'name_of_the_table';
```

- autovacuum_work_mem 파라미터는 autovacuum_max_workers 파라미터와 함께 작동합니다. autovacuum_max_workers 중 각각의 작업자는 할당된 메모리를 사용할 수 있습니다. 작은 테이블이 많이 있는 경우에는 autovacuum_max_workers를 더 많이 할당하고 autovacuum_work_mem을 더 적게 할당합니다. 큰 테이블이 많이 있는 경우(100GB 이상)에는 메모리를 더 많이 할당하고 작업자 프로세스를 더 적게 할당합니다. 가장 큰 테이블에서 성공적으로 작업을 수행하려면 충분한 메모리를 할당해 두어야 합니다. 따라서 작업자 프로세스와 메모리를 합한 양이 할당하려는 전체 메모리 양과 같아야 합니다.

트랜잭션 ID 랩어라운드의 가능성 감소

경우에 따라 Autovacuum과 관련된 파라미터 그룹 설정이 트랜잭션 ID 랩어라운드를 방지하기에 충분히 공격적이지 않을 수 있습니다. 이를 해결하기 위해 RDS for PostgreSQL은 자동 정리 파라미터 값을 자동으로 조정하는 메커니즘을 제공합니다. 적응형 자동 정리는 RDS for PostgreSQL의 기능입니다. TransactionID wraparound에 대한 자세한 설명은 PostgreSQL 설명서에 나와 있습니다.

동적 파라미터 rds.adaptive_autovacuum이 켜짐으로 설정된 RDS for PostgreSQL 인스턴스의 경우 적응형 자동 정리가 기본적으로 켜져 있습니다. 이 설정을 항상 활성화해 놓는 것이 좋습니다. 그러나 적응형 Autovacuum 파라미터 튜닝을 끄려면 rds.adaptive_autovacuum 파라미터를 0 또는 OFF로 설정합니다.

Amazon RDS Amazon RDS가 자동 정리 파라미터를 조정하더라도 트랜잭션 ID 랩어라운드는 계속 가능합니다. 트랜잭션 ID 랩어라운드에 대한 Amazon CloudWatch 경보를 구현하는 것이 좋습니다. 자세

한 내용은 AWS 데이터베이스 블로그의 RDS for PostgreSQL에서 트랜잭션 ID 랩어라운드에 대한 조기 경고 시스템 구축 게시물을 참조하세요.

적응형 자동 정리 파라미터 튜닝을 사용 설정한 경우 Amazon RDS는 CloudWatch 지표 `MaximumUsedTransactionIDs`가 `autovacuum_freeze_max_age` 파라미터 값 또는 500,000,000 중 큰 값에 도달하면 자동 정리 파라미터를 조정하기 시작합니다.

테이블이 계속 트랜잭션 ID 랩어라운드 방향으로 향하면 Amazon RDS는 Autovacuum의 파라미터를 계속 조정합니다. 이러한 각각의 조정은 랩어라운드를 피하기 위해 Autovacuum에 더 많은 리소스를 할애합니다. Amazon RDS는 다음 자동 정리 관련 파라미터를 업데이트합니다.

- autovacuum_vacuum_cost_delay
- autovacuum_vacuum_cost_limit
- `autovacuum_work_mem`
- autovacuum_naptime

RDS는 새 값이 Autovacuum을 보다 공격적으로 만드는 경우에만 이러한 파라미터를 수정합니다. 파라미터는 DB 인스턴스의 메모리에서 수정됩니다. 파라미터 그룹의 값은 변경되지 않습니다. 현재 인 메모리 설정을 보려면 PostgreSQL SHOW SQL 명령을 사용하십시오.

Amazon RDS가 이러한 Autovacuum 파라미터를 수정하면 영향받은 DB 인스턴스에 대한 이벤트를 생성합니다. 이 이벤트는 AWS Management Console 및 Amazon RDS API에서 볼 수 있습니다. `MaximumUsedTransactionIDs` CloudWatch 지표가 임계값 미만의 값을 반환하면 Amazon RDS는 메모리의 자동 정리 관련 파라미터를 파라미터 그룹에 지정된 값으로 다시 설정합니다. 그런 다음 이 변경에 해당하는 다른 이벤트를 생성합니다.

데이터베이스의 테이블을 vacuum해야 하는지 여부를 결정

다음 쿼리를 사용하여 데이터베이스의 고정되지 않은 트랜잭션 수를 표시할 수 있습니다. 데이터베이스 `datfrozenxid` 행의 `pg_database` 열은 해당 데이터베이스에 나타나는 정상 트랜잭션 ID의 하한값입니다. 이 열은 데이터베이스 내 테이블 단위 `relfrozenxid` 값 중 최소값입니다.

```
SELECT datname, age(datfrozenxid) FROM pg_database ORDER BY age(datfrozenxid) desc
 limit 20;
```

예를 들어 앞의 쿼리를 실행하면 다음과 같은 결과가 나올 수 있습니다.

```
datname    | age
```

```
mydb       | 1771757888
template0  | 1721757888
template1  | 1721757888
rdsadmin   | 1694008527
postgres   | 1693881061
(5 rows)
```

데이터베이스의 수명이 20억 트랜잭션 ID에 도달하면 트랜잭션 ID(XID) 랩어라운드가 발생하고 데이터베이스는 읽기 전용이 됩니다. 이 쿼리를 사용하면 지표를 생성하여 하루에 몇 번 실행되도록 할 수 있습니다. 기본적으로 autovacuum은 트랜잭션 수명을 200,000,000(autovacuum_freeze_max_age) 미만으로 유지하도록 설정됩니다.

샘플 모니터링 전략은 다음과 같습니다.

- autovacuum_freeze_max_age 값을 2억 개 트랜잭션으로 설정하십시오.
- 테이블이 5억 개의 고정되지 않은 트랜잭션에 도달하면 낮은 심각도 경보가 트리거됩니다. 이 값은 타당한 값이지만 autovacuum이 계속 수행되고 있지 않음을 나타낼 수 있습니다.
- 테이블 수명이 10억이 되면 조치를 취해야 할 경보로 처리되어야 합니다. 성능상의 이유로 수명을 autovacuum_freeze_max_age에 더 가깝게 유지하려는 경우가 대부분입니다. 다음 권장 사항을 사용하여 조사하는 것이 좋습니다.
- 테이블이 15억 개의 vacuum되지 않은 트랜잭션에 도달하면 높은 심각도 경보가 트리거됩니다. 데이터베이스가 트랜잭션 ID를 사용하는 속도에 따라 이 경보는 시스템에서 autovacuum을 실행할 시간이 부족함을 나타낼 수 있습니다. 이 경우 즉시 이를 해결하는 것이 좋습니다.

테이블이 지속적으로 이 임계값을 위반하면 autovacuum 파라미터를 추가로 수정합니다. 기본적으로 수동 VACUUM을 사용하면(비용에 따른 지연이 비활성화됨)은 기본 autovacuum을 사용할 때보다 더 적극적이지만 시스템 전체에 더 많이 침입할 수 있는 상태이기도 합니다.

다음과 같이 하는 것이 좋습니다.

- 모니터링 메커니즘을 숙지하고 활성화하여 가장 오래된 트랜잭션의 수명을 확인합니다.

  트랜잭션 ID 랩어라운드에 대해 경고하는 프로세스 생성에 대한 자세한 내용은 AWS 데이터베이스 블로그 게시물 Amazon RDS for PostgreSQL의 트랜잭션 ID 랩어라운드용 조기 경고 시스템 구현을 참조하세요.
- 더 많이 사용되는 테이블의 경우 autovacuum을 사용하는 것 이외에 유지 관리 기간 동안 수동 vacuum freeze를 정기적으로 수행합니다. 수동 vacuum freeze 수행에 대한 자세한 내용은 수동 vacuum freeze 수행 단원을 참조하십시오.

현재 Autovacuum을 수행할 수 있는 테이블 결정

vacuum을 수행해야 하는 테이블이 하나이거나 두 개인 경우가 많습니다. `relfrozenxid` 값이 `autovacuum_freeze_max_age`의 트랜잭션 수보다 큰 테이블은 항상 Autovacuum의 대상이 됩니다. 그렇지 않은 경우 VACUUM이 "vacuum 임계값"을 초과하여 튜플 수가 더 이상 사용되지 않는 경우 테이블이 vacuum됩니다.

autovacuum 임계값은 다음과 같이 정의되어 있습니다.

```
Vacuum-threshold = vacuum-base-threshold + vacuum-scale-factor * number-of-tuples
```

여기서 `vacuum base threshold`는 `autovacuum_vacuum_threshold`이고, `vacuum scale factor`는 `autovacuum_vacuum_scale_factor`이며, `number of tuples`는 `pg_class.reltuples`입니다.

데이터베이스에 연결되어 있는 상태에서 다음 쿼리를 실행하여 autovacuum이 vacuum 가능한 대상으로 분류하는 테이블 목록을 확인합니다.

```
WITH vbt AS (SELECT setting AS autovacuum_vacuum_threshold FROM
pg_settings WHERE name = 'autovacuum_vacuum_threshold'),
vsf AS (SELECT setting AS autovacuum_vacuum_scale_factor FROM
pg_settings WHERE name = 'autovacuum_vacuum_scale_factor'),
fma AS (SELECT setting AS autovacuum_freeze_max_age FROM pg_settings WHERE name =
 'autovacuum_freeze_max_age'),
sto AS (select opt_oid, split_part(setting, '=', 1) as param,
split_part(setting, '=', 2) as value from (select oid opt_oid, unnest(reloptions)
 setting from pg_class) opt)
SELECT '"'||ns.nspname||'"."'||c.relname||'"' as relation,
pg_size_pretty(pg_table_size(c.oid)) as table_size,
age(relfrozenxid) as xid_age,
coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
 autovacuum_freeze_max_age,
(coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) * c.reltuples)
AS autovacuum_vacuum_tuples, n_dead_tup as dead_tuples FROM
pg_class c join pg_namespace ns on ns.oid = c.relnamespace
join pg_stat_all_tables stat on stat.relid = c.oid join vbt on (1=1) join vsf on (1=1)
 join fma on (1=1)
left join sto cvbt on cvbt.param = 'autovacuum_vacuum_threshold' and c.oid =
 cvbt.opt_oid
left join sto cvsf on cvsf.param = 'autovacuum_vacuum_scale_factor' and c.oid =
 cvsf.opt_oid
```

```
left join sto cfma on cfma.param = 'autovacuum_freeze_max_age' and c.oid = cfma.opt_oid
WHERE c.relkind = 'r' and nspname <> 'pg_catalog'
AND (age(relfrozenxid) >= coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
OR coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) *
c.reltuples <= n_dead_tup)
ORDER BY age(relfrozenxid) DESC LIMIT 50;
```

현재 Autovacuum이 실행 중인지 여부 및 실행 기간 확인

테이블을 수동으로 vacuum해야 하는 경우 autovacuum이 현재 실행 중인지 확인합니다. 실행 중이면 더 효율적으로 실행되도록 파라미터를 수정하거나 VACUUM을 수동으로 실행할 수 있도록 일시적으로 autovacuum을 종료해야 합니다.

다음 쿼리를 사용하여 autovacuum이 실행 중인지 여부와 얼마 동안 실행되고 있는지, 다른 세션에 대해 대기하고 있는지 확인합니다.

```
SELECT datname, usename, pid, state, wait_event, current_timestamp - xact_start AS
 xact_runtime, query
FROM pg_stat_activity
WHERE upper(query) LIKE '%VACUUM%'
ORDER BY xact_start;
```

쿼리를 실행하면 다음과 유사한 출력이 표시됩니다.

```
 datname | usename  |  pid  | state  | wait_event |      xact_runtime       | query
 --------+----------+-------+--------+------------+-------------------------
+-----------------------------------------------------------------------------------------------
 mydb    | rdsadmin | 16473 | active |            | 33 days 16:32:11.600656 |
 autovacuum: VACUUM ANALYZE public.mytable1 (to prevent wraparound)
 mydb    | rdsadmin | 22553 | active |            | 14 days 09:15:34.073141 |
 autovacuum: VACUUM ANALYZE public.mytable2 (to prevent wraparound)
 mydb    | rdsadmin | 41909 | active |            | 3 days 02:43:54.203349  |
 autovacuum: VACUUM ANALYZE public.mytable3
 mydb    | rdsadmin |   618 | active |            | 00:00:00                |
 SELECT datname, usename, pid, state, wait_event, current_timestamp - xact_start AS
 xact_runtime, query+
         |          |       |        |            |                         | FROM
 pg_stat_activity
           +
```

```
        |          |       |        |            |                          | WHERE
query like '%VACUUM%'
         +
        |          |       |        |            |                          | ORDER BY
xact_start;
      +
```

몇 가지 문제로 인해 autovacuum 세션이 오래(며칠간) 실행될 수 있습니다. 이 문제는 대부분 maintenance_work_mem 파라미터 값이 테이블 크기 또는 업데이트 속도에 대해 너무 낮게 설정된 경우입니다.

다음 공식을 사용하여 maintenance_work_mem 파라미터 값을 설정하는 것이 좋습니다.

```
GREATEST({DBInstanceClassMemory/63963136*1024},65536)
```

짧은 기간 동안 실행되는 autovacuum 세션에서도 문제를 표시할 수 있습니다.

- 워크로드에 autovacuum_max_workers가 충분하지 않다고 표시될 수 있습니다. 이 경우 작업자 수를 명시해야 합니다.
- 인덱스 손상(autovacuum에 충돌이 발생하여 동일한 관계에서 다시 시작되지만 진행되지 않음)이 있다고 표시될 수 있습니다. 이 경우에는 vacuum freeze verbose *table* 매뉴얼을 실행하여 정확한 원인을 확인합니다.

수동 vacuum freeze 수행

vacuum 프로세스가 실행되고 있는 테이블에서 수동 vacuum을 수행하려는 경우가 있습니다. 이 작업은 수명이 20억 개 트랜잭션에 도달하거나 모니터링 중인 임계값을 초과한 테이블을 파악해 둔 경우 유용합니다.

다음 단계는 지침으로 이 프로세스를 여러 가지로 변형할 수 있습니다. 예를 들어 테스트 중에 maintenance_work_mem 파라미터 값이 너무 작게 설정되었고 테이블에 작업을 즉시 수행해야 한다고 가정해보겠습니다. 그러나 지금은 인스턴스를 반송하고 싶지 않을 수도 있습니다. 이전 세션의 쿼리를 사용하여 어떤 테이블이 문제이고 오랜 기간 동안 실행 중인 autovacuum 세션이 있는지 확인합니다. maintenance_work_mem 파라미터 설정도 변경해야 하지만 즉시 조치를 취해 문제가 되는 테이블을 vacuum해야 하기도 합니다. 이 경우 어떤 작업을 수행해야 하는지가 다음 절차에 나와 있습니다.

vacuum freeze를 수동으로 수행하려면

1. vacuum할 테이블이 포함되어 있는 데이터베이스에 세션 두 개를 엽니다. 두 번째 세션의 경우 "screen"을 사용하거나 연결이 끊긴 경우 세션을 유지하는 다른 유틸리티를 사용합니다.
2. 첫 번째 세션에서는 테이블에서 실행 중인 autovacuum 세션의 프로세스 ID(PID)를 가져옵니다.

   다음 쿼리를 실행하여 autovacuum 세션의 PID를 가져옵니다.

   ```
   SELECT datname, usename, pid, current_timestamp - xact_start
   AS xact_runtime, query
   FROM pg_stat_activity WHERE upper(query) LIKE '%VACUUM%' ORDER BY
   xact_start;
   ```

3. 세션 2에서 이 작업에 필요한 메모리 양을 계산합니다. 이 예제에서는 이 작업에 메모리를 최대 2GB까지 사용할 수 있는 것으로 보고 현재 세션의 maintenance_work_mem을 2GB로 설정합니다.

   ```
   SET maintenance_work_mem='2 GB';
   SET
   ```

4. 세션 2에서 테이블에 대한 `vacuum freeze verbose` 명령을 실행하십시오. 현재 PostgreSQL에서 이 작업에 대한 진행률 보고가 없음에도 작업을 확인할 수 있기 때문에 상세 정보 표시 설정이 유용하게 사용됩니다.

   ```
   \timing on
   Timing is on.
   vacuum freeze verbose pgbench_branches;
   ```

   ```
   INFO:  vacuuming "public.pgbench_branches"
   INFO:  index "pgbench_branches_pkey" now contains 50 row versions in 2 pages
   DETAIL:  0 index row versions were removed.
   0 index pages have been deleted, 0 are currently reusable.
   CPU 0.00s/0.00u sec elapsed 0.00 sec.
   INFO:  index "pgbench_branches_test_index" now contains 50 row versions in 2 pages
   DETAIL:  0 index row versions were removed.
   0 index pages have been deleted, 0 are currently reusable.
   CPU 0.00s/0.00u sec elapsed 0.00 sec.
   INFO:  "pgbench_branches": found 0 removable, 50 nonremovable row versions
        in 43 out of 43 pages
   DETAIL:  0 dead row versions cannot be removed yet.
   There were 9347 unused item pointers.
   ```

```
0 pages are entirely empty.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
VACUUM
Time: 2.765 ms
```

5. 세션 1에서 autovacuum이 vacuum 세션을 차단한 경우 pg_stat_activity에서 vacuum 세션에 대한 대기를 나타내는 T를 확인할 수 있습니다. 이 경우 다음과 같이 autovacuum 프로세스를 종료합니다.

```
SELECT pg_terminate_backend('the_pid');
```

> **Note**
>
> Amazon RDS의 일부 하위 버전은 위의 명령을 사용하여 autovacuum 프로세스를 종료할 수 없으며 `ERROR: 42501: must be a superuser to terminate superuser process LOCATION: pg_terminate_backend, signalfuncs.c:227` 오류와 함께 작업이 실패합니다.

이때 세션이 시작됩니다. 이 테이블이 작업 목록에서 가장 상위에 있을 것이므로 autovacuum이 즉시 재시작됩니다.

6. 세션 2에서 `vacuum freeze verbose` 명령을 시작한 다음 세션 1에서 autovacuum 프로세스를 종료합니다.

Autovacuum이 실행 중인 경우 테이블 인덱스 다시 지정

인덱스가 손상되면 autovacuum은 계속해서 테이블을 처리하려 하고 실패합니다. 이 경우 수동 vacuum을 시도하면 다음과 비슷한 오류 메시지가 표시됩니다.

```
postgres=>  vacuum freeze pgbench_branches;
ERROR: index "pgbench_branches_test_index" contains unexpected
   zero page at block 30521
HINT: Please REINDEX it.
```

인덱스가 손상된 상태에서 테이블에 대해 autovacuum을 실행하려고 하면 이미 실행 중인 autovacuum 세션이 있음을 확인하게 됩니다. REINDEX 명령을 실행하면 테이블에 대한 단독 잠금을 해제합니다. 쓰기 작업과 해당 특정 인덱스를 사용하는 읽기 작업도 차단됩니다.

테이블에서 autovacuum을 실행할 때 테이블 인덱스를 다시 지정하려면

1. vacuum할 테이블이 포함되어 있는 데이터베이스에 세션 두 개를 엽니다. 두 번째 세션의 경우 "screen"을 사용하거나 연결이 끊긴 경우 세션을 유지하는 다른 유틸리티를 사용합니다.
2. 첫 번째 세션에서는 테이블에서 실행 중인 autovacuum 세션의 PID를 가져옵니다.

   다음 쿼리를 실행하여 autovacuum 세션의 PID를 가져옵니다.

   ```
   SELECT datname, usename, pid, current_timestamp - xact_start
   AS xact_runtime, query
   FROM pg_stat_activity WHERE upper(query) like '%VACUUM%' ORDER BY
   xact_start;
   ```

3. 세션 2에서 reindex 명령을 실행합니다.

   ```
   \timing on
   Timing is on.
   reindex index pgbench_branches_test_index;
   REINDEX
     Time: 9.966 ms
   ```

4. 세션 1에서 autovacuum이 프로세스를 차단한 경우 pg_stat_activity에서 vacuum 세션에 대한 대기를 나타내는 "T"를 확인할 수 있습니다. 이 경우에는 autovacuum 프로세스를 종료합니다.

   ```
   SELECT pg_terminate_backend('the_pid');
   ```

   이때 세션이 시작됩니다. 이 테이블이 작업 목록에서 가장 상위에 있을 것이므로 autovacuum이 즉시 다시 시작된다는 점을 알아 두어야 합니다.
5. 세션 2에서 명령을 시작한 다음 세션 1에서 autovacuum 프로세스를 종료합니다.

대용량 인덱스를 사용하여 autovacuum 관리

작업 중에 autovacuum은 테이블에서 실행되는 동안 여러 vacuum 단계를 수행합니다. 테이블을 정리하기 전에 먼저 모든 인덱스에 vacuum을 실행합니다. 여러 개의 대용량 인덱스를 제거할 경우, 이 단계는 상당한 시간과 리소스를 사용합니다. 따라서 테이블의 인덱스 수를 제어하고 사용하지 않는 인덱스를 제거하는 것이 가장 좋습니다.

이 프로세스에서는 먼저 전체 인덱스 크기를 확인합니다. 그런 다음, 다음 예제에 나온 것처럼 제거할 수 있는 사용하지 않는 인덱스가 있는지 확인합니다.

테이블 및 해당 인덱스의 크기를 확인하려면

```
postgres=> select pg_size_pretty(pg_relation_size('pgbench_accounts'));
pg_size_pretty
6404 MB
(1 row)
```

```
postgres=> select pg_size_pretty(pg_indexes_size('pgbench_accounts'));
pg_size_pretty
11 GB
(1 row)
```

이 예제에서는 인덱스 크기가 테이블보다 큽니다. 이러한 차이로 인해 인덱스가 팽창하거나 사용되지 않아 성능 문제가 발생하여 autovacuum 및 삽입 작업에 영향을 미칠 수 있습니다.

사용하지 않는 인덱스를 확인하려면

pg_stat_user_indexes 보기를 사용하면 인덱스가 idx_scan 열에 사용되는 빈도를 확인할 수 있습니다. 다음 예제를 보면 사용하지 않은 인덱스는 idx_scan 값이 0입니다.

```
postgres=> select * from pg_stat_user_indexes where relname = 'pgbench_accounts' order
 by idx_scan desc;

relid  | indexrelid | schemaname | relname          | indexrelname          | idx_scan
 | idx_tup_read | idx_tup_fetch
-------+------------+------------+------------------+-----------------------+----------
+--------------+--------------
16433  | 16454      | public     | pgbench_accounts | index_f               | 6
 | 6            | 0
16433  | 16450      | public     | pgbench_accounts | index_b               | 3
 | 199999       | 0
16433  | 16447      | public     | pgbench_accounts | pgbench_accounts_pkey | 0
 | 0            | 0
16433  | 16452      | public     | pgbench_accounts | index_d               | 0
 | 0            | 0
16433  | 16453      | public     | pgbench_accounts | index_e               | 0
 | 0            | 0
16433  | 16451      | public     | pgbench_accounts | index_c               | 0
 | 0            | 0
16433  | 16449      | public     | pgbench_accounts | index_a               | 0
 | 0            | 0
(7 rows)
```

```
postgres=> select schemaname, relname, indexrelname, idx_scan from pg_stat_user_indexes
 where relname = 'pgbench_accounts' order by idx_scan desc;

schemaname  | relname          | indexrelname          | idx_scan
------------+------------------+-----------------------+----------
public      | pgbench_accounts | index_f               | 6
public      | pgbench_accounts | index_b               | 3
public      | pgbench_accounts | pgbench_accounts_pkey | 0
public      | pgbench_accounts | index_d               | 0
public      | pgbench_accounts | index_e               | 0
public      | pgbench_accounts | index_c               | 0
public      | pgbench_accounts | index_a               | 0
(7 rows)
```

> **Note**
>
> 이러한 통계는 통계가 재설정된 시점부터 증분됩니다. 사업 분기 말에만 사용되거나 특정 보고서에만 사용되는 인덱스가 있는 경우를 가정해 보겠습니다. 통계가 재설정된 이후로 이 인덱스가 사용되지 않았을 수 있습니다. 자세한 내용은 Statistics Functions(통계 함수)를 참조하세요. 고유성을 적용하는 데 사용되는 인덱스는 스캔이 실행되지 않으므로 사용하지 않는 인덱스로 식별해선 안 됩니다. 사용하지 않는 인덱스를 식별하려면 애플리케이션 및 해당 쿼리에 대한 심층적인 지식이 있어야 합니다.

데이터베이스의 통계가 마지막으로 재설정된 시간을 확인하려면 pg_stat_database를 사용하세요.

```
postgres=> select datname, stats_reset from pg_stat_database where datname =
 'postgres';

datname   | stats_reset
----------+-------------------------------
postgres  | 2022-11-17 08:58:11.427224+00
(1 row)
```

테이블에 최대한 신속하게 Vacuum을 실행하는 방법

RDS for PostgreSQL 12 이상

대형 테이블에 인덱스가 너무 많으면 DB 인스턴스가 트랜잭션 ID 랩어라운드(XID)에 가까워질 수 있는데, 이때 XID 카운터가 0으로 래핑됩니다. 이 옵션을 선택하지 않으면 데이터가 손실될 수 있습니다. 그러나 인덱스를 정리하지 않고도 테이블에 신속하게 vacuum을 실행할 수 있습니다. RDS for PostgreSQL 12에서는 `INDEX_CLEANUP` 절과 함께 VACUUM을 사용할 수 있습니다.

```
postgres=> VACUUM (INDEX_CLEANUP FALSE, VERBOSE TRUE) pgbench_accounts;

INFO: vacuuming "public.pgbench_accounts"
INFO: table "pgbench_accounts": found 0 removable, 8 nonremovable row versions in 1 out
 of 819673 pages
DETAIL: 0 dead row versions cannot be removed yet, oldest xmin: 7517
Skipped 0 pages due to buffer pins, 0 frozen pages.
CPU: user: 0.01 s, system: 0.00 s, elapsed: 0.01 s.
```

autovacuum 세션이 이미 실행 중인 경우, 해당 세션을 종료하여 수동 VACUUM을 시작해야 합니다. 수동 vacuum freeze 수행에 대한 자세한 내용은 수동 vacuum freeze 수행 섹션을 참조하세요.

> **Note**
>
> 주기적인 인덱스 정리를 건너뛰면 인덱스 팽창이 발생하여 스캔 성능이 다운그레이드될 수 있습니다. 인덱스는 데드 행을 유지하고 테이블은 데드 라인 포인터를 유지합니다. 따라서 `pg_stat_all_tables.n_dead_tup`는 인덱스 정리가 포함된 수동 VACUUM 또는 autovacuum이 실행될 때까지 증가합니다. 가장 좋은 방법은 이 프로시저만 사용하여 트랜잭션 ID 랩어라운드를 방지하는 것입니다.

RDS for PostgreSQL 11 이상

그러나 RDS for PostgreSQL 11 이하 버전에서 vacuum을 더 신속하게 완료할 수 있는 유일한 방법은 테이블의 인덱스 수를 줄이는 것입니다. 인덱스를 삭제하면 쿼리 계획에 영향을 미칠 수 있습니다. 사용하지 않는 인덱스를 먼저 삭제한 다음, XID 랩어라운드가 매우 가까워졌을 때 인덱스를 삭제하는 것이 좋습니다. vacuum 프로세스가 완료되면 이러한 인덱스를 다시 생성할 수 있습니다.

Autovacuum에 영향을 주는 기타 파라미터

이 쿼리를 사용하면 autovacuum 및 해당 동작에 직접 영향을 주는 일부 파라미터 값이 표시됩니다. autovacuum 파라미터는 PostgreSQL 설명서에 자세히 설명되어 있습니다.

```
SELECT name, setting, unit, short_desc
```

```
FROM pg_settings
WHERE name IN (
'autovacuum_max_workers',
'autovacuum_analyze_scale_factor',
'autovacuum_naptime',
'autovacuum_analyze_threshold',
'autovacuum_analyze_scale_factor',
'autovacuum_vacuum_threshold',
'autovacuum_vacuum_scale_factor',
'autovacuum_vacuum_threshold',
'autovacuum_vacuum_cost_delay',
'autovacuum_vacuum_cost_limit',
'vacuum_cost_limit',
'autovacuum_freeze_max_age',
'maintenance_work_mem',
'vacuum_freeze_min_age');
```

모두 autovacuum에 영향을 주지만 가장 중요한 사항 몇 가지는 다음과 같습니다.

- maintenance_work_mem
- autovacuum_freeze_max_age
- autovacuum_max_workers
- autovacuum_vacuum_cost_delay
- autovacuum_vacuum_cost_limit

테이블 수준 Autovacuum 파라미터 설정

Autovacuum이 관련된 스토리지 파라미터를 테이블 수준에서 설정할 수 있습니다. 이 방법은 전체 데이터베이스의 동작을 변경하는 방법보다 더 나을 수 있습니다. 큰 테이블에 적극적인 설정을 지정해야 하지만 autovacuum이 모든 테이블에서 이와 같은 방식으로 작동하지 않도록 하려는 경우가 있을 수 있습니다.

이 쿼리를 사용하면 현재 어떤 테이블에 테이블 수준 옵션을 사용 중인지가 표시됩니다.

```
SELECT relname, reloptions
FROM pg_class
WHERE reloptions IS NOT null;
```

이 쿼리가 유용한 경우는 테이블 하나가 나머지 테이블보다 훨씬 더 큰 경우입니다. 300GB 테이블 하나와 1GB 미만의 테이블 30개가 있다고 가정하십시오. 이 경우 더 큰 테이블에 특정 파라미터를 설정하여 전체 시스템의 동작이 변경되지 않도록 할 수 있습니다.

```
ALTER TABLE mytable set (autovacuum_vacuum_cost_delay=0);
```

이렇게 하면 시스템의 리소스를 더 많이 사용하는 대신 이 테이블의 비용에 따른 autovacuum 지연이 비활성화됩니다. 일반적으로 매시간 `autovacuum_cost_limit`에 도달한 `autovacuum_vacuum_cost_delay`의 autovacuum이 일시 중지됩니다. 자세한 내용은 비용에 따른 vacuum 수행에 대한 PostgreSQL 설명서에서 확인할 수 있습니다.

autovacuum 및 vacuum 활동 로그

autovacuum 활동에 대한 정보는 `rds.force_autovacuum_logging_level` 파라미터에 지정된 레벨을 기반으로 `postgresql.log`에 전송됩니다. 다음은 이 파라미터와 PostgreSQL 버전에 허용되는 해당 값이 기본 설정인 값입니다.

- `disabled`(PostgreSQL 10, PostgreSQL 9.6)
- `debug5`, `debug4`, `debug3`, `debug2`, `debug1`
- `info`(PostgreSQL 12, PostgreSQL 11)
- `notice`
- `warning`(PostgreSQL 13 이상)
- `error`, 로그, `fatal`, `panic`

`rds.force_autovacuum_logging_level`은 `log_autovacuum_min_duration` 파라미터와 함께 작동합니다. `log_autovacuum_min_duration` 파라미터 값은 autovacuum 작업이 기록되는 임계값(밀리초) 이상입니다. `-1`로 설정하면 아무것도 기록하지 않지만 `0`으로 설정하면 모든 작업이 기록됩니다. `rds.force_autovacuum_logging_level`과 마찬가지로 `log_autovacuum_min_duration`의 기본값은 다음과 같이 버전에 따라 다릅니다.

- `10000 ms` - PostgreSQL 14, PostgreSQL 13, PostgreSQL 12, PostgreSQL 11
- `(empty)` - PostgreSQL 10 및 PostgreSQL 9.6 기본값 없음

`rds.force_autovacuum_logging_level`을 `WARNING`로 설정하는 것이 좋습니다. 또한 `log_autovacuum_min_duration`을 1000에서 5000까지의 값으로 설정하는 것이 좋습니다. 5,000

밀리초 이상 걸리는 5000개의 기록 활동 설정. -1을 제외한 모든 설정은 충돌하는 잠금 또는 동시에 삭제된 관계로 인해 autovacuum 작업을 건너뛴 경우에도 메시지를 기록합니다. 자세한 내용은 PostgreSQL 설명서의 자동 Vacuuming을 참조하세요.

문제를 해결하려면 `rds.force_autovacuum_logging_level` 매개 변수를 상세 표시 정보에 대해 debug1부터 debug5까지의 디버그 레벨 중 하나로 변경할 수 있습니다. 디버그 설정은 단기 문제 해결 목적으로만 사용하는 것이 좋습니다. 자세한 내용은 PostgreSQL의 로그 시기 문서를 참조하세요.

> **Note**
>
> PostgreSQL을 사용하면 `rds_superuser` 계정에서 `pg_stat_activity`의 autovacuum 세션을 볼 수 있습니다. 예: 명령의 실행을 차단하거나, 수동으로 실행한 vacuum 명령보다 느리게 실행되는 autovacuum 세션을 식별 및 종료 가능

잘못된 데이터베이스를 사용한 autovacuum 동작 이해

DROP DATABASE 작업 도중에 중단된 데이터베이스를 유효하지 않은 것으로 나타내는 새 `-2` 값이 `pg_database` 카탈로그의 `datconnlimit` 열에 도입되었습니다.

이 새 값은 다음과 같은 RDS for PostgreSQL 버전에서 사용할 수 있습니다.

- 15.4 이상의 모든 버전
- 14.9 이상 버전
- 13.12 이상 버전
- 12.16 이상 버전
- 11.21 이상 버전

잘못된 데이터베이스는 유효한 데이터베이스의 기능을 중지하는 autovacuum 기능에 영향을 주지 않습니다. autovacuum은 잘못된 데이터베이스를 무시합니다. 따라서 PostgreSQL 환경의 모든 유효한 데이터베이스에 대해 정기적인 vacuum 작업이 계속해서 적절하고 효율적으로 작동합니다.

주제

- 트랜잭션 ID 모니터링
- 모니터링 쿼리 조정
- 잘못된 데이터베이스 문제 해결

트랜잭션 ID 모니터링

age(datfrozenxid) 함수는 일반적으로 데이터베이스의 트랜잭션 ID(XID) 수명을 모니터링하여 트랜잭션 ID 랩어라운드를 방지하는 데 사용됩니다.

유효하지 않은 데이터베이스는 autovacuum에서 제외되므로, 트랜잭션 ID(XID) 횟수가 최대값인 2 billion에 도달하여 - 2 billion로 순환하며 이 주기를 무한정 계속할 수 있습니다. 트랜잭션 ID 랩어라운드를 모니터링하는 일반적인 쿼리는 다음과 같습니다.

```
SELECT max(age(datfrozenxid)) FROM pg_database;
```

그러나 datconnlimit에 대한 -2 값이 도입되면 잘못된 데이터베이스가 이 쿼리의 결과를 왜곡할 수 있습니다. 이러한 데이터베이스는 유효하지 않으며 정기적인 유지 관리 검사에 포함되어서는 안 되므로, 오탐을 유발할 수 있어 실제보다 age(datfrozenxid)가 높다고 믿게 됩니다.

모니터링 쿼리 조정

정확한 모니터링을 위해서는 잘못된 데이터베이스를 제외하도록 모니터링 쿼리를 조정해야 합니다. 다음 권장 쿼리를 따르세요.

```
SELECT
    max(age(datfrozenxid))
FROM
    pg_database
WHERE
    datconnlimit <> -2;
```

이 쿼리를 사용하면 PostgreSQL 환경 전체의 트랜잭션 ID 사용 기간을 정확하게 반영하여 유효한 데이터베이스만 age(datfrozenxid) 계산에 고려합니다.

잘못된 데이터베이스 문제 해결

잘못된 데이터베이스에 연결하려고 하면 다음과 같은 오류 메시지가 표시될 수 있습니다.

```
postgres=> \c db1
connection to server at "mydb.xxxxxxxxxx.us-west-2.rds.amazonaws.com" (xx.xx.xx.xxx),
 port xxxx failed: FATAL:  cannot connect to invalid database "db1"
```

```
HINT:  Use DROP DATABASE to drop invalid databases.
Previous connection kept
```

또한, `log_min_messages` 파라미터가 `DEBUG2` 이상으로 설정된 경우 autovacuum 프로세스가 잘못된 데이터베이스를 건너뛰고 있음을 나타내는 다음과 같은 로그 항목을 확인할 수 있습니다.

```
2024-07-30 05:59:00 UTC::@:[32000]:DEBUG:  autovacuum: skipping invalid database "db6"
2024-07-30 05:59:00 UTC::@:[32000]:DEBUG:  autovacuum: skipping invalid database "db1"
```

이 문제를 해결하려면 연결 시도 중에 제공된 `HINT`를 따르세요. RDS 마스터 계정 또는 `rds_superuser` 역할이 있는 데이터베이스 계정을 사용하여 유효한 데이터베이스에 연결하고 잘못된 데이터베이스를 삭제합니다.

```
SELECT
    'DROP DATABASE ' || quote_ident(datname) || ';'
FROM
    pg_database
WHERE
    datconnlimit = -2 \gexec
```

RDS for PostgreSQL에서 공격적인 vacuum 블로커 식별 및 해결

PostgreSQL에서는 vacuum 작업이 스토리지를 회수하고 트랜잭션 ID 랩어라운드 문제를 방지하기 때문에 데이터베이스의 좋은 상태를 위해 vacuum 작업이 중요합니다. 그러나 vacuum 작업이 원하는 대로 작동하지 않게 되는 경우가 있어 성능이 저하되고, 스토리지가 팽창하고, 트랜잭션 ID 랩어라운드로 인해 DB 인스턴스의 가용성이 영향을 받을 수 있습니다. 따라서 최적의 데이터베이스 성능과 가용성을 위해서는 이러한 문제를 식별하고 해결하는 것이 필수적입니다. Autovacuum에 대해 자세히 알아보려면 Understanding autovacuum in Amazon RDS for PostgreSQL environments를 읽어보세요.

`postgres_get_av_diag()` 함수는 공격적 vacuum 진행 상황을 방지하거나 지연시키는 문제를 식별하는 데 도움이 됩니다. 제안이 제공되며, 여기에는 문제를 식별할 수 있는 경우 문제 해결을 위한 명령이, 문제를 식별할 수 없는 경우 추가 진단을 위한 지침이 포함될 수 있습니다. 공격적 vacuum 블로커는 연령이 RDS의 적응형 autovacuum 임곗값인 트랜잭션 ID 5억 개를 초과할 때 보고됩니다.

트랜잭션 ID의 연령이란 무엇인가요?

트랜잭션 ID에 대한 age() 함수는 데이터베이스(pg_database.datfrozenxid) 또는 테이블(pg_class.relfrozenxid)에 대해 가장 오래된 고정되지 않은 트랜잭션 ID 이후 발생한 트랜잭션 수를 계산합니다. 이 값은 마지막 공격적 vacuum 작업 이후의 데이터베이스 활동을 나타내며 향후 VACUUM 프로세스에 대해 가능한 워크로드를 강조합니다.

공격적 vacuum이란 무엇인가요?

공격적인 VACUUM 작업은 일반적인 VACUUM에서는 대개 건너 뛰는 페이지를 포함하여 테이블 내에 있는 모든 페이지에 대한 종합적인 스캔을 수행합니다. 이 철저한 스캔의 목표는 최대 연령에 근접한 트랜잭션 ID를 '고정'하여 트랜잭션 ID 랩어라운드라고 하는 상황을 효과적으로 방지하는 것을 목표로 합니다.

postgres_get_av_diag()가 블로커를 보고하려면 블로커의 연령이 최소 5억 개의 트랜잭션이어야 합니다.

주제

- RDS for PostgreSQL에 자동 정리 모니터링 및 진단 도구 설치
- RDS for PostgreSQL의 postgres_get_av_diag() 기능
- RDS for PostgreSQL에서 식별 가능한 vacuum 블로커 해결
- RDS for PostgreSQL에서 식별 불가능한 vacuum 블로커 해결
- RDS for PostgreSQL의 vacuum 성능 문제 해결
- RDS for PostgreSQL의 NOTICE 메시지 설명

RDS for PostgreSQL에 자동 정리 모니터링 및 진단 도구 설치

postgres_get_av_diag() 함수는 현재 다음 RDS for PostgreSQL 버전에서 사용할 수 있습니다.

- 17.2 이상의 17 버전
- 16.7 이상의 16 버전
- 15.11 이상의 15 버전
- 14.16 이상의 14 버전
- 13.19 이상의 13 버전

postgres_get_av_diag()를 사용하려면 rds_tools 확장을 만듭니다.

```
postgres=> CREATE EXTENSION rds_tools ;
CREATE EXTENSION
```

확장이 설치되어 있는지 확인합니다.

```
postgres=> \dx rds_tools
             List of installed extensions
   Name    | Version |  Schema   |                    Description
 ----------+---------+-----------
+------------------------------------------------------------
 rds_tools |   1.8   | rds_tools | miscellaneous administrative functions for RDS
 PostgreSQL
 1 row
```

함수가 만들어졌는지 확인합니다.

```
postgres=> SELECT
    proname function_name,
    pronamespace::regnamespace function_schema,
    proowner::regrole function_owner
FROM
    pg_proc
WHERE
    proname = 'postgres_get_av_diag';
    function_name     | function_schema | function_owner
----------------------+-----------------+----------------
 postgres_get_av_diag | rds_tools       | rds_superuser
(1 row)
```

RDS for PostgreSQL의 postgres_get_av_diag() 기능

`postgres_get_av_diag()` 함수는 RDS for PostgreSQL 데이터베이스에서 차단하거나 지연되는 자동 정리 프로세스에 대한 진단 정보를 검색합니다. 정확한 결과를 얻으려면 가장 오래된 트랜잭션 ID로 데이터베이스에서 쿼리를 실행해야 합니다. 가장 오래된 트랜잭션 ID로 데이터베이스를 사용하는 방법에 대한 자세한 내용은 Not connected to the database with the age of oldest transaction ID를 참조하세요.

```
SELECT
    blocker,
    DATABASE,
    blocker_identifier,
```

```
    wait_event,
    TO_CHAR(autovacuum_lagging_by, 'FM9,999,999,999') AS autovacuum_lagging_by,
    suggestion,
    suggested_action
FROM (
    SELECT
        *
    FROM
        rds_tools.postgres_get_av_diag ()
    ORDER BY
        autovacuum_lagging_by DESC) q;
```

`postgres_get_av_diag()` 함수는 다음 정보를 반환합니다.

blocker

Vacuum을 차단하는 데이터베이스 활동의 범주를 보여줍니다.

- 활성 문
- 트랜잭션의 유휴 상태
- 준비된 트랜잭션
- 논리적 복제 슬롯
- 물리적 복제 슬롯을 사용하는 읽기 전용 복제본
- 스트리밍 복제를 사용하는 읽기 전용 복제본
- 임시 테이블

데이터베이스

적용 가능하고 지원되는 경우 데이터베이스의 이름을 보여줍니다. 활동이 진행 중이고 autovacuum을 차단 중이거나 차단할 데이터베이스입니다. 연결하고 조치를 취해야 하는 데이터베이스입니다.

blocker_identifier

Autovacuum을 차단 중이거나 차단할 활동의 식별자를 보여줍니다. 식별자는 SQL 문, 준비된 트랜잭션, 읽기 전용 복제본의 IP 주소, 논리적 또는 물리적 복제 슬롯의 이름과 함께 프로세스 ID일 수 있습니다.

wait_event

차단 세션의 대기 이벤트 를 지정하고 다음 블로커에 적용할 수 있습니다.

- 활성 문
- 트랜잭션의 유휴 상태

autovacum_lagging_by

범주당 백로그 작업에서 autovacuum이 지연되는 트랜잭션 수를 보여줍니다.

suggestion

블로커를 해결하기 위한 제안을 보여줍니다. 이러한 지침에는 해당하는 경우 활동이 존재하는 데이터베이스의 이름, 해당하는 경우 세션의 프로세스 ID(PID) 및 수행할 작업이 포함됩니다.

suggested_action

블로커를 해결하기 위해 수행해야 하는 작업을 제안합니다.

RDS for PostgreSQL에서 식별 가능한 vacuum 블로커 해결

Autovacuum은 공격적인 vacuum을 수행하고 트랜잭션 ID의 연령을 RDS 인스턴스의 `autovacuum_freeze_max_age` 파라미터에 의해 지정된 임곗값 미만으로 낮춥니다. 이 연령은 Amazon CloudWatch 지표 `MaximumUsedTransactionIDs`를 사용하여 추적할 수 있습니다.

Amazon RDS 인스턴스에 대한 `autovacuum_freeze_max_age`의 설정(기본값은 2억 개의 트랜잭션 ID)을 찾으려면 다음 쿼리를 사용할 수 있습니다.

```
SELECT
    TO_CHAR(setting::bigint, 'FM9,999,999,999') autovacuum_freeze_max_age
FROM
    pg_settings
WHERE
    name = 'autovacuum_freeze_max_age';
```

`postgres_get_av_diag()`는 연령이 Amazon RDS의 적응형 autovacuum 임곗값인 트랜잭션 ID 5억 개를 초과할 때만 공격적 vacuum 블로커를 확인합니다. `postgres_get_av_diag()`가 블로커를 감지하려면 블로커의 연령이 최소 5억 개의 트랜잭션이어야 합니다.

`postgres_get_av_diag()` 함수는 다음과 같은 유형의 블로커를 식별합니다.

주제

- 활성 문
- 트랜잭션의 유휴 상태
- 준비된 트랜잭션
- 논리적 복제 슬롯
- 읽기 전용 복제본

- 임시 테이블

활성 문

PostgreSQL에서 활성 문은 현재 데이터베이스에서 실행 중인 SQL 문입니다. 여기에는 쿼리, 트랜잭션 또는 진행 중인 작업이 포함됩니다. pg_stat_activity를 통해 모니터링할 때 상태 열은 해당 PID가 있는 프로세스가 활성 상태임을 나타냅니다.

postgres_get_av_diag() 함수는 활성 문인 문을 식별할 때 다음과 유사한 출력을 표시합니다.

```
blocker               | Active statement
database              | my_database
blocker_identifier    | SELECT pg_sleep(20000);
wait_event            | Timeout:PgSleep
autovacuum_lagging_by | 568,600,871
suggestion            | Connect to database "my_database", review carefully and you
 may consider terminating the process using suggested_action. For more information, see
 Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"SELECT pg_terminate_backend (29621);"}
```

권장 조치

suggestion 열의 지침에 따라 사용자는 활성 문이 있는 데이터베이스에 연결할 수 있으며 suggested_action 열에 지정된 대로 세션을 종료하는 옵션을 주의 깊게 검토하는 것이 좋습니다. 종료가 안전한 경우 pg_terminate_backend() 함수를 사용하여 세션을 종료할 수 있습니다. 이 작업은 관리자(예: RDS 마스터 계정) 또는 필요한 pg_terminate_backend() 권한이 있는 사용자가 수행할 수 있습니다.

> **Warning**
>
> 종료된 세션은 해당 세션이 만든 (ROLLBACK) 변경 사항을 취소합니다. 요구 사항에 따라 문을 다시 실행할 수 있습니다. 그러나 autovacuum 프로세스가 공격적인 vacuum 작업을 완료한 후에만 다시 실행하는 것이 좋습니다.

트랜잭션의 유휴 상태

트랜잭션 문의 유휴는 명시적 트랜잭션을 열고(예: BEGIN 명령문을 발급하여), 일부 작업을 수행했으며, 이제 클라이언트가 더 많은 작업을 전달하거나 COMMIT, ROLLBACK또는 END(암시적 COMMIT을 유발함)를 발급하여 트랜잭션 종료 신호를 보내기를 기다리는 세션을 나타냅니다.

postgres_get_av_diag() 함수는 idle in transaction 문을 블로커로 식별할 때 다음과 유사한 출력을 표시합니다.

```
blocker               | idle in transaction
database              | my_database
blocker_identifier    | INSERT INTO tt SELECT * FROM tt;
wait_event            | Client:ClientRead
autovacuum_lagging_by | 1,237,201,759
suggestion            | Connect to database "my_database", review carefully and you
 may consider terminating the process using suggested_action. For more information, see
 Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"SELECT pg_terminate_backend (28438);"}
```

권장 조치

suggestion 열에 표시된 대로 트랜잭션 세션에 유휴가 있는 데이터베이스에 연결하고 pg_terminate_backend() 함수를 사용하여 세션을 종료할 수 있습니다. 사용자는 관리자(RDS 마스터 계정) 사용자이거나 pg_terminate_backend() 권한이 있는 사용자일 수 있습니다.

> **Warning**
>
> 종료된 세션은 해당 세션이 만든 (ROLLBACK) 변경 사항을 취소합니다. 요구 사항에 따라 문을 다시 실행할 수 있습니다. 그러나 autovacuum 프로세스가 공격적인 vacuum 작업을 완료한 후에만 다시 실행하는 것이 좋습니다.

준비된 트랜잭션

PostgreSQL은 준비된 트랜잭션이라고 하는 두 단계 커밋 전략의 일부인 트랜잭션을 허용합니다. max_prepared_transactions 파라미터를 0이 아닌 값으로 설정하면 활성화됩니다. 준비된 트랜잭션은 데이터베이스 충돌, 재시작 또는 클라이언트 연결 해제 후에도 트랜잭션이 내구성을 유지하고 계속 가능하도록 하기 위해 설계되었습니다. 일반 트랜잭션과 마찬가지로 트랜잭션 ID가 할당되며 autovacuum에 영향을 미칠 수 있습니다. 준비된 상태로 두면 autovacuum이 고정을 수행할 수 없으며 트랜잭션 ID 랩어라운드로 이어질 수 있습니다.

트랜잭션 관리자가 확인하지 않고 트랜잭션을 무기한으로 준비된 상태로 두면 고립된 준비 트랜잭션이 됩니다. 이 문제를 해결하는 유일한 방법은 COMMIT PREPARED 또는 ROLLBACK PREPARED 명령을 각각 사용하여 트랜잭션을 커밋하거나 롤백하는 것입니다.

> **Note**
>
> 준비된 트랜잭션 중에 가져온 백업에는 복원 후에도 해당 트랜잭션이 계속 포함되어 있습니다. 이러한 트랜잭션을 찾고 닫는 방법은 다음 정보를 참조하세요.

준비된 트랜잭션인 블로커를 식별하면 `postgres_get_av_diag()` 함수에 다음 출력이 표시됩니다.

```
blocker               | Prepared transaction
database              | my_database
blocker_identifier    | myptx
wait_event            | Not applicable
autovacuum_lagging_by | 1,805,802,632
suggestion            | Connect to database "my_database" and consider either COMMIT
 or ROLLBACK the prepared transaction using suggested_action. For more information, see
 Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"COMMIT PREPARED 'myptx';",[OR],"ROLLBACK PREPARED 'myptx';"}
```

권장 조치

제안 열에 언급된 대로 준비된 트랜잭션이 있는 데이터베이스에 연결합니다. `suggested_action` 열을 기반으로 `COMMIT`과 `ROLLBACK` 중 무엇을 수행할지 주의 깊게 검토하고 작업을 적절히 적용합니다.

준비된 트랜잭션을 일반적으로 모니터링하기 위해 PostgreSQL은 `pg_prepared_xacts`라는 카탈로그 보기를 제공합니다. 다음 쿼리를 사용하여 준비된 트랜잭션을 찾을 수 있습니다.

```
SELECT
    gid,
    prepared,
    owner,
    database,
    transaction AS oldest_xmin
FROM
    pg_prepared_xacts
ORDER BY
    age(transaction) DESC;
```

논리적 복제 슬롯

복제 슬롯의 목적은 사용되지 않은 변경 사항이 대상 서버에 복제될 때까지 이를 유지하는 것입니다. 자세한 내용은 PostgreSQL의 Logical replication을 참조하세요.

논리적 복제 슬롯에는 두 가지 유형이 있습니다.

비활성 논리적 복제 슬롯

복제가 종료되면 소비되지 않은 트랜잭션 로그를 제거할 수 없으며 복제 슬롯이 비활성화됩니다. 비활성 논리적 복제 슬롯은 현재 구독자가 사용하지 않지만 서버에 남아 있어 WAL 파일이 보존되고 이전 트랜잭션 로그를 제거할 수 없습니다. 이렇게 되면 LSN 정보를 덮어쓰지 않도록 시스템이 보존해야 하므로 디스크 사용량이 증가하고 특히 autovacuum이 내부 카탈로그 테이블을 정리하지 못하게 됩니다. 해결하지 않으면 카탈로그 팽창, 성능 저하 및 랩어라운드 vacuum 위험 증가로 이어질 수 있으며 이로 인해 트랜잭션 가동 중지 시간이 발생할 수도 있습니다.

활성이지만 느린 논리적 복제 슬롯

논리적 복제의 성능 저하로 인해 카탈로그의 데드 튜플 제거가 지연되는 경우가 있습니다. 이러한 복제 지연으로 인해 `catalog_xmin`의 업데이트 속도가 느려지고 카탈로그 팽창 및 랩어라운드 vacuum이 발생할 수 있습니다.

`postgres_get_av_diag()` 함수는 논리적 복제 슬롯을 블로커로 찾았을 때 다음과 유사한 출력을 표시합니다.

```
blocker               | Logical replication slot
database              | my_database
blocker_identifier    | slot1
wait_event            | Not applicable
autovacuum_lagging_by | 1,940,103,068
suggestion            | Ensure replication is active and resolve any lag for the slot
 if active. If inactive, consider dropping it using the command in suggested_action.
 For more information, see Working with PostgreSQL autovacuum in the Amazon RDS User
 Guide.
suggested_action      | {"SELECT pg_drop_replication_slot('slot1') FROM
 pg_replication_slots WHERE active = 'f';"}
```

권장 조치

이 문제를 해결하려면 대상 스키마 또는 적용 프로세스를 종료할 수 있는 데이터에 문제가 있는지 복제 구성을 확인합니다. 가장 일반적인 이유는 다음과 같습니다.

- 열 누락
- 호환되지 않는 데이터 유형
- 데이터 불일치
- 누락된 테이블

문제가 인프라 문제와 관련된 경우:

- 네트워크 문제 - 호환되지 않는 네트워크 상태의 Amazon RDS DB 관련 문제를 해결하려면 어떻게 해야 하나요?
- 데이터베이스 또는 DB 인스턴스를 사용할 수 없는 이유는 다음과 같습니다.
  - 복제본 인스턴스의 스토리지가 부족함 - 스토리지 추가에 대한 자세한 내용은 Amazon RDS DB instances run out of storage를 검토하세요.
  - 호환되지 않는 파라미터 - 문제를 해결하는 방법에 대한 자세한 내용은 호환되지 않는 파라미터 상태에서 멈춘 Amazon RDS DB 인스턴스를 해결하려면 어떻게 해야 하나요?를 검토하세요.

인스턴스가 AWS 네트워크 외부 또는 AWS EC2에 있는 경우 관리자에게 가용성 또는 인프라 관련 문제를 해결하는 방법을 문의하세요.

비활성 슬롯 삭제

> **Warning**
>
> 주의: 복제 슬롯을 삭제하기 전에 복제가 진행 중이지 않고 비활성 상태이며 복구할 수 없는 상태인지 주의 깊게 확인합니다. 슬롯을 성급히 삭제하면 복제가 중단되거나 데이터가 손실될 수 있습니다.

복제 슬롯이 더 이상 필요하지 않음을 확인한 후 autovacuum이 계속되도록 슬롯을 삭제합니다. `active = 'f'` 조건은 비활성 슬롯만 삭제되도록 합니다.

```
SELECT pg_drop_replication_slot('slot1') WHERE active ='f'
```

읽기 전용 복제본

Amazon RDS 읽기 전용 복제본에 대해 `hot_standby_feedback` 설정이 활성화된 경우 기본 데이터베이스의 autovacuum이 읽기 전용 복제본에서 실행되는 쿼리에 여전히 필요할 수 있는 데드 행을 제거하는 것을 방지합니다. 이는 복제 슬롯을 사용하거나 사용하지 않고 관리되는 복제본을 포함하여 모

든 유형의 물리적 읽기 전용 복제본에 영향을 미칩니다. 대기 복제본에서 실행되는 쿼리가 쿼리 충돌 및 취소를 방지하면서 기본에서 해당 행을 계속 사용 가능하도록 요구하기 때문에 이 동작이 필요합니다.

물리적 복제 슬롯을 사용하는 읽기 전용 복제본

물리적 복제 슬롯을 사용하는 읽기 전용 복제본은 RDS for PostgreSQL에서 복제의 신뢰성과 안정성을 크게 향상시킵니다. 이러한 슬롯은 복제본이 처리할 때까지 기본 데이터베이스가 필수 Write-Ahead Log 파일을 유지하여 네트워크 중단 중에도 데이터 일관성을 유지하도록 합니다.

RDS for PostgreSQL 버전 14부터 모든 복제본은 복제 슬롯을 사용합니다. 이전 버전에서는 교차 리전 복제본만 복제 슬롯을 사용했습니다.

`postgres_get_av_diag()` 함수는 물리적 복제 슬롯을 사용하는 읽기 전용 복제본을 블로커로 찾았을 때 다음과 유사한 출력을 표시합니다.

```
blocker               | Read replica with physical replication slot
database              |
blocker_identifier    | rds_us_west_2_db_xxxxxxxxxxxxxxxxxxxx
wait_event            | Not applicable
autovacuum_lagging_by | 554,080,689
suggestion            | Run the following query on the replica
 "rds_us_west_2_db_xxxxxxxxxxxxxxxxxxx" to find the long running query:

                      | SELECT * FROM pg_catalog.pg_stat_activity WHERE
 backend_xmin::text::bigint = 757989377;

                      | Review carefully and you may consdier terminating the query on
 read replica using suggested_action. For more information, see Working with PostgreSQL
 autovacuum in the Amazon RDS User Guide.                                    +
          |
suggested_action      | {"SELECT pg_terminate_backend(pid) FROM
 pg_catalog.pg_stat_activity WHERE backend_xmin::text::bigint = 757989377;","
                                                                             +
                      | [OR]

                                                +
                      | ","Disable hot_standby_feedback","

                                                +
                      | [OR]

                                                +
```

```
                      | ","Delete the read replica if not needed"}
```

스트리밍 복제를 사용하는 읽기 전용 복제본

Amazon RDS를 사용하면 버전 13까지 이전 버전에서 물리적 복제 슬롯 없이 읽기 전용 복제본을 설정할 수 있습니다. 이 접근 방식은 기본이 WAL 파일을 더 공격적으로 재활용할 수 있도록 하여 오버헤드를 줄입니다. 이는 디스크 공간이 제한되고 가끔 ReplicaLag를 허용할 수 있는 환경에서 유용합니다. 그러나 슬롯이 없으면 WAL 파일이 누락되지 않도록 대기가 동기화된 상태로 유지되어야 합니다. Amazon RDS는 아카이브된 WAL 파일을 사용하여 지연될 경우 복제본이 따라잡는 데 도움이 되지만, 이 프로세스는 주의 깊은 모니터링이 필요하며 느릴 수 있습니다.

`postgres_get_av_diag()` 함수는 스트리밍 읽기 전용 복제본을 블로커로 찾았을 때 다음과 유사한 출력을 표시합니다.

```
blocker               | Read replica with streaming replication slot
database              | Not applicable
blocker_identifier    | xx.x.x.xxx/xx
wait_event            | Not applicable
autovacuum_lagging_by | 610,146,760
suggestion            | Run the following query on the replica "xx.x.x.xxx" to find the
 long running query:
                                                                                +
                      | SELECT * FROM pg_catalog.pg_stat_activity WHERE
 backend_xmin::text::bigint = 348319343;

               +
                      | Review carefully and you may consdier terminating the query on
 read replica using suggested_action. For more information, see Working with PostgreSQL
 autovacuum in the Amazon RDS User Guide.                                       +
                      |
suggested_action      | {"SELECT pg_terminate_backend(pid) FROM
 pg_catalog.pg_stat_activity WHERE backend_xmin::text::bigint = 348319343;","

                       +
                      | [OR]

                                                                                +
                      | ","Disable hot_standby_feedback","

                                                                                +
```

```
                        | [OR]

                                                                                +

                        | ","Delete the read replica if not needed"}
```

권장 조치

suggested_action 열에서 권장하는 대로 이러한 옵션을 주의 깊게 검토하여 autovacuum 차단을 해제합니다.

- 쿼리 종료 - 제안 열의 지침에 따라 suggested_action 열에 지정된 대로 읽기 전용 복제본에 연결할 수 있습니다. 세션을 종료하는 옵션을 주의 깊게 검토하는 것이 좋습니다. 종료가 안전한 것으로 간주되는 경우 pg_terminate_backend( ) 함수를 사용하여 세션을 종료할 수 있습니다. 이 작업은 관리자(예: RDS 마스터 계정) 또는 필요한 pg_terminate_backend() 권한이 있는 사용자가 수행할 수 있습니다.

  읽기 전용 복제본에서 다음 SQL 명령을 실행하여 기본의 vacuum이 오래된 행을 정리하지 못하게 하는 쿼리를 종료할 수 있습니다. backend_xmin의 값은 함수의 출력에 보고됩니다.

  ```
  SELECT
      pg_terminate_backend(pid)
  FROM
      pg_catalog.pg_stat_activity
  WHERE
      backend_xmin::text::bigint = backend_xmin;
  ```

- 상시 대기 피드백 비활성화 - hot_standby_feedback 파라미터가 심각한 vacuum 지연을 일으키는 경우 해당 파라미터를 비활성화하는 것을 고려합니다.

  hot_standby_feedback 파라미터를 사용하면 읽기 전용 복제본이 기본에 쿼리 활동을 알려 기본이 대기에서 사용 중인 테이블이나 행을 정리하지 못하도록 할 수 있습니다. 이렇게 하면 대기의 쿼리 안정성이 보장되지만 기본의 vacuum이 크게 지연될 수 있습니다. 이 기능을 비활성화하면 대기가 따라잡기를 기다리지 않고 기본이 vacuum 작업을 진행할 수 있습니다. 그러나 기본이 정리한 행에 대기가 액세스하려고 하면 대기에서 쿼리 취소 또는 실패가 발생할 수 있습니다.

- 필요하지 않은 경우 읽기 전용 복제본 삭제 - 읽기 전용 복제본이 더 이상 필요하지 않은 경우 삭제할 수 있습니다. 이렇게 하면 연결된 복제 오버헤드가 제거되고 기본이 복제본에 의해 지연되지 않고 트랜잭션 로그를 재활용할 수 있습니다.

임시 테이블

TEMPORARY 키워드를 사용하여 만든 임시 테이블은 pg_temp_xxx와 같은 임시 스키마에 상주하며 해당 테이블을 만든 세션에서만 액세스할 수 있습니다. 세션이 종료되면 임시 테이블이 삭제됩니다. 그러나 이러한 테이블은 PostgreSQL의 autovacuum 프로세스에는 보이지 않으며 테이블을 만든 세션에서 수동으로 vacuum 처리해야 합니다. 다른 세션에서 임시 테이블을 정리하려고 해도 아무런 효과가 없습니다.

비정상적인 상황에서는 임시 테이블이 활성 세션 없이 존재합니다. 치명적인 충돌, 네트워크 문제 또는 유사한 이벤트로 인해 소유 세션이 예기치 않게 종료되는 경우 임시 테이블이 정리되지 않아 '고립된' 테이블로 남겨질 수 있습니다. PostgreSQL autovacuum 프로세스가 고립된 임시 테이블을 감지하면 다음 메시지를 로깅합니다.

```
LOG: autovacuum: found orphan temp table \"%s\".\"%s\" in database \"%s\"
```

postgres_get_av_diag() 함수는 임시 테이블을 블로커로 식별할 때 다음과 유사한 출력을 표시합니다. 함수가 임시 테이블과 관련된 출력을 올바르게 표시하려면 해당 테이블이 있는 것과 동일한 데이터베이스 내에서 실행해야 합니다.

```
blocker               | Temporary table
database              | my_database
blocker_identifier    | pg_temp_14.ttemp
wait_event            | Not applicable
autovacuum_lagging_by | 1,805,802,632
suggestion            | Connect to database "my_database". Review carefully, you
 may consider dropping temporary table using command in suggested_action. For more
 information, see Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"DROP TABLE ttemp;"}
```

권장 조치

출력의 suggestion 열에 제공된 지침에 따라 autovacuum 실행을 막는 임시 테이블을 식별하고 제거합니다. 다음 명령을 사용하여 postgres_get_av_diag()에서 보고한 임시 테이블을 삭제합니다. postgres_get_av_diag() 함수에서 제공하는 출력을 기반으로 테이블 이름을 바꿉니다.

```
DROP TABLE my_temp_schema.my_temp_table;
```

다음 쿼리를 사용하여 임시 테이블을 식별할 수 있습니다.

```
SELECT
```

```
    oid,
    relname,
    relnamespace::regnamespace,
    age(relfrozenxid)
FROM
    pg_class
WHERE
relpersistence = 't'
ORDER BY
    age(relfrozenxid) DESC;
```

RDS for PostgreSQL에서 식별 불가능한 vacuum 블로커 해결

이 섹션에서는 vacuum 작업이 진행되지 않는 다른 이유를 살펴봅니다. 이러한 문제는 현재 `postgres_get_av_diag()` 함수를 사용하여 직접 식별할 수 없습니다.

주제

- 유효하지 않은 페이지
- 인덱스 불일치
- 매우 높은 트랜잭션 속도

유효하지 않은 페이지

유효하지 않은 페이지 오류는 PostgreSQL이 해당 페이지에 액세스하는 동안 페이지의 체크섬에서 불일치를 감지할 때 발생합니다. 내용을 읽을 수 없어 autovacuum이 튜플을 고정시킬 수 없습니다. 이로 인해 정리 프로세스가 중지됩니다. 다음 오류가 PostgreSQL의 로그에 기록됩니다.

```
WARNING:  page verification failed, calculated checksum YYYYY but expected XXXX
ERROR:  invalid page in block ZZZZZ of relation base/XXXXX/XXXXX
CONTEXT:  automatic vacuum of table myschema.mytable
```

객체 유형 확인

```
ERROR: invalid page in block 4305910 of relation base/16403/186752608
WARNING: page verification failed, calculated checksum 50065 but expected 60033
```

오류 메시지에서 `base/16403/186752608` 경로는 다음 정보를 제공합니다.

- 'base'는 PostgreSQL 데이터 디렉터리 아래의 디렉터리 이름입니다.

- '16403'은 pg_database 시스템 카탈로그에서 조회할 수 있는 데이터베이스 OID입니다.
- '186752608'은 pg_class 시스템 카탈로그에서 스키마 및 객체 이름을 조회하는 데 사용할 수 있는 relfilenode입니다.

영향을 받는 데이터베이스에서 다음 쿼리의 출력을 확인하여 객체 유형을 확인할 수 있습니다. 다음 쿼리는 oid가 186752608인 객체 정보를 검색합니다. OID를 발생한 오류와 관련된 OID로 바꿉니다.

```
SELECT
    relname AS object_name,
    relkind AS object_type,
    nspname AS schema_name
FROM
    pg_class c
    JOIN pg_namespace n ON c.relnamespace = n.oid
WHERE
    c.oid = 186752608;
```

자세한 내용은 PostgreSQL 설명서의 pg_class에서 pg_class의 relkind 열에 표시된 지원되는 모든 객체 유형을 참조하세요.

지침

이 문제에 대한 가장 효과적인 솔루션은 특정 Amazon RDS 인스턴스의 구성과 불일치 페이지의 영향을 받는 데이터 유형에 따라 달라집니다.

객체 유형이 인덱스인 경우:

인덱스를 다시 구축하는 것이 좋습니다.

- **CONCURRENTLY** 옵션 사용 - PostgreSQL 버전 12 전에는 인덱스를 다시 구축하려면 테이블에 대한 액세스를 제한하는 배타적인 테이블 잠금이 필요했습니다. PostgreSQL 버전 12 이상에서는 CONCURRENTLY 옵션을 사용하여 행 수준 잠금을 허용하므로 테이블의 가용성이 크게 향상됩니다. 다음은 명령입니다.

  ```
  REINDEX INDEX ix_name CONCURRENTLY;
  ```

  CONCURRENTLY는 중단이 덜 시키지만 사용량이 많은 테이블에서는 속도가 느릴 수 있습니다. 가능하면 트래픽이 적은 기간에 인덱스를 구축하는 것이 좋습니다.

  자세한 내용은 PostgreSQL REINDEX 설명서를 참조하세요.

- **INDEX_CLEANUP FALSE** 옵션 사용 - 인덱스가 크고 완료하는 데 상당한 시간이 필요할 것으로 예상되는 경우 인덱스를 제외하면서 수동 VACUUM FREEZE를 실행하여 autovacuum 차단을 해제할 수 있습니다. 이 기능은 PostgreSQL 버전 12 이상에서 사용할 수 있습니다.

  인덱스를 우회하면 일관되지 않은 인덱스의 vacuum 프로세스를 건너뛰고 랩어라운드 문제를 완화할 수 있습니다. 그러나 이렇게 해도 기본 유효하지 않은 페이지 문제는 해결되지 않습니다. 유효하지 않은 페이지 문제를 완전히 해결하려면 인덱스를 다시 구축해야 합니다.

객체 유형이 구체화된 뷰인 경우:

구체화된 뷰에서 유효하지 않은 페이지 오류가 발생하면 영향을 받는 데이터베이스에 로그인하고 새로 고쳐 유효하지 않은 페이지를 해결합니다.

구체화된 뷰를 새로 고침:

```
REFRESH MATERIALIZED VIEW schema_name.materialized_view_name;
```

새로 고침이 실패하면 다시 만들기 시도:

```
DROP MATERIALIZED VIEW schema_name.materialized_view_name;
CREATE MATERIALIZED VIEW schema_name.materialized_view_name AS query;
```

구체화된 뷰를 새로 고치거나 다시 만들면 기본 테이블 데이터에 영향을 주지 않고 복원됩니다.

다른 모든 객체 유형의 경우:

다른 모든 객체 유형의 경우 AWS Support에 문의하세요.

인덱스 불일치

논리적으로 일치하지 않는 인덱스는 autovacuum이 진행되는 것을 방지할 수 있습니다. 다음 오류 또는 유사한 오류는 인덱스의 vacuum 단계 동안 또는 SQL 문에서 인덱스에 액세스할 때 로깅됩니다.

```
ERROR: right sibling's left-link doesn't match:block 5 links to 10 instead of expected
 2 in index ix_name
```

```
ERROR: failed to re-find parent key in index "XXXXXXXXXX" for deletion target page XXX
CONTEXT:  while vacuuming index index_name of relation schema.table
```

지침

인덱스를 다시 구축하거나 수동 VACUUM FREEZE에서 INDEX_CLEANUP을 사용하여 인덱스를 건너뜁니다. 인덱스를 다시 구축하는 방법에 대한 자세한 내용은 객체 유형이 인덱스인 경우를 참조하세요.

- CONCURRENTLY 옵션 사용 – PostgreSQL 버전 12 전에는 인덱스를 다시 구축하려면 테이블에 대한 액세스를 제한하는 배타적인 테이블 잠금이 필요했습니다. PostgreSQL 버전 12 이상에서는 CONCURRENTLY 옵션을 사용하여 행 수준 잠금을 허용하므로 테이블의 가용성이 크게 향상됩니다. 다음은 명령입니다.

  ```
  REINDEX INDEX ix_name CONCURRENTLY;
  ```

  CONCURRENTLY는 중단이 적지만 사용량이 많은 테이블에서는 속도가 느릴 수 있습니다. 가능하면 트래픽이 적은 기간에 인덱스를 구축하는 것이 좋습니다. 자세한 내용은 PostgreSQL 설명서의 REINDEX를 참조하세요.
- INDEX_CLEANUP FALSE 옵션 사용 - 인덱스가 크고 완료하는 데 상당한 시간이 필요할 것으로 예상되는 경우 인덱스를 제외하면서 수동 VACUUM FREEZE를 실행하여 autovacuum 차단을 해제할 수 있습니다. 이 기능은 PostgreSQL 버전 12 이상에서 사용할 수 있습니다.

  인덱스를 우회하면 일관되지 않은 인덱스의 vacuum 프로세스를 건너뛰고 랩어라운드 문제를 완화할 수 있습니다. 그러나 이렇게 해도 기본 유효하지 않은 페이지 문제는 해결되지 않습니다. 유효하지 않은 페이지 문제를 완전히 해결하려면 인덱스를 다시 구축해야 합니다.

매우 높은 트랜잭션 속도

PostgreSQL에서 높은 트랜잭션 속도는 autovacuum의 성능에 상당한 영향을 미쳐 데드 튜플의 정리가 느려지고 트랜잭션 ID 랩어라운드 위험이 증가할 수 있습니다. 두 기간의 `max(age(datfrozenxid))` 차이(일반적으로 초당)를 측정하여 트랜잭션 속도를 모니터링할 수 있습니다. 또한 RDS 성능 개선 도우미의 다음 카운터 지표를 사용하여 총 트랜잭션 수인 트랜잭션 속도(xact_commit과 xact_rollback의 합계)를 측정할 수 있습니다.

카운터	유형	Unit	지표
xact_commit	트랜잭션	초당 커밋 수	db.Transactions.xact_commit
xact_rollback	트랜잭션	초당 롤백 수	db.Transactions.xact_rollback

빠른 증가는 트랜잭션 로드가 높음을 나타내며, 이는 autovacuum에 부담을 주어 팽창, 잠금 경합 및 잠재적 성능 문제를 일으킬 수 있습니다. 이는 다음과 같은 몇 가지 방법으로 autovacuum 프로세스에 부정적인 영향을 미칠 수 있습니다.

- 테이블 활동: Vacuum 처리 중인 특정 테이블에 대량의 트랜잭션이 발생하여 지연이 발생할 수 있습니다.
- 시스템 리소스: 전체 시스템에 과부하가 발생하여 autovacuum이 효율적으로 작동하는 데 필요한 리소스에 액세스하기 어려울 수 있습니다.

Autovacuum이 더 효과적으로 작동하고 작업을 따라잡을 수 있도록 하려면 다음 전략을 고려하세요.

1. 가능하면 트랜잭션 속도를 줄입니다. 가능한 경우 유사한 트랜잭션을 배치 처리하거나 그룹화하는 것을 고려합니다.
2. 사용량이 적은 시간에 야간, 주간 또는 격주로 수동 `VACUUM FREEZE` 작업을 사용하여 자주 업데이트되는 테이블을 타게팅합니다.
3. 인스턴스 클래스를 스케일 업하여 높은 트랜잭션 볼륨과 autovacuum을 처리할 수 있도록 더 많은 시스템 리소스를 할당하는 것을 고려합니다.

RDS for PostgreSQL의 vacuum 성능 문제 해결

이 섹션에서는 느린 vacuum 성능에 영향을 미치는 요인과 이러한 문제를 해결하는 방법을 설명합니다.

주제

- 큰 인덱스 vacuum
- 테이블 또는 데이터베이스가 너무 많아 vacuum 작업을 할 수 없음
- 공격적 vacuum(랩어라운드 방지용) 실행 중

큰 인덱스 vacuum

VACUUM은 초기화, 힙 스캔, 인덱스 및 힙 vacuum, 인덱스 정리, 힙 잘라내기 및 최종 정리와 같은 순차적 단계를 거칩니다. 힙 스캔 중에 프로세스는 페이지를 정리하고 조각 모음을 하여 고정합니다. 힙 스캔이 완료되면 VACUUM은 인덱스를 정리하고 빈 페이지를 운영 체제로 반환하며 빈 공간 맵 vacuum 및 통계 업데이트와 같은 최종 정리 작업을 수행합니다.

인덱스를 vacuum할 때 `maintenance_work_mem`(또는 `autovacuum_work_mem`)이 인덱스를 처리하기에 충분하지 않은 경우 여러 번 전달해야 할 수 있습니다. PostgreSQL 16 이하에서는 데드 튜플

ID 저장을 위한 1GB 메모리 제한이 있어 큰 인덱스에서는 여러 번의 전달이 강제될 때가 많았습니다. PostgreSQL 17에는 단일 할당 배열을 사용하는 대신 메모리를 동적으로 할당하는 TidStore가 도입되었습니다. 이렇게 하면 1GB 제약이 제거되고, 메모리를 더 효율적으로 사용하며, 인덱스당 여러 인덱스 스캔의 필요성이 줄어듭니다.

사용 가능한 메모리가 전체 인덱스 처리를 한 번에 수용할 수 없는 경우에도 큰 인덱스는 PostgreSQL 17에서 여러 번 전달해야 할 수 있습니다. 일반적으로 큰 인덱스에는 여러 번 전달해야 하는 데드 튜플이 더 많이 포함되어 있습니다.

느린 vacuum 작업 탐지

postgres_get_av_diag() 함수는 메모리 부족으로 인해 vacuum 작업이 느리게 실행되는 시기를 탐지할 수 있습니다. 이 함수에 대한 자세한 내용은 RDS for PostgreSQL에 자동 정리 모니터링 및 진단 도구 설치 단원을 참조하십시오.

사용 가능한 메모리가 단일 패스에서 인덱스 vacuuming을 완료하기에 충분하지 않은 경우 postgres_get_av_diag() 함수는 다음 알림을 발행합니다.

rds_tools 1.8

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound and
 it might be slow.
```

```
NOTICE: The current setting of autovacuum_work_mem is "XXX" and might not be
 sufficient. Consider increasing the setting, and if necessary, scaling up the Amazon
 RDS instance class for more memory.
        Additionally, review the possibility of manual vacuum with exclusion of indexes
 using (VACUUM (INDEX_CLEANUP FALSE, VERBOSE TRUE) table_name;).
```

rds_tools 1.9

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound and
 it might be slow.
```

```
NOTICE: The current setting of autovacuum_work_mem is XX might not be sufficient.
 Consider increasing the setting to XXX, and if necessary, scaling up the RDS instance
 class for more
        memory. The suggested value is an estimate based on the current number of dead
 tuples for the table being vacuumed, which might not fully reflect the latest state.
 Additionally, review the possibility of manual
```

```
        vacuum with exclusion of indexes using (VACUUM (INDEX_CLEANUP FALSE, VERBOSE
TRUE) table_name;). For more information, see
        Working with PostgreSQL autovacuum in the Amazon Amazon RDS User Guide
        .
```

> **Note**
>
> 이 postgres_get_av_diag() 함수는 인덱스 vacuum 작업에 필요한 메모리의 양을 추정하기 위해 pg_stat_all_tables.n_dead_tup에 의존합니다.

postgres_get_av_diag() 함수가 autovacuum_work_mem 부족으로 인해 여러 인덱스 스캔이 필요한 느린 vacuum 작업을 식별하면 다음 메시지가 생성됩니다.

```
NOTICE: Your vacuum is performing multiple index scans due to insufficient
 autovacuum_work_mem:XXX for index vacuuming.
        For more information, see Working with PostgreSQL autovacuum in the Amazon
 Amazon RDS User Guide.
```

지침

수동 VACUUM FREEZE를 사용해 다음 해결 방법을 적용하여 테이블 고정의 속도를 높일 수 있습니다.

Vacuum 작업을 위한 메모리 증가

postgres_get_av_diag() 함수에서 제안한 대로 인스턴스 수준에서 잠재적 메모리 제약을 해결하기 위해 autovacuum_work_mem 파라미터를 늘리는 것이 좋습니다. autovacuum_work_mem은 동적 파라미터이지만 새 메모리 설정이 적용되려면 autovacuum 대몬이 작업자를 다시 시작해야 합니다. 이를 수행하는 방법은 다음과 같습니다.

1. 새 설정이 있는지 확인합니다.
2. 현재 autovacuum을 실행 중인 프로세스를 종료합니다.

이 접근 방식을 사용하면 조정된 메모리 할당이 새 autovacuum 작업에 적용됩니다.

보다 즉각적인 결과를 얻으려면 세션 내에서 maintenance_work_mem 설정이 증가된 VACUUM FREEZE 작업을 수동으로 수행하는 것을 고려하세요.

```
SET maintenance_work_mem TO '1GB';
VACUUM FREEZE VERBOSE table_name;
```

Amazon RDS를 사용하고 있고 maintenance_work_mem 또는 autovacuum_work_mem에 대해 더 높은 값을 지원하는 데 추가 메모리가 필요한 경우 메모리가 더 많은 인스턴스 클래스로 업그레이드하는 것을 고려해 보세요. 이를 통해 수동 및 자동 vacuum 작업을 모두 개선하는 데 필요한 리소스를 제공하여 전반적인 vacuum 및 데이터베이스 성능을 개선할 수 있습니다.

INDEX_CLEANUP 비활성화

PostgreSQL 버전 12 이상의 수동 VACUUM에서는 인덱스 정리 단계를 건너뛸 수 있고, PostgreSQL 버전 14 이상의 긴급 autovacuum에서는 vacuum_failsafe_age 파라미터를 기반으로 자동으로 이 작업을 수행합니다.

> **Warning**
>
> 인덱스 정리를 건너뛰면 인덱스 팽창이 발생하여 쿼리 성능에 부정적인 영향을 미칠 수 있습니다. 이를 완화하려면 영향을 받는 인덱스를 유지 관리 기간 동안 다시 인덱싱하거나 vacuum하는 것이 좋습니다.

큰 인덱스 처리에 대한 추가 지침은 대용량 인덱스를 사용하여 autovacuum 관리 의 설명서를 참조하세요.

병렬 인덱스 vacuum 작업

PostgreSQL 13부터 인덱스는 각 인덱스에 하나의 vacuum 작업자 프로세스가 할당된 수동 VACUUM을 사용하여 기본적으로 병렬로 vacuum 및 정리가 가능합니다. 그러나 vacuum 작업이 병렬 실행에 적합한지 PostgreSQL이 확인하려면 다음과 같은 특정 기준을 충족해야 합니다.

- 인덱스가 최소 2개 있어야 합니다.
- max_parallel_maintenance_workers 파라미터 값을 최소 2로 설정합니다.
- 인덱스 크기는 min_parallel_index_scan_size 한도를 초과해야 하며, 기본값은 512KB입니다.

Amazon RDS 인스턴스에서 사용할 수 vCPU의 수와 테이블의 인덱스 수에 따라 max_parallel_maintenance_workers 설정을 조정하여 vacuum 작업 처리 시간을 최적화할 수 있습니다.

자세한 내용은 Parallel vacuuming in Amazon RDS for PostgreSQL and Amazon Aurora PostgreSQL을 참조하세요.

테이블 또는 데이터베이스가 너무 많아 vacuum 작업을 할 수 없음

PostgreSQL의 The Autovacuum Daemon 설명서에서 언급한 것처럼 autovacuum 대몬은 여러 프로세스를 통해 작동합니다. 여기에는 시스템 내 각 데이터베이스에 대해 autovacuum 작업자 프로세스를 시작하는 영구 autovacuum 런처가 포함됩니다. 런처는 데이터베이스당 약 `autovacuum_naptime`초마다 이러한 작업자가 시작하도록 일정을 예약합니다.

데이터베이스가 'N'개이면 새 작업자가 대략 [`autovacuum_naptime`/N초]마다 시작합니다. 그러나 총 동시 작업자 수는 `autovacuum_max_workers` 설정에 따라 제한됩니다. Vacuum 작업이 필요한 데이터베이스 또는 테이블 수가 이 한도를 초과하는 경우 작업자를 사용할 수 있게 되는 즉시 다음 데이터베이스 또는 테이블이 처리됩니다.

많은 대형 테이블 또는 데이터베이스에서 동시에 vacuum 작업이 필요한 경우 사용 가능한 모든 autovacuum 작업자가 장기간 점유되어 다른 테이블 및 데이터베이스에 대한 유지 관리가 지연될 수 있습니다. 트랜잭션 속도가 높은 환경에서는 이 병목 현상이 빠르게 에스컬레이션되어 Amazon RDS 인스턴스 내에서 vacuum 랩어라운드 문제가 발생할 수 있습니다.

`postgres_get_av_diag()`가 많은 수의 테이블 또는 데이터베이스를 감지하면 다음과 같은 권장 사항을 제공합니다.

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound and
 it might be slow.
```

```
NOTICE: The current setting of autovacuum_max_workers:3 might not be sufficient.
 Consider increasing the setting and, if necessary, consider scaling up the Amazon RDS
 instance class for more workers.
```

지침

autovacuum_max_workers 증가

Vacuum 작업의 속도를 높이기 위해 더 많은 동시 autovacuum 작업자를 허용하도록 `autovacuum_max_workers` 파라미터를 조정하는 것이 좋습니다. 성능 병목 현상이 지속되면 Amazon RDS 인스턴스를 더 많은 vCPU가 있는 클래스로 스케일 업하는 것이 좋습니다. 이렇게 하면 병렬 처리 기능을 더욱 개선할 수 있습니다.

공격적 vacuum(랩어라운드 방지용) 실행 중

PostgreSQL의 데이터베이스(MaximumUsedTransactionIDs) 연령은 공격적 vacuum(랩어라운드 방지용)이 성공적으로 완료될 때만 감소합니다. 이 vacuum 작업이 완료될 때까지 트랜잭션 속도에 따라 연령이 계속 증가합니다.

`postgres_get_av_diag()` 함수는 공격적인 vacuum을 감지하면 다음 `NOTICE`를 생성합니다. 그러나 vacuum의 활성 상태가 최소 2분간 지속된 후에만 이 출력이 트리거됩니다.

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound,
 monitor autovacuum performance.
```

공격적 vacuum에 대한 자세한 내용은 When an aggressive vacuum is already running 섹션을 참조하세요.

다음 쿼리를 사용하여 공격적인 vacuum이 진행 중인지 확인할 수 있습니다.

```
SELECT
    a.xact_start AS start_time,
    v.datname "database",
    a.query,
    a.wait_event,
    v.pid,
    v.phase,
    v.relid::regclass,
    pg_size_pretty(pg_relation_size(v.relid)) AS heap_size,
    (
        SELECT
            string_agg(pg_size_pretty(pg_relation_size(i.indexrelid)) || ':' ||
 i.indexrelid::regclass || chr(10), ', ')
        FROM
            pg_index i
        WHERE
            i.indrelid = v.relid
    ) AS index_sizes,
    trunc(v.heap_blks_scanned * 100 / NULLIF(v.heap_blks_total, 0)) AS step1_scan_pct,
    v.index_vacuum_count || '/' || (
        SELECT
            count(*)
        FROM
            pg_index i
        WHERE
            i.indrelid = v.relid
    ) AS step2_vacuum_indexes,
    trunc(v.heap_blks_vacuumed * 100 / NULLIF(v.heap_blks_total, 0)) AS
 step3_vacuum_pct,
    age(CURRENT_TIMESTAMP, a.xact_start) AS total_time_spent_sofar
FROM
    pg_stat_activity a
```

```
    INNER JOIN pg_stat_progress_vacuum v ON v.pid = a.pid;
```

출력의 쿼리 열을 확인하여 공격적인 vacuum(랩어라운드 방지용)인지 확인할 수 있습니다. '랩어라운드 방지용'이라는 문구는 공격적인 vacuum임을 나타냅니다.

```
query                 | autovacuum: VACUUM public.t3 (to prevent wraparound)
```

예를 들어, 트랜잭션 연령이 10억인 블로커가 있고 동일한 트랜잭션 연령에서 랩어라운드 방지용으로 공격적인 vacuum이 필요한 테이블이 있다고 가정해 보겠습니다. 또한 트랜잭션 연령이 7억 5,000만인 또 다른 블로커가 있습니다. 트랜잭션 연령이 1억인 블로커를 정리한 후에 트랜잭션 연령이 7억 5,000만으로 즉시 줄어들지 않습니다. 공격적인 vacuum이 필요한 테이블 또는 연령이 7억 5,000만이 넘는 트랜잭션이 완료될 때까지 트랜잭션 연령이 높게 유지됩니다. 이 기간 동안 PostgreSQL 클러스터의 트랜잭션 연령이 계속 증가합니다. Vacuum 프로세스가 완료되면 트랜잭션 연령이 7억 5,000만으로 감소하지만 추가 vacuum 작업이 완료될 때까지 다시 증가하기 시작합니다. 이 주기는 이러한 조건이 지속되는 한, 트랜잭션 연령이 결국 `autovacuum_freeze_max_age`에서 지정된 Amazon RDS 인스턴스에 대해 구성된 수준으로 떨어질 때까지 계속됩니다.

RDS for PostgreSQL의 NOTICE 메시지 설명

`postgres_get_av_diag()` 함수는 다음 NOTICE 메시지를 제공합니다.

연령이 모니터링 임곗값에 아직 도달하지 않은 경우

`postgres_get_av_diag()`가 블로커를 식별하기 위한 모니터링 임곗값은 기본적으로 5억 개의 트랜잭션입니다. `postgres_get_av_diag()`가 다음 NOTICE를 생성하면 트랜잭션 연령이 아직 이 임곗값에 도달하지 않았음을 나타냅니다.

```
NOTICE: postgres_get_av_diag() checks for blockers that prevent aggressive vacuums
 only, it does so only after exceeding dvb_threshold which is 500,000,000 and age of
 this PostgreSQL cluster is currently at 2.
```

연령이 가장 많은 트랜잭션 ID를 가진 데이터베이스에 연결되지 않음

`postgres_get_av_diag()` 함수는 연령이 가장 많은 트랜잭션 ID를 가진 데이터베이스에 연결할 때 가장 정확한 출력을 제공합니다. `postgres_get_av_diag()`에서 보고한 연령이 가장 많은 트랜잭션 ID를 가진 데이터베이스는 사용자의 경우 'my_database'와 다릅니다. 올바른 데이터베이스에 연결되지 않은 경우 다 NOTICE가 생성됩니다.

```
NOTICE: You are not connected to the database with the age of oldest transaction
 ID. Connect to my_database database and run postgres_get_av_diag() for accurate
 reporting.
```

연령이 가장 많은 트랜잭션을 가진 데이터베이스에 연결하는 것은 다음과 같은 이유로 중요합니다.

- 임시 테이블 블로커 식별: 임시 테이블의 메타데이터는 각 데이터베이스마다 다르기 때문에 일반적으로 테이블이 만들어진 데이터베이스에서 찾을 수 있습니다. 그러나 임시 테이블이 최상위 블로커이고 수명이 가장 많은 트랜잭션을 가진 데이터베이스에 있는 경우 이는 오해의 소지가 있을 수 있습니다. 올바른 데이터베이스에 연결하면 임시 테이블 블로커를 정확하게 식별할 수 있습니다.
- 느린 vacuum 진단: 인덱스 메타데이터 및 테이블 수 정보는 데이터베이스별로 다르며 느린 vacuum 문제를 진단하는 데 필요합니다.

연령을 기준으로 가장 오래된 트랜잭션을 가진 데이터베이스가 rdsadmin 또는 template0 데이터베이스임

경우에 따라 `rdsadmin` 또는 `template0` 데이터베이스가 연령이 가장 많은 트랜잭션 ID를 가진 데이터베이스로 식별될 수 있습니다. 이 경우 `postgres_get_av_diag()`는 다음 NOTICE를 발행합니다.

```
NOTICE: The database with the age of oldest transaction ID is rdsadmin or template0,
 reach out to support if the reported blocker is in rdsadmin or template0.
```

나열된 블로커가 이 두 데이터베이스 중 하나에서 시작되지 않는지 확인합니다. `rdsadmin` 또는 `template0`에 블로커가 있는 것으로 보고되는 경우 이러한 데이터베이스는 사용자가 액세스할 수 없으며 개입이 필요하므로 지원팀에 문의하세요.

`rdsadmin` 또는 `template0` 데이터베이스가 최상위 블로커를 포함할 가능성은 매우 낮습니다.

공격적 vacuum이 이미 실행 중인 경우

`postgres_get_av_diag()` 함수는 공격적인 vacuum 프로세스가 실행 중일 때 보고하도록 설계되었지만, vacuum이 최소 1분 동안 활성 상태인 경우에만 이 출력을 트리거합니다. 이러한 의도적인 지연은 오탐지의 가능성을 줄이는 데 도움이 됩니다. 이 함수는 대기함으로써 효과적이고 중요한 vacuum만 보고되도록 하므로 vacuum 활동을 보다 정확하고 안정적으로 모니터링할 수 있습니다.

`postgres_get_av_diag()` 함수는 진행 중인 공격적인 vacuum이 하나 이상인 것을 감지하면 다음 NOTICE를 생성합니다.

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound,
 monitor autovacuum performance.
```

NOTICE에 표시된 대로 vacuum의 성능을 계속 모니터링합니다. 공격적 vacuum에 대한 자세한 내용은 공격적 vacuum(랩어라운드 방지용) 실행 중 섹션을 참조하세요.

Autovacuum이 꺼져 있는 경우

데이터베이스 인스턴스에서 autovacuum이 비활성화된 경우 `postgres_get_av_diag()` 함수는 다음 NOTICE를 생성합니다.

```
NOTICE: Autovacuum is OFF, we strongly recommend to enable it, no restart is
 necessary.
```

자동 정리는 원활한 데이터베이스 작업을 보장하는 RDS for PostgreSQL DB 인스턴스의 중요한 기능입니다. 오래된 행 버전을 자동으로 제거하고, 스토리지 공간을 회수하고, 테이블 팽창을 방지하여 테이블과 인덱스를 효율적으로 유지함으로써 최적의 성능을 발휘하도록 합니다. 또한 Amazon RDS 인스턴스에서 트랜잭션을 중지할 수 있는 트랜잭션 ID 랩어라운드를 방지합니다. Autovacuum을 비활성화하면 데이터베이스 성능과 안정성이 장기적으로 저하될 수 있습니다. 항상 켜두는 것이 좋습니다. 자세한 내용은 Understanding autovacuum in RDS for PostgreSQL environments를 참조하세요.

> **Note**
>
> Autovacuum을 끄더라도 공격적인 vacuum이 중지되지 않습니다. 테이블이 `autovacuum_freeze_max_age` 임곗값에 도달하면 여전히 공격적 vacuum이 발생합니다.

남은 트랜잭션 수가 매우 적음

`postgres_get_av_diag()` 함수는 랩어라운드 vacuum이 임박하면 다음 NOTICE를 생성합니다. 이 NOTICE는 Amazon RDS 인스턴스가 잠재적으로 새 트랜잭션을 거부하기까지 1억 개의 트랜잭션이 남았을 때 발행됩니다.

```
WARNING: Number of transactions remaining is critically low, resolve issues with
 autovacuum or perform manual VACUUM FREEZE before your instance stops accepting
 transactions.
```

데이터베이스 가동 중지 시간을 방지하려면 즉각적인 조치가 필요합니다. Vacuum 작업을 면밀히 모니터링하고 영향을 받는 데이터베이스에서 VACUUM FREEZE를 수동으로 시작하여 트랜잭션 실패를 방지해야 합니다.

RDS for PostgreSQL에서 지원되는 로깅 메커니즘 작업

PostgreSQL DB 인스턴스에 발생하는 기록 활동을 설정할 수 있는 파라미터, 확장 및 기타 구성 항목이 몇 가지 있습니다. 여기에는 다음이 포함됩니다.

- log_statement 파라미터는 PostgreSQL 데이터베이스에서 사용자 활동 로그에 사용할 수 있습니다. RDS for PostgreSQL 로깅 및 로그를 모니터링하는 방법에 대한 자세한 내용은 RDS for PostgreSQL 데이터베이스 로그 파일 섹션을 참조하세요.
- rds.force_admin_logging_level 파라미터는 Amazon RDS 내부 사용자(rdsadmin)가 DB 인스턴스의 데이터베이스에서 수행한 작업을 기록합니다. PostgreSQL 오류 로그에 출력을 기록합니다. 허용되는 값은 disabled, debug5, debug4, debug3, debug2, debug1, info, notice, warning, error, 로그, fatal 및 panic입니다. 기본값은 disabled입니다.
- PostgreSQL 오류 로그에서 다양한 autovacuum 작업을 캡처하도록 rds.force_autovacuum_logging_level 파라미터를 설정할 수 있습니다. 자세한 내용은 autovacuum 및 vacuum 활동 로그 섹션을 참조하세요.
- PostgreSQL Audit(pgAudit) 확장 프로그램은 세션 수준 또는 객체 수준에서 활동을 캡처하도록 설치 및 구성할 수 있습니다. 자세한 내용은 pgAudit를 사용하여 데이터베이스 활동 로깅 섹션을 참조하세요.
- log_fdw 확장을 통해 SQL을 사용하여 데이터베이스 엔진 로그에 액세스할 수 있습니다. 자세한 정보는 log_fdw 확장으로 SQL을 사용하여 DB 로그에 액세스의 내용을 참조하세요.
- pg_stat_statements 라이브러리가 RDS for PostgreSQL 버전 10 이상의 shared_preload_libraries 파라미터에 대한 기본값으로 지정됩니다. 실행 중인 쿼리를 분석할 때 사용할 수 있는 라이브러리입니다. pg_stat_statements가 DB 파라미터 그룹에 설정되는 것을 확인하세요. 이 라이브러리에서 제공하는 정보를 사용하여 RDS for PostgreSQL DB 인스턴스를 모니터링하는 방법에 대한 자세한 내용은 RDS PostgreSQL에 대한 SQL 통계 섹션을 참조하세요.
- log_hostname 파라미터는 각 클라이언트 연결의 호스트 이름을 로그에 캡처합니다. RDS for PostgreSQL 버전 12 이상에서 이 파라미터는 기본적으로 off로 설정됩니다. 이 파라미터를 켰다면 세션 연결 시간을 모니터링해야 합니다. 이 파라미터를 켜면 서비스는 도메인 이름 시스템(DNS) 역방향 조회 요청을 사용하여, 연결할 클라이언트의 호스트 이름을 얻은 다음 PostgreSQL 로그에 추가합니다. 이렇게 되면 세션 연결 중에 상당한 영향이 발생합니다. 이 파라미터는 문제를 해결해야 할 때만 켜는 것이 좋습니다.

일반적으로 로깅의 목적은 DBA가 성능을 모니터링하고 튜닝하고 문제를 해결할 수 있도록 하는 것입니다. 대부분의 로그가 Amazon CloudWatch 또는 성능 개선 도우미로 자동 업로드됩니다. 여기서 로그는 DB 인스턴스에 대한 전체 지표를 제공하기 위해 정렬되고 그룹화됩니다. Amazon RDS 모니터링 및 지표에 대한 자세한 내용은 Amazon RDS 인스턴스에서 지표 모니터링 섹션을 참조하세요.

PostgreSQL을 사용한 임시 파일 관리

PostgreSQL에서 복잡한 쿼리는 여러 가지 정렬 또는 해시 연산을 동시에 수행할 수 있으며, 각 연산은 인스턴스 메모리를 사용하여 `work_mem` 파라미터에 지정된 값까지 결과를 저장합니다. 인스턴스 메모리가 충분하지 않은 경우 결과를 저장하기 위한 임시 파일이 생성됩니다. 이러한 임시 파일은 쿼리 실행을 완료하기 위해 디스크에 기록됩니다. 나중에 이러한 파일은 쿼리가 완료된 후 자동으로 제거됩니다. RDS for PostgreSQL에서는 이러한 파일이 데이터 볼륨의 Amazon EBS에 저장됩니다. 자세한 내용은 Amazon RDS DB 인스턴스 스토리지를 참조하세요. CloudWatch에 게시되는 `FreeStorageSpace` 지표를 계속 모니터링하여 DB 인스턴스에 충분한 스토리지 여유 공간이 있는지 확인합니다. 자세한 내용은 `FreeStorageSpace` 단원을 참조하세요.

임시 파일 사용량을 늘리는 다중 동시 쿼리가 포함된 워크로드에서는 Amazon RDS 최적화된 읽기 인스턴스를 사용하는 것이 좋습니다. 이러한 인스턴스는 SSD(Solid State Drive) 블록 스토리지에 기반한 NVMe(Non-Volatile Memory Express)를 사용하여 임시 파일을 배치할 수 있습니다. 자세한 내용은 Amazon RDS Optimized Reads로 RDS for PostgreSQL 쿼리 성능 개선 단원을 참조하십시오.

다음과 같은 파라미터와 함수를 사용하여 인스턴스의 임시 파일을 관리할 수 있습니다.

- **`temp_file_limit`** - 이 파라미터는 temp_files 크기(KB)를 초과하는 모든 쿼리를 취소합니다. 이러한 한도는 쿼리가 무한으로 실행되어 임시 파일이 디스크 공간을 사용하는 현상을 방지합니다. `log_temp_files` 파라미터의 결과를 사용하여 값을 추정할 수 있습니다. 가장 좋은 방법은 워크로드 동작을 검사하고 추정치에 따라 한도를 설정하는 것입니다. 다음 예는 한도를 초과했을 때 쿼리가 취소되는 방식을 보여줍니다.

```
postgres=>select * from pgbench_accounts, pg_class, big_table;
```

```
ERROR: temporary file size exceeds temp_file_limit (64kB)
```

- **`log_temp_files`** - 이 파라미터는 세션의 임시 파일이 제거될 경우 postgresql.log로 메시지를 보냅니다. 이 파라미터는 쿼리가 성공적으로 완료된 후 로그를 생성합니다. 따라서 활성 상태의 장기 실행 쿼리의 문제를 해결하는 데에는 도움이 되지 않을 수도 있습니다.

다음 예시에서는 쿼리가 성공적으로 완료되었을 때 임시 파일이 정리되는 동안 항목이 postgresql.log 파일에 기록되는 것을 보여줍니다.

```
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:LOG:
 temporary file: path "base/pgsql_tmp/pgsql_tmp31236.5", size 140353536
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:STATEMENT:
  select a.aid from pgbench_accounts a, pgbench_accounts b where a.bid=b.bid order by
 a.bid limit 10;
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:LOG:
 temporary file: path "base/pgsql_tmp/pgsql_tmp31236.4", size 180428800
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:STATEMENT:
  select a.aid from pgbench_accounts a, pgbench_accounts b where a.bid=b.bid order by
 a.bid limit 10;
```

- **pg_ls_tmpdir** - RDS for PostgreSQL 13 이상에서 제공되는 이 함수를 사용하면 현재 임시 파일 사용량을 확인할 수 있습니다. 완료된 쿼리는 함수 결과에 나타나지 않습니다. 다음 예제에서는 이 함수의 결과를 볼 수 있습니다.

```
postgres=>select * from pg_ls_tmpdir();
```

```
      name       |    size    |      modification
-----------------+------------+------------------------
 pgsql_tmp8355.1 | 1072250880 | 2023-02-06 22:54:56+00
 pgsql_tmp8351.0 | 1072250880 | 2023-02-06 22:54:43+00
 pgsql_tmp8327.0 | 1072250880 | 2023-02-06 22:54:56+00
 pgsql_tmp8351.1 |  703168512 | 2023-02-06 22:54:56+00
 pgsql_tmp8355.0 | 1072250880 | 2023-02-06 22:54:00+00
 pgsql_tmp8328.1 |  835031040 | 2023-02-06 22:54:56+00
 pgsql_tmp8328.0 | 1072250880 | 2023-02-06 22:54:40+00
(7 rows)
```

```
postgres=>select query from pg_stat_activity where pid = 8355;

query
--------------------------------------------------------------------------------
```

```
select a.aid from pgbench_accounts a, pgbench_accounts b where a.bid=b.bid order by
 a.bid
(1 row)
```

파일 이름에는 임시 파일을 생성한 세션의 처리 ID(PID)가 포함됩니다. 다음 예와 같은 고급 쿼리는 각 PID에 대한 임시 파일의 합계를 수행합니다.

```
postgres=>select replace(left(name, strpos(name, '.')-1),'pgsql_tmp','') as pid,
 count(*), sum(size) from pg_ls_tmpdir() group by pid;
```

```
 pid  | count |   sum
------+-------+------------------
 8355 |     2 | 2144501760
 8351 |     2 | 2090770432
 8327 |     1 | 1072250880
 8328 |     2 | 2144501760
(4 rows)
```

- **pg_stat_statements** - pg_stat_statements 파라미터를 활성화하면 호출당 평균 임시 파일 사용량을 볼 수 있습니다. 쿼리의 query_id를 식별하고 이를 사용하여 다음 예와 같이 임시 파일 사용량을 검사할 수 있습니다.

```
postgres=>select queryid from pg_stat_statements where query like 'select a.aid from
 pgbench%';
```

```
       queryid
----------------------
 -7170349228837045701
(1 row)
```

```
postgres=>select queryid, substr(query,1,25), calls, temp_blks_read/calls
 temp_blks_read_per_call, temp_blks_written/calls temp_blks_written_per_call from
 pg_stat_statements where queryid = -7170349228837045701;
```

```
       queryid        |          substr           | calls | temp_blks_read_per_call |
 temp_blks_written_per_call
----------------------+---------------------------+-------+-------------------------
+---------------------------
 -7170349228837045701 | select a.aid from pgbench |    50 |                  239226 |
                    388678
(1 row)
```

- **Performance Insights** - 성능 개선 도우미 대시보드에서 temp_bytes 및 temp_files 메트릭을 켜서 임시 파일 사용량을 확인할 수 있습니다. 그런 다음, 이 두 지표의 평균을 확인하고 두 지표가 쿼리 워크로드에 어떻게 상응하는지 확인할 수 있습니다. 성능 개선 도우미 내부의 보기에는 임시 파일을 생성 중인 쿼리가 구체적으로 표시되지 않습니다. 그러나 성능 개선 도우미를 `pg_ls_tmpdir`에 대해 표시된 쿼리와 결합하면 쿼리 워크로드의 문제를 해결하고, 분석하고, 변경 사항을 확인할 수 있습니다.

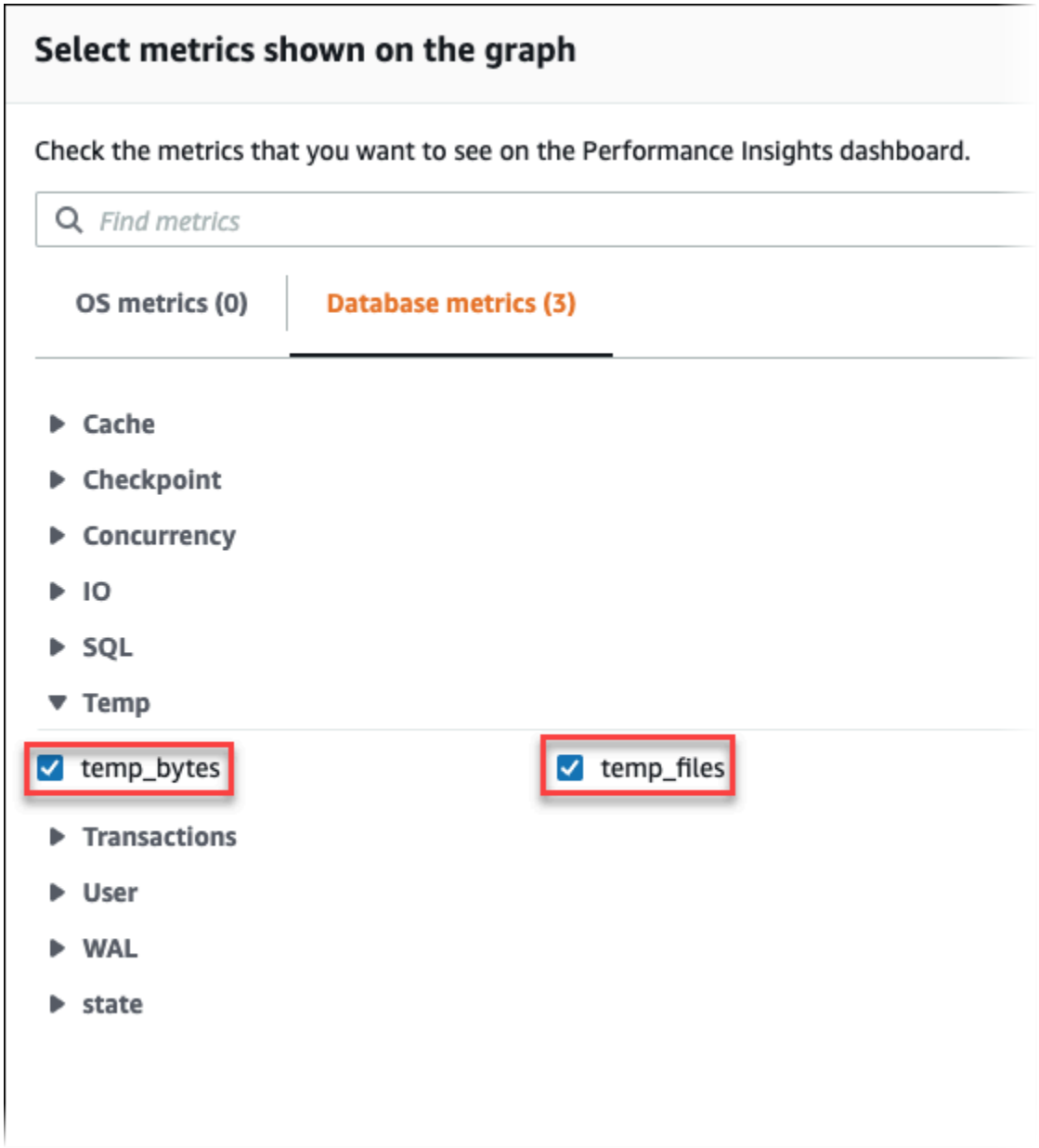
  성능 개선 도우미를 사용하여 지표를 분석하고 쿼리를 분석하는 방법에 대한 자세한 내용은 성능 개선 도우미 대시보드를 사용한 지표 분석 섹션을 참조하세요.

  성능 개선 도우미를 사용하여 임시 파일 사용량을 보는 예제는 성능 개선 도우미를 사용하여 임시 파일 사용량 확인 섹션을 참조하세요.

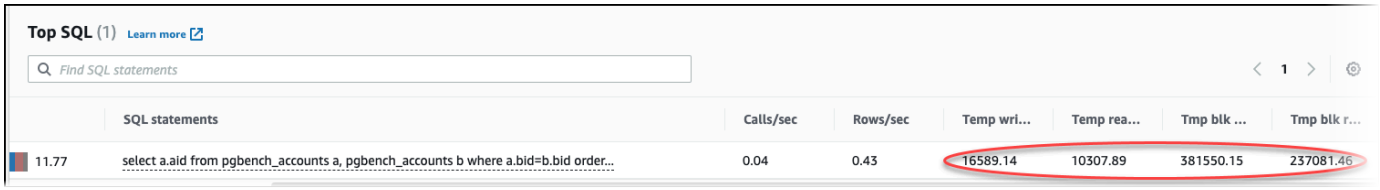
성능 개선 도우미를 사용하여 임시 파일 사용량 확인

성능 개선 도우미를 통해 지표 temp_bytes 및 temp_files를 설정하여 임시 파일 사용량을 볼 수 있습니다. 성능 개선 도우미 보기에는 임시 파일을 생성하는 특정 쿼리가 표시되지 않지만, `pg_ls_tmpdir`에 대해 표시된 쿼리와 성능 개선 도우미를 결합하면 쿼리 워크로드를 분석하고, 변경 사항을 확인하고, 문제를 해결할 수 있습니다.

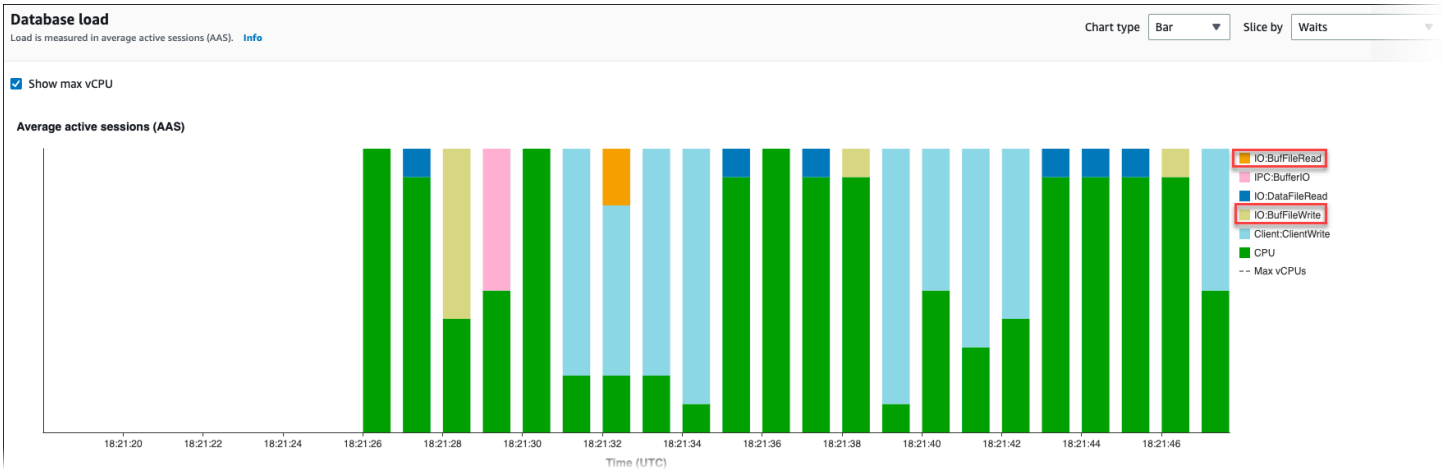
1. 성능 개선 도우미 대시보드에서 지표 관리를 선택합니다.
2. 다음 이미지에 나와 있는 것처럼 데이터베이스 지표를 선택하고, temp_bytes 및 temp_files를 선택합니다.



3. 상위 SQL 탭에서 기본 설정 아이콘을 선택합니다.
4. 기본 설정 창에서 상위 SQL 탭에 표시할 다음 통계를 켜고 계속을 선택합니다.
   - 임시 쓰기/초
   - 임시 읽기/초
   - 임시 대량 쓰기/호출
   - 임시 대량 읽기/호출
5. 다음 예제에 나온 것처럼, 임시 파일은 `pg_ls_tmpdir`에 대해 표시된 쿼리와 함께 통합될 때 분할됩니다.



`IO:BufFileRead` 및 `IO:BufFileWrite` 이벤트는 대개 워크로드의 상위 쿼리에서 임시 파일이 생성될 때 발생합니다. 데이터베이스 부하 및 상위 SQL 섹션의 AAS(상위 활성 세션)를 검토하여 `IO:BufFileRead` 및 `IO:BufFileWrite`에서 대기 중인 상위 쿼리를 식별하는 데 성능 개선 도우미를 사용할 수 있습니다.



성능 개선 도우미를 사용하여 상위 쿼리와 대기 이벤트에서 부하를 분석하는 방법에 대한 자세한 내용은 상위 SQL(Top SQL) 탭 개요 단원을 참조하세요. 임시 파일 사용량 및 관련 대기 이벤트를 늘리는 쿼리를 식별하고 조정해야 합니다. 이러한 대기 이벤트 및 수정에 대한 자세한 내용은 IO:BufFileRead 및 IO:BufFileWrite를 참조하세요.

> **Note**
>
> 이 `work_mem` 파라미터는 정렬 작업의 메모리가 부족하여 결과가 임시 파일에 기록되는 시점을 제어합니다. 이 파라미터의 설정을 기본값보다 높게 변경하면 모든 데이터베이스 세션에서 더 많은 메모리를 사용할 수 있으므로 변경하지 않는 것이 좋습니다. 또한 복잡한 조인 및 정렬을 수행하는 단일 세션에서는 각 작업이 메모리를 사용하는 병렬 작업을 수행할 수 있습니다. 여러 조인 및 정렬이 포함된 대용량 보고서가 있을 경우 `SET work_mem` 명령을 사용하여 세션 수준에서 이 파라미터를 설정하는 것이 가장 좋습니다. 그러면 변경 사항이 현재 세션에만 적용되고 값이 전체적으로 변경되지 않습니다.

pgBadger를 사용한 PostgreSQL의 로그 분석

pgBadger 등의 로그 분석기를 사용하여 PostgreSQL 로그를 분석할 수 있습니다. pgBadger 문서에는 %l 패턴(세션 또는 프로세스의 로그 라인)은 접두사에 포함되어야 한다고 나와 있습니다. 그러나 현재 RDS `log_line_prefix`를 pgBadger에 파라미터로 사용하더라도 여전히 보고서를 생성해야 합니다.

예를 들어 다음 명령은 pgBadger를 사용하여 2014-02-04 일자의 Amazon RDS for PostgreSQL 로그 파일을 정확한 형식으로 나타내고 있습니다.

```
./pgbadger -f stderr -p '%t:%r:%u@%d:[%p]:' postgresql.log.2014-02-04-00
```

PGSnapper를 사용하여 PostgreSQL 모니터링

PGSnapper를 사용하여 Amazon RDS for PostgreSQL 성능 관련 통계 및 지표의 주기적인 수집을 지원할 수 있습니다. 자세한 내용은 Monitor Amazon RDS for PostgreSQL performance using PGSnapper(PGSnapper를 사용하여 Amazon RDS for PostgreSQL 성능 모니터링)를 참조하세요.

RDS for PostgreSQL DB 인스턴스에 파라미터로 작업

일부 경우, 사용자 정의 파라미터 그룹을 지정하지 않고 RDS for PostgreSQL DB 인스턴스를 생성할 수 있습니다. 이 경우 선택한 PostgreSQL 버전에 대한 기본 파라미터 그룹을 사용하여 DB 인스턴스가 생성됩니다. 예를 들어, PostgreSQL 13.3을 사용하여 RDS for PostgreSQL DB 인스턴스를 생성한다고 가정해 보겠습니다. 이 경우 DB 인스턴스는 PostgreSQL 13 릴리스용 파라미터 그룹(`default.postgres13`)의 값을 사용하여 생성됩니다.

사용자 지정 DB 파라미터 그룹을 생성할 수도 있습니다. RDS for PostgreSQL DB 인스턴스용 설정을 기본값에서 수정하려면 이 작업을 수행해야 합니다. 자세한 방법은 Amazon RDS의 파라미터 그룹을 참조하세요.

RDS for PostgreSQL DB 인스턴스에서 설정을 여러 가지 방법으로 추적할 수 있습니다. AWS Management Console, AWS CLI 또는 Amazon RDS API를 사용할 수 있습니다. 다음과 같이 인스턴스의 PostgreSQL `pg_settings` 테이블에서 값을 쿼리할 수도 있습니다.

```
SELECT name, setting, boot_val, reset_val, unit
 FROM pg_settings
 ORDER BY name;
```

이 쿼리에서 반환된 값에 대한 자세한 내용은 PostgreSQL 설명서의 `pg_settings` 섹션을 참조하세요.

RDS for PostgreSQL DB 인스턴스에서 max_connections 및 shared_buffers에 대한 설정을 변경할 때 특히 주의해야 합니다. 예를 들어 max_connections 또는 shared_buffers에 대한 설정을 수정해서 실제 워크로드에 비해 너무 높은 값을 사용한다고 가정하겠습니다. 이 경우 RDS for PostgreSQL DB 인스턴스는 시작되지 않습니다. 이 경우 postgres.log에 다음과 같은 오류가 기록됩니다.

```
2018-09-18 21:13:15 UTC::@:[8097]:FATAL:  could not map anonymous shared memory: Cannot
 allocate memory
2018-09-18 21:13:15 UTC::@:[8097]:HINT:  This error usually means that PostgreSQL's
 request for a shared memory segment
exceeded available memory or swap space. To reduce the request size (currently
 3514134274048 bytes), reduce
PostgreSQL's shared memory usage, perhaps by reducing shared_buffers or
 max_connections.
```

그러나 PostgreSQL DB 파라미터 그룹의 기본 RDS에 포함된 설정 값은 변경할 수 없습니다. 파라미터에 대한 설정을 변경하려면 먼저 사용자 지정 DB 파라미터 그룹을 생성합니다. 그런 다음 해당 사용자 정의 그룹의 설정을 변경한 다음 사용자 정의 파라미터 그룹을 RDS for PostgreSQL DB 인스턴스에 적용합니다. 자세한 내용은 Amazon RDS의 파라미터 그룹를 참조하세요.

RDS for PostgreSQL에는 두 가지 유형의 파라미터가 있습니다.

- 정적 파라미터 - 정적 파라미터를 사용하려면 변경 후 RDS for PostgreSQL DB 인스턴스를 재부팅해야 새 값이 적용됩니다.
- 동적 파라미터 - 동적 파라미터는 설정을 변경한 후 재부팅할 필요가 없습니다.

> **Note**
>
> RDS for PostgreSQL DB 인스턴스가 자체 사용자 정의 DB 파라미터 그룹을 사용하는 경우, 실행 중인 DB 인스턴스에서 동적 파라미터 값을 변경할 수 있습니다. AWS Management Console, AWS CLI 또는 Amazon RDS API를 사용하여 이 작업을 수행할 수 있습니다.

권한이 있는 경우 ALTER DATABASE, ALTER ROLE 및 SET 명령을 사용하여 파라미터 값을 변경할 수도 있습니다.

RDS for PostgreSQL DB 인스턴스 파라미터 목록

다음 표에는 RDS for PostgreSQL DB 인스턴스에서 사용할 수 있는 일부 파라미터가 나와 있습니다. 사용 가능한 모든 파라미터를 보려면 describe-db-parameters AWS CLI 명령을 사용하세요. 예를 들어 RDS for PostgreSQL 버전 13의 기본 파라미터 그룹에서 사용할 수 있는 모든 파라미터 목록을 가져오려면 다음을 실행합니다.

```
aws rds describe-db-parameters --db-parameter-group-name default.postgres13
```

콘솔을 사용할 수도 있습니다. Amazon RDS 메뉴에서 파라미터 그룹을 선택한 다음 AWS 리전에서 사용 가능한 파라미터 그룹 중에서 파라미터 그룹을 선택합니다.

파라미터 이름	적용 유형	설명
application_name	동적	애플리케이션 이름이 통계 및 로그에 표시되도록 설정합니다.
archive_command	동적	WAL 파일을 보관하기 위해 호출할 셸 명령을 설정합니다.
array_nulls	동적	어레이의 NULL 요소 입력을 활성화합니다.
authentication_timeout	동적	클라이언트 인증 완료를 위한 최대 허용 시간을 설정합니다.
autovacuum	동적	autovacuum 서브프로세스를 시작합니다.
autovacuum_analyze_scale_factor	동적	분석 전 삽입, 업데이트 또는 삭제되는 튜플 수를 reltuples 분수 값으로 지정합니다.
autovacuum_analyze_threshold	동적	분석 전 삽입, 업데이트 또는 삭제되는 튜플의 최소 수를 지정합니다.
autovacuum_freeze_max_age	정적	트랜잭션 ID 랩어라운드를 방지하기 위한 테이블의 autovacuum 기간을 지정합니다.
autovacuum_naptime	동적	autovacuum 실행 간 절전 시간을 지정합니다.

파라미터 이름	적용 유형	설명
autovacuum_max_workers	정적	autovacuum 작업자 프로세스를 동시에 실행할 수 있는 최대 수를 설정합니다.
autovacuum_vacuum_cost_delay	동적	autovacuum에서 vacuum 코스트 지연 시간(밀리초)을 지정합니다.
autovacuum_vacuum_cost_limit	동적	autovacuum에서 지연 시간 이전에 이용 가능한 vacuum 코스트 값을 지정합니다.
autovacuum_vacuum_scale_factor	동적	vacuum 전 업데이트 또는 삭제되는 튜플 수를 reltuples 분수 값으로 지정합니다.
autovacuum_vacuum_threshold	동적	vacuum 전 업데이트 또는 삭제되는 튜플의 최소 수를 지정합니다.
backslash_quote	동적	문자열 리터럴에서 백슬래시(\)의 허용 여부를 설정합니다.
bgwriter_delay	동적	라운드 사이에 백그라운드 라이터의 절전 시간을 지정합니다.
bgwriter_lru_maxpages	동적	백그라운드 라이터가 라운드마다 LRU 페이지를 작성할 최대 수를 지정합니다.
bgwriter_lru_multiplier	동적	라운드마다 해제할 평균 버퍼 사용량의 배수를 지정합니다.
bytea_output	동적	바이트의 출력 형식을 설정합니다.
check_function_bodies	동적	CREATE FUNCTION 도중 함수 본문을 검사합니다.
checkpoint_completion_target	동적	체크포인트 도중 변경된 버퍼 플러시에 사용된 시간으로 체크포인트 간격의 분수 값입니다.
checkpoint_segments	동적	로그 세그먼트에서 자동 write-ahead log(WAL) 체크포인트의 최대 간격을 설정합니다.

파라미터 이름	적용 유형	설명
checkpoint_timeout	동적	자동 WAL 체크포인트 사이의 최대 시간을 설정합니다.
checkpoint_warning	동적	체크포인트 세그먼트가 이 파라미터 값보다 더 빨리 채워지는 경우 경고를 활성화합니다.
client_connection_check_interval	동적	쿼리를 실행하는 동안 연결 끊김 검사 사이의 시간 간격을 설정합니다.
client_encoding	동적	클라이언트 문자 세트 인코딩을 설정합니다.
client_min_messages	동적	클라이언트에게 보여지는 메시지 수준을 설정합니다.
commit_delay	동적	트랜잭션 커밋부터 디스크에 대한 WAL 플러시까지 지연 시간(밀리초)을 설정합니다.
commit_siblings	동적	commit_delay 실행 전에 동시에 열려 있는 트랜잭션 최소 개수를 설정합니다.
constraint_exclusion	동적	planner가 제약 조건을 사용하여 쿼리를 최적화하도록 활성화합니다.
cpu_index_tuple_cost	동적	인덱스 스캔 중 각 인덱스 항목을 처리하는 데 따른 planner의 예상 코스트를 설정합니다.
cpu_operator_cost	동적	각 연산자 또는 함수 호출을 처리하는 데 따른 planner의 예상 코스트를 설정합니다.
cpu_tuple_cost	동적	각 튜플(행)을 처리하는 데 따른 planner의 예상 코스트를 설정합니다.
cursor_tuple_fraction	동적	planner가 예상하는 검색할 커서 행의 분수 값을 설정합니다.
datestyle	동적	날짜와 시간 값에 대한 표시 형식을 설정합니다.

파라미터 이름	적용 유형	설명
deadlock_timeout	동적	교착 상태 여부를 확인하기 이전 잠금 대기 시간을 설정합니다.
debug_pretty_print	동적	구문과 실행 계획 트리를 들여쓰기 하여 표시합니다.
debug_print_parse	동적	각 쿼리의 구문 분석 트리를 기록합니다.
debug_print_plan	동적	각 쿼리의 실행 계획을 기록합니다.
debug_print_rewritten	동적	각 쿼리에서 재작성된 구문 분석 트리를 기록합니다.
default_statistics_target	동적	기본 통계 대상을 설정합니다.
default_tablespace	동적	테이블과 인덱스를 생성할 기본 테이블스페이스를 설정합니다.
default_transaction_deferrable	동적	새로운 트랜잭션의 기본 deferrable 상태를 설정합니다.
default_transaction_isolation	동적	새로운 트랜잭션마다 트랜잭션 격리 수준을 설정합니다.
default_transaction_read_only	동적	새로운 트랜잭션의 기본 읽기 전용 상태를 설정합니다.
default_with_oids	동적	새로운 테이블을 생성할 때 객체 ID(OID)가 기본적으로 포함됩니다.
effective_cache_size	동적	디스크 캐시 크기에 대한 planner의 가정을 설정합니다.
effective_io_concurrency	동적	디스크 하위 시스템에서 효율적으로 동시에 처리할 수 있는 요청 수를 지정합니다.

파라미터 이름	적용 유형	설명
enable_bitmapscan	동적	planner가 비트맵 스캔 계획을 사용할 수 있도록 활성화합니다.
enable_hashagg	동적	planner가 해시된 집계 계획을 사용할 수 있도록 활성화합니다.
enable_hashjoin	동적	planner가 해시 조인 계획을 사용할 수 있도록 활성화합니다.
enable_indexscan	동적	planner가 인덱스 스캔 계획을 사용할 수 있도록 활성화합니다.
enable_material	동적	planner가 구체화를 사용할 수 있도록 활성화합니다.
enable_mergejoin	동적	planner가 병합 조인 계획을 사용할 수 있도록 활성화합니다.
enable_nestloop	동적	planner가 중첩 루프 조인 계획을 사용할 수 있도록 활성화합니다.
enable_seqscan	동적	planner가 순차적 스캔 계획을 사용할 수 있도록 활성화합니다.
enable_sort	동적	planner가 명시적 정렬 단계를 사용할 수 있도록 활성화합니다.
enable_tidscan	동적	planner가 TID 스캔 계획을 사용할 수 있도록 활성화합니다.
escape_string_warning	동적	일반 문자열 리터럴의 백슬래시(\) 이스케이프에 대해 경고합니다.
extra_float_digits	동적	부동 소수점으로 표시할 자릿수를 설정합니다.
from_collapse_limit	동적	서브 쿼리가 축소되지 않는 FROM 목록 크기를 설정합니다.

파라미터 이름	적용 유형	설명
fsync	동적	업데이트를 디스크와 강제로 동기화합니다.
full_page_writes	동적	체크포인트 후 최초 변경 시 전체 페이지를 WAL에 기입합니다.
geqo	동적	유전적 쿼리 최적화를 활성화합니다.
geqo_effort	동적	GEQO: 다른 GEQO 파라미터의 기본값을 설정하는 데 사용됩니다.
geqo_generations	동적	GEQO: 알고리즘의 반복 횟수입니다.
geqo_pool_size	동적	GEQO: 모집단의 개체 수입니다.
geqo_seed	동적	GEQO: 무작위 경로 선택을 위한 시드(seed)를 지정합니다.
geqo_selection_bias	동적	GEQO: 모집단 내 선택적 압력을 지정합니다.
geqo_threshold	동적	GEQO가 사용되는 FROM 항목의 임계값을 설정합니다.
gin_fuzzy_search_limit	동적	정확한 GIN 기준 검색에 허용되는 최대 결과 수를 설정합니다.
hot_standby_feedback	동적	핫 스탠바이가 피드백 메시지를 기본 또는 업스트림 스탠바이로 전송하는지 여부를 결정합니다.
intervalstyle	동적	간격 값에 대한 표시 형식을 설정합니다.
join_collapse_limit	동적	JOIN 구문이 결합되지 않는 FROM 목록 크기를 설정합니다.
lc_messages	동적	메시지 표시 언어를 설정합니다.
lc_monetary	동적	통화 금액의 형식으로 사용할 로캘을 설정합니다.
lc_numeric	동적	숫자의 형식으로 사용할 로캘을 설정합니다.

파라미터 이름	적용 유형	설명
lc_time	동적	날짜와 시간 값의 형식으로 사용할 로캘을 설정합니다.
log_autovacuum_min_duration	동적	autovacuum 작업이 기록되는 최소 실행 시간을 설정합니다.
log_checkpoints	동적	각 체크포인트를 기록합니다.
log_connections	동적	성공한 연결을 모두 기록합니다.
log_disconnections	동적	지속 시간을 포함해 세션 종료를 기록합니다.
log_duration	동적	완료된 개별 SQL 문의 지속 시간을 기록합니다.
log_error_verbosity	동적	기록된 메시지의 세부 사항을 설정합니다.
log_executor_stats	동적	실행기 성능 통계를 서버 로그에 기록합니다.
log_filename	동적	로그 파일의 이름 패턴을 설정합니다.
log_file_mode	동적	로그 파일에 대한 파일 권한을 설정합니다. 기본값은 0644입니다.
log_hostname	동적	연결 로그에 호스트 이름을 기록합니다. PostgreSQL 12 이상 버전에서는 이 파라미터가 기본적으로 '해제' 상태입니다. 이 파라미터를 켜면 연결은 DNS 역방향 조회를 사용하여 연결 로그에 캡처되는 호스트 이름을 가져옵니다. 이 파라미터를 켰다면, 연결을 설정하는 데 걸리는 시간이 달라지는지 모니터링해야 합니다.
log_line_prefix	동적	각 로그 행에 접두사가 붙은 정보를 제어합니다.
log_lock_waits	동적	오랜 잠금 대기 시간을 기록합니다.
log_min_duration_statement	동적	문이 기록되는 최소 실행 시간을 설정합니다.

파라미터 이름	적용 유형	설명
log_min_error_statement	동적	이 수준 이상으로 오류 원인이 되는 모든 문을 기록합니다.
log_min_messages	동적	기록되는 메시지 수준을 설정합니다.
log_parser_stats	동적	구문 분석기 성능 통계를 서버 로그에 기록합니다.
log_planner_stats	동적	planner 성능 통계를 서버 로그에 기록합니다.
log_rotation_age	동적	N분 후에 자동 로그 파일 로테이션이 일어납니다.
log_rotation_size	동적	N킬로바이트 후에 자동 로그 파일 로테이션이 일어납니다.
log_statement	동적	기록할 문 유형을 설정합니다.
log_statement_stats	동적	누적 성능 통계를 서버 로그에 기록합니다.
log_temp_files	동적	이 킬로바이트 수치보다 큰 임시 파일의 사용을 기록합니다.
log_timezone	동적	로그 메시지에 사용할 표준 시간대를 설정합니다.
log_truncate_on_rotation	동적	로그 순환 중에 동일한 이름의 기존 로그 파일을 잘라냅니다.
logging_collector	정적	하위 프로세스를 시작하여 stderr 출력 및/또는 csvlog를 로그 파일로 캡처합니다.
maintenance_work_mem	동적	유지 관리 작업에 사용할 최대 메모리를 설정합니다.
max_connections	정적	동시에 접속할 수 있는 최대 수를 설정합니다.
max_files_per_process	정적	서버 프로세스마다 파일을 동시에 열 수 있는 최대 수를 설정합니다.
max_locks_per_transaction	정적	하나의 트랜잭션에서 사용할 수 있는 최대 잠금 횟수를 설정합니다.

파라미터 이름	적용 유형	설명
max_pred_locks_per_transaction	정적	하나의 트랜잭션에서 사용할 수 있는 최대 술어(predicate) 잠금 횟수를 설정합니다.
max_prepared_transactions	정적	트랜잭션을 동시에 준비할 수 있는 최대 수를 설정합니다.
max_stack_depth	동적	최대 스택 깊이(KB)를 설정합니다.
max_standby_archive_delay	동적	핫 스탠바이 서버가 아카이브 WAL 데이터를 처리할 때 쿼리 취소까지 걸리는 최대 지연 시간을 설정합니다.
max_standby_streaming_delay	동적	핫 스탠바이 서버가 스트리밍 WAL 데이터를 처리할 때 쿼리 취소까지 걸리는 최대 지연 시간을 설정합니다.

max_wal_size	동적	검사 점을 트리거하는 WAL 크기(MB)를 설정합니다.
		• RDS for PostgreSQL 15 이전 버전의 경우 2GB입니다.
		• RDS for PostgreSQL 16 이상 버전의 경우 RDS for PostgreSQL은 인스턴스에 할당된 스토리지 크기를 기반으로 자동으로 구성됩니다.
		  • 100GB 이상의 스토리지가 할당된 인스턴스의 경우 6GB입니다.
		  • 할당된 스토리지가 100GB 미만인 인스턴스의 경우 2GB입니다.
		
		Amazon RDS for PostgreSQL DB 인스턴스에서 다음 명령을 사용하여 현재 값을 확인합니다.
		`SHOW max_wal_size;`

파라미터 이름	적용 유형	설명
min_wal_size	동적	WAL을 축소할 최소 크기를 설정합니다. PostgreSQL 버전 9.6 이하의 경우 min_wal_size 는 16MB 단위입니다. PostgreSQL 버전 10 이상의 경우 min_wal_size 는 1MB 단위입니다.
quote_all_identifiers	동적	SQL 조각 생성 시 모든 식별자에 인용 부호(")를 추가합니다.
random_page_cost	동적	비순차적으로 가져온 디스크 페이지에 대한 planner의 예상 코스트를 설정합니다. 이 파라미터는 쿼리 계획 관리(QPM)가 켜져 있지 않으면 값이 없습니다. QPM이 켜져 있으면 이 파라미터의 기본값은 4입니다.
rds.adaptive_autovacuum	동적	트랜잭션 ID 임계값이 초과될 때마다 autovacuum 파라미터를 자동으로 조정합니다.
rds.force_ssl	동적	SSL 연결을 사용해야 합니다. RDS for PostgreSQL 버전 15의 기본값은 1(켜짐)로 설정되어 있습니다. 다른 모든 RDS for PostgreSQL 메이저 버전 14 이상에는 기본값이 0(꺼짐)로 설정되어 있습니다.
rds.local_volume_spill_enabled	정적	로컬 볼륨에 논리적 유출 파일 쓰기를 활성화합니다.
rds.log_retention_period	동적	Amazon RDS가 n분보다 오래된 PostgreSQL 로그를 삭제하도록 로그 보존을 설정합니다.
rds.rds_superuser_reserved_connections	정적	rds_superuser용으로 예약된 연결 슬롯 수를 설정합니다. 이 파라미터는 버전 15 이상에서만 사용할 수 있습니다. 자세한 내용은 reserved_connections PostgreSQL 설명서를 참조하세요.

파라미터 이름	적용 유형	설명
rds.replica_identity_full	동적	이 파라미터를 on으로 설정하면 모든 데이터베이스 테이블에 대해 복제본 ID 설정이 FULL로 재정의됩니다. 즉, REPLICA IDENTITY FULL 설정에 관계없이 모든 열 값이 Write Ahead Log(WAL)에 기록됩니다. **Note** 이 파라미터를 켜면 추가 WAL 로깅으로 인해 데이터베이스 인스턴스 IOPS가 증가할 수 있습니다.
rds.restrict_password_commands	정적	암호를 관리할 수 있는 사람을 rds_password 역할이 있는 사용자로 제한합니다. 암호 제한을 활성화하려면 이 파라미터를 1로 설정합니다. 기본값은 0입니다.
search_path	동적	스키마로 한정되지 않은 이름의 스키마 검색 순서를 설정합니다.
seq_page_cost	동적	순차적으로 가져온 디스크 페이지에 대한 planner의 예상 코스트를 설정합니다.
session_replication_role	동적	트리거 및 다시 쓰기 규칙에 대한 세션 동작을 설정합니다.
shared_buffers	정적	서버에서 사용할 공유 메모리 버퍼의 수를 설정합니다.
shared_preload_libraries	정적	RDS for PostgreSQL DB 인스턴스에 미리 로드할 공유 라이브러리를 나열합니다. 지원되는 값으로는 auto_explain, orafce, pgaudit, pglogical, pg_bigm, pg_cron, pg_hint_plan, pg_prewarm, pg_similarity, pg_stat_statements, pg_tle, pg_transport, plprofiler, plrust가 있습니다.

파라미터 이름	적용 유형	설명
ssl	동적	SSL 연결을 활성화합니다.
sql_inheritance	동적	다양한 명령에서 서브테이블이 기본적으로 상속됩니다.
ssl_renegotiation_limit	동적	암호화 키를 재협상하기 전에 전송 및 수신할 트래픽 양을 설정합니다.
standard_conforming_strings	동적	... 문자열에서 백슬래시가 리터럴로 처리됩니다.
statement_timeout	동적	모든 문에 허용되는 최대 지속 시간을 설정합니다.
synchronize_seqscans	동적	동기 방식의 순차적 스캔을 활성화합니다.
synchronous_commit	동적	현재 트랜잭션 동기화 수준을 설정합니다.
tcp_keepalives_count	동적	TCP keepalive의 최대 재전송 횟수를 지정합니다.
tcp_keepalives_idle	동적	TCP keepalive의 실행 주기를 지정합니다.
tcp_keepalives_interval	동적	TCP keepalive의 재전송 주기를 지정합니다.
temp_buffers	동적	각 세션에서 사용하는 임시 버퍼의 최대 수를 설정합니다.
temp_file_limit	동적	임시 파일이 증가할 수 있는 최대 크기(KB)를 설정합니다.
temp_tablespaces	동적	임시 테이블 및 정렬 파일에 사용할 테이블스페이스를 설정합니다.

파라미터 이름	적용 유형	설명
timezone	동적	타임스탬프를 표시 및 해석할 시간대를 설정합니다. IANA(Internet Assigned Numbers Authority)에서는 https://www.iana.org/time-zones에서 일 년에 여러 번 새로운 표준 시간대를 게시합니다. RDS에서 PostgreSQL의 새로운 마이너 유지 관리 릴리스를 릴리스할 때마다 릴리스 시점의 최신 표준 시간대 데이터가 함께 제공됩니다. 최신 RDS for PostgreSQL 버전을 사용하면 RDS의 최신 표준 시간대 데이터를 갖게 됩니다. DB 인스턴스에 최신 표준 시간대 데이터가 있는지 확인하려면 상위 DB 엔진 버전으로 업그레이드하는 것이 좋습니다. PostgreSQL DB 인스턴스의 표준 시간대 테이블은 수동으로 수정할 수 없습니다. RDS는 실행 중인 DB 인스턴스의 표준 시간대 데이터를 수정하거나 재설정하지 않습니다. 새 표준 시간대 데이터는 데이터베이스 엔진 버전 업그레이드를 수행할 때만 설치됩니다.
track_activities	동적	명령 실행에 대한 정보를 수집합니다.
track_activity_query_size	정적	pg_stat_activity.current_query에 예약되는 크기(바이트)를 설정합니다.
track_counts	동적	데이터베이스 작업에 관한 통계를 수집합니다.
track_functions	동적	데이터베이스 작업에 관한 함수 수준 통계를 수집합니다.
track_io_timing	동적	데이터베이스 I/O 작업에 관한 시간 통계를 수집합니다.
transaction_deferrable	동적	잠재적 직렬화 오류 없이 시작될 때까지 직렬화가 가능한 읽기 전용 트랜잭션의 지연 여부를 결정합니다.

파라미터 이름	적용 유형	설명
transaction_isolation	동적	현재 트랜잭션 격리 수준을 설정합니다.
transaction_read_only	동적	현재 트랜잭션의 읽기 전용 상태를 설정합니다.
transform_null_equals	동적	expr=NULL을 expr IS NULL로 처리합니다.
update_process_title	동적	프로세스 제목을 업데이트하여 활성 SQL 명령을 표시합니다.
vacuum_cost_delay	동적	vacuum 코스트 지연 시간(밀리초)을 지정합니다.
vacuum_cost_limit	동적	지연 시간 이전에 이용 가능한 vacuum 코스트 값을 지정합니다.
vacuum_cost_page_dirty	동적	vacuum으로 페이지 변경 시 부과되는 vacuum 코스트를 지정합니다.
vacuum_cost_page_hit	동적	버퍼 캐시에서 발견되는 페이지에 대한 vacuum 코스트를 지정합니다.
vacuum_cost_page_miss	동적	버퍼 캐시에서 발견되지 않는 페이지에 대한 vacuum 코스트를 지정합니다.
vacuum_defer_cleanup_age	동적	vacuum 및 hot cleanup을 연기해야 하는 트랜잭션 수를 지정합니다(있는 경우).
vacuum_freeze_min_age	동적	vacuum에서 테이블 행을 동결해야 하는 최소 기간을 지정합니다.
vacuum_freeze_table_age	동적	vacuum에서 전체 테이블을 스캔하여 튜플을 동결해야 하는 기간을 지정합니다.
wal_buffers	정적	WAL 기능을 위해 공유 메모리에서 사용할 디스크 페이지 버퍼 수를 설정합니다.

파라미터 이름	적용 유형	설명
wal_writer_delay	동적	WAL 플러시 사이에 WAL 작성기의 절전 시간을 지정합니다.
work_mem	동적	쿼리 작업 공간에 사용할 최대 메모리를 설정합니다.
xmlbinary	동적	XML에서 바이너리 값의 인코딩 방식을 설정합니다.
xmloption	동적	암시적 구문 분석 및 직렬화 작업에서 XML 데이터를 문서 또는 내용 조각으로 간주할지 여부를 설정합니다.

Amazon RDS는 모든 파라미터에서 기본 PostgreSQL 단위를 사용합니다. 다음 표는 각 파라미터의 PostgreSQL 기본 단위를 나타냅니다.

파라미터 이름	단위
archive_timeout	s
authentication_timeout	s
autovacuum_naptime	s
autovacuum_vacuum_cost_delay	ms
bgwriter_delay	ms
checkpoint_timeout	s
checkpoint_warning	s
deadlock_timeout	ms
effective_cache_size	8KB
lock_timeout	ms

파라미터 이름	단위
log_autovacuum_min_duration	ms
log_min_duration_statement	ms
log_rotation_age	분
log_rotation_size	KB
log_temp_files	KB
maintenance_work_mem	KB
max_stack_depth	KB
max_standby_archive_delay	ms
max_standby_streaming_delay	ms
post_auth_delay	s
pre_auth_delay	s
segment_size	8KB
shared_buffers	8KB
statement_timeout	ms
ssl_renegotiation_limit	KB
tcp_keepalives_idle	s
tcp_keepalives_interval	s
temp_file_limit	KB
work_mem	KB
temp_buffers	8KB

파라미터 이름	단위
vacuum_cost_delay	ms
wal_buffers	8KB
wal_receiver_timeout	ms
wal_segment_size	B
wal_sender_timeout	ms
wal_writer_delay	ms
wal_receiver_status_interval	s

RDS for PostgreSQL의 대기 이벤트를 사용한 튜닝

대기 이벤트는 RDS for PostgreSQL의 중요한 튜닝 도구입니다. 세션이 리소스를 기다리는 이유와 어떤 작업을 수행하는지 알 수 있다면 병목 현상을 더 잘 줄일 수 있습니다. 이 섹션의 정보를 통해 가능한 원인과 해결 조치를 찾을 수 있습니다. 이 섹션에서는 기본적인 PostgreSQL 튜닝 개념에 대해서도 설명합니다.

이 섹션에서 다루는 대기 이벤트는 RDS for PostgreSQL에만 해당됩니다.

주제

- RDS for PostgreSQL 튜닝을 위한 필수 개념
- RDS for PostgreSQL 대기 이벤트
- Client:ClientRead
- Client:ClientWrite
- CPU
- IO:BufFileRead 및 IO:BufFileWrite
- IO:DataFileRead
- IO:WALWrite
- IPC:parallel 대기 이벤트
- Lock:advisory
- Lock:extend
- Lock:Relation
- Lock:transactionid
- Lock:tuple
- LWLock:BufferMapping (LWLock:buffer_mapping)
- LWLock:BufferIO(IPC:BufferIO)
- LWLock:buffer_content (BufferContent)
- LWLock:lock_manager (LWLock:lockmanager)
- LWLock:pg_stat_statements
- Timeout:PgSleep
- Timeout:VacuumDelay

RDS for PostgreSQL 튜닝을 위한 필수 개념

RDS for PostgreSQL 데이터베이스를 튜닝하기 전에 대기 이벤트가 무엇인지, 왜 발생하는지 확인해 보세요. 또한 RDS for PostgreSQL의 기본 메모리 및 디스크 아키텍처도 검토하세요. 유용한 아키텍처 다이어그램은 PostgreSQL 위키북을 참조하세요.

주제

- RDS for PostgreSQL 대기 이벤트
- RDS for PostgreSQL 메모리
- RDS for PostgreSQL 프로세스

RDS for PostgreSQL 대기 이벤트

대기 이벤트는 세션이 리소스를 대기 중임을 나타냅니다. 예를 들어 대기 이벤트 `Client:ClientRead`는 RDS for PostgreSQL이 클라이언트로부터 데이터를 수신하기 위해 대기 중일 때 발생합니다. 세션은 일반적으로 대기하는 리소스는 다음과 같습니다.

- 버퍼에 대한 단일 스레드 액세스(예: 세션이 버퍼 수정을 시도하는 경우)
- 현재 다른 세션에 의해 잠겨 있는 행
- 데이터 파일 읽기
- 로그 파일 쓰기

예를 들어 쿼리를 충족하기 위해 세션에서 전체 테이블 스캔을 수행할 수 있습니다. 데이터가 아직 메모리에 없는 경우 세션은 디스크 I/O가 완료될 때까지 기다립니다. 버퍼를 메모리로 읽으면 다른 세션이 동일한 버퍼에 액세스하기 때문에 세션을 기다려야 할 수 있습니다. 데이터베이스는 미리 정의된 대기 이벤트를 사용하여 대기를 기록합니다. 이러한 이벤트는 카테고리로 그룹화됩니다.

대기 이벤트 자체는 성능 문제를 나타내지 않습니다. 예를 들어 요청된 데이터가 메모리에 없으면 디스크에서 데이터를 읽어야 합니다. 한 세션이 업데이트를 위해 행을 잠그면 다른 세션은 해당 행의 잠금이 해제될 때까지 기다려 업데이트할 수 있습니다. 커밋을 수행하려면 로그 파일에 대한 쓰기가 완료될 때까지 대기해야 합니다. 대기는 데이터베이스의 정상적인 기능에 필수적입니다.

반면 많은 수의 대기 이벤트는 일반적으로 성능 문제를 나타냅니다. 이러한 경우 대기 이벤트 데이터를 사용하여 세션이 시간을 소비하는 위치를 결정할 수 있습니다. 예를 들어 일반적으로 실행하는 데 몇 분 정도 걸리는 보고서가 몇 시간 동안 실행되는 경우 총 대기 시간에 가장 많이 기여하는 대기 이벤트를 식별할 수 있습니다. 최상위 대기 이벤트의 원인을 확인할 수 있는 경우 성능을 향상시키는 변경 작

업을 수행할 수 있습니다. 예를 들어 세션이 다른 세션에 의해 잠긴 행에서 대기 중인 경우 잠금 세션을 종료할 수 있습니다.

RDS for PostgreSQL 메모리

RDS for PostgreSQL 메모리는 공유 메모리와 로컬 메모리로 구분됩니다.

주제

- RDS for PostgreSQL의 공유 메모리
- RDS for PostgreSQL의 로컬 메모리

RDS for PostgreSQL의 공유 메모리

RDS for PostgreSQL은 인스턴스가 시작될 때 공유 메모리를 할당합니다. 공유 메모리는 여러 하위 영역으로 나뉩니다. 다음 섹션에서는 가장 중요한 항목에 대해 설명합니다.

주제

- 공유 버퍼
- 미리 쓰기 로그(WAL) 버퍼

공유 버퍼

공유 버퍼 풀은 애플리케이션 연결에서 사용 중이거나 사용 중인 모든 페이지를 보관하는 RDS for PostgreSQL 메모리 영역입니다. 페이지는 디스크 블록의 메모리 버전입니다. 공유 버퍼 풀은 디스크에서 읽은 데이터 블록을 캐시합니다. 이 풀은 디스크에서 데이터를 다시 읽어야 할 필요성을 줄여 데이터베이스가 더 효율적으로 연산하게 합니다.

모든 테이블과 인덱스는 고정된 크기의 페이지 배열로 저장됩니다. 각 블록에는 행에 해당하는 여러 튜플이 포함되어 있습니다. 튜플은 모든 페이지에 저장할 수 있습니다.

공유 버퍼 풀에는 유한 메모리가 있습니다. 새 요청에 메모리에 없는 페이지가 필요하고 메모리가 더 이상 존재하지 않는 경우 RDS for PostgreSQL은 요청을 수용하기 위해 자주 사용되지 않는 페이지를 삭제합니다. 제거 정책은 클럭 스윕 알고리즘에 의해 구현됩니다.

`shared_buffers` 파라미터는 서버가 데이터 캐싱에 전념하는 메모리 양을 결정합니다. 기본값은 DB 인스턴스에 사용 가능한 메모리를 기반으로 `{DBInstanceClassMemory/32768}`바이트로 설정됩니다.

미리 쓰기 로그(WAL) 버퍼

미리 쓰기 로그(WAL) 버퍼는 RDS for PostgreSQL이 나중에 영구 스토리지에 쓰는 트랜잭션 데이터를 보유합니다. WAL 메커니즘을 사용하여 RDS for PostgreSQL이 다음을 수행할 수 있습니다.

- 장애 발생 후 데이터 복구
- 디스크에 빈번한 쓰기를 방지하여 디스크 I/O 감소

클라이언트가 데이터를 변경하면 RDS for PostgreSQL이 변경 사항을 WAL 버퍼에 기록합니다. 클라이언트가 `COMMIT`을 발행하면 WAL 라이터 프로세스는 트랜잭션 데이터를 WAL 파일에 씁니다.

`wal_level` 파라미터는 WAL에 기록되는 정보의 양을 결정하며, 가능한 값은 `minimal`, `replica`, `logical`입니다.

RDS for PostgreSQL의 로컬 메모리

모든 백엔드 프로세스는 쿼리 처리를 위해 로컬 메모리를 할당합니다.

주제

- 작업 메모리 영역
- 유지 관리 작업 메모리 영역
- 임시 버퍼 영역

작업 메모리 영역

작업 메모리 영역은 정렬 및 해시를 수행하는 쿼리에 대한 임시 데이터를 보유합니다. 예를 들어, 다음을 포함하는 쿼리 `ORDER BY` 절은 정렬을 수행합니다. 쿼리는 해시 조인 및 집계에서 해시 테이블을 사용합니다.

`work_mem` 파라미터는 임시 디스크 파일에 쓰기 전에 내부 정렬 작업 및 해시 테이블에서 사용할 메모리 양이며, 메가바이트 단위로 측정됩니다. 기본값은 4MB입니다. 여러 세션을 동시에 실행할 수 있으며 각 세션은 유지 관리 작업을 병렬로 실행할 수 있습니다. 이러한 이유로 사용되는 총 작업 메모리는 `work_mem` 설정의 배수가 될 수 있습니다.

유지 관리 작업 메모리 영역

유지 관리 작업 메모리 영역은 유지 관리 작업을 위해 데이터를 캐시합니다. 이러한 작업에는 베큠, 인덱스 생성 및 외래 키 추가가 포함됩니다.

maintenance_work_mem 파라미터는 유지 관리 작업에서 사용할 최대 메모리 양을 지정하며, 메가바이트 단위로 측정됩니다. 기본값은 64MB입니다. 데이터베이스 세션은 한 번에 하나의 유지 관리 작업만 실행할 수 있습니다.

임시 버퍼 영역

임시 버퍼 영역에서는 각 데이터베이스 세션에 대한 임시 테이블을 캐시합니다.

각 세션은 사용자가 지정한 한도까지 필요에 따라 임시 버퍼를 할당합니다. 세션이 끝나면 서버는 버퍼를 지웁니다.

temp_buffers 파라미터는 각 세션에서 사용하는 임시 버퍼의 최대 수를 설정하며, 메가바이트 단위로 측정됩니다. 기본값은 8MB입니다. 세션 내에서 임시 테이블을 처음 사용하기 전에 temp_buffers 값을 변경할 수 있습니다.

RDS for PostgreSQL 프로세스

RDS for PostgreSQL은 여러 프로세스를 사용합니다.

주제

- Postmaster 프로세스
- 백엔드 프로세스
- 백그라운드 프로세스

Postmaster 프로세스

Postmaster 프로세스는 RDS for PostgreSQL을 시작할 때 시작되는 첫 번째 프로세스입니다. 포스트마스터 프로세스는 다음과 같은 주요 책임이 있습니다.

- 백그라운드 프로세스 포크 및 모니터링
- 클라이언트 프로세스에서 인증 요청을 수신하고 데이터베이스에서 요청을 처리하도록 허용하기 전에 인증 요청을 인증합니다.

백엔드 프로세스

Postmaster가 클라이언트 요청을 인증하면 postmaster는 postgres 프로세스라고도 하는 새 백엔드 프로세스를 생성합니다. 하나의 클라이언트 프로세스가 정확히 하나의 백엔드 프로세스에 연결됩니다. 클라이언트 프로세스와 백엔드 프로세스는 포스트마스터 프로세스의 개입 없이 직접 통신합니다.

백그라운드 프로세스

Postmaster 프로세스는 서로 다른 백엔드 작업을 수행하는 여러 프로세스를 생성합니다. 몇 가지 중요한 사항은 다음과 같습니다.

- WAL 라이터

  RDS for PostgreSQL 미리 쓰기 로깅(WAL) 버퍼의 데이터를 로그 파일에 씁니다. 미리 쓰기 로깅의 원칙은 데이터베이스가 변경 사항을 설명하는 로그 레코드를 디스크에 기록하기 전까지는 데이터베이스가 데이터 파일에 변경 사항을 쓸 수 없다는 것입니다. WAL 메커니즘은 디스크 I/O를 줄이고, RDS for PostgreSQL 장애 후 로그를 사용하여 데이터베이스를 복구할 수 있도록 합니다.
- 백그라운드 라이터

  이 프로세스는 메모리 버퍼에서 데이터 파일로 더티(수정된) 페이지를 주기적으로 씁니다. 백엔드 프로세스가 메모리에서 페이지를 수정하면 페이지가 더티(dirty) 상태가 됩니다.
- Autovacuum 데몬

  데몬은 다음 구성 요소로 이루어져 있습니다.

  - Autovacuum 시작 관리자
  - Autovacuum 작업자 프로세스

  Autovacuum은 삽입되고 업데이트되거나 삭제된 튜플 수가 많은 테이블이 있는지 확인합니다. 데몬에는 다음과 같은 책임이 있습니다.

  - 업데이트되거나 삭제된 행이 차지하는 디스크 공간 복구 또는 재사용
  - 플래너가 사용하는 통계 업데이트
  - 트랜잭션 ID 랩어라운드로 인해 이전 데이터 손실 방지

  Autovacuum 기능은 `VACUUM`과 `ANALYZE` 명령을 자동화합니다. `VACUUM`에는 다음과 같은 표준 및 전체 변형이 있습니다. 스탠다드 베큠은 다른 데이터베이스 연산과 병행하여 실행됩니다. `VACUUM FULL`에는 작업 중인 테이블에 독점적인 잠금이 필요합니다. 따라서 동일한 테이블에 액세스하는 연산과 병렬로 실행할 수 없습니다. `VACUUM`은 상당한 양의 I/O 트래픽을 생성하여 다른 활성 세션의 성능이 저하될 수 있습니다.

RDS for PostgreSQL 대기 이벤트

다음 표에는 성능 문제를 가장 일반적으로 나타내는 RDS for PostgreSQL에 대한 대기 이벤트가 나열되어 있으며 가장 일반적인 원인과 해결 조치가 요약되어 있습니다.

대기 이벤트	정의
Client:ClientRead	이 이벤트는 RDS for PostgreSQL이 클라이언트에서 데이터를 수신하기 위해 대기 중일 때 발생합니다.
Client:ClientWrite	이 이벤트는 RDS for PostgreSQL이 클라이언트에 데이터를 쓰기 위해 대기 중일 때 발생합니다.
CPU	이 이벤트는 스레드가 CPU에서 활성 상태이거나 CPU를 대기 중일 때 발생합니다.
IO:BufFileRead 및 IO:BufFileWrite	이러한 이벤트는 RDS for PostgreSQL이 임시 파일을 만들 때 발생합니다.
IO:DataFileRead	이 이벤트는 공유 메모리에서 페이지를 사용할 수 없기 때문에 연결이 백엔드 프로세스에서 저장소에서 필요한 페이지를 읽도록 대기할 때 발생합니다.
IO:WALWrite	이 이벤트는 RDS for PostgreSQL이 미리 쓰기 로그(WAL) 버퍼가 WAL 파일에 쓰여지기를 기다리는 동안 발생합니다.
Lock:advisory	이 이벤트는 PostgreSQL 애플리케이션이 잠금을 사용하여 여러 세션에서 활동을 조정할 때 발생합니다.
Lock:extend	이 이벤트는 백엔드 프로세스가 릴레이션을 확장하기 위해 릴레이션을 잠그기를 기다리는 동안 다른 프로세스에서 동일한 목적으로 해당 릴레이션에 대한 잠금이 있는 경우에 발생합니다.
Lock:Relation	이 이벤트는 쿼리가 현재 다른 트랜잭션에 의해 잠긴 테이블 또는 뷰에 대한 잠금을 얻기 위해 대기 중일 때 발생합니다.
Lock:transactionid	이 이벤트는 트랜잭션이 행 수준 잠금을 대기 중일 때 발생합니다.

대기 이벤트	정의
Lock:tuple	이 이벤트는 백엔드 프로세스가 튜플에 대한 잠금 획득을 대기 중일 때 발생합니다.
LWLock:BufferMapping (LWLock:buffer_mapping)	이 이벤트는 세션이 데이터 블록을 공유 버퍼 풀의 버퍼와 연결하기 위해 대기 중일 때 발생합니다.
LWLock:BufferIO(IPC:BufferIO)	이 이벤트는 RDS for PostgreSQL이 페이지에 동시에 액세스하려고 할 때 다른 프로세스가 입출력(I/O) 작업을 완료할 때까지 기다리는 경우에 발생합니다.
LWLock:buffer_content (BufferContent)	이 대기 이벤트는 다른 세션에서 특정 데이터 페이지에 대해 쓰기 잠금을 설정한 동안 세션에서 메모리 내 해당 페이지 읽기 또는 쓰기를 대기 중일 때 발생합니다.
LWLock:lock_manager (LWLock:lockmanager)	이 이벤트는 RDS for PostgreSQL 엔진이 빠른 경로 잠금이 불가능할 때 잠금을 할당, 확인 및 할당 해제하기 위해 공유 잠금 메모리 영역을 유지 관리하는 경우에 발생합니다.
Timeout:PgSleep	이 이벤트는 서버 프로세스가 `pg_sleep` 함수 및 절전 시간 초과가 만료될 때까지 대기할 때 발생합니다.
Timeout:VacuumDelay	이 이벤트는 예상 비용 한도에 도달하여 진공 프로세스가 휴면 상태임을 나타냅니다.

Client:ClientRead

`Client:ClientRead` 이벤트는 RDS for PostgreSQL이 클라이언트에서 데이터를 수신하기 위해 대기 중일 때 발생합니다.

주제

- 지원되는 엔진 버전
- 컨텍스트

- 대기 증가의 가능한 원인
- 작업

지원되는 엔진 버전

이 대기 이벤트 정보는 RDS for PostgreSQL 버전 10 이상에서 지원됩니다.

컨텍스트

RDS for PostgreSQL DB 인스턴스가 클라이언트로부터 데이터를 수신하기 위해 대기 중입니다. RDS for PostgreSQL DB 인스턴스는 클라이언트로부터 데이터를 수신해야 클라이언트에 더 많은 데이터를 보낼 수 있습니다. 클라이언트에서 데이터를 수신하기 전에 인스턴스가 대기하는 시간이 `Client:ClientRead` 이벤트입니다.

대기 증가의 가능한 원인

상위 대기에서 나타나는 `Client:ClientRead` 이벤트의 일반적인 원인은 다음을 포함합니다.

네트워크 대기 시간 증가

RDS for PostgreSQL DB 인스턴스와 클라이언트 간에 네트워크 지연 시간이 늘어날 수 있습니다. 네트워크 지연 시간이 높을수록 DB 인스턴스가 클라이언트에서 데이터를 수신하는 데 필요한 시간이 늘어납니다.

클라이언트에 대한 로드 증가

클라이언트에 CPU 압력 또는 네트워크 포화가 있을 수 있습니다. 클라이언트 로드가 증가하면 클라이언트에서 RDS for PostgreSQL DB 인스턴스로의 데이터 전송이 지연될 수 있습니다.

과도한 네트워크 왕복

RDS for PostgreSQL DB 인스턴스와 클라이언트 간의 네트워크 왕복 횟수가 많으면 클라이언트에서 RDS for PostgreSQL DB 인스턴스로의 데이터 전송을 지연시킬 수 있습니다.

대량 복사 연산

복사 연산 중에 데이터가 클라이언트의 파일 시스템에서 RDS for PostgreSQL DB 인스턴스로 전송됩니다. DB 인스턴스에 대량의 데이터를 전송하면 클라이언트에서 DB 인스턴스로의 데이터 전송이 지연될 수 있습니다.

유휴 클라이언트 연결

클라이언트가 `idle in transaction` 상태의 RDS for PostgreSQL DB 인스턴스에 연결하는 경우 DB 인스턴스는 클라이언트가 더 많은 데이터를 보내거나 명령을 실행할 때까지 기다릴 수 있습니다. 이 상태의 연결은`Client:ClientRead` 이벤트 증가로 이어질 수 있습니다.

연결 풀링에 사용되는 PGBouncer

PGBouncer에는 `pkt_buf`와 같은 낮은 수준의 네트워크 구성 설정이 있으며, 기본적으로 4,096으로 설정됩니다. 워크로드가 PGBouncer를 통해 4,096바이트보다 큰 쿼리 패킷을 보내는 경우, `pkt_buf` 설정을 8,192까지 늘리는 것이 좋습니다. 새 설정으로 인해 `Client:ClientRead` 이벤트의 수가 줄어들지 않는 경우 `pkt_buf` 설정을 16,384 또는 32,768과 같이 더 큰 값으로 설정하는 것이 좋습니다. 쿼리 텍스트가 큰 경우 더 큰 설정이 특히 유용할 수 있습니다.

작업

대기 이벤트의 원인에 따라 다른 작업을 권장합니다.

주제

- 인스턴스와 동일한 가용 영역 및 VPC 서브넷에 클라이언트를 배치합니다.
- 클라이언트 확장
- 현재 세대 인스턴스 사용
- 네트워크 대역폭 향상
- 최대 네트워크 성능 모니터링
- '트랜잭션에서 유휴' 상태의 트랜잭션 모니터링

인스턴스와 동일한 가용 영역 및 VPC 서브넷에 클라이언트를 배치합니다.

네트워크 대기 시간을 줄이고 네트워크 처리량(throughput)을 늘리려면 RDS for PostgreSQL DB 인스턴스와 동일한 가용 영역 및 Virtual Private Cloud(VPC) 서브넷에 클라이언트를 배치합니다. 클라이언트가 가능한 한 DB 인스턴스에 지리적으로 가까이 있는지 확인합니다.

클라이언트 확장

Amazon CloudWatch 또는 기타 호스트 지표를 사용하여 클라이언트가 현재 CPU 또는 네트워크 대역폭 또는 둘 다에 의해 제한되어 있는지 확인합니다. 클라이언트가 제한된 경우 그에 따라 클라이언트를 확장합니다.

현재 세대 인스턴스 사용

점보 프레임을 지원하는 DB 인스턴스 클래스를 사용하지 않는 경우도 있습니다. Amazon EC2 애플리케이션을 실행하는 경우 클라이언트에 현재 세대 인스턴스를 사용하는 것이 좋습니다. 또한 클라이언트 운영 체제에서 최대 전송 단위(MTU)를 구성합니다. 이 기술은 네트워크 왕복 수를 줄이고 네트워크 처리량을 늘릴 수 있습니다. 자세한 내용은 Amazon EC2 사용 설명서의 점보 프레임(9001 MTU)을 참조하세요.

DB 인스턴스 클래스에 대한 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요. Amazon EC2 인스턴스 유형과 동일한 DB 인스턴스 클래스를 확인하려면 `db.`가 Amazon EC2 인스턴스 유형 이름 앞에 있어야 합니다. 예를 들어, `r5.8xlarge` Amazon EC2 인스턴스는 `db.r5.8xlarge` DB 인스턴스 클래스입니다.

네트워크 대역폭 향상

DB 인스턴스의 수신 및 발신 네트워크 트래픽을 모니터링하는 `NetworkReceiveThroughput` 및 `NetworkTransmitThroughput` Amazon CloudWatch 지표를 사용합니다. 이러한 지표는 네트워크 대역폭이 워크로드에 충분한지 확인하는 데 도움이 될 수 있습니다.

네트워크 대역폭이 충분하지 않으면 늘리세요. 만약 AWS 클라이언트 또는 DB 인스턴스가 네트워크 대역폭 제한에 도달했다면. 대역폭을 늘리는 유일한 방법은 DB 인스턴스 크기를 늘리는 것입니다. 자세한 내용은 DB 인스턴스 클래스 유형 단원을 참조하십시오.

CloudWatch 지표에 대한 자세한 내용은 Amazon RDS에 대한 Amazon CloudWatch 지표 섹션을 참조하세요.

최대 네트워크 성능 모니터링

Amazon EC2 클라이언트를 사용하는 경우 Amazon EC2는 집계 인바운드 및 아웃바운드 네트워크 대역폭을 포함하여 네트워크 성능 지표에 대한 최댓값을 제공합니다. 또한 연결 추적을 제공하여 패킷이 예상대로 반환되고 도메인 이름 시스템(DNS)과 같은 서비스에 대한 링크-로컬 서비스 액세스를 제공합니다. 이러한 최댓값을 모니터링하려면 현재 향상된 네트워킹 드라이버를 사용하고 클라이언트의 네트워크 성능을 모니터링하세요.

자세한 내용은 Amazon EC2 사용 설명서의 Amazon EC2 인스턴스의 네트워크 성능 모니터링과 Amazon EC2 사용 설명서의 Amazon EC2 인스턴스의 네트워크 성능 모니터링을 참조하세요.

'트랜잭션에서 유휴' 상태의 트랜잭션 모니터링

`idle in transaction` 연결의 수가 점점 늘어나고 있는지 확인하세요. 이를 수행하려면 `state`열의 `pg_stat_activity`테이블을 모니니터링하세요. 다음과 유사한 쿼리를 실행하여 연결 소스를 식별할 수 있습니다.

```
select client_addr, state, count(1) from pg_stat_activity
where state like 'idle in transaction%'
group by 1,2
order by 3 desc
```

Client:ClientWrite

`Client:ClientWrite` 이벤트는 RDS for PostgreSQL이 클라이언트에 데이터를 쓰기 위해 대기 중일 때 발생합니다.

주제

- 지원되는 엔진 버전
- 컨텍스트
- 대기 증가의 가능한 원인
- 작업

지원되는 엔진 버전

이 대기 이벤트 정보는 RDS for PostgreSQL 버전 10 이상에서 지원됩니다.

컨텍스트

클라이언트 프로세스는 클러스터가 더 많은 데이터를 보낼 수 있게 되기 전 반드시 RDS for PostgreSQL DB 클러스터로부터 받은 모든 데이터를 읽어야 합니다. 클라이언트에서 데이터를 수신하기 전에 클러스터가 대기하는 시간은 `Client:ClientWrite` 이벤트를 트리거합니다.

RDS for PostgreSQL DB 인스턴스와 클라이언트 간의 네트워크 처리량(throughput)이 감소하면 이 이벤트가 발생할 수 있습니다. 클라이언트에 대한 CPU 압력 및 네트워크 포화 상태시에도 이 이벤트가 발생할 수 있습니다. CPU 압력은 CPU가 완전히 활용되고 CPU 시간을 기다리는 작업이 있을 때입니다. 네트워크 포화는 데이터베이스와 클라이언트 간의 네트워크가 처리할 수 있는 것보다 많은 데이터를 전달하는 경우입니다.

대기 증가의 가능한 원인

상위 대기에서 나타나는 `Client:ClientWrite` 이벤트의 일반적인 원인은 다음을 포함합니다.

네트워크 대기 시간 증가

RDS for PostgreSQL DB 인스턴스와 클라이언트 간에 네트워크 지연 시간이 늘어날 수 있습니다. 네트워크 대기 시간이 높을수록 클라이언트가 데이터를 수신하는 데 필요한 시간이 늘어납니다.

클라이언트에 대한 로드 증가

클라이언트에 CPU 압력 또는 네트워크 포화가 있을 수 있습니다. 클라이언트에 대한 로드가 증가하면 RDS for MySQL DB 인스턴스에서 데이터 수신이 지연됩니다.

클라이언트에 전송되는 대용량 데이터

RDS for PostgreSQL DB 인스턴스가 많은 양의 데이터를 클라이언트에 전송하고 있을 수 있습니다. 클라이언트는 클러스터가 데이터를 전송하는 만큼 빠르게 데이터를 수신하지 못할 수 있습니다. 큰 테이블 복사와 같은 활동 시 `Client:ClientWrite` 이벤트가 증가할 수 있습니다.

작업

대기 이벤트의 원인에 따라 다른 작업을 권장합니다.

주제

- 클러스터와 동일한 가용 영역 및 VPC 서브넷에 클라이언트를 배치합니다.
- 현재 세대 인스턴스 사용
- 클라이언트에 전송해야 하는 데이터 양 감소
- 클라이언트 확장

클러스터와 동일한 가용 영역 및 VPC 서브넷에 클라이언트를 배치합니다.

네트워크 대기 시간을 줄이고 네트워크 처리량(throughput)을 늘리려면 RDS for PostgreSQL DB 인스턴스와 동일한 가용 영역 및 Virtual Private Cloud(VPC) 서브넷에 클라이언트를 배치합니다.

현재 세대 인스턴스 사용

점보 프레임을 지원하는 DB 인스턴스 클래스를 사용하지 않는 경우도 있습니다. Amazon EC2 애플리케이션을 실행하는 경우 클라이언트에 현재 세대 인스턴스를 사용하는 것이 좋습니다. 또한 클라이언트 운영 체제에서 최대 전송 단위(MTU)를 구성합니다. 이 기술은 네트워크 왕복 수를 줄이고 네트워크 처리량을 늘릴 수 있습니다. 자세한 내용은 Amazon EC2 사용 설명서의 점보 프레임(9001 MTU)을 참조하세요.

DB 인스턴스 클래스에 대한 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요. Amazon EC2 인스턴스 유형과 동일한 DB 인스턴스 클래스를 확인하려면 db.가 Amazon EC2 인스턴스 유형 이름 앞에 있어야 합니다. 예를 들어, r5.8xlarge Amazon EC2 인스턴스는 db.r5.8xlarge DB 인스턴스 클래스입니다.

클라이언트에 전송해야 하는 데이터 양 감소

가능하면 애플리케이션을 조정하여 RDS for PostgreSQL DB 인스턴스가 클라이언트에 보내는 데이터의 양을 줄이세요. 이러한 조정을 수행하면 클라이언트에서 CPU 및 네트워크 경합을 줄일 수 있습니다.

클라이언트 확장

Amazon CloudWatch 또는 기타 호스트 지표를 사용하여 클라이언트가 현재 CPU 또는 네트워크 대역폭 또는 둘 다에 의해 제한되어 있는지 확인합니다. 클라이언트가 제한된 경우 그에 따라 클라이언트를 확장합니다.

CPU

이 이벤트는 스레드가 CPU에서 활성 상태이거나 CPU를 대기 중일 때 발생합니다.

주제

- 지원되는 엔진 버전
- 컨텍스트
- 대기 증가의 가능한 원인
- 작업

지원되는 엔진 버전

이 대기 이벤트 정보는 모든 버전의 RDS for PostgreSQL과 관련이 있습니다.

컨텍스트

중앙 처리 장치는 명령을 실행하는 컴퓨터의 구성 요소입니다. 예를 들어 CPU 명령은 연산 연산을 수행하고 메모리에서 데이터를 교환합니다. 쿼리가 데이터베이스 엔진을 통해 수행하는 명령 수를 늘리면 쿼리 실행 시간이 늘어납니다. CPU 스케줄링은 프로세스에 CPU 시간을 부여합니다. 스케줄링은 운영 체제 커널에 의해 조정됩니다.

주제

- 이 대기 시간이 언제 발생하는지 확인하는 방법
- DBloadCPU 지표
- os.cpuUtilization 지표
- CPU 스케줄링의 원인

이 대기 시간이 언제 발생하는지 확인하는 방법

CPU 대기 이벤트는 백엔드 프로세스가 CPU에서 활성 상태이거나 CPU를 기다리고 있음을 나타냅니다. 쿼리에 다음 정보가 표시될 때 발생한다는 것을 알고 있습니다.

- `pg_stat_activity.state` 열이 `active` 값을 갖고 있습니다.
- `pg_stat_activity`의 `wait_event_type`과 `wait_event` 열은 모두 `null`입니다.

CPU를 사용하거나 대기 중인 백엔드 프로세스를 보려면 다음 쿼리를 실행하세요.

```
SELECT *
FROM   pg_stat_activity
WHERE  state = 'active'
AND    wait_event_type IS NULL
AND    wait_event IS NULL;
```

DBloadCPU 지표

CPU 성능 개선 도우미의 지표는 `DBLoadCPU`입니다. `DBLoadCPU`의 값은 Amazon CloudWatch 지표 `CPUUtilization`의 값과 다를 수 있습니다. 후자의 지표는 데이터베이스 인스턴스에 대한 HyperVisor에서 수집됩니다.

os.cpuUtilization 지표

성능 개선 도우미 운영 시스템 지표는 CPU 사용률에 대한 자세한 정보를 제공합니다. 예를 들어 다음 지표가 표시될 수 있습니다.

- `os.cpuUtilization.nice.avg`
- `os.cpuUtilization.total.avg`
- `os.cpuUtilization.wait.avg`

- os.cpuUtilization.idle.avg

성능 개선 도우미는 데이터베이스 엔진의 CPU 사용량을 os.cpuUtilization.nice.avg와 같이 보고합니다.

CPU 스케줄링의 원인

운영 체제(OS) 커널은 CPU의 스케줄링을 처리합니다. CPU가 활성 상태이면 프로세스는 스케줄링될 때까지 기다려야 할 수 있습니다. CPU는 계산을 수행하는 동안 활성 상태입니다. 실행 중이 아닌 유휴 스레드, 즉 메모리 I/O에서 대기하는 유휴 스레드가 있는 경우에도 CPU는 활성 상태입니다. 일반적인 데이터베이스 워크로드에서는 이 유형의 I/O가 지배적입니다.

다음 조건이 충족되면 프로세스가 CPU에 스케줄링될 때까지 대기할 수 있습니다.

- CloudWatch CPUUtilization 지표가 100%에 가깝습니다.
- 평균 로드가 vCPU 수보다 크므로 부하가 많음을 나타냅니다. 성능 개선 도우미의 OS 지표 섹션에서 loadAverageMinute 지표를 찾을 수 있습니다.

대기 증가의 가능한 원인

이 이벤트가 정상 상태보다 많이 발생하여 성능 문제를 나타낼 수 있는 경우 일반적인 원인은 다음과 같습니다.

주제

- 갑작스러운 스파이크의 원인
- 장기 고주파의 원인
- 코너 케이스

갑작스러운 스파이크의 원인

갑작스런 스파이크의 가장 큰 원인은 다음과 같습니다.

- 애플리케이션이 데이터베이스에 대한 동시 연결을 너무 많이 열었습니다. 이 시나리오를 '연결 폭풍'이라고 합니다.
- 다음 방법 중 하나로 애플리케이션 워크로드가 변경되었습니다.
  - 새 쿼리
  - 데이터 집합 크기 증가

  - 인덱스 유지 관리 또는 생성
  - 새로운 함수
  - 새로운 연산자
  - 병렬 쿼리 실행의 증가
- 쿼리 실행 계획이 변경되었습니다. 경우에 따라 변경으로 인해 버퍼가 증가할 수 있습니다. 예를 들어, 쿼리는 이전에 인덱스를 사용했을 때의 순차적 스캔을 사용하고 있습니다. 이 경우 쿼리는 동일한 목표를 달성하기 위해 더 많은 CPU가 필요합니다.

장기 고주파의 원인

장기간에 걸쳐 재발하는 이벤트의 가장 큰 원인은 다음과 같습니다.

- 너무 많은 백엔드 프로세스가 CPU에서 동시에 실행되고 있습니다. 이러한 프로세스는 병렬 작업자가 될 수 있습니다.
- 쿼리는 많은 수의 버퍼가 필요하기 때문에 차선적으로 수행됩니다.

코너 케이스

실제 원인으로 판명될 가능성이 있는 원인이 없다면 다음과 같은 상황이 발생할 수 있습니다.

- CPU가 프로세스를 안팎으로 교환하고 있습니다.
- 방대한 페이지 기능이 해제된 경우 CPU가 페이지 테이블 항목을 관리하고 있을 수 있습니다. 이 메모리 관리 기능은 마이크로, 스몰, 미디엄 DB 인스턴스 클래스를 제외한 모든 DB 인스턴스 클래스에 대해 기본적으로 사용 설정되어 있습니다. 자세한 내용은 RDS for PostgreSQL용 방대한 페이지 단원을 참조하십시오.

작업

CPU 대기 이벤트가 데이터베이스 활동을 지배하는 경우 반드시 성능 문제를 나타내지는 않습니다. 성능이 저하되는 경우에만 이 이벤트에 응답하세요.

주제

- 데이터베이스로 인해 CPU 증가가 발생하는지 조사합니다.
- 연결 수 증가 여부 확인
- 워크로드 변경에 대응

데이터베이스로 인해 CPU 증가가 발생하는지 조사합니다.

성능 개선 도우미의 `os.cpuUtilization.nice.avg` 지표를 검토합니다. 이 값이 CPU 사용량보다 훨씬 작다면 데이터베이스가 아닌 프로세스가 CPU의 주요 기여자입니다.

연결 수 증가 여부 확인

Amazon CloudWatch의 `DatabaseConnections` 지표를 검토합니다. 작업은 CPU 대기 이벤트 증가 시기 동안 증가 또는 감소 여부에 따라 달라집니다.

연결이 증가했습니다.

연결 수가 증가했다면, CPU를 사용하는 백엔드 프로세스 수를 vCPU 수와 비교하세요. 가능한 시나리오는 다음과 같습니다.

- CPU를 사용하는 백엔드 프로세스 수가 vCPU 수보다 적습니다.

  이 경우 연결 수는 문제가 되지 않습니다. 그러나 계속해서 CPU 사용률을 줄이려고 할 수도 있습니다.
- CPU를 사용하는 백엔드 프로세스 수가 vCPU 수보다 적습니다.

  이 경우에는 다음 옵션을 고려하세요.
  - 데이터베이스에 연결된 백엔드 프로세스 수를 줄입니다. 예를 들어 RDS 프록시와 같은 연결 풀링 솔루션을 구현합니다. 자세한 내용은 Amazon RDS Proxy 섹션을 참조하세요.
  - 인스턴스 크기를 업그레이드하여 vCPU 수를 늘립니다.
  - 해당하는 경우 일부 읽기 전용 워크로드를 리더 노드로 리디렉션하세요.

연결이 증가하지 않았습니다.

성능 개선 도우미의 `blks_hit` 지표를 검토합니다. `blks_hit`의 증가와 CPU 사용량 사이의 상관관계를 찾습니다. 가능한 시나리오는 다음과 같습니다.

- CPU 사용량 및 `blks_hit`이 상관관계가 있습니다.

  이 경우 CPU 사용량에 연결된 최상위 SQL 문을 찾아 계획 변경을 찾습니다. 다음과 같은 기술을 사용할 수 있습니다.
  - 계획을 수동으로 설명하고 예상 실행 계획과 비교합니다.
  - 초당 블록 히트와 초당 로컬 블록 히트가 증가하는지 확인합니다. 성능 개선 도우미의 대시보드의 상위 SQL 섹션에서 환경설정을 선택합니다.

- CPU 사용량 및 `blks_hit`이 상관관계가 없습니다.

  이 경우 다음 중 한 가지라도 발생하는지 확인합니다.

  - 애플리케이션이 데이터베이스에 급속하게 연결되고 데이터베이스와의 연결이 끊어지고 있습니다.

    `log_connections`와 `log_disconnections`를 켜 이 동작을 진단하고, 그런 다음 PostgreSQL 로그를 분석합니다. `pgbadger` 로그 분석기 사용을 고려하세요. 자세한 내용은 https://github.com/darold/pgbadger 섹션을 참조하세요.

  - OS에 과부하가 걸렸습니다.

    이 경우 성능 개선 도우미는 백엔드 프로세스가 평소보다 오랜 시간 동안 CPU를 사용하고 있음을 보여줍니다. 성능 개선 도우미 `os.cpuUtilization` 지표 또는 CloudWatch `CPUUtilization` 지표에서 증거를 찾습니다. 운영 체제에 과부하가 걸린 경우 향상된 모니터링 지표를 살펴보고 더 자세히 진단하세요. 특히 프로세스 목록과 각 프로세스에서 소비되는 CPU 비율을 살펴보세요.

  - 상위 SQL 문이 너무 많은 CPU를 소비하고 있습니다.

    CPU 사용량에 연결된 문을 검사하여 CPU를 더 적게 사용할 수 있는지 확인합니다. `EXPLAIN` 명령을 실행하고 가장 큰 영향을 미치는 계획 노드에 중점을 둡니다. PostgreSQL 실행 계획 비주얼라이저를 사용하는 것이 좋습니다. http://explain.dalibo.com/ 섹션을 참조해 이 도구를 사용해 보세요.

워크로드 변경에 대응

워크로드가 변경된 경우 다음 변경 유형을 찾습니다.

새 쿼리

새 쿼리가 예상되는지 확인합니다. 그렇다면 실행 계획과 초당 실행 횟수가 예상되는지 확인합니다.

데이터 집합 크기 증가

파티셔닝이 아직 구현되지 않은 경우 파티셔닝이 도움이 될지 확인합니다. 이 전략은 쿼리가 검색해야 하는 페이지 수를 줄일 수 있습니다.

인덱스 유지 관리 또는 생성

유지 관리 일정이 예상되는지 확인합니다. 모범 사례는 피크 활동 이외의 유지 관리 활동을 예약하는 것입니다.

새로운 함수

테스트 중에 이러한 함수가 예상대로 작동하는지 확인합니다. 특히 초당 실행 횟수가 예상되는지 확인합니다.

새로운 연산자

테스트 중에 이러한 함수가 예상대로 작동하는지 확인합니다.

병렬 쿼리 실행 증가

다음 상황 중 한 가지라도 발생했는지 확인합니다.

- 포함된 관계식 또는 인덱스의 크기가 갑자기 커져 `min_parallel_table_scan_size` 또는 `min_parallel_index_scan_size`와 크게 다릅니다.
- `parallel_setup_cost` 또는 `parallel_tuple_cost`에 최근 변경 사항이 적용되었습니다.
- `max_parallel_workers` 또는 `max_parallel_workers_per_gather`에 최근 변경 사항이 적용되었습니다.

IO:BufFileRead 및 IO:BufFileWrite

`IO:BufFileRead` 및 `IO:BufFileWrite` 이벤트는 RDS for PostgreSQL이 임시 파일을 만들 때 발생합니다. 연산에 현재 정의된 작업 메모리 파라미터보다 많은 메모리가 필요한 경우 임시 데이터를 영구 스토리지에 씁니다. 이 작업을 '디스크로 유출'이라고도 합니다.

주제

- 지원되는 엔진 버전
- 컨텍스트
- 대기 증가의 가능한 원인
- 작업

지원되는 엔진 버전

이 대기 이벤트 정보는 모든 RDS for PostgreSQL 버전에서 지원됩니다.

컨텍스트

`IO:BufFileRead`와 `IO:BufFileWrite`는 작업 메모리 영역 및 유지 관리 작업 메모리 영역과 관련이 있습니다. 이러한 로컬 메모리 영역에 대한 자세한 내용은 PostgreSQL 설명서에서 리소스 소비를 참조하세요.

기본값은 work_mem 또는 4MB입니다. 한 세션이 병렬로 연산을 수행하는 경우 병렬 처리를 처리하는 각 작업자는 4MB의 메모리를 사용합니다. 이런 이유로, work_mem을 신중하게 설정하세요. 값을 너무 많이 늘리면 많은 세션을 실행하는 데이터베이스가 너무 많은 메모리를 소비할 수 있습니다. 값을 너무 낮게 설정하면 RDS for PostgreSQL이 로컬 스토리지에 임시 파일을 만듭니다. 이러한 임시 파일의 디스크 I/O는 성능을 저하시킬 수 있습니다.

다음과 같은 일련의 이벤트를 관찰하면 데이터베이스에서 임시 파일이 생성될 수 있습니다.

1. 갑작스럽고 급격한 가용성 감소
2. 여유 공간을 위한 신속한 복구

'체인톱' 패턴도 볼 수 있습니다. 이 패턴은 데이터베이스에서 지속적으로 작은 파일을 생성한다는 것을 의미할 수 있습니다.

대기 증가의 가능한 원인

일반적으로 이러한 대기 이벤트는 work_mem 또는 maintenance_work_mem 파라미터가 할당하는 것보다 더 많은 메모리를 소모하는 연산에 의해 발생합니다. 보정을 위해 연산은 임시 파일에 기록합니다. IO:BufFileRead와 IO:BufFileWrite 이벤트의 일반적인 원인에는 다음이 포함됩니다.

작업 메모리 영역에 존재하는 것보다 많은 메모리가 필요한 쿼리

다음 특성을 가진 쿼리는 작업 메모리 영역을 사용합니다.

- 해시 조인.
- ORDER BY 절
- GROUP BY 절
- DISTINCT
- 윈도 함수
- CREATE TABLE AS SELECT
- 구체화 뷰 새로고침

작업 메모리 영역에 존재하는 것보다 많은 메모리가 필요한 명령문

다음 명령문에서는 유지 관리 작업 메모리 영역을 사용합니다.

- CREATE INDEX

- CLUSTER

작업

대기 이벤트의 원인에 따라 다른 작업을 권장합니다.

주제

- 문제 식별
- 조인 쿼리 검토
- ORDER BY와 GROUP BY 쿼리를 검토합니다.
- DISTINCT 연산 사용을 피하세요.
- GROUP BY 함수 대신 윈도 함수를 사용하는 것이 좋습니다.
- 구체화된 뷰 및 CTAS 명령문 조사
- 인덱스를 다시 구축할 때 pg_repack 사용
- 테이블을 클러스터링할 때 maintenance_work_mem 증가
- 메모리를 조정하여 IO:BufFileRead 및 IO:BufFileWrite 방지

문제 식별

성능 개선 도우미가 켜져 있지 않고 `IO:BufFileRead`와 `IO:BufFileWrite`가 정상보다 더 자주 발생한다고 의심되는 상황을 가정합니다. 문제의 원인을 식별하기 위해 지정한 임계값 KB를 초과하는 임시 파일을 생성하는 모든 쿼리를 로깅하도록 `log_temp_files` 파라미터를 설정할 수 있습니다. 기본적으로 `log_temp_files`는 `-1`로 설정되어 있어 이 로깅 기능을 해제합니다. 이 파라미터를 `0`로 설정하면 RDS for PostgreSQL이 모든 임시 파일을 로깅합니다. `1024`으로 설정하면 RDS for PostgreSQL은 1MB보다 큰 임시 파일을 생성하는 모든 쿼리를 로깅합니다. `log_temp_files`에 관한 자세한 내용은 PostgreSQL 문서의 오류 보고 및 로깅을 참조하세요.

조인 쿼리 검토

쿼리에서 조인이 사용될 가능성이 높습니다. 예를 들어 다음 쿼리는 네 개의 테이블을 조인합니다.

```
SELECT *
       FROM "order"
 INNER JOIN order_item
       ON (order.id = order_item.order_id)
 INNER JOIN customer
```

```
       ON (customer.id = order.customer_id)
 INNER JOIN customer_address
       ON (customer_address.customer_id = customer.id AND
           order.customer_address_id = customer_address.id)
 WHERE customer.id = 1234567890;
```

임시 파일 사용량 스파이크의 원인은 쿼리 자체의 문제입니다. 예를 들어, 끊어진 절은 조인을 제대로 필터링하지 않을 수 있습니다. 다음 예에서 두 번째 내부 조인을 고려해 보겠습니다.

```
SELECT *
       FROM "order"
 INNER JOIN order_item
       ON (order.id = order_item.order_id)
 INNER JOIN customer
       ON (customer.id = customer.id)
 INNER JOIN customer_address
       ON (customer_address.customer_id = customer.id AND
           order.customer_address_id = customer_address.id)
 WHERE customer.id = 1234567890;
```

위의 쿼리가 실수로 `customer.id`를 `customer.id`에 조인하여 모든 고객과 모든 주문 사이에 데카르트 프로덕트를 생성합니다. 이러한 유형의 우발적인 조인은 큰 임시 파일을 생성합니다. 테이블의 크기에 따라 테카르트 쿼리(Cartesian query)는 스토리지를 채울 수도 있습니다. 다음 조건이 충족되면 애플리케이션에 데카르트 조인(Cartesian join)이 있을 수 있습니다.

- 스토리지 가용성이 급격히, 크게 줄어들고 복구가 빨라집니다.
- 인덱스가 생성되지 않습니다.
- `CREATE TABLE FROM SELECT` 문이 발행되고 있지 않습니다.
- 구체화 뷰가 새로 고침이 되지 않습니다.

적절한 키를 사용하여 테이블이 조인되고 있는지 확인하려면 쿼리 및 객체 관계형 매핑 지시어를 검사합니다. 애플리케이션의 특정 쿼리가 항상 호출되는 것은 아니며 일부 쿼리는 동적으로 생성됩니다.

ORDER BY와 GROUP BY 쿼리를 검토합니다.

경우에 따라 `ORDER BY` 절에 임시 파일이 과도하게 발생할 수 있습니다. 다음 지침을 참고하세요.

- 정렬이 필요할 때 `ORDER BY` 절의 열만 포함합니다. 이 지침은 `ORDER BY` 절에서 수천 개의 행을 반환하고 많은 열을 지정하는 쿼리에 특히 중요합니다.

- 오름차순 또는 내림차순이 동일한 열과 일치할 때 ORDER BY 절을 가속하기 위해 인덱스를 생성하는 것이 좋습니다. 부분 인덱스는 작기 때문에 선호됩니다. 작은 인덱스를 더 빠르게 읽고 탐색할 수 있습니다.
- null 값을 허용할 수 있는 열에 대한 인덱스를 만드는 경우 null 값을 인덱스의 끝에 저장할지 아니면 인덱스의 시작 부분에 저장할지 고려해야 합니다.

  가능한 경우 결과 집합을 필터링하여 정렬해야 하는 행 수를 줄입니다. WITH 절의 명령문 또는 하위 쿼리를 사용할 경우, 내부 쿼리가 결과 집합을 생성하여 외부 쿼리로 전달한다는 것을 기억하세요. 쿼리가 필터링할 수 있는 행이 많을수록 쿼리 정렬이 할 일이 줄어듭니다.
- 전체 결과 집합을 가져올 필요가 없는 경우 LIMIT 절을 사용하세요. 예를 들어 상위 5개 행만 원하는 경우 LIMIT 절을 사용하는 쿼리는 결과를 계속 생성하지 않습니다. 이렇게 하면 쿼리에 메모리와 임시 파일이 더 적게 필요합니다.

GROUP BY 절을 사용하는 쿼리는 임시 파일이 필요할 수도 있습니다. GROUP BY 쿼리는 다음과 같은 함수를 사용하여 값을 요약합니다.

- COUNT
- AVG
- MIN
- MAX
- SUM
- STDDEV

GROUP BY 쿼리를 튜닝하려면 ORDER BY 쿼리의 권장 사항을 따르세요.

DISTINCT 연산 사용을 피하세요.

가능하면 중복된 행을 제거하기 위해 DISTINCT 연산을 사용하지 마세요. 쿼리가 반환하는 불필요하고 중복된 행이 많을수록 DISTINCT 연산에 시간이 오래 걸리게 됩니다. 가능하면 WHERE 절에 필터를 추가합니다. 다른 테이블에 동일한 필터를 사용하는 경우에도 마찬가지입니다. 쿼리를 필터링하고 조인하면 성능이 향상되고 리소스 사용이 줄어듭니다. 또한 잘못된 보고와 결과를 방지할 수 있습니다.

DISTINCT를 동일 테이블의 여러 행에서 사용해야 하는 경우 복합 인덱스를 만드는 것이 좋습니다. 인덱스에서 여러 열을 그룹화하면 고유한 행을 평가하는 시간을 단축할 수 있습니다. 또한 RDS for PostgreSQL 버전 10 이상을 사용하는 경우, CREATE STATISTICS 명령을 통해 다중 열의 통계를 상호 연관시킬 수 있습니다.

GROUP BY 함수 대신 윈도 함수를 사용하는 것이 좋습니다.

GROUP BY를 사용하여 결과 집합을 변경한 다음 집계된 결과를 검색합니다. 윈도 함수를 사용하면 결과 집합을 변경하지 않고 데이터를 집계할 수 있습니다. 윈도 함수는 OVER 절을 통해 다른 행과 상호 연관시키는 쿼리에 의해 정의된 집합에서 계산을 수행합니다. 윈도 함수의 모든 GROUP BY 함수를 사용할 수 있으며, 다음과 같은 함수도 사용할 수 있습니다.

- RANK
- ARRAY_AGG
- ROW_NUMBER
- LAG
- LEAD

윈도 함수에 의해 생성된 임시 파일 수를 최소화하려면 두 개의 별개의 집계가 필요할 때 동일한 결과 집합에 대한 중복을 제거합니다. 다음과 같은 쿼리를 가정하겠습니다.

```
SELECT sum(salary) OVER (PARTITION BY dept ORDER BY salary DESC) as sum_salary
     , avg(salary) OVER (PARTITION BY dept ORDER BY salary ASC) as avg_salary
  FROM empsalary;
```

WINDOW 절을 사용하여 다음과 같이 쿼리를 다시 작성할 수 있습니다.

```
SELECT sum(salary) OVER w as sum_salary
         , avg(salary) OVER w as_avg_salary
    FROM empsalary
  WINDOW w AS (PARTITION BY dept ORDER BY salary DESC);
```

기본적으로 RDS for PostgreSQL 실행 플래너는 유사한 노드를 통합하므로 연산을 복제하지 않습니다. 그러나 윈도 블록에 대한 명시적 선언을 사용하면 쿼리를 보다 쉽게 유지할 수 있습니다. 또한 중복을 방지하여 성능을 향상시킬 수 있습니다.

구체화된 뷰 및 CTAS 명령문 조사

구체화된 뷰가 새로 고쳐지면 쿼리가 실행됩니다. 이 쿼리에는 GROUP BY, ORDER BY, 또는 DISTINCT 같은 연산이 포함될 수 있습니다. 새로 고침을 하는 동안 많은 수의 임시 파일과 IO:BufFileWrite 및 IO:BufFileRead 대기 이벤트를 관찰할 수 있습니다. 마찬가지로, SELECT 문을 기반으로 테이블을 생성할 때도 마찬가지입니다. CREATE TABLE 문은 쿼리를 실행합니다. 필요한 임시 파일을 줄이려면 쿼리를 최적화하세요.

인덱스를 다시 구축할 때 pg_repack 사용

인덱스를 생성하면 엔진이 결과 세트를 정렬합니다. 테이블의 크기가 커지고 인덱싱된 열의 값이 다양해짐에 따라 임시 파일에 더 많은 공간이 필요합니다. 대부분의 경우 유지 관리 작업 메모리 영역을 수정하지 않으면 큰 테이블에 대한 임시 파일이 생성되지 않도록 할 수 없습니다. `maintenance_work_mem`에 대한 자세한 내용은 PostgreSQL 설명서의 https://www.postgresql.org/docs/current/runtime-config-resource.html을 참조하십시오.

큰 인덱스를 다시 생성할 때 가능한 해결 방법은 pg_repack 확장을 사용하는 것입니다. 자세한 내용은 pg_repack 문서의 최소한의 잠금으로 PostgreSQL 데이터베이스의 테이블 재구성을 참조하세요. RDS for PostgreSQL DB 인스턴스에서 확장을 설정하는 방법은 pg_repack 확장을 사용하여 테이블 및 인덱스에서 부풀림을 줄입니다. 섹션을 참조하세요.

테이블을 클러스터링할 때 maintenance_work_mem 증가

`CLUSTER` 명령은 index_name에 의해 지정된 기존 인덱스를 기반으로 하는 table_name에 의해 지정된 테이블을 클러스터링합니다. RDS for PostgreSQL은 지정된 인덱스의 순서와 일치하도록 테이블을 물리적으로 다시 만듭니다.

마그네틱 스토리지가 널리 보급되었을 때 스토리지 처리량이 제한적이었기 때문에 클러스터링이 일반적이었습니다. SSD 기반 스토리지가 일반적이므로 클러스터링은 인기가 덜합니다. 그러나 테이블을 클러스터링하는 경우에도 테이블 크기, 인덱스, 쿼리 등에 따라 성능이 약간 향상될 수 있습니다.

`CLUSTER` 명령을 실행하거나 `IO:BufFileWrite` 및 `IO:BufFileRead` 대기 이벤트를 관찰할 수 있는 경우, `maintenance_work_mem`을 조정합니다. 메모리 크기를 상당히 크게 늘립니다. 값이 높으면 엔진이 클러스터링 작업에 더 많은 메모리를 사용할 수 있음을 의미합니다.

메모리를 조정하여 IO:BufFileRead 및 IO:BufFileWrite 방지

일부 상황에서는 메모리를 조정해야 합니다. 목표는 다음과 같이 적절한 파라미터를 사용하여 다음 소비 영역에서 메모리 균형을 맞추는 것입니다.

- `work_mem` 값
- `shared_buffers` 값 디스카운트 후 남은 메모리
- 열리고 사용 중인 최대 연결은 `max_connections`로 제한됩니다.

메모리 조정에 대한 자세한 내용은 PostgreSQL 설명서에서 리소스 소비를 참조하세요.

작업 메모리 영역 크기 증가

경우에 따라 세션에서 사용하는 메모리를 늘리는 것이 유일한 방법입니다. 쿼리가 올바르게 작성되고 조인에 올바른 키를 사용하는 경우 work_mem 값을 증가시키는 것이 좋습니다.

쿼리가 생성하는 임시 파일 수를 확인하려면 log_temp_files를 0에 설정하세요. work_mem 값을 로그에서 식별된 최대값으로 늘리면 쿼리가 임시 파일을 생성하지 못하도록 합니다. 하지만 work_mem은 각 연결 또는 병렬 워커에 대해 계획 노드당 최대값을 설정합니다. 데이터베이스에 5,000개의 연결이 있고 각 연결이 256MiB 메모리를 사용하는 경우 엔진에 1.2TiB의 RAM이 필요합니다. 따라서 인스턴스의 메모리가 부족할 수 있습니다.

공유 버퍼 풀에 충분한 메모리 예약

데이터베이스는 작업 메모리 영역뿐만 아니라 공유 버퍼 풀과 같은 메모리 영역을 사용합니다. work_mem을 늘리기 전에 이러한 추가 메모리 영역의 요구 사항을 고려하는 것이 좋습니다.

예를 들어 RDS for PostgreSQL 인스턴스 클래스가 db.r5.2xlarge라고 가정합니다. 이 클래스에는 64GiB의 메모리가 있습니다. 기본적으로 메모리의 25%는 공유 버퍼 풀에 예약되어 있습니다. 공유 메모리 영역에 할당된 양을 빼면 16,384MB가 남아 있습니다. 운영 체제와 엔진에도 메모리가 필요하므로 나머지 메모리를 작업 메모리 영역에만 할당하지 마세요.

work_mem에 할당할 수 있는 메모리는 인스턴스 클래스에 따라 달라집니다. 더 큰 인스턴스 클래스를 사용하는 경우 더 많은 메모리를 사용할 수 있습니다. 하지만 앞의 예에서는 16GiB를 초과하여 사용할 수 없습니다. 그렇지 않으면 메모리가 부족할 때 인스턴스를 사용할 수 없게 됩니다. 인스턴스를 사용할 수 없는 상태에서 복구하기 위해 RDS for PostgreSQL 자동화 서비스가 자동으로 다시 시작됩니다.

연결 수 관리

데이터베이스 인스턴스에 5,000개의 동시 연결이 있다고 가정합니다. 각 연결은 최소 4MiB의 work_mem을 사용합니다. 연결의 메모리 소비량이 많으면 성능이 저하될 수 있습니다. 다음과 같은 옵션이 있습니다.

- vCPU가 더 많은 대규모 인스턴스 클래스로 업그레이드하세요.
- 연결 프록시 또는 풀러를 사용하여 동시 데이터베이스 연결 수를 줄이세요.

프록시의 경우 Amazon RDS 프록시, PGBouncer 또는 애플리케이션에 기반한 연결 풀러를 고려하세요. 이 솔루션은 CPU 부하를 완화합니다. 또한 모든 연결에 작업 메모리 영역이 필요할 때 위험을 줄일 수 있습니다. 데이터베이스 연결 수가 적으면 work_mem 값을 늘릴 수 있습니다. 이런 방법으로 IO:BufFileRead와 IO:BufFileWrite 대기 이벤트의 발생을 줄일 수 있습니다. 또한 작업 메모리 영역을 기다리는 쿼리의 속도가 크게 향상됩니다.

IO:DataFileRead

`IO:DataFileRead` 이벤트는 공유 메모리에서 페이지를 사용할 수 없기 때문에 저장소에서 필요한 페이지를 읽기 위해 백엔드 프로세스에서 연결이 대기할 때 발생합니다.

주제

- 지원되는 엔진 버전
- 컨텍스트
- 대기 증가의 가능한 원인
- 작업

지원되는 엔진 버전

이 대기 이벤트 정보는 모든 RDS for PostgreSQL 버전에서 지원됩니다.

컨텍스트

모든 쿼리 및 데이터 조작(DML) 작업은 버퍼 풀의 페이지에 액세스합니다. 읽기를 유도할 수 있는 명령문은 `SELECT`,`UPDATE` 및 `DELETE`가 있습니다. 예를 들어, 하나의 `UPDATE`는 테이블 또는 인덱스에서 페이지를 읽을 수 있습니다. 요청하거나 업데이트되는 페이지가 공유 버퍼 풀에 없는 경우 이 읽기는 `IO:DataFileRead` 이벤트를 생성합니다.

공유 버퍼 풀은 유한하기 때문에 채워질 수 있습니다. 이 경우 메모리에 없는 페이지에 대한 요청은 데이터베이스가 디스크에서 블록을 읽도록 강제로 합니다. 만약 `IO:DataFileRead` 이벤트가 자주 발생한다면 공유 버퍼 풀이 너무 작아서 워크로드를 수용할 수 없는 것일 수도 있습니다. 이 문제는 버퍼 풀에 맞지 않는 많은 수의 행을 읽는 `SELECT` 쿼리에서 극심하게 발생합니다. 버퍼 풀에 대한 자세한 내용은 PostgreSQL 설명서의 리소스 소비를 참조하세요.

대기 증가의 가능한 원인

`IO:DataFileRead` 이벤트의 일반적인 원인에는 다음이 포함됩니다.

연결 스파이크

동일한 수의 Io:DataFileRead 대기 이벤트를 생성하는 여러 연결을 찾을 수 있습니다. 이 경우 `IO:DataFileRead` 이벤트의 스파이크 갑작스럽고 큰 증가)가 발생할 수 있습니다.

순차 검사를 수행하는 SELECT 및 DML 문

애플리케이션에서 새 작업을 수행하고 있을 수 있습니다. 또는 새 실행 계획으로 인해 기존 작업이 변경될 수 있습니다. 이러한 경우 테이블(특히 큰 테이블)이 더 큰 `seq_scan` 값을 가진 테이블을 찾으세요. `pg_stat_user_tables`를 쿼리하여 탐색 더 많은 읽기 작업을 생성하는 쿼리를 추적하려면 `pg_stat_statements` 확장 프로그램을 사용하세요.

대용량 데이터 세트를 위한 CTAS 및 CREATE 인덱스

CTAS는 `CREATE TABLE AS SELECT` 문입니다. 대용량 데이터 세트를 소스로 사용하여 CTAS를 실행하거나 큰 테이블에 인덱스를 만드는 경우 `IO:DataFileRead` 이벤트가 발생할 수 있습니다. 인덱스를 만들 때 데이터베이스는 순차 스캔을 사용하여 전체 객체를 읽어야 할 수 있습니다. CTAS는 페이지가 메모리에 없을 때 `IO:DataFile` 리드를 생성합니다.

여러 베큠 작업자가 동시에 실행

베큠 작업자는 수동 또는 자동으로 트리거될 수 있습니다. 공격적인 베큠 전략을 채택하는 것이 좋습니다. 그러나 테이블에 업데이트되거나 삭제된 행이 많으면 `IO:DataFileRead` 대기가 늘어납니다. 공간을 회수한 후 `IO:DataFileRead`에 소비되는 베큠 시간이 감소합니다.

대용량 데이터 수집

애플리케이션이 대용량 데이터를 수집할 때 `ANALYZE` 연산이 더 자주 발생할 수 있습니다. `ANALYZE` 프로세스는 autovacuum 시작 관리자에 의해 트리거되거나 수동으로 호출될 수 있습니다.

`ANALYZE` 연산은 테이블의 하위 집합을 읽습니다. 스캔해야 하는 페이지 수는 `default_statistics_target` 값에 30을 곱하여 계산합니다. 자세한 내용은 PostgreSQL 설명서를 참조하세요. `default_statistics_target` 파라미터 1에서 10,000 사이의 값을 허용합니다. 여기서 기본값은 100입니다.

리소스 부족

인스턴스 네트워크 대역폭 또는 CPU가 사용되는 경우 `IO:DataFileRead` 이벤트가 더 자주 발생할 수 있습니다.

작업

대기 이벤트의 원인에 따라 다른 작업을 권장합니다.

주제

- 대기를 생성하는 쿼리에 대한 술어 필터 확인
- 유지 보수 작업의 영향 최소화
- 많은 연결 수에 대응

대기를 생성하는 쿼리에 대한 술어 필터 확인

IO:DataFileRead 대기 이벤트를 생성 중인 특정 쿼리를 식별한다고 가정합니다. 다음 기법을 사용하여 식별할 수 있습니다.

- 성능 개선 도우미
- pg_stat_statements 확장에서 제공하는 것과 같은 카탈로그 뷰
- 카탈로그 뷰 pg_stat_all_tables, 주기적으로 증가하는 물리적 읽기 횟수가 표시되는 경우
- pg_statio_all_tables 뷰, _read 카운터의 증가를 보여주는 경우

이 쿼리의 술어(WHERE 절)에 사용되는 필터를 결정하는 것이 좋습니다. 아래 지침을 따르세요.

- EXPLAIN 명령을 실행합니다. 출력에서 어떤 유형의 스캔이 사용되는지 식별합니다. 순차 스캔이 반드시 문제를 의미하지는 않습니다. 순차 스캔을 사용하는 쿼리는 필터를 사용하는 쿼리와 비교할 때 자연스럽게 더 많은 IO:DataFileRead 이벤트를 생성합니다.

  WHERE 절에 열거된 열이 인덱싱되었는지 확인합니다. 그렇지 않은 경우 이 열에 대한 인덱스를 만드는 것이 좋습니다. 이 접근법은 순차적 스캔을 피하고 IO:DataFileRead 이벤트를 줄입니다. 쿼리에 제한적인 필터가 있지만 순차적 스캔을 생성하는 경우 적절한 인덱스가 사용되고 있는지 평가합니다.
- 쿼리가 매우 큰 테이블에 액세스하고 있는지 확인합니다. 경우에 따라 테이블을 분할하면 성능이 향상되어 쿼리가 필요한 파티션만 읽을 수 있습니다.
- 조인 작업에서 카디널리티(총 행 수)를 조사합니다. WHERE 절을 위해 필터에 전달하는 값이 얼마나 제한적인지 확인하세요. 가능한 경우 쿼리를 조정하여 계획의 각 단계에서 전달되는 행 수를 줄입니다.

유지 보수 작업의 영향 최소화

VACUUM과 ANALYZE 같은 유지 관리 작업이 중요합니다. 이러한 유지 관리 작업과 관련된 IO:DataFileRead 대기 이벤트를 찾을 수 있으므로 끄지 않는 것이 좋습니다. 다음 방법을 사용하면 이러한 작업의 효과를 최소화할 수 있습니다.

- 사용량이 적은 시간대에 수동으로 유지 관리 작업을 실행합니다. 이 기술은 데이터베이스가 자동 연산의 임계값에 도달하지 못하도록 합니다.
- 매우 큰 테이블의 경우 테이블을 분할하는 것이 좋습니다. 이 기술은 유지보수 작업의 오버헤드를 줄여줍니다. 데이터베이스는 유지 관리가 필요한 파티션에만 액세스합니다.
- 대량의 데이터를 수집할 때는 자동 분석 기능을 사용하지 않도록 설정하는 것이 좋습니다.

다음 공식이 true이면 테이블에 대해 autovacuum 기능이 자동으로 트리거됩니다.

```
pg_stat_user_tables.n_dead_tup > (pg_class.reltuples x autovacuum_vacuum_scale_factor)
 + autovacuum_vacuum_threshold
```

뷰 pg_stat_user_tables와 카탈로그 pg_class에는 여러 개의 행이 있습니다. 한 행은 테이블의 한 행에 대응할 수 있습니다. 이 공식은 reltuples가 특정 테이블을 위한 것이라고 가정합니다. 파라미터 autovacuum_vacuum_scale_factor(기본적으로 0.20)와 autovacuum_vacuum_threshold(기본적으로 50개의 튜플)는 일반적으로 전체 인스턴스에 대해 전역으로 설정됩니다. 그러나 특정 테이블에 대해 다른 값을 설정할 수 있습니다.

주제

- 불필요한 공간을 소비하는 테이블 찾기
- 불필요하게 공간을 소비하는 인덱스 찾기
- Autovacuum이 가능한 테이블 찾기

불필요한 공간을 소비하는 테이블 찾기

불필요하게 공간을 소비하는 테이블을 찾으려면 PostgreSQLpgstattuple 확장의 함수를 사용할 수 있습니다. 이 확장(모듈)은 모든 RDS for PostgreSQL DB 인스턴스에서 기본적으로 사용할 수 있으며, 다음 명령을 사용하여 인스턴스에서 인스턴스화할 수 있습니다.

```
CREATE EXTENSION pgstattuple;
```

이 확장에 대한 자세한 내용은 PostgreSQL 설명서의 pgstattuple를 참조하세요.

애플리케이션에서 테이블 및 인덱스 팽창을 검사할 수 있습니다. 자세한 내용은 테이블 및 인덱스 팽창 진단을 참조하세요.

불필요하게 공간을 소비하는 인덱스 찾기

다음 쿼리를 실행하여 부풀려진 인덱스를 찾고 읽기 권한이 있는 테이블에서 불필요하게 소비되는 공간을 추정할 수 있습니다.

```
-- WARNING: rows with is_na = 't' are known to have bad statistics ("name" type is not
 supported).
-- This query is compatible with PostgreSQL 8.2 and later.

SELECT current_database(), nspname AS schemaname, tblname, idxname,
 bs*(relpages)::bigint AS real_size,
  bs*(relpages-est_pages)::bigint AS extra_size,
  100 * (relpages-est_pages)::float / relpages AS extra_ratio,
  fillfactor, bs*(relpages-est_pages_ff) AS bloat_size,
  100 * (relpages-est_pages_ff)::float / relpages AS bloat_ratio,
  is_na
  -- , 100-(sub.pst).avg_leaf_density, est_pages, index_tuple_hdr_bm,
  -- maxalign, pagehdr, nulldatawidth, nulldatahdrwidth, sub.reltuples, sub.relpages
  -- (DEBUG INFO)
FROM (
  SELECT coalesce(1 +
       ceil(reltuples/floor((bs-pageopqdata-pagehdr)/(4+nulldatahdrwidth)::float)), 0
       -- ItemIdData size + computed avg size of a tuple (nulldatahdrwidth)
    ) AS est_pages,
    coalesce(1 +
       ceil(reltuples/floor((bs-pageopqdata-pagehdr)*fillfactor/
(100*(4+nulldatahdrwidth)::float))), 0
    ) AS est_pages_ff,
    bs, nspname, table_oid, tblname, idxname, relpages, fillfactor, is_na
    -- , stattuple.pgstatindex(quote_ident(nspname)||'.'||quote_ident(idxname)) AS
 pst,
    -- index_tuple_hdr_bm, maxalign, pagehdr, nulldatawidth, nulldatahdrwidth,
 reltuples
    -- (DEBUG INFO)
  FROM (
    SELECT maxalign, bs, nspname, tblname, idxname, reltuples, relpages, relam,
 table_oid, fillfactor,
      ( index_tuple_hdr_bm +
          maxalign - CASE -- Add padding to the index tuple header to align on MAXALIGN
            WHEN index_tuple_hdr_bm%maxalign = 0 THEN maxalign
            ELSE index_tuple_hdr_bm%maxalign
          END
        + nulldatawidth + maxalign - CASE -- Add padding to the data to align on
 MAXALIGN
```

```
            WHEN nulldatawidth = 0 THEN 0
            WHEN nulldatawidth::integer%maxalign = 0 THEN maxalign
            ELSE nulldatawidth::integer%maxalign
          END
      )::numeric AS nulldatahdrwidth, pagehdr, pageopqdata, is_na
      -- , index_tuple_hdr_bm, nulldatawidth -- (DEBUG INFO)
    FROM (
      SELECT
        i.nspname, i.tblname, i.idxname, i.reltuples, i.relpages, i.relam, a.attrelid
AS table_oid,
        current_setting('block_size')::numeric AS bs, fillfactor,
        CASE -- MAXALIGN: 4 on 32bits, 8 on 64bits (and mingw32 ?)
          WHEN version() ~ 'mingw32' OR version() ~ '64-bit|x86_64|ppc64|ia64|amd64'
THEN 8
          ELSE 4
        END AS maxalign,
        /* per page header, fixed size: 20 for 7.X, 24 for others */
        24 AS pagehdr,
        /* per page btree opaque data */
        16 AS pageopqdata,
        /* per tuple header: add IndexAttributeBitMapData if some cols are null-able */
        CASE WHEN max(coalesce(s.null_frac,0)) = 0
          THEN 2 -- IndexTupleData size
          ELSE 2 + (( 32 + 8 - 1 ) / 8)
          -- IndexTupleData size + IndexAttributeBitMapData size ( max num filed per
index + 8 - 1 /8)
        END AS index_tuple_hdr_bm,
        /* data len: we remove null values save space using it fractionnal part from
stats */
        sum( (1-coalesce(s.null_frac, 0)) * coalesce(s.avg_width, 1024)) AS
nulldatawidth,
        max( CASE WHEN a.atttypid = 'pg_catalog.name'::regtype THEN 1 ELSE 0 END ) > 0
AS is_na
      FROM pg_attribute AS a
        JOIN (
          SELECT nspname, tbl.relname AS tblname, idx.relname AS idxname,
            idx.reltuples, idx.relpages, idx.relam,
            indrelid, indexrelid, indkey::smallint[] AS attnum,
            coalesce(substring(
              array_to_string(idx.reloptions, ' ')
               from 'fillfactor=([0-9]+)')::smallint, 90) AS fillfactor
          FROM pg_index
            JOIN pg_class idx ON idx.oid=pg_index.indexrelid
            JOIN pg_class tbl ON tbl.oid=pg_index.indrelid
```

```
            JOIN pg_namespace ON pg_namespace.oid = idx.relnamespace
          WHERE pg_index.indisvalid AND tbl.relkind = 'r' AND idx.relpages > 0
        ) AS i ON a.attrelid = i.indexrelid
        JOIN pg_stats AS s ON s.schemaname = i.nspname
          AND ((s.tablename = i.tblname AND s.attname =
 pg_catalog.pg_get_indexdef(a.attrelid, a.attnum, TRUE))
          -- stats from tbl
          OR  (s.tablename = i.idxname AND s.attname = a.attname))
          -- stats from functional cols
        JOIN pg_type AS t ON a.atttypid = t.oid
      WHERE a.attnum > 0
      GROUP BY 1, 2, 3, 4, 5, 6, 7, 8, 9
    ) AS s1
  ) AS s2
    JOIN pg_am am ON s2.relam = am.oid WHERE am.amname = 'btree'
) AS sub
-- WHERE NOT is_na
ORDER BY 2,3,4;
```

Autovacuum이 가능한 테이블 찾기

Autovacuum이 가능한 테이블을 찾으려면 다음 쿼리를 실행합니다.

```
--This query shows tables that need vacuuming and are eligible candidates.
--The following query lists all tables that are due to be processed by autovacuum.
-- During normal operation, this query should return very little.
WITH  vbt AS (SELECT setting AS autovacuum_vacuum_threshold
              FROM pg_settings WHERE name = 'autovacuum_vacuum_threshold')
    , vsf AS (SELECT setting AS autovacuum_vacuum_scale_factor
              FROM pg_settings WHERE name = 'autovacuum_vacuum_scale_factor')
    , fma AS (SELECT setting AS autovacuum_freeze_max_age
              FROM pg_settings WHERE name = 'autovacuum_freeze_max_age')
    , sto AS (SELECT opt_oid, split_part(setting, '=', 1) as param,
                split_part(setting, '=', 2) as value
              FROM (SELECT oid opt_oid, unnest(reloptions) setting FROM pg_class) opt)
SELECT
    '"'||ns.nspname||'"."'||c.relname||'"' as relation
    , pg_size_pretty(pg_table_size(c.oid)) as table_size
    , age(relfrozenxid) as xid_age
    , coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
 autovacuum_freeze_max_age
    , (coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
         coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) *
 c.reltuples)
```

```
          as autovacuum_vacuum_tuples
    , n_dead_tup as dead_tuples
FROM pg_class c
JOIN pg_namespace ns ON ns.oid = c.relnamespace
JOIN pg_stat_all_tables stat ON stat.relid = c.oid
JOIN vbt on (1=1)
JOIN vsf ON (1=1)
JOIN fma on (1=1)
LEFT JOIN sto cvbt ON cvbt.param = 'autovacuum_vacuum_threshold' AND c.oid =
 cvbt.opt_oid
LEFT JOIN sto cvsf ON cvsf.param = 'autovacuum_vacuum_scale_factor' AND c.oid =
 cvsf.opt_oid
LEFT JOIN sto cfma ON cfma.param = 'autovacuum_freeze_max_age' AND c.oid = cfma.opt_oid
WHERE c.relkind = 'r'
AND nspname <> 'pg_catalog'
AND (
    age(relfrozenxid) >= coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
    or
    coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
      coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) * c.reltuples
 <= n_dead_tup
    -- or 1 = 1
)
ORDER BY age(relfrozenxid) DESC;
```

많은 연결 수에 대응

Amazon CloudWatch를 모니터링할 때 DatabaseConnections 지표 스파이크를 확인할 수도 있습니다. 이 증가는 데이터베이스에 대한 연결 수가 증가했음을 나타냅니다. 다음과 같이 하는 것이 좋습니다.

- 애플리케이션이 각 인스턴스에서 열 수 있는 연결 수를 제한합니다. 애플리케이션에 내장된 연결 풀 기능이 있는 경우 적절한 연결 수를 설정합니다. 인스턴스의 vCPU가 효과적으로 병렬화할 수 있는 항목에 따라 숫자를 기준으로 합니다.

  애플리케이션에서 연결 풀 기능을 사용하지 않는 경우 Amazon RDS 프록시 또는 대안을 사용하는 것이 좋습니다. 이 접근 방식을 사용하면 애플리케이션이 로드 밸런서와 여러 연결을 열 수 있습니다. 그런 다음 밸런서는 데이터베이스와의 제한된 수의 연결을 열 수 있습니다. 병렬로 실행되는 연결 수가 적을수록 DB 인스턴스는 커널에서 컨텍스트 전환을 덜 수행합니다. 쿼리는 더 빠르게 진행되므로 대기 이벤트 수가 줄어듭니다. 자세한 내용은 Amazon RDS Proxy 단원을 참조하십시오.

- 가능하면 항상 RDS for PostgreSQL용 읽기 전용 복제본을 활용하세요. 애플리케이션이 읽기 전용 작업을 실행할 때는 이러한 요청을 읽기 전용 복제본으로 보내세요. 이 방법은 기본 (라이터) 노드의 I/O 부담을 줄입니다.
- DB 인스턴스를 확장하는 것이 좋습니다. 대용량 인스턴스 클래스는 더 많은 메모리를 제공하므로 RDS for PostgreSQL에 페이지를 저장할 수 있는 더 큰 공유 버퍼 풀이 제공됩니다. 크기가 클수록 DB 인스턴스에 연결을 처리할 수 있는 vCPU가 늘어납니다. 더 많은 vCPU가 `IO:DataFileRead` 대기 이벤트를 생성 중인 연산이 쓰기를 하고 있을 때 특히 유용합니다.

IO:WALWrite

주제

- 지원되는 엔진 버전
- 컨텍스트
- 대기 증가의 가능한 원인
- 작업

지원되는 엔진 버전

이 대기 이벤트 정보는 RDS for PostgreSQL 버전 10 이상의 모든 버전에서 지원됩니다.

컨텍스트

미리 쓰기 로그 데이터를 생성하는 데이터베이스의 활동은 먼저 WAL 버퍼를 채운 다음 비동기적으로 디스크에 씁니다. 대기 이벤트 `IO:WALWrite`는 SQL 세션이 트랜잭션의 COMMIT 호출을 해제할 수 있도록 WAL 데이터가 디스크에 쓰기를 완료하기를 기다리고 있을 때 생성됩니다.

대기 증가의 가능한 원인

이 대기 이벤트가 자주 발생하는 경우 워크로드와 워크로드가 수행하는 업데이트 유형 및 빈도를 검토해야 합니다. 특히 다음 활동 유형을 찾습니다.

과중한 DML 활동

데이터베이스 테이블의 데이터는 즉시 변경되지 않습니다. 한 테이블에 대한 삽입은 다른 클라이언트가 동일한 테이블을 삽입하거나 업데이트할 때까지 기다려야 할 수 있습니다. 데이터 값 변경(INSERT, UPDATE, DELETE, COMMIT, ROLLBACK TRANSACTION)을 위한 데이터 조작 언어(DML) 문으로 인해 경합이 발생하여 미리 쓰기 로그 파일이 버퍼가 플러시되기를 기다리고 있을 수

있습니다. 이 상황은 과중한 DML 활동을 나타내는 다음과 같은 Amazon RDS 성능 개선 도우미 지표에 캡처됩니다.

- `tup_inserted`
- `tup_updated`
- `tup_deleted`
- `xact_rollback`
- `xact_commit`

이러한 지표에 대한 자세한 내용은 Amazon RDS for PostgreSQL용 성능 개선 도우미 카운터 섹션을 참조하세요.

잦은 체크포인트 활동

체크포인트가 자주 실행되면 WAL 파일의 수가 늘어납니다. RDS for PostgreSQL에서는 전체 페이지 쓰기가 항상 '켜져' 있습니다. 전체 페이지 쓰기는 데이터 손실을 방지하는 데 도움이 됩니다. 하지만 체크포인트가 너무 자주 발생하면 시스템에 전반적인 성능 문제가 발생할 수 있습니다. DML 활동이 많은 시스템에서는 특히 그렇습니다. 경우에 따라 '체크포인트가 너무 자주 발생합니다'라는 오류 메시지가 `postgresql.log`에 표시될 수 있습니다.

체크포인트를 조정할 때는 비정상 종료 시 복구에 필요한 예상 시간에 맞춰 성능을 신중하게 조정하는 것이 좋습니다.

작업

이 대기 이벤트 수를 줄이려면 다음 작업을 수행하는 것이 좋습니다.

주제

- 커밋 수 저감
- 체크포인트 모니터링
- 스케일 업 IO
- 전용 로그 볼륨(DLV)

커밋 수 저감

커밋 수를 줄이려면 명령문을 트랜잭션 블록으로 결합하면 됩니다. Amazon RDS 성능 개선 도우미를 사용하여 실행 중인 쿼리 유형을 검사합니다. 대규모 유지 관리 작업을 사용량이 적은 시간대로 옮길

수도 있습니다. 예를 들어 프로덕션 외 시간에 인덱스를 생성하거나 pg_repack 작업을 사용할 수 있습니다.

체크포인트 모니터링

RDS for PostgreSQL DB 인스턴스가 체크포인트용 WAL 파일에 얼마나 자주 쓰는지 확인하기 위해 모니터링할 수 있는 두 가지 파라미터가 있습니다.

- log_checkpoints - 이 파라미터는 기본적으로 'on'으로 설정되어 있습니다. 그러면 메시지가 각 체크포인트의 PostgreSQL 로그로 전송됩니다. 이러한 로그 메시지에는 기록된 버퍼 수, 쓰기 소요 시간 및 지정된 체크포인트에 대해 추가, 제거 또는 재활용된 WAL 파일 수가 포함됩니다.

  이 파라미터에 대한 자세한 내용은 PostgreSQL 설명서의 오류 보고 및 로깅을 참조하세요.
- checkpoint_warning - 이 파라미터는 초과할 경우 경고가 생성되는 체크포인트 빈도의 임계값(초)을 설정합니다. 기본적으로 이 파라미터는 RDS for PostgreSQL에서 설정되지 않습니다. 이 파라미터의 값을 설정하여 RDS for PostgreSQL DB 인스턴스의 데이터베이스 변경 사항이 WAL 파일이 처리할 크기가 아닌 속도로 작성될 때 경고를 받도록 할 수 있습니다. 예를 들어 이 파라미터를 30으로 설정한다고 가정해 보겠습니다. RDS for PostgreSQL 인스턴스가 30초 빈도보다 더 자주 변경 사항을 작성해야 하는 경우 "체크포인트가 너무 자주 발생합니다"라는 경고가 PostgreSQL 로그로 전송됩니다. 이는 max_wal_size 값을 늘려야 함을 나타낼 수 있습니다.

  자세한 내용은 PostgreSQL 설명서에서 Write Ahead Log를 참조하세요.

스케일 업 IO

이 입출력(IO) 대기 이벤트 유형은 더 빠른 IO를 제공하도록 초당 입출력 작업 처리량(IOPS)을 조정하여 해결할 수 있습니다. CPU를 확장하는 것보다 IO를 확장하는 것이 좋습니다. 늘어난 CPU는 더 많은 작업을 처리할 수 있고 따라서 IO 병목 현상이 더욱 악화될 수 있으므로 CPU를 확장하면 I/O 경합이 더 심해질 수 있기 때문입니다. 일반적으로 크기 조정 작업을 수행하기 전에 워크로드 조정을 고려하는 것이 좋습니다.

전용 로그 볼륨(DLV)

Amazon RDS 콘솔, AWS CLI 또는 Amazon RDS API를 사용하여 프로비저닝된 IOPS(PIOPS) 스토리지를 사용하는 DB 인스턴스 전용 로그 볼륨(DLV)을 사용할 수 있습니다. DLV는 데이터베이스 테이블이 들어 있는 볼륨과 분리된 스토리지 볼륨으로 PostgreSQL 데이터베이스 트랜잭션 로그를 옮깁니다. 자세한 내용은 전용 로그 볼륨(DLV) 단원을 참조하십시오.

IPC:parallel 대기 이벤트

다음 IPC:parallel wait events는 세션이 병렬 쿼리 실행 작업과 관련된 프로세스 간 통신을 기다리고 있음을 나타냅니다.

- IPC:BgWorkerStartup - 병렬 워커 프로세스가 시작 시퀀스를 완료할 때까지 프로세스가 기다리고 있습니다. 병렬 쿼리 실행을 위해 워커를 초기화할 때 발생합니다.
- IPC:BgWorkerShutdown - 병렬 워커 프로세스가 종료 시퀀스를 완료할 때까지 프로세스가 기다리고 있습니다. 병렬 쿼리 실행의 정리 단계에서 발생합니다.
- IPC:ExecuteGather - 프로세스가 쿼리 실행 중에 병렬 워커 프로세스로부터 데이터 수신을 기다리고 있습니다. 리더 프로세스가 워커로부터 결과를 수집해야 할 때 발생합니다.
- IPC:ParallelFinish - 병렬 워커가 실행을 완료하고 최종 결과를 보고할 때까지 프로세스가 기다리고 있습니다. 병렬 쿼리 실행의 완료 단계에서 발생합니다.

주제

- 지원되는 엔진 버전
- 컨텍스트
- 대기 증가의 가능한 원인
- 작업

지원되는 엔진 버전

이 대기 이벤트 정보는 모든 Aurora PostgreSQL 버전에서 지원됩니다.

컨텍스트

PostgreSQL에서 병렬 쿼리를 실행할 때는 단일 쿼리를 처리하기 위해 여러 프로세스가 함께 작동합니다. 쿼리가 병렬화에 적합한 것으로 확인되면 리더 프로세스는 max_parallel_workers_per_gather 파라미터 설정에 따라 하나 이상의 병렬 워커 프로세스와 조정합니다. 리더 프로세스는 워커들을 대상으로 작업을 나누고, 각 워커는 각자가 맡은 데이터를 독립적으로 처리하며, 결과는 리더 프로세스로 다시 수집됩니다.

> **Note**
>
> 각 병렬 워커는 전체 사용자 세션과 유사한 리소스 요구 사항이 있는 별도의 프로세스로 작동합니다. 즉, 워커가 4개인 병렬 쿼리는 비병렬 쿼리에 비해 리소스(CPU, 메모리, I/O 대역폭)를

최대 5배까지 소비할 수 있습니다. 리더 프로세스와 각 워커 프로세스 모두 자체 리소스 할당을 유지하기 때문입니다. 예를 들어, work_mem과 같은 설정은 각 워커에게 개별적으로 적용되어 모든 프로세스에서 총 메모리 사용량을 크게 증가시킬 수 있습니다.

병렬 쿼리 아키텍처는 세 가지 주요 구성 요소로 구성됩니다.

- 리더 프로세스: 병렬 작업을 시작하고, 워크로드를 나누고, 워커 프로세스와 조정하는 주요 프로세스입니다.
- 워커 프로세스: 쿼리의 일부를 병렬로 실행하는 백그라운드 프로세스입니다.
- 수집/수집 병합: 여러 워커 프로세스의 결과를 리더에게 다시 결합하는 작업입니다.

병렬 실행 중에 프로세스는 프로세스 간 통신(IPC) 메커니즘을 통해 서로 통신해야 합니다. 이러한 IPC 대기 이벤트는 여러 단계에서 발생합니다.

- 워커 시작: 병렬 워커가 초기화될 때
- 데이터 교환: 워커가 데이터를 처리하고 리더에게 결과를 보낼 때
- 워커 종료: 병렬 실행이 완료되고 워커가 종료될 때
- 동기화 포인트: 프로세스가 다른 프로세스와 조정하거나 다른 프로세스가 작업을 완료하기를 기다려야 할 때

이러한 대기 이벤트를 이해하는 것은 병렬 쿼리 실행과 관련된 성능 문제를 진단하는 데 매우 중요합니다. 특히 여러 병렬 쿼리가 동시에 실행될 수 있는 동시성이 높은 환경에서는 더욱 그렇습니다.

대기 증가의 가능한 원인

병렬 관련 IPC 대기 이벤트 증가에 여러 요인이 기여할 수 있습니다.

병렬 쿼리의 높은 동시성

많은 병렬 쿼리가 동시에 실행되면 리소스 경합이 발생하고 IPC 작업 대기 시간이 늘어날 수 있습니다. 특히 트랜잭션 볼륨 또는 분석 워크로드가 많은 시스템에서 흔히 발생합니다.

최적화되지 않은 병렬 쿼리 계획

쿼리 플래너가 비효율적인 병렬 계획을 선택하면 불필요한 병렬화 또는 워커 간의 작업 분산 부실이 발생할 수 있습니다. 이로 인해 특히 IPC:ExecuteGather 및 IPC:ParallelFinish 이벤트

의 IPC 대기 시간이 늘어날 수 있습니다. 이러한 계획 문제는 오래된 통계 및 테이블/인덱스 팽창으로 인해 발생하는 경우가 많습니다.

병렬 워커의 빈번한 시작 및 종료

병렬 워커를 자주 시작하고 종료하는 수명이 짧은 쿼리는 `IPC:BgWorkerStartup` 및 `IPC:BgWorkerShutdown` 이벤트를 증가시킬 수 있습니다. 이는 작고 병렬화 가능한 쿼리가 많은 OLTP 워크로드에서 흔히 볼 수 있습니다.

리소스 제약 조건

CPU, 메모리 또는 I/O 용량이 제한되면 병렬 실행에서 병목 현상이 발생하여 모든 IPC 이벤트에서 대기 시간이 늘어날 수 있습니다. 예를 들어 CPU가 포화 상태인 경우 워커 프로세스를 시작하거나 할당된 작업을 처리하는 데 시간이 더 오래 걸릴 수 있습니다.

복잡한 쿼리 구조

여러 수준의 병렬 처리(예: 병렬 조인 후 병렬 집계)가 있는 쿼리는 더 복잡한 IPC 패턴을 초래하고 특히 `IPC:ExecuteGather` 이벤트의 경우 대기 시간이 증가할 수 있습니다.

대규모 결과 세트

대규모 결과 세트를 생성하는 쿼리는 리더 프로세스가 워커 프로세스로부터 결과를 수집하고 처리하는 데 더 많은 시간을 소비하므로 `IPC:ExecuteGather` 대기 시간이 늘어날 수 있습니다.

이러한 요인을 이해하면 Aurora PostgreSQL에서 병렬 쿼리 실행과 관련된 성능 문제를 진단하고 해결하는 데 도움이 될 수 있습니다.

작업

병렬 쿼리와 관련하여 대기가 발생하면 일반적으로 백엔드 프로세스가 병렬 워커 프로세스를 조정하거나 대기 중임을 의미합니다. 이러한 대기는 병렬 계획을 실행하는 동안 흔히 발생합니다. 병렬 워커 사용량을 모니터링하고, 파라미터 설정을 검토하고, 쿼리 실행 및 리소스 할당을 조정하여 이러한 대기의 영향을 조사하고 완화할 수 있습니다.

주제

- 비효율적인 병렬 처리가 있는지 쿼리 계획 분석
- 병렬 쿼리 사용량 모니터링
- 병렬 쿼리 설정 검토 및 조정
- 리소스 할당 최적화
- 연결 관리 조사

- 유지 관리 작업 검토 및 최적화

비효율적인 병렬 처리가 있는지 쿼리 계획 분석

병렬 쿼리 실행은 종종 시스템 불안정, CPU 급증 및 예측할 수 없는 쿼리 성능 변동으로 이어질 수 있습니다. 병렬 처리가 실제로 특정 워크로드를 개선하는지를 철저히 분석하는 것이 중요합니다. EXPLAIN ANALYZE를 사용하여 병렬 쿼리 실행 계획을 검토합니다.

세션 수준에서 병렬 처리를 일시적으로 비활성화하여 계획 효율성을 비교합니다.

```
SET max_parallel_workers_per_gather = 0;
EXPLAIN ANALYZE <your_query>;
```

병렬 처리를 다시 활성화하고 비교합니다.

```
RESET max_parallel_workers_per_gather;
EXPLAIN ANALYZE <your_query>;
```

병렬 처리를 비활성화했을 때 결과가 더 좋거나 더 일관된 경우 SET 명령을 사용하여 세션 수준에서 특정 쿼리에 대해 병렬 처리를 비활성화하는 것이 좋습니다. 더 광범위한 영향을 미치려면 DB 파라미터 그룹에서 관련 파라미터를 조정하여 인스턴스 수준에서 병렬 처리를 비활성화할 수 있습니다. 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요.

병렬 쿼리 사용량 모니터링

다음 쿼리를 사용하여 병렬 쿼리 활동 및 용량에 대한 가시성을 확보합니다.

활성 병렬 워커 프로세스를 확인합니다.

```
SELECT
    COUNT(*)
FROM
    pg_stat_activity
WHERE
    backend_type = 'parallel worker';
```

이 쿼리는 활성 병렬 워커 프로세스의 수를 보여줍니다. 값이 크면 `max_parallel_workers`가 큰 값으로 구성되어 있음을 나타낼 수 있으며 이 값을 줄이는 것이 좋습니다.

동시 병렬 쿼리를 확인합니다.

```
SELECT
    COUNT(DISTINCT leader_pid)
FROM
    pg_stat_activity
WHERE
    leader_pid IS NOT NULL;
```

이 쿼리는 병렬 쿼리를 시작한 고유한 리더 프로세스의 수를 반환합니다. 여기서 숫자가 크면 여러 세션이 병렬 쿼리를 동시에 실행하고 있다는 뜻이며 CPU와 메모리에 대한 수요가 증가할 수 있습니다.

병렬 쿼리 설정 검토 및 조정

다음 파라미터를 검토하여 워크로드에 적절한지 확인합니다.

- `max_parallel_workers`: 모든 세션의 총 병렬 워커 수입니다.
- `max_parallel_workers_per_gather`: 쿼리당 최대 워커 수입니다.

OLAP 워크로드의 경우 이러한 값을 늘리면 성능이 향상될 수 있습니다. OLTP 워크로드의 경우 일반적으로 낮은 값이 선호됩니다.

```
SHOW max_parallel_workers;
SHOW max_parallel_workers_per_gather;
```

리소스 할당 최적화

CPU 사용률을 모니터링하고, 사용률이 지속적으로 높고 애플리케이션이 병렬 쿼리의 이점을 누릴 수 있는 경우 vCPU 수를 조정하는 것이 좋습니다. 병렬 작업에 적절한 메모리를 사용할 수 있는지 확인합니다.

- 성능 개선 도우미 지표를 사용하여 시스템이 CPU 바운드인지 확인합니다.

- 각 병렬 워커는 자체 work_mem을 사용합니다. 총 메모리 사용량이 인스턴스 한도 내에 있는지 확인합니다.

병렬 쿼리는 비병렬 쿼리보다 훨씬 많은 리소스를 사용할 수 있습니다. 각 워커 프로세스는 시스템에 추가 사용자 세션과 거의 동일한 영향을 미치는 완전히 별개의 프로세스이기 때문입니다. 이 설정에 대한 값을 선택할 때, 그리고 work_mem과 같이 리소스 사용률을 제어하는 다른 설정을 구성할 때 이 점을 고려해야 합니다. 자세한 내용은 PostgreSQL 설명서를 참조하세요. work_mem과 같은 리소스 한도는 각 워커에 개별적으로 적용됩니다. 즉, 전체 사용률이 단일 프로세스보다 모든 프로세스에서 훨씬 높을 수 있습니다.

워크로드가 심하게 병렬화된 경우 vCPU를 늘리거나 메모리 파라미터를 조정하는 것이 좋습니다.

연결 관리 조사

연결 소진이 발생하는 경우 애플리케이션 연결 풀링 전략을 검토합니다. 아직 사용하지 않는 경우 애플리케이션 수준에서 연결 풀링을 구현하는 것이 좋습니다.

유지 관리 작업 검토 및 최적화

인덱스 생성 및 기타 유지 관리 작업을 조정하여 리소스 경합을 방지합니다. 이러한 작업은 사용량이 적은 시간으로 예약하는 것이 좋습니다. 사용자 쿼리 로드가 많은 기간에는 과도한 유지 관리(예: 병렬 인덱스 빌드)를 예약하지 마세요. 이러한 작업은 병렬 워커를 소비하고 정규 쿼리의 성능에 영향을 미칠 수 있습니다.

Lock:advisory

Lock:advisory 이벤트는 PostgreSQL 애플리케이션이 잠금을 사용하여 여러 세션에서 활동을 조정할 때 발생합니다.

주제

- 관련 엔진 버전
- 컨텍스트
- 원인
- 작업

관련 엔진 버전

이 대기 이벤트 정보는 RDS for PostgreSQL 버전 9.6 이상과 관련이 있습니다.

컨텍스트

PostgreSQL 권고 잠금은 사용자의 애플리케이션 코드에 의해 명시적으로 잠기거나 잠금 해제된 애플리케이션 수준의 협력 잠금입니다. 애플리케이션은 PostgreSQL 권고 잠금을 사용하여 여러 세션에서 활동을 조정할 수 있습니다. 일반, 객체 또는 행 레벨 잠금과는 달리 애플리케이션은 잠금 수명을 완벽하게 제어할 수 있습니다. 자세한 내용은 PostgreSQL 설명서의 권고 잠금을 참조하세요.

권고 잠금은 트랜잭션이 종료되기 전에 해제되거나 트랜잭션 간에 세션에 의해 유지될 수 있습니다. `CREATE INDEX` 문에 의해 획득한 테이블에 대한 액세스 독점 잠금과 같은 암시적 시스템 적용 잠금에는 해당되지 않습니다.

권고 잠금을 획득(잠금) 및 릴리즈(잠금 해제) 하는 데 사용되는 함수에 대한 설명은 PostgreSQL 설명서의 권고 잠금 함수을 참조하세요.

명시적인 잠금은 일반 PostgreSQL 잠금 시스템에 구현되며 `pg_locks` 시스템 뷰에서 볼 수 있습니다.

원인

이 잠금 유형은 명시적으로 사용하는 애플리케이션에 의해서만 제어됩니다. 쿼리의 일부로 각 행에 대해 획득된 권고 사항 잠금은 잠금이 급증하거나 장기 축적을 일으킬 수 있습니다.

이러한 효과는 쿼리에서 반환되는 것보다 많은 행에 대한 잠금을 획득하는 방식으로 쿼리가 실행될 때 발생합니다. 애플리케이션은 결국 모든 잠금을 해제해야 하지만 반환되지 않은 행에 잠금을 획득하면 애플리케이션에서 모든 잠금을 찾을 수 없습니다.

다음 예는 PostgreSQL 설명서의 권고 잠금에 자세히 설명되어 있습니다.

```
SELECT pg_advisory_lock(id) FROM foo WHERE id > 12345 LIMIT 100;
```

이 예에서, `LIMIT` 절은 행이 이미 내부적으로 선택되고 ID 값이 잠긴 후에만 쿼리의 출력을 중지할 수 있습니다. 이는 데이터 볼륨이 증가함에 따라 플래너가 개발 중에 테스트되지 않은 다른 실행 계획을 선택하게 되면 갑자기 발생할 수 있습니다. 이 경우 빌드업은 애플리케이션이 잠긴 모든 ID 값에 대해 `pg_advisory_unlock`을 명시적으로 호출하기 때문에 발생합니다. 그러나 이 경우 반환되지 않은 행에서 획득한 잠금 집합을 찾을 수 없습니다. 잠금은 세션 수준에서 획득되기 때문에 트랜잭션이 끝날 때 자동으로 해제되지 않습니다.

차단된 잠금 시도에서 스파이크가 발생할 수 있는 또 다른 원인은 의도하지 않은 충돌입니다. 이러한 충돌에서 애플리케이션의 관련 없는 부분은 실수로 동일한 잠금 ID 공간을 공유합니다.

작업

권고 잠금의 애플리리케이션 사용을 검토하고 애플리케이션 흐름에서 각 유형의 권고 잠금을 획득하고 해제하는 위치와 시기를 자세히 검토합니다.

세션이 너무 많은 잠금을 획득하고 있는지 아니면 장기 실행 세션에서 잠금을 조기에 해제하지 않아 잠금이 느리게 축적되는지 확인합니다. pg_terminate_backend(pid)를 사용하는 세션을 종료하여 세션 수준 잠금의 느린 축적을 수정할 수 있습니다.

권고 잠금을 대기하는 클라이언트는 pg_stat_activity에서 wait_event_type=Lock 및 wait_event=advisory와 표시됩니다. locktype=advisory 및 granted=f를 탐색하며 동일한 pid를 위해 pg_locks 시스템 뷰를 쿼리하여 특정 잠금 값을 얻을 수 있습니다.

그런 다음, 예에 표시된 대로 granted=t를 가진 동일한 명시적인 잠금을 위해 pg_locks를 쿼리하여 차단 세션을 식별할 수 있습니다.

```
SELECT blocked_locks.pid AS blocked_pid,
         blocking_locks.pid AS blocking_pid,
         blocked_activity.usename AS blocked_user,
         blocking_activity.usename AS blocking_user,
         now() - blocked_activity.xact_start AS blocked_transaction_duration,
         now() - blocking_activity.xact_start AS blocking_transaction_duration,
         concat(blocked_activity.wait_event_type,':',blocked_activity.wait_event) AS
 blocked_wait_event,
         concat(blocking_activity.wait_event_type,':',blocking_activity.wait_event) AS
 blocking_wait_event,
         blocked_activity.state AS blocked_state,
         blocking_activity.state AS blocking_state,
         blocked_locks.locktype AS blocked_locktype,
         blocking_locks.locktype AS blocking_locktype,
         blocked_activity.query AS blocked_statement,
         blocking_activity.query AS blocking_statement
    FROM pg_catalog.pg_locks blocked_locks
    JOIN pg_catalog.pg_stat_activity blocked_activity ON blocked_activity.pid =
 blocked_locks.pid
    JOIN pg_catalog.pg_locks blocking_locks
        ON blocking_locks.locktype = blocked_locks.locktype
        AND blocking_locks.DATABASE IS NOT DISTINCT FROM blocked_locks.DATABASE
        AND blocking_locks.relation IS NOT DISTINCT FROM blocked_locks.relation
        AND blocking_locks.page IS NOT DISTINCT FROM blocked_locks.page
        AND blocking_locks.tuple IS NOT DISTINCT FROM blocked_locks.tuple
        AND blocking_locks.virtualxid IS NOT DISTINCT FROM blocked_locks.virtualxid
```

```
        AND blocking_locks.transactionid IS NOT DISTINCT FROM
blocked_locks.transactionid
        AND blocking_locks.classid IS NOT DISTINCT FROM blocked_locks.classid
        AND blocking_locks.objid IS NOT DISTINCT FROM blocked_locks.objid
        AND blocking_locks.objsubid IS NOT DISTINCT FROM blocked_locks.objsubid
        AND blocking_locks.pid != blocked_locks.pid
    JOIN pg_catalog.pg_stat_activity blocking_activity ON blocking_activity.pid =
blocking_locks.pid
    WHERE NOT blocked_locks.GRANTED;
```

모든 명시적인 잠금 API 함수에는 두 개의 인수 집합이 있으며, 하나의 `bigint` 인수 또는 두 개의 `integer` 인수입니다.

- 하나의 API 함수의 경우 `bigint` 인수, 상위 32비트는`pg_locks.classid` 인수, 더 낮은 32비트는 `pg_locks.objid` 인수입니다.
- 두 개가 포함된 API 함수의 경우 `integer` 인수, 첫 번째 인수는 `pg_locks.classid`, 두 번째 인수는 `pg_locks.objid`입니다.

`pg_locks.objsubid` 값은 어떤 API 양식을 사용했는지를 나타냅니다. 1은 하나의 `bigint` 인수를 의미하며, 2는 두 개의 `integer` 인수를 의미합니다.

Lock:extend

`Lock:extend` 이벤트는 백엔드 프로세스가 릴레이션을 확장하기 위해 릴레이션을 잠그기를 기다리는 동안 다른 프로세스에서 동일한 목적으로 해당 릴레이션에 대한 잠금이 있는 경우에 발생합니다.

주제

- 지원되는 엔진 버전
- 컨텍스트
- 대기 증가의 가능한 원인
- 작업

지원되는 엔진 버전

이 대기 이벤트 정보는 모든 RDS for PostgreSQL 버전에서 지원됩니다.

컨텍스트

이벤트 Lock:extend는 백엔드 프로세스가 관계를 확장하는 동안 다른 백엔드 프로세스가 잠금을 유지하는 관계를 확장하기 위해 대기 중임을 나타냅니다. 한 번에 하나의 프로세스만 관계를 확장할 수 있기 때문에 시스템은 Lock:extend 대기 이벤트를 생성합니다. INSERT,COPY, 및 UPDATE 연산은 이 이벤트를 생성할 수 있습니다.

대기 증가의 가능한 원인

Lock:extend 이벤트가 정상보다 많이 나타나 성능 문제를 나타내는 경우 일반적인 원인은 다음과 같습니다.

동일한 테이블에 대한 동시 삽입 또는 업데이트 서지

동일한 테이블에 삽입하거나 업데이트하는 쿼리의 동시 세션 수가 증가할 수 있습니다.

네트워크 대역폭 부족

DB 인스턴스의 네트워크 대역폭이 현재 워크로드의 스토리지 통신 요구 사항에 따라 충분하지 않을 수 있습니다. 이로 인해 Lock:extend 이벤트에서 스토리지 지연 시간이 증가할 수 있습니다.

작업

대기 이벤트의 원인에 따라 다른 작업을 권장합니다.

주제

- 동일한 관계식으로 동시 삽입 및 업데이트 감소
- 네트워크 대역폭 향상

동일한 관계식으로 동시 삽입 및 업데이트 감소

먼저, 증가가 있는지 확인합니다. tup_inserted와 tup_updated 지표 및 이 대기 이벤트의 증가가 수반됩니다. 그렇다면 삽입 및 업데이트 작업에 대한 경합이 높은 관계식을 확인합니다. 이를 확인하려면 n_tup_ins 및 n_tup_upd 필드의 값을 위한 pg_stat_all_tables 뷰를 쿼리하세요. pg_stat_all_tables 뷰에 대한 자세한 내용은 PostgreSQL 설명서의 pg_stat_statements를 참조하세요.

차단 및 차단된 쿼리에 대한 자세한 정보를 위해서 다음 예와 같이 pg_stat_activity를 쿼리하세요.

```
SELECT
    blocked.pid,
    blocked.usename,
    blocked.query,
    blocking.pid AS blocking_id,
    blocking.query AS blocking_query,
    blocking.wait_event AS blocking_wait_event,
    blocking.wait_event_type AS blocking_wait_event_type
FROM pg_stat_activity AS blocked
JOIN pg_stat_activity AS blocking ON blocking.pid = ANY(pg_blocking_pids(blocked.pid))
where
blocked.wait_event = 'extend'
and blocked.wait_event_type = 'Lock';

   pid  | usename  |            query             | blocking_id |
            blocking_query                          | blocking_wait_event |
 blocking_wait_event_type
  ------+----------+------------------------------+-------------
+-------------------------------------------------------------------
+---------------------+--------------------------
   7143 |  myuser  | insert into tab1 values (1); |        4600 | INSERT INTO tab1 (a)
 SELECT s FROM generate_series(1,1000000) s; | DataFileExtend      | IO
```

Lock:extend 이벤트 증가에 기여하는 관계를 파악한 후 다음 기술을 사용하여 경합을 줄입니다.

- 파티셔닝을 사용하여 동일한 테이블에 대한 경합을 줄일 수 있는지 확인합니다. 삽입되거나 업데이트된 튜플을 다른 파티션으로 분리하면 경합을 줄일 수 있습니다. 파티셔닝에 대한 자세한 내용은 pg_partman 확장자를 사용하여 PostgreSQL 파티션 관리하기 섹션을 참조하세요.
- 대기 이벤트가 주로 업데이트 활동으로 인한 경우 관계식의 채우기 요소 값을 줄이는 것이 좋습니다. 이렇게 하면 업데이트 중에 새 블록에 대한 요청을 줄일 수 있습니다. 필팩터는 테이블 페이지를 포장하기 위한 최대 공간을 결정하는 테이블의 저장 파라미터입니다. 이 값은 페이지의 전체 공간의 백분율로 표시됩니다. 필팩터 파라미터에 대한 자세한 내용은 PostgreSQL 설명서의 테이블 생성을 참조하세요.

> **Important**
>
> 필팩터를 변경하는 경우 이 값을 변경하면 워크로드에 따라 성능에 부정적인 영향을 줄 수 있으므로 시스템을 테스트하는 것이 좋습니다.

네트워크 대역폭 향상

쓰기 지연 시간이 증가하는지 확인하려면 CloudWatch의 `WriteLatency` 지표를 확인하세요. 그렇다면, `WriteThroughput` 및 `ReadThroughput` Amazon CloudWatch 지표를 사용하여 DB 인스턴스의 스토리지 관련 트래픽을 모니터링하세요. 이러한 지표는 네트워크 대역폭이 워크로드의 스토리지 활동에 충분한지 확인하는 데 도움이 될 수 있습니다.

네트워크 대역폭이 충분하지 않으면 늘리세요. 만약 클라이언트 또는 DB 인스턴스가 네트워크 대역폭 제한에 도달했다면. 대역폭을 늘리는 유일한 방법은 DB 인스턴스 크기를 늘리는 것입니다.

CloudWatch 지표에 대한 자세한 내용은 Amazon RDS에 대한 Amazon CloudWatch 지표 섹션을 참조하세요. 각 DB 인스턴스 클래스에 대한 네트워크 성능에 관한 자세한 내용은 Amazon RDS에 대한 Amazon CloudWatch 지표 섹션을 참조하세요.

Lock:Relation

`Lock:Relation` 이벤트는 쿼리가 현재 다른 트랜잭션에 의해 잠긴 테이블 또는 뷰(관계식)에 대한 잠금을 얻기 위해 대기 중일 때 발생합니다.

주제

- 지원되는 엔진 버전
- 컨텍스트
- 대기 증가의 가능한 원인
- 작업

지원되는 엔진 버전

이 대기 이벤트 정보는 모든 RDS for PostgreSQL 버전에서 지원됩니다.

컨텍스트

대부분의 PostgreSQL 명령은 암시적으로 잠금을 사용하여 테이블의 데이터에 대한 동시 액세스를 제어합니다. 애플리케이션 코드에서 `LOCK` 명령과 함께 이러한 잠금을 명시적으로 사용할 수도 있습니다. 많은 잠금 모드는 서로 호환되지 않으며 동일한 객체에 액세스하려고 할 때 트랜잭션을 차단할 수 있습니다. 이 경우 RDS for PostgreSQL은 `Lock:Relation` 이벤트를 생성합니다. 다음은 몇 가지 일반적인 예입니다.

- `ACCESS EXCLUSIVE`와 같은 독점 잠금은 모든 동시 액세스를 차단할 수 있습니다. 데이터 정의 언어(DDL) 작업(예: `DROP TABLE`,`TRUNCATE`,`VACUUM FULL`, 및`CLUSTER`)은 명시적으로 `ACCESS`

EXCLUSIVE 잠금을 획득합니다. 또한 ACCESS EXCLUSIVE는 모드를 명시적으로 식별하지 않는 LOCK TABLE 문을 위한 기본 잠금 모드입니다.

- 데이터 조작 언어(DML) 명령문 UPDATE,DELETE, 및 INSERT와 충돌하는 테이블에 ROW EXCLUSIVE 잠금을 획득하는 CREATE INDEX (without CONCURRENT)를 사용하세요.

테이블 수준 잠금 및 충돌하는 잠금 모드에 대한 자세한 내용은 PostgreSQL 설명서의 명시적 잠금을 참조하세요.

쿼리 및 트랜잭션 차단은 일반적으로 다음 중 한 가지 방법으로 잠금 해제합니다.

- 쿼리 차단 - 애플리케이션이 쿼리를 취소하거나 사용자가 프로세스를 종료할 수 있습니다. 세션의 명령문 시간 초과 또는 교착 상태 감지 메커니즘으로 인해 엔진이 쿼리를 강제로 종료할 수도 있습니다.
- 트랜잭션 차단 - 트랜잭션이 ROLLBACK이나 COMMIT 문을 실행할 때 차단을 중지합니다. 롤백은 클라이언트 또는 네트워크 문제로 세션의 연결이 끊어지거나 종료된 경우에도 자동으로 수행됩니다. 세션은 데이터베이스 엔진이 종료될 때, 시스템의 메모리가 부족할 때 종료될 수 있습니다.

대기 증가의 가능한 원인

Lock:Relation 이벤트가 평소보다 더 자주 발생하면 성능 문제를 나타낼 수 있습니다. 일반적인 원인은 다음과 같습니다.

테이블 잠금 충돌로 인한 동시 세션 증가

잠금 모드가 충돌하는 동일한 테이블을 하거나 잠그는 쿼리의 동시 세션 수가 증가할 수 있습니다.

유지 관리 작업

VACUUM과 ANALYZE 같은 상태 유지 관리 작업은 충돌하는 잠금 수를 크게 늘릴 수 있습니다. VACUUM FULL은 하나의 ACCESS EXCLUSIVE 잠금을, ANALYSE는 하나의 SHARE UPDATE EXCLUSIVE 잠금을 획득합니다. 두 가지 유형의 잠금은 모두 Lock:Relation 대기 이벤트를 발생시킬 수 있습니다. 구체화된 뷰 새로 고침과 같은 애플리케이션 데이터 유지 관리 작업은 차단된 질의 및 트랜잭션을 증가시킬 수도 있습니다.

리더 인스턴스 잠금

라이터와 리더가 보유한 관계 잠금 간에 충돌이 있을 수 있습니다. 현재 ACCESS EXCLUSIVE 관계 잠금만 리더 인스턴스에 복제됩니다. 그러나 ACCESS EXCLUSIVE 관계 잠금은 리더가 보유한 모든 ACCESS SHARE 관계 잠금과 충돌합니다. 이로 인해 리더에서 잠금 관계 대기 이벤트가 증가할 수 있습니다.

작업

대기 이벤트의 원인에 따라 다른 작업을 권장합니다.

주제

- SQL 문 차단의 영향 감소
- 유지 보수 작업의 영향 최소화

SQL 문 차단의 영향 감소

SQL 문 차단의 영향을 줄이려면 가능한 경우 애플리케이션 코드를 수정하세요. 다음은 블록을 줄이기 위한 두 가지 일반적인 기술입니다.

- NOWAIT 옵션 사용 - SELECT나 LOCK 문 같은 일부 SQL 명령은 이 옵션을 지원합니다. NOWAIT 지시어는 잠금을 즉시 획득할 수 없는 경우 잠금 요청 쿼리를 취소합니다. 이 기술은 차단 세션이 그 뒤에 차단된 세션이 쌓이지 않게 하는 데 도움이 될 수 있습니다.

  예: 트랜잭션 A가 트랜잭션 B가 보유한 잠금을 기다리고 있다고 가정합니다. 이제 B가 트랜잭션 C에 의해 잠긴 테이블에 대한 잠금을 요청하면 트랜잭션 C가 완료될 때까지 트랜잭션 A가 차단될 수 있습니다. 그러나 트랜잭션 B가 NOWAIT를 사용하는 경우 C에서 잠금을 요청하면 빠르게 실패하고 트랜잭션 A가 계속해서 기다릴 필요가 없도록 할 수 있습니다.
- SET lock_timeout 사용 - SQL 문이 관계식 잠금을 획득하기 위해 대기하는 시간을 제한하는 lock_timeout 값을 설정하세요. 지정된 제한 시간 내에 잠금을 획득하지 않으면 잠금을 요청하는 트랜잭션이 취소됩니다. 세션 수준에서 이 값을 설정합니다.

유지 보수 작업의 영향 최소화

VACUUM과 ANALYZE 같은 유지 관리 작업이 중요합니다. 이러한 유지 관리 작업과 관련된 Lock:Relation 대기 이벤트를 찾을 수 있으므로 끄지 않는 것이 좋습니다. 다음 방법을 사용하면 이러한 작업의 효과를 최소화할 수 있습니다.

- 사용량이 적은 시간대에 수동으로 유지 관리 작업을 실행합니다.
- Autovacuum 작업으로 인한 Lock:Relation 대기를 줄이려면 필요한 autovacuum 튜닝을 수행하세요. Autovacuum 튜닝에 대한 자세한 내용은 Amazon RDS 사용 설명서의 Amazon RDS에서 PostgreSQL Autovacuum 사용을 참조하세요.

Lock:transactionid

Lock:transactionid 이벤트는 트랜잭션이 행 수준 잠금을 대기 중일 때 발생합니다.

주제

- 지원되는 엔진 버전
- 컨텍스트
- 대기 증가의 가능한 원인
- 작업

지원되는 엔진 버전

이 대기 이벤트 정보는 모든 RDS for PostgreSQL 버전에서 지원됩니다.

컨텍스트

이벤트 Lock:transactionid 트랜잭션이 동시에 실행 중인 트랜잭션에 이미 부여된 행 수준 잠금을 획득하려고 할 때 발생합니다. Lock:transactionid 대기 이벤트를 보여주는 세션이 이 잠금으로 인해 차단되었습니다. 차단 트랜잭션이 종료된 후 COMMIT 또는ROLLBACK 문, 차단된 트랜잭션을 진행할 수 있습니다.

RDS for PostgreSQL의 다중 버전 동시성 제어 의미 체계는 리더가 라이터를 차단하지 않고 라이터는 리더를 차단하지 않도록 보장합니다. 행 수준 충돌이 발생하려면 차단 및 차단된 트랜잭션이 다음 유형의 충돌하는 명령문을 실행해야 합니다.

- UPDATE
- SELECT … FOR UPDATE
- SELECT … FOR KEY SHARE

명령문 SELECT … FOR KEY SHARE는 특별한 경우입니다. 데이터베이스가 FOR KEY SHARE 절을 사용해 참조 무결성의 성능을 최적화합니다. 행의 행 수준 잠금은 행을 참조하는 다른 테이블에서 INSERT, UPDATE, DELETE 명령을 차단할 수 있습니다.

대기 증가의 가능한 원인

이 이벤트가 정상보다 많이 나타나는 경우 일반적으로 원인은 다음 조건과 결합된 UPDATE, SELECT … FOR UPDATE, 또는 SELECT … FOR KEY SHARE 문입니다.

주제

- 높은 동시성
- 트랜잭션의 유휴 상태
- 장기 실행 트랜잭션

높은 동시성

RDS for PostgreSQL은 세분화된 행 수준 잠금 의미 체계를 사용할 수 있습니다. 다음 조건이 충족되면 행 수준 충돌 확률이 높아집니다.

- 동일한 행에 대해 높은 동시 워크로드가 경합됩니다.
- 동시성이 증가합니다.

트랜잭션의 유휴 상태

가끔 `pg_stat_activity.state` 열에 `idle in transaction` 값이 표시됩니다. 이 값은 트랜잭션을 시작했지만 아직 `COMMIT` 또는 `ROLLBACK`을 실행하지 않은 세션에 대해 표시됩니다. 만약 `pg_stat_activity.state` 값이 `active`가 아니라면, `pg_stat_activity`에 표시된 쿼리는 실행을 마친 가장 최근의 쿼리입니다. 차단 세션은 열려 있는 트랜잭션이 잠금을 유지하고 있기 때문에 쿼리를 능동적으로 처리하지 않습니다.

유휴 트랜잭션이 행 수준 잠금을 획득한 경우 다른 세션에서 잠금을 획득하지 못할 수 있습니다. 이 조건으로 인해 `Lock:transactionid` 대기 이벤트가 자주 발생합니다. 문제를 진단하려면 `pg_stat_activity`와 `pg_locks`의 출력을 검사하세요.

장기 실행 트랜잭션

오랫동안 실행되는 트랜잭션은 오랜 시간 동안 잠금을 얻습니다. 이러한 장기 잠금은 다른 트랜잭션의 실행을 차단할 수 있습니다.

작업

행 잠금은 `UPDATE`, `SELECT … FOR UPDATE`, 또는 `SELECT … FOR KEY SHARE` 문 사이의 충돌입니다. 솔루션을 시도하기 전에 이러한 명령문이 동일한 행에서 실행되는 시점을 확인하세요. 다음 섹션에 설명된 전략을 선택하려면 이 정보를 사용하세요.

주제

- 동시성에 대응

- 유휴 트랜잭션에 대응
- 장기 실행 트랜잭션에 대응

동시성에 대응

동시성이 문제가 되는 경우 다음 기술 중 하나를 시도해 보세요.

- 애플리케이션의 동시성을 낮춥니다. 예를 들어 활성 세션 수를 줄일 수 있습니다.
- 연결 풀을 구현합니다. RDS 프록시로 연결을 풀링하는 방법에 대한 자세한 내용은 Amazon RDS Proxy 섹션을 참조하세요.
- UPDATE 문과 SELECT … FOR UPDATE 문의 경쟁 방지를 위한 애플리케이션 또는 데이터 모델을 설계하세요. SELECT … FOR KEY SHARE 문으로 액세스하는 외래 키 수를 줄일 수도 있습니다.

유휴 트랜잭션에 대응

pg_stat_activity.state가 idle in transaction을 나타낸다면, 다음 전략을 시도해 보세요.

- 가능하면 자동 커밋을 켭니다. 이 접근 방식은 트랜잭션이 COMMIT이나 ROLLBACK을 대기하는 동안 다른 트랜잭션을 차단하는 것을 방지합니다.
- COMMIT, ROLLBACK, 또는 END가 누락된 코드 경로 검색
- 애플리케이션의 예외 처리 논리에 항상 유효한 end of transaction으로 향하는 경로가 있는지 확인하세요.
- COMMIT 및 ROLLBACK과의 트랜잭션을 종료한 후 애플리케이션이 쿼리 결과를 처리하는지 확인하세요.

장기 실행 트랜잭션에 대응

장기 실행 트랜잭션으로 인해 Lock:transactionid가 자주 발생하는 경우 다음 전략을 시도해 보세요.

- 장기 실행 트랜잭션에서 행 잠금을 차단합니다.
- 가능하면 자동 커밋을 구현하여 쿼리 길이를 제한합니다.

Lock:tuple

Lock:tuple 이벤트는 백엔드 프로세스가 튜플에 대한 잠금 획득을 대기 중일 때 발생합니다.

주제

- 지원되는 엔진 버전
- 컨텍스트
- 대기 증가의 가능한 원인
- 작업

지원되는 엔진 버전

이 대기 이벤트 정보는 모든 RDS for PostgreSQL 버전에서 지원됩니다.

컨텍스트

이벤트 `Lock:tuple`은 다른 백엔드가 동일한 튜플에서 충돌하는 잠금을 보유하는 동안 백엔드가 튜플에 대한 잠금을 얻기 위해 대기 중임을 나타냅니다. 다음 테이블에서는 세션이 `Lock:tuple` 이벤트를 생성하는 시나리오를 가정합니다.

Time	세션 1	세션 2	세션 3
t1	트랜잭션을 시작합니다.		
t2	1행을 업데이트합니다.		
t3		1행을 업데이트합니다. 세션은 튜플에 대한 배타적 잠금을 획득한 다음 세션 1이 커밋하거나 롤백하여 잠금을 해제할 때까지 기다립니다.	
t4			1행을 업데이트합니다. 세션은 세션 2가 튜플에서 배타적 잠금을 해제할 때까지 기다립니다.

또는 벤치마킹 도구 pgbench를 사용하여 이 대기 이벤트를 시뮬레이션할 수 있습니다. 테이블의 동일한 행을 사용자 정의 SQL 파일로 업데이트하도록 많은 수의 동시 세션을 구성합니다.

충돌하는 잠금 모드에 대한 자세한 내용은 PostgreSQL 설명서의 명시적 잠금을 참조하세요. pgbench에 관한 더 자세한 내용은 PostgreSQL 설명서의 pgbench 섹션을 참조하세요.

대기 증가의 가능한 원인

이 이벤트가 정상보다 많이 발생해 성능 문제를 일으킬 수 있는 경우 일반적인 원인은 다음과 같습니다.

- 많은 수의 동시 세션이 UPDATE 또는 DELETE 문을 실행하여 동일한 튜플에 대해 충돌하는 잠금을 얻으려고 시도합니다.
- 높은 동시 세션이 FOR UPDATE 또는 FOR NO KEY UPDATE 잠금 모드를 통해 SELECT 문을 실행하고 있습니다.
- 다양한 요인으로 인해 애플리케이션이나 연결 풀이 더 많은 세션을 열어 동일한 작업을 실행할 수 있습니다. 새 세션이 동일한 행을 수정하려고 하면 DB 로드가 급증할 수 있으며, Lock:tuple이 표시될 수 있습니다.

자세한 내용은 PostgreSQL 설명서의 행 수준 잠금을 참조하세요.

작업

대기 이벤트의 원인에 따라 다른 작업을 권장합니다.

주제

- 애플리케이션 로직 조사
- 차단 세션 찾기
- 높은 동시성 저감
- 병목 현상 해결

애플리케이션 로직 조사

차단 세션이 오랜 시간 동안 idle in transaction 상태인지 확인하세요. 그렇다면 차단 세션을 단기 솔루션으로 종료하는 것이 좋습니다. pg_terminate_backend 함수를 사용할 수 있습니다. 이 함수에 대한 자세한 내용은 PostgreSQL 설명서의 서버 신호 전송 함수를 참조하세요.

장기 솔루션의 경우 다음을 수행합니다.

- 애플리케이션 로직을 조정합니다.
- idle_in_transaction_session_timeout 파라미터를 사용합니다. 이 파라미터는 지정된 시간보다 오랫동안 유휴 상태인 열린 트랜잭션으로 세션을 종료합니다. 자세한 내용은 PostgreSQL 설명서의 클라이언트 인증을 참조하세요.

- autocommit을 최대한 많이 사용합니다. 자세한 내용은 PostgreSQL 설명서의 AUTOCOMMIT 설정을 참조하세요.

차단 세션 찾기

`Lock:tuple` 대기 이벤트가 발생하는 동안 어떤 잠금이 서로 의존하는지 파악하여 차단 및 차단된 세션을 식별합니다. 자세한 내용은 PostgreSQL 위키의 잠금 종속성 정보를 참조하세요.

다음 예에서는 `tuple` 필터링과 `wait_time` 정렬을 통해 모든 세션을 쿼리합니다.

```
SELECT blocked_locks.pid AS blocked_pid,
         blocking_locks.pid AS blocking_pid,
         blocked_activity.usename AS blocked_user,
         blocking_activity.usename AS blocking_user,
         now() - blocked_activity.xact_start AS blocked_transaction_duration,
         now() - blocking_activity.xact_start AS blocking_transaction_duration,
         concat(blocked_activity.wait_event_type,':',blocked_activity.wait_event) AS
 blocked_wait_event,
         concat(blocking_activity.wait_event_type,':',blocking_activity.wait_event) AS
 blocking_wait_event,
         blocked_activity.state AS blocked_state,
         blocking_activity.state AS blocking_state,
         blocked_locks.locktype AS blocked_locktype,
         blocking_locks.locktype AS blocking_locktype,
         blocked_activity.query AS blocked_statement,
         blocking_activity.query AS blocking_statement
    FROM pg_catalog.pg_locks blocked_locks
    JOIN pg_catalog.pg_stat_activity blocked_activity ON blocked_activity.pid =
 blocked_locks.pid
    JOIN pg_catalog.pg_locks blocking_locks
        ON blocking_locks.locktype = blocked_locks.locktype
        AND blocking_locks.DATABASE IS NOT DISTINCT FROM blocked_locks.DATABASE
        AND blocking_locks.relation IS NOT DISTINCT FROM blocked_locks.relation
        AND blocking_locks.page IS NOT DISTINCT FROM blocked_locks.page
        AND blocking_locks.tuple IS NOT DISTINCT FROM blocked_locks.tuple
        AND blocking_locks.virtualxid IS NOT DISTINCT FROM blocked_locks.virtualxid
        AND blocking_locks.transactionid IS NOT DISTINCT FROM
 blocked_locks.transactionid
        AND blocking_locks.classid IS NOT DISTINCT FROM blocked_locks.classid
        AND blocking_locks.objid IS NOT DISTINCT FROM blocked_locks.objid
        AND blocking_locks.objsubid IS NOT DISTINCT FROM blocked_locks.objsubid
        AND blocking_locks.pid != blocked_locks.pid
```

```
    JOIN pg_catalog.pg_stat_activity blocking_activity ON blocking_activity.pid =
blocking_locks.pid
    WHERE NOT blocked_locks.GRANTED;
```

높은 동시성 저감

Lock:tuple 이벤트는 지속적으로 발생할 수 있으며, 특히 작업 부하가 많은 시간에 발생할 수 있습니다. 이 경우 매우 바쁜 행에 대해 높은 동시성을 줄이는 것이 좋습니다. 종종 몇 개의 행만 대기열이나 불리언 로직을 제어하므로 이러한 행을 매우 바쁘게 만듭니다.

비즈니스 요구 사항, 애플리케이션 로직 및 워크로드 유형에 따라 다양한 접근 방식을 사용하여 동시성을 줄일 수 있습니다. 예를 들면, 다음을 수행할 수 있습니다.

- 테이블 및 데이터 로직을 재설계하여 높은 동시성을 줄입니다.
- 행 수준에서 높은 동시성을 줄이기 위해 애플리케이션 로직을 변경합니다.
- 행 수준 잠금으로 쿼리를 활용하고 재설계합니다.
- NOWAIT 절을 사용해 연산을 재시도합니다.
- 낙관적 및 하이브리드 잠금 로직 동시성 제어를 사용하는 것이 좋습니다.
- 데이터베이스 격리 수준을 변경하는 것이 좋습니다.

병목 현상 해결

Lock:tuple은 CPU 부족 또는 Amazon EBS 대역폭의 최대 사용량과 같은 병목 현상이 발생시킬 수 있습니다. 병목 현상을 줄이려면 다음 방법을 고려하세요.

- 인스턴스 클래스 유형을 확장하세요.
- 리소스 집약적인 쿼리를 최적화하세요.
- 애플리케이션 로직을 변경하세요.
- 거의 액세스하지 않는 데이터를 아카이브하세요.

LWLock:BufferMapping (LWLock:buffer_mapping)

이 이벤트는 세션이 데이터 블록을 공유 버퍼 풀의 버퍼와 연결하기 위해 대기 중일 때 발생합니다.

> **Note**
>
> RDS for PostgreSQL 버전 13 이상에서 이 이벤트의 이름은 `LWLock:BufferMapping`입니다. RDS for PostgreSQL 버전 12 이전에서 이 이벤트의 이름은 `LWLock:buffer_mapping`입니다.

주제

- 지원되는 엔진 버전
- 컨텍스트
- 원인
- 작업

지원되는 엔진 버전

이 대기 이벤트 정보는 RDS for PostgreSQL 버전 9.6 이상과 관련이 있습니다.

컨텍스트

공유 버퍼 풀은 프로세스에서 사용 중이거나 사용되었던 모든 페이지를 보관하는 PostgreSQL 메모리 영역입니다. 프로세스에 페이지가 필요한 경우 페이지를 공유 버퍼 풀로 읽습니다. `shared_buffers` 파라미터는 공유 버퍼 크기를 설정하고 테이블 및 인덱스 페이지를 저장할 메모리 영역을 예약합니다. 이 파라미터를 변경하는 경우 데이터베이스를 다시 시작해야 합니다.

`LWLock:buffer_mapping` 대기 이벤트는 다음 시나리오에서 발생합니다.

- 프로세스가 버퍼 테이블에서 페이지를 검색하고 공유 버퍼 매핑 잠금을 획득합니다.
- 프로세스가 페이지를 버퍼 풀로 로드하고 배타적 버퍼 매핑 잠금을 획득합니다.
- 프로세스가 페이지를 버퍼 풀로 로드하고 배타적 버퍼 매핑 잠금을 획득합니다.

원인

이 이벤트가 정상보다 많이 발생하여 성능 문제가 발생할 수 있는 경우 데이터베이스가 공유 버퍼 풀에 페이징 및 페이징 중입니다. 일반적인 원인은 다음과 같습니다.

- 대규모 쿼리
- 부풀린 인덱스 및 테이블

- 전체 테이블 스캔
- 작업 세트보다 작은 공유 풀 크기

작업

대기 이벤트의 원인에 따라 다른 작업을 권장합니다.

주제

- 버퍼 관련 지표 모니터링
- 인덱싱 전략 평가
- 신속하게 할당해야 하는 버퍼 수 저감

버퍼 관련 지표 모니터링

`LWLock:buffer_mapping`이 스파이크를 기다렸다가 버퍼 적중률을 조사합니다. 이러한 지표를 사용하여 버퍼 캐시에서 발생하는 상황을 더 잘 이해할 수 있습니다. 다음 지표를 검토합니다.

`blks_hit`

이 성능 개선 도우미 카운터 지표는 공유 버퍼 풀에서 검색된 블록 수를 나타냅니다. `LWLock:buffer_mapping` 대기 이벤트가 발생하면 `blks_hit`에서 스파이크가 발생할 수 있습니다.

`blks_read`

이 성능 개선 도우미 카운터 지표는 공유 버퍼 풀에서 I/O를 읽어야 하는 블록 수를 나타냅니다. `LWLock:buffer_mapping` 대기 이벤트의 리드업에서 `blks_read`의 스파이크가 발생할 수 있습니다.

인덱싱 전략 평가

인덱싱 전략의 성능 저하가 아닌지 확인하려면 다음을 알아보세요.

인덱스 팽창

인덱스와 테이블 팽창으로 인해 불필요한 페이지가 공유 버퍼로 읽히지 않는지 확인합니다. 테이블에 사용되지 않는 행이 포함된 경우 데이터를 보관하고 테이블에서 행을 제거하는 것이 좋습니다. 그런 다음 크기가 조정된 테이블의 인덱스를 다시 작성할 수 있습니다.

자주 사용하는 쿼리의 인덱스

최적의 인덱스가 있는지 확인하려면 성능 개선 도우미에서 DB 엔진 지표를 모니터링하세요. `tup_returned` 지표는 읽은 행의 수를 보여줍니다. `tup_fetched` 지표는 클라이언트에게 반환되는 행 수를 보여줍니다 `tup_returned`가 `tup_fetched`보다 훨씬 큰 경우, 데이터가 제대로 인덱스화되지 않을 수 있습니다. 또한 테이블 통계가 최신 상태가 아닐 수도 있습니다.

신속하게 할당해야 하는 버퍼 수 저감

`LWLock:buffer_mapping` 대기 이벤트를 줄이려면, 신속하게 할당해야 하는 버퍼 수를 줄이세요. 한 가지 전략은 소규모 배치 작업을 수행하는 것입니다. 테이블을 분할하여 더 작은 배치를 달성할 수 있습니다.

LWLock:BufferIO(IPC:BufferIO)

`LWLock:BufferIO` 이벤트는 RDS for PostgreSQL이 페이지에 동시에 액세스하려고 하는 다른 프로세스들이 입출력 (I/O) 작업을 완료할 때까지 기다리는 경우에 발생합니다. 그 목적은 동일한 페이지를 공유 버퍼로 읽는 것입니다.

주제

- 관련 엔진 버전
- 컨텍스트
- 원인
- 작업

관련 엔진 버전

이 대기 이벤트 정보는 모든 RDS for PostgreSQL 버전과 관련이 있습니다. RDS for PostgreSQL 12 이하 버전의 경우 이 대기 이벤트의 이름이 lwlock:buffer_io인 반면, RDS for PostgreSQL 13 버전에서는 lwlock:bufferio로 이름이 지정됩니다. RDS for PostgreSQL 14 버전부터는 BufferIO 대기 이벤트가 `LWLock`에서 `IPC` 대기 이벤트 유형(IPC:BufferIO)으로 변경되었습니다.

컨텍스트

각 공유 버퍼에는 매번 블록(또는 페이지)이 공유 버퍼 풀 외부에서 회수되야 하는 `LWLock:BufferIO` 대기 이벤트와 연결된 I/O 잠금이 있습니다.

이 잠금은 모두 동일한 블록에 액세스해야 하는 여러 세션을 처리하는 데 사용됩니다. 이 블록은 shared_buffers 파라미터로 정의된 공유 버퍼 풀 외부에서 읽어야 합니다.

공유 버퍼 풀 내에서 페이지를 읽는 즉시 LWLock:BufferIO 잠금은 해제됩니다.

> **Note**
>
> LWLock:BufferIO 대기 이벤트는 IO:DataFileRead 대기 이벤트에 선행됩니다. IO:DataFileRead 대기 이벤트는 스토리지에서 데이터를 읽는 동안 발생합니다.

경량 잠금에 대한 자세한 내용은 잠금 개요를 참조하세요.

원인

상위 대기에서 나타나는 LWLock:BufferIO 이벤트의 일반적인 원인은 다음을 포함합니다.

- I/O 작업이 보류 중인 동일한 페이지에 액세스하려고 시도하는 여러 백엔드 또는 연결
- 공유 버퍼 풀의 크기 간의 비율(shared_buffers 파라미터로 정의됨) 및 현재 워크로드에 필요한 버퍼 수
- 공유 버퍼 풀의 크기가 다른 작업에서 제거되는 페이지 수와 균형이 맞지 않습니다.
- 엔진이 공유 버퍼 풀에 필요한 것보다 많은 페이지를 읽어야 하는 크거나 부풀린 인덱스
- DB 엔진이 테이블에서 필요한 것보다 더 많은 페이지를 읽도록 하는 인덱스의 부족
- 체크포인트가 너무 자주 발생하거나 수정된 페이지를 너무 많이 플러시해야 함
- 같은 페이지에서 작업을 수행하려고 시도하는 데이터베이스 연결의 갑작스러운 급증

작업

대기 이벤트의 원인에 따라 다른 작업을 권장합니다.

- BufferCacheHitRatio 및 LWLock:BufferIO 대기 이벤트의 급격한 감소 간의 상관관계에 대한 Amazon CloudWatch 지표가 관찰됩니다. 이 효과는 공유 버퍼 설정이 작음을 의미할 수 있습니다. DB 인스턴스 클래스를 늘리거나 확장해야 할 수 있습니다. 워크로드를 더 많은 리더 노드로 분할할 수 있습니다.
- LWLock:BufferIO가 BufferCacheHitRatio 지표 강하와 동시에 발생한다면 워크로드 피크 시간을 기준으로 max_wal_size와 checkpoint_timeout을 튜닝하세요. 그런 다음 어떤 쿼리가 발생하는지 식별합니다.

- 사용되지 않는 인덱스가 있는지 확인한 다음 제거합니다.
- 분할된 테이블(분할된 인덱스도 있음)을 사용합니다. 이렇게 하면 인덱스 재정렬을 낮게 유지하고 영향을 줄일 수 있습니다.
- 불필요하게 열을 인덱싱하지 마세요.
- 연결 풀을 사용하여 갑작스러운 데이터베이스 연결 스파이크를 방지합니다.
- 데이터베이스에 대한 최대 연결 수를 모범 사례로 제한합니다.

LWLock:buffer_content (BufferContent)

`LWLock:buffer_content` 이벤트는 다른 세션에서 특정 데이터 페이지에 대해 쓰기 잠금을 설정한 동안 세션에서 메모리 내 해당 페이지 읽기 또는 쓰기를 대기 중일 때 발생합니다. RDS for PostgreSQL 13 이상에서는 이 대기 이벤트를 `BufferContent`라고 합니다.

주제

- 지원되는 엔진 버전
- 컨텍스트
- 대기 증가의 가능한 원인
- 작업

지원되는 엔진 버전

이 대기 이벤트 정보는 모든 RDS for PostgreSQL 버전에서 지원됩니다.

컨텍스트

데이터를 읽거나 조작하기 위해 PostgreSQL 공유 메모리 버퍼를 통해 데이터에 액세스합니다. 버퍼에서 읽기 위해 프로세스는 공유 모드에서 버퍼 콘텐츠에 대한 경량 잠금(LWLock) 을 가져옵니다. 버퍼에 쓰려면 배타적 모드에서 해당 잠금을 가져옵니다. 공유 잠금을 사용하면 다른 프로세스가 동시에 해당 콘텐츠에 대한 공유 잠금을 획득할 수 있습니다. 배타적 잠금은 다른 프로세스에서 모든 유형의 잠금을 얻지 못하게 합니다.

`LWLock:buffer_content`(`BufferContent`) 이벤트는 여러 프로세스가 특정 버퍼의 내용을 잠그려고 시도하고 있음을 나타냅니다.

대기 증가의 가능한 원인

LWLock:buffer_content(BufferContent) 이벤트가 정상보다 많이 나타나 성능 문제를 나타내는 경우 일반적인 원인은 다음과 같습니다.

동일한 데이터에 대한 동시 업데이트 증가

동일한 테이블에 삽입하거나 업데이트하는 쿼리의 동시 세션 수가 증가할 수 있습니다. 이 경합은 인덱스가 많은 테이블에서 더 두드러질 수 있습니다.

워크로드 데이터가 메모리에 없습니다.

활성 워크로드에서 처리 중인 데이터가 메모리에 없으면 이러한 대기 이벤트가 증가할 수 있습니다. 이 효과는 잠금을 유지하는 프로세스가 디스크 I/O 작업을 수행하는 동안 더 오래 유지할 수 있기 때문입니다.

외래 키 제약 조건을 과도하게 사용

외래 키 제약 조건은 프로세스가 버퍼 콘텐츠 잠금에 보관하는 시간을 늘릴 수 있습니다. 이 효과는 해당 키가 업데이트되는 동안 읽기 작업에서 참조된 키에 대해 공유 버퍼 콘텐츠 잠금이 필요하기 때문입니다.

작업

대기 이벤트의 원인에 따라 다른 작업을 권장합니다. Amazon RDS 성능 개선 도우미를 사용하거나 pg_stat_activity 뷰를 쿼리하여 LWLock:buffer_content(BufferContent) 이벤트를 식별할 수 있습니다.

주제

- 인메모리 효율성 향상
- 외래 키 제약 조건 사용 감소
- 사용되지 않는 인덱스 삭제
- 시퀀스 사용 시 캐시 크기 늘리기

인메모리 효율성 향상

활성 워크로드 데이터가 메모리에 있을 확률을 높이려면 테이블을 분할하거나 인스턴스 클래스를 확장하세요. DB 인스턴스 클래스에 대한 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요.

외래 키 제약 조건 사용 감소

외래 키 제약 조건을 사용을 위해 많은 수의 LWLock:buffer_content(BufferContent) 대기 이벤트를 경험하는 워크로드를 조사합니다. 불필요한 외래 키 제약 조건을 제거합니다.

사용되지 않는 인덱스 삭제

많은 수의 LWLock:buffer_content(BufferContent) 대기 이벤트를 경험하는 워크로드를 위해 사용하지 않는 인덱스를 식별하고 제거합니다.

시퀀스 사용 시 캐시 크기 늘리기

테이블에서 시퀀스를 사용하는 경우 캐시 크기를 늘려 시퀀스 페이지와 인덱스 페이지에서 경합을 제거합니다. 각 시퀀스는 공유 메모리에서 단일 페이지입니다. 캐시는 연결별로 사전 정의됩니다. 많은 동시 세션에서 시퀀스 값을 받는 경우 이 방법으로는 워크로드를 처리하기에 충분하지 않을 수 있습니다.

LWLock:lock_manager (LWLock:lockmanager)

이 이벤트는 RDS for PostgreSQL 엔진이 빠른 경로 잠금이 불가능할 때 잠금을 할당, 확인 및 할당 해제하기 위해 공유 잠금 메모리 영역을 유지 관리하는 경우에 발생합니다.

주제

- 지원되는 엔진 버전
- 컨텍스트
- 대기 증가의 가능한 원인
- 작업

지원되는 엔진 버전

이 대기 이벤트 정보는 RDS for PostgreSQL 버전 9.6 이상과 관련이 있습니다. 버전 13 이전의 RDS for PostgreSQL에서 이 대기 이벤트의 이름은 LWLock:lock_manager입니다. 버전 13 이상의 RDS for PostgreSQL에서 이 대기 이벤트의 이름은 LWLock:lockmanager입니다.

컨텍스트

SQL 문을 실행하면 RDS for PostgreSQL 동시 작업 중에 데이터베이스의 구조, 데이터 및 무결성을 보호하기 위해 잠금을 기록합니다. 엔진은 빠른 경로 잠금 또는 빠르지 않은 경로 잠금을 사용하여 이 목

표를 달성할 수 있습니다. 빠르지 않은 경로 잠금은 빠른 경로 잠금보다 비용이 많이 들고 오버헤드가 더 많이 발생합니다.

빠른 경로 잠금

자주 수행되고 해제되지만 충돌이 거의 발생하지 않는 잠금의 오버헤드를 줄이기 위해 백엔드 프로세스에서 빠른 경로 잠금을 사용할 수 있습니다. 데이터베이스는 다음 기준을 충족하는 잠금에 대해 이 메커니즘을 사용합니다.

- DEFAULT 잠금 방법을 사용합니다.
- 공유 관계가 아닌 데이터베이스 관계에 대한 잠금을 나타냅니다.
- 그들은 충돌할 가능성이 없는 약한 잠금입니다.
- 엔진은 충돌하는 잠금이 존재하지 않을 수 있는지 신속하게 확인할 수 있습니다.

다음 조건 중 하나에 부합할 때 엔진은 빠른 경로 잠금을 사용할 수 없습니다.

- 이 잠금이 이전 기준을 충족하지 않습니다.
- 백엔드 프로세스에 사용할 수 있는 슬롯이 더 이상 없습니다.

빠른 경로 잠금을 위해 쿼리를 조정하려는 경우 다음 쿼리를 사용할 수 있습니다.

```
SELECT count(*), pid, mode, fastpath
  FROM pg_locks
 WHERE fastpath IS NOT NULL
 GROUP BY 4,3,2
 ORDER BY pid, mode;
 count | pid  |      mode       | fastpath
-------+------+-----------------+----------
16 | 9185 | AccessShareLock | t
336 | 9185 | AccessShareLock | f
1 | 9185 | ExclusiveLock   | t
```

다음 쿼리는 데이터베이스의 합계만 표시합니다.

```
SELECT count(*), mode, fastpath
  FROM pg_locks
 WHERE fastpath IS NOT NULL
 GROUP BY 3,2
 ORDER BY mode,1;
```

```
count |      mode       | fastpath
------+-----------------+----------
16 | AccessShareLock | t
337 | AccessShareLock | f
1 | ExclusiveLock   | t
(3 rows)
```

고속 경로 잠금에 대한 자세한 내용은 잠금 관리자 README의 빠른 경로와 PostgreSQL 설명서의 pg-locks를 참조하세요.

잠금 관리자의 배율 조정 문제 예

이 예에서는 이름이 purchases인 테이블이 매일로 분할된 5년짜리 데이터를 저장합니다. 각 파티션에는 두 개의 인덱스가 있습니다. 다음과 같은 일련의 이벤트가 발생합니다.

1. 며칠 분량의 데이터를 쿼리하면 데이터베이스가 많은 파티션을 읽어야 합니다.
2. 데이터베이스는 각 파티션에 대한 잠금 항목을 만듭니다. 파티션 인덱스가 최적기 액세스 경로의 일부인 경우 데이터베이스도 해당 인덱스에 대한 잠금 항목을 만듭니다.
3. 동일한 백엔드 프로세스에 대해 요청된 잠금 항목 수가 FP_LOCK_SLOTS_PER_BACKEND의 값인 16보다 높은 경우 잠금 관리자는 비고속 경로 잠금 방법을 사용합니다.

최신 애플리케이션에는 수백 개의 세션이 있을 수 있습니다. 동시 세션이 적절한 파티션 정리 없이 부모를 쿼리하는 경우 데이터베이스는 수백 또는 수천 개의 고속 경로 잠금을 생성할 수 있습니다. 일반적으로 이 동시성이 vCPU 수보다 높으면 LWLock:lock_manager 대기 이벤트가 표시됩니다.

> **Note**
>
> LWLock:lock_manager 대기 이벤트는 데이터베이스 스키마의 파티션 또는 인덱스 수와 관련이 없습니다. 대신 데이터베이스가 제어해야 하는 비고속 경로 잠금 수와 관련이 있습니다.

대기 증가의 가능한 원인

LWLock:lock_manager 대기 이벤트가 정상보다 더 발생하여 성능 문제를 나타낼 수 있으며 갑작스런 스파이크의 가장 큰 원인은 다음과 같습니다.

- 동시 활성 세션은 빠른 경로 잠금을 사용하지 않는 쿼리를 실행하고 있습니다. 이러한 세션도 최대 vCPU를 초과합니다.

- 많은 수의 동시 활성 세션이 심하게 분할된 테이블에 액세스하고 있습니다. 각 파티션에는 여러 개의 인덱스가 있습니다.
- 데이터베이스에 연결 폭풍이 발생합니다. 기본적으로 일부 애플리케이션 및 연결 풀 소프트웨어는 데이터베이스 속도가 느릴 때 더 많은 연결을 만듭니다. 이 관행은 문제를 악화시킵니다. 연결 스톰이 발생하지 않도록 연결 풀 소프트웨어를 튜닝합니다.
- 많은 수의 세션이 파티션을 정리하지 않고 상위 테이블을 쿼리합니다.
- 데이터 정의 언어 (DDL), 유지 관리 명령은 자주 액세스하거나 수정되는 사용 중인 관계식 또는 튜플을 독점적으로 잠급니다.

작업

CPU 대기 이벤트가 발생하는 것이 반드시 성능 문제를 나타내지는 않습니다. 성능이 저하되고 이 대기 이벤트가 DB 로드를 지배하는 경우에만 이 이벤트에 응답합니다.

주제

- 파티션 정리 사용
- 불필요한 인덱스 삭제
- 빠른 경로 잠금을 위한 쿼리 조정
- 다른 대기 이벤트 조정
- 하드웨어 병목 현상 저감
- 연결 풀러 사용
- RDS for PostgreSQL 버전 업그레이드

파티션 정리 사용

파티션 정리는 선언적으로 분할된 테이블을 위한 쿼리 최적화 전략으로, 테이블 검색에서 불필요한 파티션을 제외하여 성능을 향상시킵니다. 파티션 정리는 기본적으로 활성화되어 있습니다. 꺼져 있으면 다음과 같이 켭니다.

```
SET enable_partition_pruning = on;
```

WHERE 절에 파티셔닝에 사용되는 열이 들어 있으면 쿼리는 파티션 정리를 활용할 수 있습니다. 더 자세한 내용은 PostgreSQL 설명서의 파티션 정리를 참조하세요.

불필요한 인덱스 삭제

데이터베이스에 사용되지 않거나 거의 사용되지 않는 인덱스가 포함되어 있을 수 있습니다. 이 경우 삭제하는 것이 좋습니다. 다음 중 하나를 수행하세요.

- 불필요한 인덱스를 찾아내려면, PostgreSQL 위키의 사용되지 않는 인덱스를 읽어보세요.
- PG 컬렉터를 실행합니다. 이 SQL 스크립트는 데이터베이스 정보를 수집하여 통합 HTML 보고서로 표시합니다. '사용되지 않은 인덱스' 섹션을 확인하세요. 자세한 내용은 AWS Labs GitHub 리포지토리의 pg-collector를 참조하세요.

빠른 경로 잠금을 위한 쿼리 조정

쿼리에서 빠른 경로 잠금을 사용하는지 확인하려면 `pg_locks` 테이블의 `fastpath` 열을 쿼리하세요. 쿼리에서 빠른 경로 잠금을 사용하지 않는 경우 쿼리당 관계 수를 16 미만으로 줄이세요.

다른 대기 이벤트 조정

`LWLock:lock_manager`가 최상위 대기 목록에서 첫 번째 또는 두 번째일 경우, 다음 대기 이벤트도 목록에 나타나는지 확인합니다.

- `Lock:Relation`
- `Lock:transactionid`
- `Lock:tuple`

위의 이벤트가 목록에서 높게 표시되는 경우 이러한 대기 이벤트를 먼저 조정하는 것이 좋습니다. 이러한 이벤트는 `LWLock:lock_manager` 드라이버가 될 수 있습니다.

하드웨어 병목 현상 저감

CPU 부족 또는 Amazon EBS 대역폭의 최대 사용량과 같은 하드웨어 병목 현상이 발생할 수 있습니다. 이 경우에는 하드웨어 병목 현상을 줄이는 것이 좋습니다. 다음 조치를 고려해 보세요.

- 인스턴스 클래스를 확장하세요.
- 대량의 CPU와 메모리를 사용하는 쿼리를 최적화하세요.
- 애플리케이션 로직을 변경하세요.
- 데이터를 아카이빙하세요.

CPU, 메모리 및 EBS 네트워크 대역폭에 대한 자세한 내용은 Amazon RDS 인스턴스 유형을 참조하세요.

연결 풀러 사용

총 활성 연결 수가 최대 vCPU를 초과하는 경우 인스턴스 유형이 지원할 수 있는 것보다 많은 OS 프로세스에 CPU가 필요합니다. 이 경우에는 연결 풀을 사용하거나 튜닝하는 것이 좋습니다. 인스턴스 유형별 vCPU 수에 대한 자세한 내용은 Amazon RDS 인스턴스 유형을 참조하세요.

연결 풀링에 대한 자세한 내용은 다음 리소스를 참조하세요.

- Amazon RDS Proxy
- pgbouncer
- PostgreSQL 설명서의 연결 풀 및 데이터 원본

RDS for PostgreSQL 버전 업그레이드

현재 버전의 RDS for PostgreSQL이 12보다 낮은 경우 버전 12 이상으로 업그레이드하세요. PostgreSQL 버전 12 이상에는 향상된 파티션 메커니즘이 있습니다. 버전 12의 더 자세한 정보는 PostgreSQL 릴리스 12.0을 참조하세요. RDS for PostgreSQL 업그레이드에 대한 자세한 내용은 RDS for PostgreSQL DB 엔진 업그레이드 섹션을 참조하세요.

LWLock:pg_stat_statements

LWLock:pg_stat_statements 대기 이벤트는 `pg_stat_statements` 확장이 SQL 문을 추적하는 해시 테이블에서 배타적 잠금을 수행할 때 발생합니다. 다음 시나리오에서 발생할 수 있습니다.

- 추적된 문 수가 구성된 `pg_stat_statements.max` 파라미터 값에 도달하고 더 많은 항목을 위한 공간을 확보해야 하는 경우, 확장은 직접 호출 수를 기준으로 정렬을 수행하고 가장 적게 실행된 문 중 5%를 제거한 다음 해시를 나머지 항목으로 다시 채웁니다.
- `pg_stat_statements`가 디스크의 `pgss_query_texts.stat` 파일에 대해 `garbage collection` 작업을 수행하고 파일을 다시 쓰는 경우

주제

- 지원되는 엔진 버전
- 컨텍스트
- 대기 증가의 가능한 원인

- 작업

지원되는 엔진 버전

이 대기 이벤트 정보는 모든 RDS for PostgreSQL 버전에서 지원됩니다.

컨텍스트

pg_stat_statements 확장 이해 - pg_stat_statements 확장은 해시 테이블에서 SQL 문 실행 통계를 추적합니다. 이 확장은 `pg_stat_statements.max` 파라미터에서 정의된 한도까지 SQL 문을 추적합니다. 이 파라미터는 추적할 수 있는 문의 최대 수를 결정하며, 이는 pg_stat_statements 뷰의 최대 행 수에 해당합니다.

문 통계 지속성 - 이 확장은 다음을 통해 인스턴스 재시작 시 문 통계를 유지합니다.

- pg_stat_statements.stat라는 파일에 데이터 쓰기
- pg_stat_statements.save 파라미터를 사용하여 지속성 동작 제어

pg_stat_statements.save가 다음과 같이 설정된 경우:

- on(기본값): 통계는 종료 시 저장되고 서버 시작 시 다시 로드됩니다.
- off: 통계는 종료 시 저장되지 않으며 서버 시작 시 다시 로드되지 않습니다.

쿼리 텍스트 저장소 - 이 확장은 추적된 쿼리의 텍스트를 `pgss_query_texts.stat`라는 파일에 저장합니다. 이 파일은 폐영역 회수가 발생하기 전에 추적된 모든 SQL 문의 평균 크기의 두 배까지 늘어날 수 있습니다. 이 확장에는 정리 작업 및 `pgss_query_texts.stat` 파일 재작성 중에 해시 테이블에 대한 배타적 잠금이 필요합니다.

문 할당 취소 프로세스 - 추적된 문 수가 `pg_stat_statements.max` 한도에 도달하고 새 문을 추적해야 하는 경우 확장은 다음을 수행합니다.

- 해시 테이블에서 배타적 잠금(LWLock:pg_stat_statements)을 수행합니다.
- 기존 데이터를 로컬 메모리에 로드합니다.
- 직접 호출 수를 기반으로 빠른 정렬을 수행합니다.
- 가장 적게 호출된 문(하위 5%)을 제거합니다.
- 해시 테이블을 나머지 항목으로 다시 채웁니다.

문 할당 취소 모니터링 - PostgreSQL 14 이상에서는 pg_stat_statements_info 뷰를 사용하여 문 할당 취소를 모니터링할 수 있습니다. 이 뷰에는 새 문을 위한 공간을 마련하기 위해 문을 할당 취소한 횟수를 보여주는 dealloc 열이 포함되어 있습니다.

문의 할당 취소가 자주 발생하면 디스크에서 pgss_query_texts.stat 파일의 폐영역 회수가 더 자주 발생합니다.

대기 증가의 가능한 원인

LWLock:pg_stat_statements 대기 시간 증가의 일반적인 원인은 다음과 같습니다.

- 애플리케이션에서 사용하는 고유 쿼리 수가 증가함
- 사용되는 고유 쿼리 수 대비 pg_stat_statements.max 파라미터 값이 작음

작업

대기 이벤트의 원인에 따라 다른 작업을 권장합니다. Amazon RDS Performance Insights를 사용하거나 pg_stat_activity 뷰를 쿼리하여 LWLock:pg_stat_statements 이벤트를 식별할 수 있습니다.

다음 pg_stat_statements 파라미터를 조정하여 추적 동작을 제어하고 LWLock:pg_stat_ 문 대기 이벤트를 줄입니다.

주제

- pg_stat_statements.track 파라미터 비활성화
- pg_stat_statements.max 파라미터 증가
- pg_stat_statements.track_utility 파라미터 비활성화

pg_stat_statements.track 파라미터 비활성화

LWLock:pg_stat_statements 대기 이벤트가 데이터베이스 성능에 부정적인 영향을 미치고 근본 원인을 식별하기 위해 pg_stat_statements 뷰를 추가로 분석하기 전에 신속한 솔루션이 필요한 경우 pg_stat_statements.track 파라미터를 none으로 설정하여 파라미터를 비활성화할 수 있습니다. 그러면 문 통계 수집이 비활성화됩니다.

pg_stat_statements.max 파라미터 증가

할당 취소를 줄이고 디스크에서 pgss_query_texts.stat 파일의 폐영역 회수를 최소화하려면 pg_stat_statements.max 파라미터 값을 늘립니다. 기본값은 5,000입니다.

> **Note**
>
> pg_stat_statements.max 파라미터는 정적입니다. 이 파라미터에 변경 사항을 적용하려면 DB 인스턴스를 다시 시작해야 합니다.

pg_stat_statements.track_utility 파라미터 비활성화

pg_stat_statements 뷰를 분석하여 pg_stat_statements에서 추적하는 리소스를 가장 많이 소비하는 유틸리티 명령을 확인할 수 있습니다.

pg_stat_statements.track_utility 파라미터는 모듈이 SELECT, INSERT, UPDATE, DELETE 및 MERGE를 제외한 모든 명령을 포함하는 유틸리티 명령을 추적하는지를 제어합니다. 이 파라미터는 기본적으로 on로 설정되어 있습니다.

예를 들어 애플리케이션이 본질적으로 고유한 많은 저장점 쿼리를 사용하는 경우 문 할당 취소가 증가할 수 있습니다. 이를 해결하기 위해 pg_stat_statements.track_utility 파라미터를 비활성화하여 pg_stat_statements가 저장점 쿼리를 추적하지 않도록 할 수 있습니다.

> **Note**
>
> pg_stat_statements.track_utility 파라미터는 동적 파라미터입니다. 데이터베이스 인스턴스를 다시 시작하지 않고도 값을 변경할 수 있습니다.

Example pg_stat_statements의 고유한 저장점 쿼리 예시

```
                    query                          |       queryid
---------------------------------------------------+---------------------
 SAVEPOINT JDBC_SAVEPOINT_495701                   | -7249565344517699703
 SAVEPOINT JDBC_SAVEPOINT_1320                     | -1572997038849006629
 SAVEPOINT JDBC_SAVEPOINT_26739                    |  54791337410474486
 SAVEPOINT JDBC_SAVEPOINT_1294466                  |  8170064357463507593
 ROLLBACK TO SAVEPOINT JDBC_SAVEPOINT_65016        | -33608214779996400
 SAVEPOINT JDBC_SAVEPOINT_14185                    | -2175035613806809562
 SAVEPOINT JDBC_SAVEPOINT_45837                    | -6201592986750645383
 SAVEPOINT JDBC_SAVEPOINT_1324                     |  6388797791882029332
```

PostgreSQL 17에는 유틸리티 명령 추적을 위한 몇 가지 향상된 기능이 도입되었습니다.

- 이제 Savepoint 이름이 상수로 표시됩니다.
- 이제 두 단계 커밋 명령의 글로벌 트랜잭션 ID(GID)가 상수로 표시됩니다.
- DEALLOCATE 문의 이름이 상수로 표시됩니다.
- 이제 CALL 파라미터가 상수로 표시됩니다.

Timeout:PgSleep

`Timeout:PgSleep` 이벤트는 서버 프로세스가 `pg_sleep` 함수를 가리키고 및 절전 시간 초과가 만료될 때까지 대기할 때 발생합니다.

주제

- 지원되는 엔진 버전
- 대기 증가의 가능한 원인
- 작업

지원되는 엔진 버전

이 대기 이벤트 정보는 모든 RDS for PostgreSQL 버전에서 지원됩니다.

대기 증가의 가능한 원인

이 대기 이벤트는 애플리리케이션, 저장된 함수 또는 사용자가 다음 함수 중 하나를 호출하는 SQL 문을 실행할 때 발생합니다.

- `pg_sleep`
- `pg_sleep_for`
- `pg_sleep_until`

위의 함수는 지정된 시간(초)이 경과할 때까지 실행을 지연시킵니다. 예를 들어, `SELECT pg_sleep(1)`은 1초 동안 일시 중지됩니다. 자세한 내용은 PostgreSQL 설명서의 실행 지연을 참조하세요.

작업

`pg_sleep` 함수를 실행 중인 명령문을 식별합니다. 함수의 사용이 적절한지 확인합니다.

Timeout:VacuumDelay

`Timeout:VacuumDelay` 이벤트는 vacuum I/O의 비용 한도가 초과되었고 vacuum 프로세스가 절전 모드로 전환되었음을 나타냅니다. vacuum 작업은 해당 비용 지연 파라미터에 지정된 기간 동안 중지되었다가 작업을 재개합니다. 수동 vacuum 명령의 경우 `vacuum_cost_delay` 파라미터에 지연이 지정됩니다. autovacuum 대몬(deamon)의 경우 `autovacuum_vacuum_cost_delay parameter.`에 지연이 지정됩니다.

주제

- 지원되는 엔진 버전
- 컨텍스트
- 대기 증가의 가능한 원인
- 작업

지원되는 엔진 버전

이 대기 이벤트 정보는 모든 RDS for PostgreSQL 버전에서 지원됩니다.

컨텍스트

PostgreSQL에는 autovacuum 대몬(daemon)과 수동 vacuum 명령이 모두 있습니다. RDS for PostgreSQL DB 인스턴스의 경우 autovacuum 프로세스가 기본적으로 켜져 있습니다. 수동 vacuum 명령은 필요에 따라 사용됩니다(예: 죽은 튜플의 테이블을 지우거나 새 통계를 생성하기 위해).

vacuum이 진행 중일 때 PostgreSQL은 시스템이 다양한 I/O 작업을 수행할 때의 예상 비용을 내부 카운터를 사용하여 추적합니다. 카운터가 비용 한도 파라미터에 지정된 값에 도달하면 작업을 수행하는 프로세스는 비용 지연 파라미터에 지정된 짧은 기간 동안 절전 상태로 전환됩니다. 그런 다음 카운터를 재설정하고 작업을 계속합니다.

vacuum 프로세스에는 리소스 소비 조절에 사용할 수 있는 파라미터가 있습니다. autovacuum과 수동 vacuum 명령에는 비용 한도 값을 설정하기 위한 자체 파라미터가 있습니다. 또한 비용 지연, 즉 한도에 도달했을 때 vacuum을 절전 상태로 전환하는 데 걸리는 시간을 지정하는 자체 파라미터도 있습니다. 비용 지연 파라미터는 이렇게 리소스 소비 제한 메커니즘으로 작동합니다. 다음 목록에서 이러한 파라미터에 대한 설명을 찾을 수 있습니다.

autovacuum 대몬(daemon)의 제한에 영향을 주는 파라미터

- autovacuum_vacuum_cost_limit - 자동 vacuum 작업에 사용할 비용 한도 값을 지정합니다. 이 파라미터의 설정을 늘리면 vacuum 프로세스가 더 많은 리소스를 사용할 수 있고 Timeout:VacuumDelay 대기 이벤트가 줄어듭니다.
- autovacuum_vacuum_cost_delay - 자동 vacuum 작업에 사용할 비용 지연 값을 지정합니다. 기본값은 2밀리초입니다. 지연 파라미터를 0으로 설정하면 제한 메커니즘이 해제되므로 Timeout:VacuumDelay 대기 이벤트가 나타나지 않습니다.

자세한 내용은 PostgreSQL 설명서의 자동 Vacuuming을 참조하세요.

수동 vacuum 프로세스의 제한에 영향을 미치는 파라미터

- vacuum_cost_limit - vacuum 프로세스가 절전 상태로 전환되는 임계값입니다. 기본적으로 한도는 200입니다. 이 숫자는 다양한 리소스에 필요한 추가 I/O의 누적 예상 비용을 나타냅니다. 이 값을 늘리면 Timeout:VacuumDelay 대기 이벤트 수가 줄어듭니다.
- vacuum_cost_delay - vacuum 비용 한도에 도달했을 때 vacuum 프로세스가 절전 상태가 되는 시간입니다. 기본 설정은 0이며, 이는 이 기능이 해제되어 있음을 의미합니다. 이 값을 정수 값으로 설정하면 이 기능을 켜는 데 걸리는 시간을 밀리초 단위로 지정할 수 있지만 기본 설정을 그대로 두는 것이 좋습니다.

vacuum_cost_delay 파라미터에 대한 자세한 내용은 PostgreSQL 설명서에서 리소스 소비를 참조하세요.

RDS for PostgreSQL에서 autovacuum을 구성하고 사용하는 자세한 방법은 Amazon RDS for PostgreSQL에서 PostgreSQL 자동 정리 사용을 참조하세요.

대기 증가의 가능한 원인

Timeout:VacuumDelay는 vacuum의 절전 시간을 제어하는 비용 한도 파라미터 설정(vacuum_cost_limit, autovacuum_vacuum_cost_limit)과 비용 지연 파라미터(vacuum_cost_delay, autovacuum_vacuum_cost_delay) 간의 균형에 영향을 받습니다. 비용 한도 파라미터 값을 높이면 vacuum이 절전 상태로 전환되기 전에 더 많은 리소스를 사용할 수 있습니다. 그러면 Timeout:VacuumDelay 대기 이벤트가 줄어듭니다. 지연 파라미터 중 하나를 늘리면 Timeout:VacuumDelay 대기 이벤트가 더 자주, 더 오랜 시간 동안 발생합니다.

autovacuum_max_workers 파라미터 설정은 Timeout:VacuumDelay의 수도 늘릴 수 있습니다. 각각의 추가 autovacuum 작업자 프로세스는 내부 카운터 메커니즘에 기여하므로 단일 autovacuum 작

업자 프로세스를 사용할 때보다 한도에 더 빨리 도달할 수 있습니다. 비용 한도에 더 빨리 도달할수록 비용 지연이 더 자주 적용되어 `Timeout:VacuumDelay` 대기 이벤트가 더 많이 발생합니다. 자세한 내용은 PostgreSQL 설명서의 autovacuum_max_workers를 참조하세요.

큰 객체(예: 500GB 이상)도 이 대기 이벤트를 발생시킬 수 있습니다. vacuum이 큰 객체의 처리를 완료하는 데 시간이 걸릴 수 있기 때문입니다.

작업

vacuum 작업이 예상대로 완료되면 문제 해결이 필요하지 않습니다. 즉, 이 대기 이벤트가 반드시 문제를 나타내는 것은 아닙니다. 이는 완료해야 하는 다른 프로세스에 리소스를 적용할 수 있도록 지연 파라미터에 지정된 시간 동안 vacuum이 절전 상태로 전환되고 있음을 나타냅니다.

vacuum 작업을 더 빨리 완료하려면 지연 파라미터를 낮추면 됩니다. 이렇게 하면 vacuum의 절전 시간이 단축됩니다.

Amazon DevOps Guru의 사전 예방 인사이트를 활용하여 RDS for PostgreSQL 튜닝

DevOps Guru의 사전 예방 인사이트는 RDS for PostgreSQL DB 인스턴스에서 문제를 일으킬 수 있는 조건을 감지하여 문제가 발생하기 전에 이를 알려줍니다. 사전 예방적 인사이트를 통해 트랜잭션 연결의 장기 유휴 상태를 경고할 수 있습니다. 트랜잭션 연결에서 유휴 상태로 오래 실행되는 문제 해결에 대한 자세한 내용은 데이터베이스가 트랜잭션 연결 시 오랫동안 유휴 상태로 실행됨 섹션을 참조하세요.

DevOps Guru는 다음과 같은 작업을 수행할 수 있습니다.

- 데이터베이스 구성을 일반적인 권장 설정과 교차 검사하여 여러 가지 일반적인 데이터베이스 문제를 방지합니다.
- 확인하지 않은 상태로 두면 나중에 더 큰 문제로 이어질 수 있는 플릿의 심각한 문제를 알려줍니다.
- 새로 발견된 문제를 알려줍니다.

모든 사전 예방 인사이트에는 문제의 원인에 대한 분석과 수정 조치를 위한 권장 사항이 포함됩니다.

Amazon DevOps Guru for Amazon RDS에 대한 자세한 내용은 Amazon DevOps Guru for Amazon RDS로 성능 이상 분석 섹션을 참조하세요.

데이터베이스가 트랜잭션 연결 시 오랫동안 유휴 상태로 실행됨

데이터베이스에 대한 연결이 1,800초 이상 `idle in transaction` 상태로 지속되었습니다.

주제

- 지원되는 엔진 버전
- 컨텍스트
- 이 문제의 잠재적 원인
- 작업
- 관련 지표

지원되는 엔진 버전

이 인사이트 정보는 RDS for PostgreSQL의 모든 버전에서 지원됩니다.

컨텍스트

해당 `idle in transaction` 상태의 트랜잭션은 다른 쿼리를 차단하는 잠금을 유지할 수 있습니다. 또한 VACUUM(autovacuum 포함)이 잘못된 행을 정리하여 인덱스 또는 테이블 팽창 또는 트랜잭션 ID 랩어라운드로 이어지는 것을 방지할 수 있습니다.

이 문제의 잠재적 원인

BEGIN 또는 START TRANSACTION을 사용하여 대화형 세션에서 시작된 트랜잭션이 COMMIT, ROLLBACK 또는 END 명령을 사용하여 종료되지 않았습니다. 이 경우 트랜잭션이 `idle in transaction` 상태로 이동합니다.

작업

`pg_stat_activity` 쿼리를 통해 유휴 트랜잭션을 찾을 수 있습니다.

SQL 클라이언트에서 다음 쿼리를 실행하여 `idle in transaction` 상태의 모든 연결을 나열하고 기간별로 정렬합니다.

```
SELECT now() - state_change as idle_in_transaction_duration, now() - xact_start as
 xact_duration,*
FROM  pg_stat_activity
WHERE state  = 'idle in transaction'
AND   xact_start is not null
ORDER BY 1 DESC;
```

인사이트의 원인에 따라 다른 조치를 취할 것을 권장합니다.

주제

- 트랜잭션 종료
- 연결 종료
- idle_in_transaction_session_timeout 파라미터 구성
- AUTOCOMMIT 상태 확인
- 애플리케이션 코드에서 트랜잭션 로직 확인

트랜잭션 종료

BEGIN 또는 START TRANSACTION을 사용하여 대화형 세션에서 트랜잭션을 시작하면 트랜잭션이 `idle in transaction` 상태로 이동합니다. COMMIT, ROLLBACK, END 명령을 실행하여 트랜잭션을 종료하거나 연결을 완전히 끊어 트랜잭션을 롤백할 때까지 이 상태로 유지됩니다.

연결 종료

다음 쿼리를 사용하여 유휴 트랜잭션과의 연결을 종료합니다.

```
SELECT pg_terminate_backend(pid);
```

pid는 연결의 프로세스 ID입니다.

idle_in_transaction_session_timeout 파라미터 구성

파라미터 그룹에서 `idle_in_transaction_session_timeout` 파라미터를 구성합니다. 이 파라미터를 구성하면 트랜잭션의 오랜 유휴 상태를 종료하기 위해 수동 개입이 필요하지 않다는 이점이 있습니다. 이 파라미터에 대한 자세한 내용은 PostgreSQL 설명서를 참조하세요.

트랜잭션이 idle_in_transaction 상태로 지정된 시간보다 오래 열려 있는 경우 연결이 종료된 후 PostgreSQL 로그 파일에 다음 메시지가 보고됩니다.

```
FATAL: terminating connection due to idle in transaction timeout
```

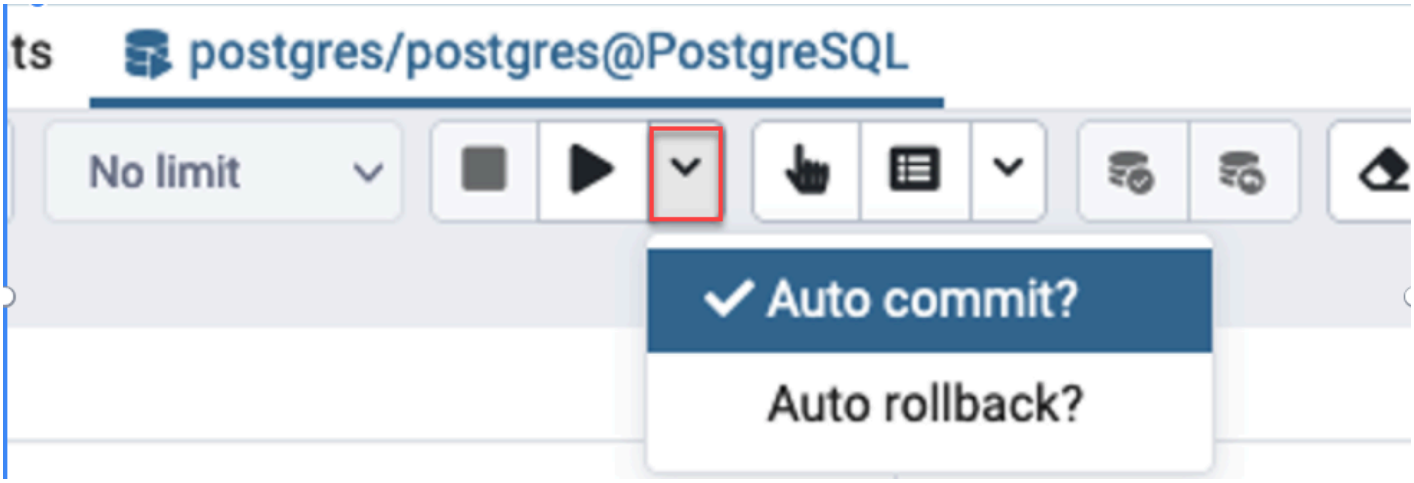
AUTOCOMMIT 상태 확인

AUTOCOMMIT은 기본적으로 활성화되어 있습니다. 하지만 클라이언트에서 실수로 비활성화된 경우에는 반드시 다시 활성화해야 합니다.

- psql 클라이언트에서 다음 명령을 실행합니다.

  ```
  postgres=> \set AUTOCOMMIT on
  ```

- pgadmin에서 아래쪽 화살표에서 AUTOCOMMIT 옵션을 선택하여 활성화합니다.



애플리케이션 코드에서 트랜잭션 로직 확인

애플리케이션 로직에 문제가 있는지 조사합니다. 다음 조치를 고려해 보세요.

- JDBC 자동 커밋이 애플리케이션에서 참으로 설정되어 있는지 확인합니다. 또한 코드에 명시적 `COMMIT` 명령을 사용하는 것도 고려해 보세요.
- 오류 처리 로직을 확인하여 오류 발생 후 트랜잭션이 종료되는지 확인합니다.
- 트랜잭션이 열려 있는 동안 애플리케이션이 쿼리에서 반환된 행을 처리하는 데 시간이 오래 걸리는지 확인합니다. 그렇다면 행을 처리하기 전에 트랜잭션을 닫도록 애플리케이션을 코딩하는 것을 고려해 보세요.
- 트랜잭션에 장기 실행 작업이 많이 포함되어 있는지 확인합니다. 그렇다면 단일 트랜잭션을 여러 트랜잭션으로 나누세요.

관련 지표

이 인사이트와 관련된 PI 지표는 다음과 같습니다.

- idle_in_transaction_count - `idle in transaction` 상태에 있는 세션 수입니다.
- idle_in_transaction_max_time - `idle in transaction` 상태에서 가장 오래 실행되는 트랜잭션의 지속 시간입니다.

Amazon RDS for PostgreSQL로 PostgreSQL 확장 사용

다양한 확장 프로그램 및 모듈을 설치하여 PostgreSQL의 기능을 확장할 수 있습니다. 예를 들어 공간 데이터로 작업하려면 PostGIS 확장을 설치하고 사용할 수 있습니다. 자세한 내용은 PostGIS 확장을 사용하여 공간 데이터 관리 섹션을 참조하세요. 또 다른 예로, 매우 큰 테이블의 데이터 입력을 개선하려는 경우 pg_partman 확장을 사용하여 데이터 분할을 고려할 수 있습니다. 자세한 내용은 pg_partman 확장자를 사용하여 PostgreSQL 파티션 관리하기를 참조하세요.

> **Note**
>
> RDS for PostgreSQL은 DB 인스턴스에 추가할 수 있는 pg_tle 확장을 통해 PostgreSQL용 신뢰할 수 있는 언어 확장을 지원합니다. 이 확장을 사용하면 개발자는 설정 및 구성 요구 사항을 간소화하는 안전한 환경에서 자체 PostgreSQL 확장을 만들 수 있습니다. pg_tle 확장을 지원하는 RDS for PostgreSQL 버전에 대해 알아보고 자세한 내용을 확인하려면 PostgreSQL용 신뢰할 수 있는 언어 확장 작업 섹션을 참조하세요.

경우에 따라 확장을 설치하는 대신 RDS for PostgreSQL DB 인스턴스의 사용자 지정 DB 파라미터 그룹의 shared_preload_libraries 목록에 특정 모듈을 추가할 수 있습니다. 일반적으로 기본 DB 클러스터 파라미터 그룹은 pg_stat_statements만 로드하지만 목록에 추가할 수 있는 다른 모듈도 몇 개 있습니다. 예를 들어, PostgreSQL pg_cron 확장을 사용하여 유지 관리 예약에 자세히 설명된 대로 pg_cron 모듈을 추가하여 스케줄링 기능을 추가할 수 있습니다. 또 다른 예로, auto_explain 모듈을 로드하여 쿼리 실행 계획을 로깅할 수 있습니다. 자세히 알아보려면 AWS 지식 센터에서 쿼리 실행 계획 로깅에 관한 문서를 참조하세요.

RDS for PostgreSQL 버전에 따라 확장 프로그램을 설치하려면 다음과 같이 rds_superuser 권한이 필요합니다.

- RDS for PostgreSQL 버전 12 및 이전 버전의 경우 확장 프로그램을 설치하려면 rds_superuser 권한이 필요합니다.
- RDS for PostgreSQL 버전 13 및 이상 버전의 경우 지정된 데이터베이스 인스턴스에 대한 생성 권한이 있는 사용자(역할)는 신뢰할 수 있는 확장 프로그램을 설치하고 사용할 수 있습니다. 신뢰할 수 있는 확장 프로그램 목록은 PostgreSQL 신뢰할 수 있는 확장 섹션을 참조하세요.

RDS for PostgreSQL DB 인스턴스에 설치할 수 있는 확장 프로그램을 rds.allowed_extensions 파라미터에 나열하여 정확하게 지정할 수도 있습니다. 자세한 내용은 PostgreSQL 확장의 설치 제한 섹션을 참조하세요.

`rds_superuser` 역할에 대한 자세한 내용은 PostgreSQL 역할 및 권한 이해 섹션을 참조하세요.

주제

- orafce 확장에서 함수 사용
- PostgreSQL에 대한 Amazon RDS 위임 확장 지원 사용
- pg_partman 확장자를 사용하여 PostgreSQL 파티션 관리하기
- pgAudit를 사용하여 데이터베이스 활동 로깅
- PostgreSQL pg_cron 확장을 사용하여 유지 관리 예약
- pglogical을 사용하여 인스턴스 간 데이터 동기화
- pgactive를 사용하여 액티브-액티브 복제 지원
- pg_repack 확장을 사용하여 테이블 및 인덱스에서 부풀림을 줄입니다.
- PLV8 확장 업그레이드 및 사용
- PL/Rust를 사용하여 Rust 언어로 PostgreSQL 함수 작성
- PostGIS 확장을 사용하여 공간 데이터 관리

orafce 확장에서 함수 사용

orafce 확장은 Oracle 데이터베이스에서 함수와 패키지의 하위 집합을 에뮬레이션하는 함수 및 연산자를 제공합니다. orafce 확장을 사용하면 Oracle 애플리케이션을 PostgreSQL로 쉽게 포팅할 수 있습니다. 이 확장은 RDS for PostgreSQL 버전 9.6.6 이상에서 지원됩니다. orafce에 대한 자세한 내용은 GitHub에서 orafce를 참조하세요.

> **Note**
>
> RDS for PostgreSQL은 orafce 확장의 일부분인 `utl_file` 패키지를 지원하지 않습니다. 이는 `utl_file` 스키마 함수가 기본 호스트에 대한 수퍼유저 권한을 필요로 하는 운영 체제 텍스트 파일의 읽기 및 쓰기 작업을 제공하기 때문입니다. 관리형 서비스에서 RDS for PostgreSQL은 호스트 액세스를 제공하지 않습니다.

orafce 확장을 사용하려면

1. DB 인스턴스를 생성할 때 사용한 기본 사용자 이름으로 DB 인스턴스에 연결합니다.

   동일한 DB 인스턴스의 다른 데이터베이스에 대해 orafce를 활성화하려면 `/c dbname` psql 명령을 사용합니다. 이 명령을 사용하면 연결을 시작한 후 기본 데이터베이스에서 변경합니다.

2. CREATE EXTENSION 문을 사용하여 orafce 확장을 활성화합니다.

```
CREATE EXTENSION orafce;
```

3. ALTER SCHEMA 문을 사용하여 oracle 스키마 소유권을 rds_superuser 역할로 이전합니다.

```
ALTER SCHEMA oracle OWNER TO rds_superuser;
```

oracle 스키마의 소유자 목록을 보려면 \dn psql 명령을 사용합니다.

PostgreSQL에 대한 Amazon RDS 위임 확장 지원 사용

PostgreSQL에 대한 Amazon RDS 위임 확장 지원을 사용하면 `rds_superuser` 역할이 아닌 사용자에게도 확장 관리를 위임할 수 있습니다. 이 위임 확장 지원을 통해 `rds_extension`이라는 새 역할이 생성되며, 이 역할이 할당되어야 해당 사용자가 다른 확장을 관리할 수 있습니다. 이 역할은 확장을 생성하고, 업데이트하고, 삭제할 수 있습니다.

RDS DB 인스턴스에 설치할 수 있는 확장을 `rds.allowed_extensions` 파라미터에 등록하여 지정할 수 있습니다. 자세한 내용은 Amazon RDS for PostgreSQL로 PostgreSQL 확장 사용을 참조하세요.

`rds.allowed_delegated_extensions` 파라미터를 사용하여 `rds_extension` 역할이 있는 사용자가 관리할 수 있는 가용 확장 목록을 제한할 수 있습니다.

위임 확장 지원은 다음 버전에서 사용할 수 있습니다.

- 모든 상위 버전
- 16.4 이상의 16 버전
- 15.8 이상의 15 버전
- 14.13 이상의 14 버전
- 13.16 이상의 13 버전
- 12.20 이상의 12 버전

주제

- 사용자에게 위임 확장 지원 활성화
- RDS 위임 확장 지원에서 PostgreSQL에 대해 사용되는 구성
- 위임 확장에 대한 지원 비활성화
- Amazon RDS 위임 확장 지원 사용의 이점
- PostgreSQL에 대한 Amazon RDS 위임 확장 지원 제한
- 특정 확장에 필요한 권한
- 보안 고려 사항
- 확장 삭제 캐스케이드 비활성화
- 위임 확장 지원을 사용하여 추가할 수 있는 확장의 예

사용자에게 위임 확장 지원 활성화

사용자에게 위임 확장 지원을 활성화하려면 다음을 수행해야 합니다.

1. 사용자에게 **rds_extension** 역할 부여 - rds_superuser 권한으로 데이터베이스에 연결하고 다음 명령을 실행합니다.

```
Postgres => grant rds_extension to user_name;
```

2. 위임된 사용자가 관리할 수 있는 확장 목록 설정 - rds.allowed_delegated_extensions를 통해 DB 클러스터 파라미터에서 rds.allowed_extensions를 사용하여 가용 확장의 하위 집합을 지정할 수 있습니다. 다음 수준 중 하나에서 이 작업을 수행할 수 있습니다.

   - 클러스터 또는 인스턴스 파라미터 그룹에서 AWS Management Console 또는 API를 통해 가능합니다. 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.
   - 데이터베이스 수준에서 다음 명령을 사용하세요.

```
alter database database_name set rds.allowed_delegated_extensions =
 'extension_name_1,
                   extension_name_2,...extension_name_n';
```

   - 사용자 수준에서 다음 명령을 사용하세요.

```
alter user user_name set rds.allowed_delegated_extensions = 'extension_name_1,
                   extension_name_2,...extension_name_n';
```

> **Note**
>
> rds.allowed_delegated_extensions 동적 파라미터를 변경한 후에는 데이터베이스를 다시 시작할 필요가 없습니다.

3. 확장 생성 프로세스 중에 생성된 객체에 대한 위임된 사용자 액세스 허용 - 특정 확장에서는 추가 권한을 부여해야 rds_extension 역할을 가진 사용자가 해당 개체에 액세스할 수 있는 개체를 만들 수 있습니다. rds_superuser는 위임된 사용자에게 해당 객체에 대한 액세스 권한을 부여해야 합니다. 이벤트 트리거를 사용하여 위임된 사용자에게 권한을 자동으로 부여하는 것도 한 가지 방법입니다. 자세한 정보는 위임 확장에 대한 지원 비활성화에서 이벤트 트리거 예제를 참조하세요.

RDS 위임 확장 지원에서 PostgreSQL에 대해 사용되는 구성

구성 이름	설명	기본 값	참고	권한을 수정하거나 부여할 수 있는 사람
rds.allowed_delegated_extensions	이 파라미터는 rds_extension 역할이 이 데이터베이스에서 관리할 수 있는 확장을 제한합니다. rds.allowed_extensions의 하위 집합이어야 합니다.	빈 문자열	• 기본적으로 이 파라미터는 빈 문자열이므로 rds_extension 을 사용하는 사용자에게 확장이 위임되지 않았음을 의미합니다. • 사용자에게 권한이 있는 경우 모든 지원되는 확장을 추가할 수 있습니다. 이렇게 하려면 rds.allowed_delegated_extensions 파라미터를 쉼표로 구분된 확장 이름(문자열)으로 설정합니다. 이 파라미터에 확장 목록을 추가하면 rds_extension 역할의 사용자가 설치할 수 있는 확장을 명시적으로 파악할 수 있습니다. • *로 설정하면 rds_allow	rds_superuser

구성 이름	설명	기본 값	참고	권한을 수정하거나 부여할 수 있는 사람
			ed_extensions 에 나열된 모든 확장이 rds_extension 역할의 사용자에게 위임된다는 의미입니다. 이 파라미터 설정에 대한 자세한 내용은 사용자에게 위임 확장 지원 활성화 섹션을 참조하세요.	
rds.allowed_extensions	이 파라미터를 사용하면 고객이 RDS DB 인스턴스에 설치할 수 있는 확장을 제한할 수 있습니다. 자세한 내용은 PostgreSQL 확장 설치 제한을 참조하세요.	"*"	기본적으로 이 파라미터는 '*'로 설정되어 있습니다. 즉, RDS for PostgreSQL과 Aurora PostgreSQL에서 지원되는 모든 확장은 필요한 권한이 있는 사용자가 만들 수 있습니다. 이 값이 비어 있으면 RDS DB 인스턴스에 확장을 설치할 수 없습니다.	관리자

구성 이름	설명	기본 값	참고	권한을 수정하거나 부여할 수 있는 사람
rds-deleg ated_ nsior ow_dr ascac	이 파라미터는 rds_extension 권한이 있는 사용자가 캐스케이드 옵션을 사용하여 확장을 삭제할 수 있는 기능을 제어합니다.	꺼	기본적으로 rds-delegated_exte nsion_all ow_drop_c ascade 는 off로 설정됩니다. 즉, rds_extension 사용자는 캐스케이드 옵션을 사용하여 확장 프로그램을 삭제할 수 없습니다. 이 기능을 부여하려면 rds.deleg ated_exte nsion_all ow_drop_c ascade 파라미터를 on으로 설정해야 합니다.	rds_superuser

위임 확장에 대한 지원 비활성화

부분적으로 비활성화

위임된 사용자는 새 확장을 만들 수는 없지만 기존 확장을 계속 업데이트할 수는 있습니다.

- DB 클러스터 파라미터 그룹에서 rds.allowed_delegated_extensions를 기본값으로 재설정합니다.
- 데이터베이스 수준에서 다음 명령을 사용하세요.

```
alter database database_name reset rds.allowed_delegated_extensions;
```

- 사용자 수준에서 다음 명령을 사용하세요.

```
alter user user_name reset rds.allowed_delegated_extensions;
```

전체적으로 비활성화

사용자로부터 rds_extension 역할을 취소하면 사용자는 표준 권한으로 되돌아갑니다. 해당 사용자는 더 이상 확장을 생성, 업데이트 또는 삭제할 수 없습니다.

```
postgres => revoke rds_extension from user_name;
```

이벤트 트리거 예제

확장 생성 시 만들어진 객체에 대한 권한 설정이 필요한 확장을 rds_extension 권한이 있는 위임된 사용자가 사용할 수 있도록 하려면, 아래 이벤트 트리거 예제를 사용자 지정하고 위임된 사용자가 전체 기능에 액세스할 수 있도록 하려는 확장만 추가하면 됩니다. 이 이벤트 트리거는 template1(기본 템플릿)에서 만들 수 있으므로, template1에서 만든 모든 데이터베이스에는 해당 이벤트 트리거가 있습니다. 위임된 사용자가 확장을 설치하면 이 트리거는 확장에서 생성된 객체에 대한 소유권을 자동으로 부여합니다.

```
CREATE OR REPLACE FUNCTION create_ext()

  RETURNS event_trigger AS $$

DECLARE

  schemaname TEXT;
  databaseowner TEXT;

  r RECORD;

BEGIN

  IF tg_tag = 'CREATE EXTENSION' and current_user != 'rds_superuser' THEN
    RAISE NOTICE 'SECURITY INVOKER';
    RAISE NOTICE 'user: %', current_user;
    FOR r IN SELECT * FROM pg_event_trigger_ddl_commands()
    LOOP
        CONTINUE WHEN r.command_tag != 'CREATE EXTENSION' OR r.object_type !=
 'extension';
```

```
        schemaname = (
            SELECT n.nspname
            FROM pg_catalog.pg_extension AS e
            INNER JOIN pg_catalog.pg_namespace AS n
            ON e.extnamespace = n.oid
            WHERE e.oid = r.objid
        );

        databaseowner = (
            SELECT pg_catalog.pg_get_userbyid(d.datdba)
            FROM pg_catalog.pg_database d
            WHERE d.datname = current_database()
        );
        RAISE NOTICE 'Record for event trigger %, objid: %,tag: %, current_user: %,
schema: %, database_owenr: %', r.object_identity, r.objid, tg_tag, current_user,
schemaname, databaseowner;
        IF r.object_identity = 'address_standardizer_data_us' THEN
            EXECUTE format('GRANT SELECT, UPDATE, INSERT, DELETE ON TABLE %I.us_gaz TO
%I WITH GRANT OPTION;', schemaname, databaseowner);
            EXECUTE format('GRANT SELECT, UPDATE, INSERT, DELETE ON TABLE %I.us_lex TO
%I WITH GRANT OPTION;', schemaname, databaseowner);
            EXECUTE format('GRANT SELECT, UPDATE, INSERT, DELETE ON TABLE %I.us_rules
TO %I WITH GRANT OPTION;', schemaname, databaseowner);
        ELSIF r.object_identity = 'dict_int' THEN
            EXECUTE format('ALTER TEXT SEARCH DICTIONARY %I.intdict OWNER TO %I;',
schemaname, databaseowner);
        ELSIF r.object_identity = 'pg_partman' THEN
            EXECUTE format('GRANT SELECT, UPDATE, INSERT, DELETE ON TABLE
%I.part_config TO %I WITH GRANT OPTION;', schemaname, databaseowner);
            EXECUTE format('GRANT SELECT, UPDATE, INSERT, DELETE ON TABLE
%I.part_config_sub TO %I WITH GRANT OPTION;', schemaname, databaseowner);
            EXECUTE format('GRANT SELECT, UPDATE, INSERT, DELETE ON TABLE
%I.custom_time_partitions TO %I WITH GRANT OPTION;', schemaname, databaseowner);
        ELSIF r.object_identity = 'postgis_topology' THEN
            EXECUTE format('GRANT SELECT, UPDATE, INSERT, DELETE ON ALL TABLES IN
SCHEMA topology TO %I WITH GRANT OPTION;', databaseowner);
            EXECUTE format('GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA topology TO
%I WITH GRANT OPTION;', databaseowner);
            EXECUTE format('GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA topology TO %I
WITH GRANT OPTION;', databaseowner);
            EXECUTE format('GRANT USAGE ON SCHEMA topology TO %I WITH GRANT OPTION;',
databaseowner);
        END IF;
    END LOOP;
```

```
  END IF;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;

CREATE EVENT TRIGGER log_create_ext ON ddl_command_end EXECUTE PROCEDURE create_ext();
```

Amazon RDS 위임 확장 지원 사용의 이점

PostgreSQL에 대한 Amazon RDS 위임 확장 지원을 사용하면 `rds_superuser` 역할이 아닌 사용자에게도 확장 관리를 안전하게 위임할 수 있습니다. 이 기능에는 다음과 같은 이점이 있습니다.

- 원하는 사용자에게 확장 관리를 쉽게 위임할 수 있습니다.
- 이 작업에는 `rds_superuser` 역할이 필요하지 않습니다.
- 동일한 DB 클러스터의 여러 데이터베이스에 대해 서로 다른 확장 세트를 지원하는 기능을 제공합니다.

PostgreSQL에 대한 Amazon RDS 위임 확장 지원 제한

- 확장 생성 프로세스 중에 생성되는 객체에서 확장이 제대로 작동하려면 추가 권한이 필요할 수 있습니다.
- `log_fdw`, `pg_cron`, `pg_tle`, `pgactive`, `pglogical`, `postgis_raster`, `postgis_tiger_geocoder`, `postgis_topology`를 포함한 일부 확장은 위임된 확장 사용자가 기본적으로 관리할 수 없습니다.

특정 확장에 필요한 권한

다음과 같은 확장을 생성, 사용 또는 업데이트하려면 위임된 사용자에게 다음과 같은 함수, 테이블 및 스키마에 대해 필요한 권한이 있어야 합니다.

소유권이나 권한이 필요한 확장	함수	표	스키마	텍스트 검색 사전	설명
addre tanda er_da s	없음	us_gaz, us_lex, us_lex, l.us_rules	없음	없음	없음
amch	bt_index_check, bt_index_ parent_check	없음	없음	없음	없음
dict_i	없음	없음	없음	intdict	없음
pg_p n	없음	custom_ti me_partitions, part_config, part_config_sub	없음	없음	없음
pg_st tatem	없음	없음	없음	없음	없음
Post(	st_tileenvelope	spatial_ref_sys	없음	없음	없음
postg aster	없음	없음	없음	없음	없음

소유권이나 권한이 필요한 확장	함수	표	스키마	텍스트 검색 사전	설명
postg opolo	없음	topology, layer	topology	없음	위임된 사용자는 데이터베이스 소유자여야 함
log_f	create_foreign_table_for_log_file	없음	없음	없음	없음
rds_t	role_password_encryption_type	없음	없음	없음	없음
postg iger_oder	없음	geocode_settings_default, geocode_settings	tiger	없음	없음
pg_fr acem	pg_freespace	없음	없음	없음	없음
pg_vi lity	pg_visibility	없음	없음	없음	없음

보안 고려 사항

rds_extension 역할이 있는 사용자는 접속 권한이 있는 모든 데이터베이스의 확장을 관리할 수 있다는 점을 기억하세요. 위임된 사용자가 단일 데이터베이스의 확장을 관리하도록 하려는 경우, 각 데이터베이스의 모든 공개 권한을 취소한 다음 해당 특정 데이터베이스의 연결 권한을 위임 사용자에게 명시적으로 부여하는 것이 좋습니다.

사용자가 여러 데이터베이스의 정보에 액세스할 수 있도록 하는 확장에는 여러 가지가 있습니다. 이러한 확장을 rds.allowed_delegated_extensions에 추가하기 전에 rds_extension 권한을 부여한 사용자가 데이터베이스 간 기능을 사용할 수 있는지 확인하세요. 예를 들어, postgres_fdw 및 dblink는 동일한 인스턴스 또는 원격 인스턴스의 여러 데이터베이스를 쿼리할 수 있는 기능을 제공합니다. log_fdw는 인스턴스의 모든 데이터베이스에 대한 postgres 엔진 로그 파일을 읽습니다. 이 로그 파일에는 여러 데이터베이스의 느린 쿼리나 오류 메시지가 포함될 수 있습니다. pg_cron은 예약된 백그라운드 작업을 DB 인스턴스에서 실행할 수 있도록 하고 작업이 다른 데이터베이스에서 실행되도록 구성할 수 있습니다.

확장 삭제 캐스케이드 비활성화

rds_extension 역할의 사용자가 캐스케이드 옵션을 사용하여 확장을 삭제하는 기능은 rds.delegated_extension_allow_drop_cascade 파라미터에 의해 제어됩니다. 기본적으로 rds-delegated_extension_allow_drop_cascade는 off로 설정됩니다. 즉, rds_extension 역할의 사용자는 아래 쿼리에 나와 있는 캐스케이드 옵션으로 확장을 삭제할 수 없습니다.

```
DROP EXTENSION CASCADE;
```

이렇게 하면 확장에 종속된 객체와 해당 객체에 종속된 모든 객체가 자동으로 삭제되기 때문입니다. 캐스케이드 옵션을 사용하려고 시도하면 오류가 발생합니다.

이 기능을 부여하려면 rds.delegated_extension_allow_drop_cascade 파라미터를 on으로 설정해야 합니다.

rds.delegated_extension_allow_drop_cascade 동적 파라미터를 변경하기 위해 데이터베이스를 다시 시작할 필요가 없습니다. 다음 수준 중 하나에서 이 작업을 수행할 수 있습니다.

- 클러스터 또는 인스턴스 파라미터 그룹에서 AWS Management Console 또는 API를 통해 가능합니다.
- 데이터베이스 수준에서 다음 명령을 사용하세요.

```
alter database database_name set rds.delegated_extension_allow_drop_cascade = 'on';
```

- 사용자 수준에서 다음 명령을 사용하세요.

```
alter role tenant_user set rds.delegated_extension_allow_drop_cascade = 'on';
```

위임 확장 지원을 사용하여 추가할 수 있는 확장의 예

- rds_tools

```
extension_test_db=> create extension rds_tools;
CREATE EXTENSION
extension_test_db=> SELECT * from rds_tools.role_password_encryption_type() where
 rolname = 'pg_read_server_files';
ERROR: permission denied for function role_password_encryption_type
```

- amcheck

```
extension_test_db=> CREATE TABLE amcheck_test (id int);
CREATE TABLE
extension_test_db=> INSERT INTO amcheck_test VALUES (generate_series(1,100000));
INSERT 0 100000
extension_test_db=> CREATE INDEX amcheck_test_btree_idx ON amcheck_test USING btree
 (id);
CREATE INDEX
extension_test_db=> create extension amcheck;
CREATE EXTENSION
extension_test_db=> SELECT bt_index_check('amcheck_test_btree_idx'::regclass);
ERROR: permission denied for function bt_index_check
extension_test_db=> SELECT bt_index_parent_check('amcheck_test_btree_idx'::regclass);
ERROR: permission denied for function bt_index_parent_check
```

- pg_freespacemap

```
extension_test_db=> create extension pg_freespacemap;
CREATE EXTENSION
extension_test_db=> SELECT * FROM pg_freespace('pg_authid');
ERROR: permission denied for function pg_freespace
extension_test_db=> SELECT * FROM pg_freespace('pg_authid',0);
ERROR: permission denied for function pg_freespace
```

- pg_visibility

```
extension_test_db=> create extension pg_visibility;
CREATE EXTENSION
extension_test_db=> select * from pg_visibility('pg_database'::regclass);
ERROR: permission denied for function pg_visibility
```

- postgres_fdw

```
extension_test_db=> create extension postgres_fdw;
CREATE EXTENSION
extension_test_db=> create server myserver foreign data wrapper postgres_fdw options
 (host 'foo', dbname 'foodb', port '5432');
ERROR: permission denied for foreign-data wrapper postgres_fdw
```

pg_partman 확장자를 사용하여 PostgreSQL 파티션 관리하기

PostgreSQL 테이블 파티셔닝은 데이터 입력 및 보고의 고성능 처리를 위한 프레임워크를 제공합니다. 대량의 데이터를 매우 빠르게 입력해야 하는 데이터베이스의 경우 파티셔닝을 사용합니다. 파티셔닝은 큰 테이블의 빠른 쿼리를 제공합니다. 파티셔닝은 I/O 리소스가 적기 때문에 데이터베이스 인스턴스에 영향을 주지 않고 데이터를 유지 관리하는 데 도움이 됩니다.

파티셔닝을 사용하면 데이터를 사용자 지정 크기의 청크로 분할하여 처리할 수 있습니다. 예를 들어, 시간별, 일별, 주별, 월별, 분기별, 연도별, 사용자 지정 또는 이러한 조합과 같은 범위의 시계열 데이터를 분할할 수 있습니다. 시계열 데이터 예제의 경우 테이블을 시간별로 분할한 경우 각 파티션에는 1시간의 데이터가 포함됩니다. 시계열 테이블을 일별로 분할한 경우 파티션에는 하루 분량의 데이터가 저장되는 식입니다. 파티션 키는 파티션의 크기를 제어합니다.

분할된 테이블에서 `INSERT` 또는 `UPDATE` SQL 명령을 사용하는 경우 데이터베이스 엔진은 데이터를 적절한 파티션으로 라우팅합니다. 데이터를 저장하는 PostgreSQL 테이블 파티션은 기본 테이블의 하위 테이블입니다.

데이터베이스 쿼리를 읽는 동안 PostgreSQL 최적화 프로그램은 쿼리 `WHERE` 절을 검사하고 가능한 경우 데이터베이스 스캔을 관련 파티션에만 보냅니다.

버전 10부터, PostgreSQL은 선언적 분할을 사용하여 테이블 분할을 구현합니다. 이를 네이티브 PostgreSQL 파티셔닝이라고도 합니다. PostgreSQL 버전 10 이전에는 트리거를 사용하여 파티션을 구현했습니다.

PostgreSQL 테이블 분할은 다음과 같은 기능을 제공합니다:

- 언제든지 새 파티션을 만들 수 있습니다.
- 가변 분할 영역 범위입니다.
- 데이터 정의 언어(DDL) 문을 사용하여 분리 및 재연결 가능한 파티션

  예를 들어 분리 가능한 파티션은 주 파티션에서 기록 데이터를 제거하지만 분석을 위해 기록 데이터를 유지하는 데 유용합니다.
- 새 파티션은 다음을 포함하여 상위 데이터베이스 테이블 속성을 상속합니다.
  - 인덱스
  - 파티션 키 열을 포함해야 하는 기본 키
  - 외래 키
  - 제약 점검
  - 참조

- 전체 테이블 또는 각 특정 파티션에 대한 인덱스 생성

개별 파티션에 대한 스키마를 변경할 수 없습니다. 그러나 분할 영역에 전파되는 상위 테이블 (예: 새 열 추가) 을 변경할 수 있습니다.

주제

- PostgreSQL pg_partman 확장 개요
- pg_partman 확장 활성화
- create_parent 함수를 사용하여 파티션 구성
- run_maintenance_proc 함수를 사용하여 파티션 유지 관리 구성

PostgreSQL pg_partman 확장 개요

PostgreSQL pg_partman 확장을 사용하여 테이블 파티션의 생성 및 유지 관리를 자동화할 수 있습니다. 자세한 내용은 pg_partman 문서의 PG 파티션 관리자를 참조하세요.

> **Note**
>
> pg_partman 확장은 RDS for PostgreSQL 버전 12.5 이상에서 지원됩니다.

각 파티션을 수동으로 생성하지 않고 다음 설정으로 pg_partman을 구성합니다.

- 분할할 테이블
- 파티션 유형
- 파티션 키
- 분할 영역 세분성
- 파티션 사전 생성 및 관리 옵션

PostgreSQL 분할 테이블을 만든 후, create_parent 함수를 호출하여 pg_partman에 등록합니다. 이렇게 하면 함수에 전달하는 파라미터를 기반으로 필요한 파티션이 생성됩니다.

pg_partman 확장은 예약된 시간에 호출하여 파티션을 자동으로 관리할 수 있는 run_maintenance_proc 함수도 제공합니다. 필요에 따라 적절한 파티션이 생성되도록 하려면 이 기능을 주기적으로 실행하도록 예약합니다(예: 매 시간). 파티션이 자동으로 삭제되게 할 수도 있습니다.

pg_partman 확장 활성화

파티션을 관리하려는 동일한 PostgreSQL DB 인스턴스 내에 여러 개의 데이터베이스가 있는 경우 각 데이터베이스별로 별도로 pg_partman 확장을 활성화해야 합니다. 특정 데이터베이스에 대해 pg_partman 확장을 활성화하려면 파티션 유지 관리 스키마를 생성한 후 다음과 같이 pg_partman 확장을 생성합니다.

```
CREATE SCHEMA partman;
CREATE EXTENSION pg_partman WITH SCHEMA partman;
```

> **Note**
>
> pg_partman 확장을 만들려면 rds_superuser 권한이 있는지 확인하세요.

다음과 같은 오류가 발생하면 계정에 rds_superuser 권한을 부여하거나 수퍼유저 계정을 사용하십시오.

```
ERROR: permission denied to create extension "pg_partman"
HINT: Must be superuser to create this extension.
```

rds_superuser 권한을 부여하려면 수퍼유저 계정으로 연결하고 다음 명령을 실행합니다.

```
GRANT rds_superuser TO user-or-role;
```

pg_partman 확장 사용 방법을 보여 주는 예제에서는 다음 예제 데이터베이스 테이블 및 파티션을 사용합니다. 이 데이터베이스는 타임스탬프를 기반으로 분할된 테이블을 사용합니다. data_mart 스키마에는 라는 열이 events 있는 테이블이 포함되어 created_at있습니다. 이 events 테이블에는 다음과 같은 설정이 포함되어 있습니다.

- 기본 키 event_id 및 created_at파티션을 안내하는 데 사용되는 열이 있어야 합니다.
- ck_valid_operation 테이블 열에 대한 값을 operation 적용하는 CHECK 제약 조건.
- 두 개의 외래 키. 하나는 외부 테이블을fk_orga_membership) organization 가리키고 다른 하나는 자체 참조 외래 키입니다.fk_parent_event_id
- 두 개의 인덱스. 여기서 하나 (idx_org_id) 는 외래 키용이고 다른 인덱스 (idx_event_type) 는 이벤트 유형용입니다.

다음 DDL 명령문은 각 파티션에 자동으로 포함되는 이러한 객체를 만듭니다.

```
CREATE SCHEMA data_mart;
CREATE TABLE data_mart.organization ( org_id BIGSERIAL,
        org_name TEXT,
        CONSTRAINT pk_organization PRIMARY KEY (org_id)
    );

CREATE TABLE data_mart.events(
        event_id        BIGSERIAL,
        operation       CHAR(1),
        value           FLOAT(24),
        parent_event_id BIGINT,
        event_type      VARCHAR(25),
        org_id          BIGSERIAL,
        created_at      timestamp,
        CONSTRAINT pk_data_mart_event PRIMARY KEY (event_id, created_at),
        CONSTRAINT ck_valid_operation CHECK (operation = 'C' OR operation = 'D'),
        CONSTRAINT fk_orga_membership
            FOREIGN KEY(org_id)
            REFERENCES data_mart.organization (org_id),
        CONSTRAINT fk_parent_event_id
            FOREIGN KEY(parent_event_id, created_at)
            REFERENCES data_mart.events (event_id,created_at)
    ) PARTITION BY RANGE (created_at);

CREATE INDEX idx_org_id     ON  data_mart.events(org_id);
CREATE INDEX idx_event_type ON  data_mart.events(event_type);
```

create_parent 함수를 사용하여 파티션 구성

pg_partman 확장을 활성화한 후에는 create_parent 함수를 사용하여 파티션 유지 관리 스키마 내에 파티션을 구성합니다. 다음 예에서는 events에서 만든 pg_partman 확장 활성화 테이블 예제를 사용합니다. create_parent 함수를 다음과 같이 호출합니다.

```
SELECT partman.create_parent(
 p_parent_table => 'data_mart.events',
 p_control      => 'created_at',
 p_type         => 'range',
 p_interval     => '1 day',
 p_premake      => 30);
```

파라미터는 다음과 같습니다.

- p_parent_table – 상위 분할 테이블입니다. 이 테이블은 이미 존재해야 하며 스키마를 포함하여 정규화되어야 합니다.
- p_control – 분할이 기반으로되는 열입니다. 데이터 유형은 정수 또는 시간 기반이어야 합니다.
- p_type - 유형은 'range' 또는 'list'입니다.
- p_interval – 각 파티션에 대한 시간 간격 또는 정수 범위입니다. 값의 예로는 1 day, 1 hour 등이 포함됩니다.
- p_premake – 새 삽입을 지원하기 위해 미리 생성할 분할 영역 수입니다.

create_parent 함수에 대한 자세한 설명은 pg_partman 설명서에서 생성 함수를 참조하세요.

run_maintenance_proc 함수를 사용하여 파티션 유지 관리 구성

파티션 유지 관리 작업을 실행하여 자동으로 새 파티션을 생성하거나, 파티션을 분리하거나, 이전 파티션을 제거할 수 있습니다. 파티션 유지 관리에는 내부 스케줄러를 시작하는 pg_partman 확장 및 pg_cron 확장의 run_maintenance_proc 함수가 사용됩니다. pg_cron 스케줄러는 데이터베이스에 정의된 SQL 문, 함수 및 프로시저를 자동으로 실행합니다.

다음 예제에서는 에서 만든 events 테이블 예제를 pg_partman 확장 활성화 사용하여 파티션 유지 관리 작업이 자동으로 실행되도록 설정합니다. 전제 조건으로 pg_cron을 DB 인스턴스의 파라미터 그룹의 shared_preload_libraries 파라미터에 추가합니다.

```
CREATE EXTENSION pg_cron;

UPDATE partman.part_config
SET infinite_time_partitions = true,
    retention = '3 months',
    retention_keep_table=true
WHERE parent_table = 'data_mart.events';
SELECT cron.schedule('@hourly', $$CALL partman.run_maintenance_proc()$$);
```

다음으로, 앞의 예제에 대한 단계별 설명을 찾을 수 있습니다.

1. DB 인스턴스와 연결된 파라미터 그룹을 수정하고 pg_cron 파라미터 값에 shared_preload_libraries 추가합니다. 이 변경 사항을 적용하려면 DB 인스턴스를 다시 시작해야 합니다. 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요.

2. CREATE EXTENSION pg_cron; 권한이 있는 계정을 사용하여 rds_superuser 명령을 실행합니다. 이렇게 하면 pg_cron 확장이 활성화됩니다. 자세한 내용은 PostgreSQL pg_cron 확장을 사용하여 유지 관리 예약 섹션을 참조하세요.
3. UPDATE partman.part_config 명령을 실행하여 data_mart.events 테이블에 대한 pg_partman 설정을 조정합니다.
4. SET 명령을 실행합니다. data_mart.events 테이블을 구성하려면 다음 절을 함께 사용합니다.
   a. infinite_time_partitions = true, – 제한 없이 자동으로 새 파티션을 만들 수 있도록 테이블을 구성합니다.
   b. retention = '3 months', – 최대 보존 기간이 3개월이 되도록 테이블을 구성합니다.
   c. retention_keep_table=true – 보존 기간이 만료될 때 테이블이 자동으로 삭제되지 않도록 테이블을 구성합니다. 대신 보존 기간보다 오래된 파티션은 상위 테이블에서만 분리됩니다.
5. SELECT cron.schedule 명령을 실행합니다. pg_cron 함수를 호출하려면 이 호출은 스케줄러가 pg_partman 유지 관리 프로시저 partman.run_maintenance_proc를 실행하는 빈도를 정의합니다. 이 예제에서는 프로시저가 매 시간 실행됩니다.

run_maintenance_proc 함수에 대한 자세한 설명은 pg_partman 설명서에서 유지 관리 함수를 참조하세요.

pgAudit를 사용하여 데이터베이스 활동 로깅

금융 기관, 정부 기관 및 많은 업계에서는 규제 요구 사항을 충족하기 위해 감사 로그를 보관해야 합니다. RDS for PostgreSQL DB 인스턴스와 함께 PostgreSQL Audit 확장(pgAudit)을 사용하면 감사자가 일반적으로 필요로 하거나 규제 요구 사항을 충족하는 데 필요한 세부 레코드를 캡처할 수 있습니다. 예를 들어 특정 데이터베이스 및 테이블의 변경 내용을 추적하고 변경한 사용자 및 기타 여러 세부 정보를 기록하도록 pgAudit 확장을 설정할 수 있습니다.

pgAudit 확장은 네이티브 PostgreSQL 로깅 인프라의 기능을 기반으로 로그 메시지를 더 자세히 확장한 것입니다. 즉, 다른 로그 메시지를 보는 것과 동일한 접근 방식을 사용하여 감사 로그를 볼 수 있습니다. PostgreSQL 로깅에 대한 자세한 내용은 RDS for PostgreSQL 데이터베이스 로그 파일 섹션을 참조하세요.

pgAudit 확장은 일반 텍스트 암호와 같은 민감한 데이터를 로그에서 삭제합니다. RDS for PostgreSQL DB 인스턴스가 RDS for PostgreSQL DB 인스턴스에 쿼리 로깅을 활성화합니다.에 설명된 대로 데이터 조작 언어(DML) 문을 로깅하도록 구성된 경우 PostgreSQL Audit 확장을 사용하여 일반 텍스트 암호 문제를 방지할 수 있습니다.

매우 구체적으로 데이터베이스 인스턴스에 대한 감사를 구성할 수 있습니다. 모든 데이터베이스와 모든 사용자를 감사할 수 있습니다. 또는 특정 데이터베이스, 사용자 및 기타 객체만 감사하도록 선택할 수 있습니다. 특정 사용자 및 데이터베이스를 감사에서 명시적으로 제외할 수도 있습니다. 자세한 내용은 감사 로깅에서 사용자 또는 데이터베이스 제외 단원을 참조하십시오.

캡처할 수 있는 세부 정보의 양을 고려하여 pgAudit를 사용하는 경우 스토리지 사용량을 모니터링하는 것이 좋습니다.

pgAudit 확장은 사용 가능한 모든 RDS for PostgreSQL 버전 RDS for PostgreSQL 버전에서 지원되는 pgAudit 버전 목록을 보려면 Aurora PostgreSQL 릴리스 정보의 Extension versions for Amazon RDS for PostgreSQL(Amazon RDS for PostgreSQL 확장 버전)을 참조하세요.

주제

- pgAudit 확장 설정
- 데이터베이스 객체 감사
- 감사 로깅에서 사용자 또는 데이터베이스 제외
- pgAudit 확장 프로그램에 대한 참조

pgAudit 확장 설정

RDS for PostgreSQL DB 인스턴스에 pgAudit 확장을 설정하려면 먼저 RDS for PostgreSQL DB 인스턴스용 사용자 지정 DB 파라미터 그룹의 공유 라이브러리에 pgAudit을 추가해야 합니다. 사용자 지정 DB 클러스터 파라미터 그룹을 만드는 방법에 관한 자세한 내용은 Amazon RDS의 파라미터 그룹 섹션을 참조하세요. 다음으로 pgAudit 확장을 설치합니다. 마지막으로, 감사하려는 데이터베이스 및 객체를 지정합니다. 이 섹션에 절차가 설명되어 있습니다. AWS Management Console 또는 AWS CLI를 사용할 수 있습니다.

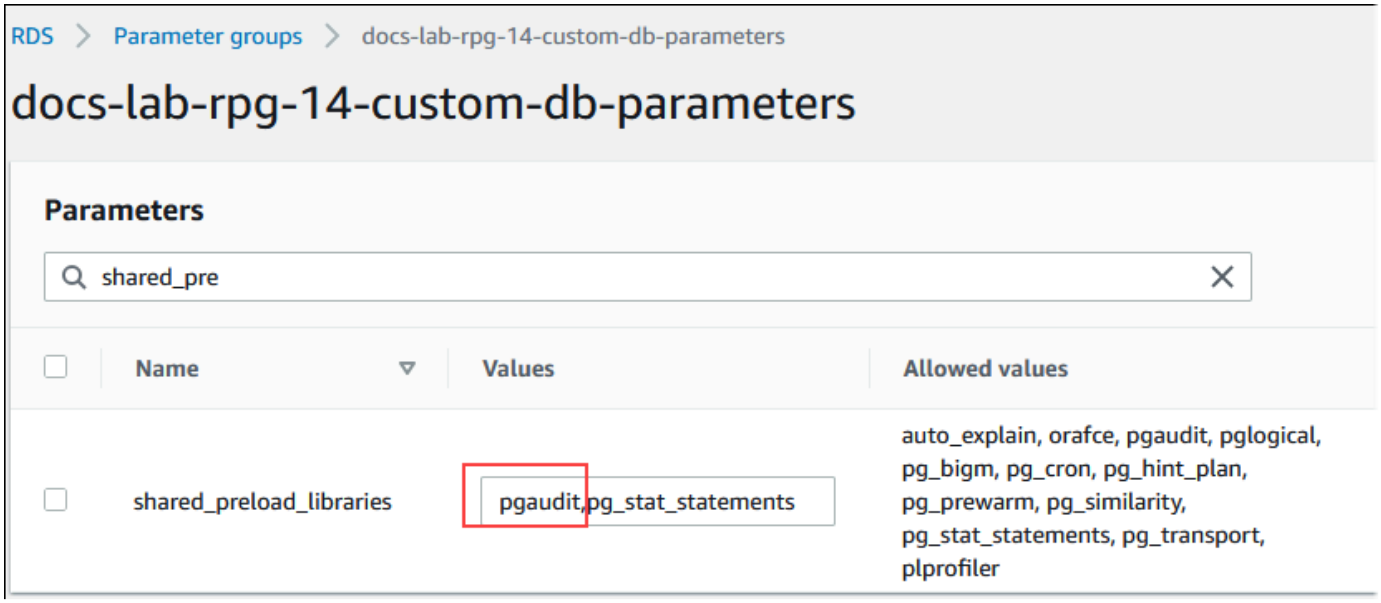
이 모든 작업을 수행하려면 `rds_superuser` 역할의 권한이 있어야 합니다.

다음 단계에서는 사용자의 RDS for PostgreSQL DB 인스턴스가 사용자 지정 DB에 연결되어 있다고 가정합니다.

콘솔

pgAudit 확장 설정 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 RDS for PostgreSQL DB 인스턴스를 선택합니다.
3. 의 구성 탭을 엽니다. RDS for PostgreSQL DB 인스턴스 인스턴스 세부 정보 중에서 파라미터 그룹 링크를 찾습니다.
4. 링크를 선택하여 와 연결된 사용자 지정 파라미터를 엽니다. RDS for PostgreSQL DB 인스턴스
5. 파라미터 검색 필드에 `shared_pre`를 입력하여 `shared_preload_libraries` 파라미터를 찾습니다.
6. 파라미터 편집을 선택하여 속성 값에 액세스합니다.
7. 값 필드의 목록에 `pgaudit`를 추가합니다. 쉼표를 사용하여 값 목록에서 항목을 구분합니다.



8. RDS for PostgreSQL DB 인스턴스를 재부팅하여 `shared_preload_libraries` 파라미터 변경 사항이 적용되도록 합니다.
9. 인스턴스를 사용할 수 있게 되면 pgAudit가 초기화되었는지 확인합니다. `psql`을 사용하여 RDS for PostgreSQL DB 인스턴스에 연결하고 다음 명령을 실행합니다.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pgaudit
(1 row)
```

10. pgAudit가 초기화되었으므로 이제 확장을 생성할 수 있습니다. 확장은 라이브러리를 초기화한 후에 생성해야 합니다. `pgaudit` 확장이 데이터 정의 언어(DDL) 문 감사를 위한 이벤트 트리거를 설치하기 때문입니다.

```
CREATE EXTENSION pgaudit;
```

11. `psql` 세션을 닫습니다.

```
labdb=> \q
```

12. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.

13. 목록에서 pgaudit.log 파라미터를 찾아 사용 사례에 적합한 값으로 설정합니다. 예를 들어, 다음 이미지처럼 pgaudit.log 파라미터를 write로 설정하면 로그에 대한 삽입, 업데이트, 삭제 및 기타 유형의 변경 사항이 캡처됩니다.



pgaudit.log 파라미터에 다음 값 중 하나를 선택할 수도 있습니다.

- 없음 - 기본값입니다. 데이터베이스 변경 사항이 로깅되지 않습니다.
- 모두 - 모든 항목(읽기, 쓰기, 함수, 역할, ddl, 기타)을 로깅합니다.
- ddl - ROLE 클래스에 포함되지 않은 모든 데이터 정의 언어(DDL) 문을 로깅합니다.
- 함수 - 함수 호출 및 DO 블록을 로깅합니다.
- 기타 – 기타 명령(예: DISCARD, FETCH, CHECKPOINT, VACUUM, SET)을 로깅합니다.
- 읽기 – 원본이 관계(예: 테이블) 또는 쿼리인 경우 SELECT 및 COPY를 로깅합니다.
- 역할 - 역할 및 권한과 관련된 문을 로깅합니다(예: GRANT, REVOKE, CREATE ROLE, ALTER ROLE, DROP ROLE).
- 쓰기 - 대상이 관계(테이블)인 경우 INSERT, UPDATE, DELETE, TRUNCATE, COPY를 로깅합니다.

14. Save changes(변경 사항 저장)를 선택합니다.
15. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
16. 데이터베이스 목록에서 RDS for PostgreSQL DB 인스턴스를 선택합니다.

AWS CLI

pgAudit 설정 방법

AWS CLI를 사용하여 pgAudit를 설정하려면 다음 절차와 같이 modify-db-parameter-group 작업을 호출하여 사용자 지정 파라미터 그룹의 감사 로그 파라미터를 수정합니다.

1. 다음 AWS CLI 명령을 사용하여 shared_preload_libraries 파라미터에 pgaudit를 추가합니다.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pgaudit,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. 다음 AWS CLI 명령으로 RDS for PostgreSQL DB 인스턴스를 재부팅하여 pgaudit 라이브러리가 초기화되도록 합니다.

```
aws rds reboot-db-instance \
    --db-instance-identifier your-instance \
    --region aws-region
```

3. 인스턴스를 사용할 수 있게 되면 pgaudit가 초기화되었는지 확인할 수 있습니다. psql을 사용하여 RDS for PostgreSQL DB 인스턴스에 연결하고 다음 명령을 실행합니다.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pgaudit
(1 row)
```

pgAudit가 초기화되었으므로 이제 확장을 생성할 수 있습니다.

```
CREATE EXTENSION pgaudit;
```

4. AWS CLI를 사용할 수 있도록 psql 세션을 닫습니다.

```
labdb=> \q
```

5. 다음 AWS CLI 명령을 사용하여 세션 감사 로깅을 통해 로깅할 문의 클래스를 지정합니다. 이 예에서는 pgaudit.log 파라미터를 write로 설정하여 로그에 대한 삽입, 업데이트 및 삭제를 캡처합니다.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=pgaudit.log,ParameterValue=write,ApplyMethod=pending-reboot" \
   --region aws-region
```

pgaudit.log 파라미터에 다음 값 중 하나를 선택할 수도 있습니다.

- 없음 - 기본값입니다. 데이터베이스 변경 사항이 로깅되지 않습니다.
- 모두 - 모든 항목(읽기, 쓰기, 함수, 역할, ddl, 기타)을 로깅합니다.
- ddl - ROLE 클래스에 포함되지 않은 모든 데이터 정의 언어(DDL) 문을 로깅합니다.
- 함수 - 함수 호출 및 DO 블록을 로깅합니다.
- 기타 – 기타 명령(예: DISCARD, FETCH, CHECKPOINT, VACUUM, SET)을 로깅합니다.
- 읽기 – 원본이 관계(예: 테이블) 또는 쿼리인 경우 SELECT 및 COPY를 로깅합니다.
- 역할 - 역할 및 권한과 관련된 문을 로깅합니다(예: GRANT, REVOKE, CREATE ROLE, ALTER ROLE, DROP ROLE).
- 쓰기 - 대상이 관계(테이블)인 경우 INSERT, UPDATE, DELETE, TRUNCATE, COPY를 로깅합니다.

다음 AWS CLI 명령으로 RDS for PostgreSQL DB 인스턴스를 재부팅합니다.

```
aws rds reboot-db-instance \
    --db-instance-identifier your-instance \
    --region aws-region
```

데이터베이스 객체 감사

RDS for PostgreSQL DB 인스턴스에 pgAudit를 설정하고 요구 사항에 맞게 구성하면 PostgreSQL 로그에 더 자세한 정보가 캡처됩니다. 예를 들어 기본 PostgreSQL 로깅 구성은 데이터베이스 테이블에서 변경 사항이 적용된 날짜 및 시간을 식별하지만 pgAudit 확장을 사용하면 확장 파라미터의 구성에 따라 스키마, 변경한 사용자 및 기타 세부 정보가 로그 항목에 포함될 수 있습니다. 감사를 설정하여 다음 방법으로 변경 사항을 추적할 수 있습니다.

- 세션마다 사용자별로 추적. 세션 수준에서 정규화된 명령 텍스트를 캡처할 수 있습니다.
- 객체마다 사용자별, 데이터베이스별로 추적

객체 감사 기능은 시스템에서 rds_pgaudit 역할을 만든 다음, 사용자 지정 파라미터 그룹의 pgaudit.role 파라미터에 이 역할을 추가하면 활성화됩니다. 기본적으로 pgaudit.role 파라미터는 설정되어 있지 않으며 유일하게 허용되는 값은 rds_pgaudit입니다. 다음 단계에서는 pgaudit가 초기화되었고 pgAudit 확장 설정의 절차에 따라 pgaudit 확장을 만든 것으로 가정합니다.

```
2022-10-07 23:36:51 UTC:52.95.4.10(14410):postgres@labdb:[1374]:LOG: statement: SELECT feedback, s.sentiment,s.confidence
FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
ORDER BY s.confidence DESC;
2022-10-07 23:36:51 UTC:52.95.4.10(14410):postgres@labdb:[1374]:LOG: AUDIT: SESSION,2,1,READ,SELECT,TABLE,public.support,"SELECT
feedback, s.sentiment,s.confidence
FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
ORDER BY s.confidence DESC;",<none>
2022-10-07 23:36:51 UTC:52.95.4.10(14410):postgres@labdb:[1374]:LOG: QUERY STATISTICS
2022-10-07 23:36:51 UTC:52.95.4.10(14410):postgres@labdb:[1374]:DETAIL: ! system usage stats:
! 0.009494 s user, 0.007442 s system, 0.141985 s elapsed
! [0.022327 s user, 0.007442 s system total]
```

이 예에서 볼 수 있듯이 LOG: AUDIT: SESSION 행은 테이블 및 해당 스키마를 비롯한 세부 정보를 제공합니다.

객체 감사를 설정하는 방법

1. psql을 사용하여 RDS for PostgreSQL DB 인스턴스에 연결합니다.

   ```
   psql --host=your-instance-name.aws-region.rds.amazonaws.com --port=5432 --
   username=postgrespostgres --password --dbname=labdb
   ```

2. 다음 명령을 사용하여 rds_pgaudit라는 데이터베이스 역할을 생성합니다.

   ```
   labdb=> CREATE ROLE rds_pgaudit;
   CREATE ROLE
   labdb=>
   ```

3. psql 세션을 닫습니다.

   ```
   labdb=> \q
   ```

   이어질 몇 단계에서는 AWS CLI를 사용하여 사용자 지정 파라미터 그룹에서 감사 로그 파라미터를 수정합니다.

4. 다음 AWS CLI 명령을 사용하여 rds_pgaudit에 pgaudit.role 파라미터를 추가합니다. 기본적으로 이 파라미터는 설정되어 있지 않으며 유일하게 허용되는 값은 rds_pgaudit입니다.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=pgaudit.role,ParameterValue=rds_pgaudit,ApplyMethod=pending-reboot"
 \
   --region aws-region
```

5. 다음 AWS CLI 명령으로 RDS for PostgreSQL DB 인스턴스를 재부팅하여 파라미터 변경 사항이 적용되도록 합니다.

```
aws rds reboot-db-instance \
    --db-instance-identifier your-instance \
    --region aws-region
```

6. 다음 명령을 실행하여 `pgaudit.role`이 `rds_pgaudit`로 설정되었는지 확인합니다.

```
SHOW pgaudit.role;
pgaudit.role
-----------------
rds_pgaudit
```

pgAudit 로깅을 테스트하기 위해 감사하려는 몇 가지 예제 명령을 실행할 수 있습니다. 예를 들어 다음과 같은 명령을 실행할 수 있습니다.

```
CREATE TABLE t1 (id int);
GRANT SELECT ON t1 TO rds_pgaudit;
SELECT * FROM t1;
id
----
(0 rows)
```

데이터베이스 로그에는 다음과 유사한 항목이 포함됩니다.

```
...
2017-06-12 19:09:49 UTC:...:rds_test@postgres:[11701]:LOG: AUDIT:
OBJECT,1,1,READ,SELECT,TABLE,public.t1,select * from t1;
...
```

로그 확인에 대한 자세한 내용은 Amazon RDS 로그 파일 모니터링 단원을 참조하십시오.

pgAudit 확장 프로그램에 대한 자세한 내용은 GitHub에서 pgAudit을 참조하세요.

감사 로깅에서 사용자 또는 데이터베이스 제외

RDS for PostgreSQL 데이터베이스 로그 파일에서 설명한 대로 PostgreSQL 로그는 스토리지 공간을 사용합니다. pgAudit 확장을 사용하면 추적하는 변경 사항에 따라 로그에 수집된 데이터 양이 다양한 수준으로 늘어납니다. 의 모든 사용자 또는 데이터베이스를 감사할 필요는 없을 수도 있습니다. RDS for PostgreSQL DB 인스턴스

스토리지에 미치는 영향을 최소화하고 불필요하게 감사 레코드를 캡처하지 않도록 사용자 및 데이터베이스를 감사에서 제외할 수 있습니다. 지정된 세션 내에서 로깅을 변경할 수도 있습니다. 다음 예에서는 그 방법을 보여줍니다.

> **Note**
>
> 세션 수준의 파라미터 설정은 RDS for PostgreSQL DB 인스턴스에 대한 사용자 지정 DB 파라미터 그룹의 설정보다 우선합니다. 데이터베이스 사용자가 감사 로깅 구성 설정을 우회하지 못하게 하려면 사용자의 권한을 변경해야 합니다.

RDS for PostgreSQL DB 인스턴스가 모든 사용자 및 데이터베이스에 대해 동일한 수준의 활동을 감사하도록 구성되어 있다고 가정하겠습니다. 그런데 사용자 `myuser`를 감사하지 않기로 결정한다면 다음 SQL 명령으로 `myuser`에 대한 감사를 해제할 수 있습니다.

```
ALTER USER myuser SET pgaudit.log TO 'NONE';
```

그런 다음 다음 쿼리를 사용하여 `pgaudit.log`의 `user_specific_settings` 열에서 파라미터가 `NONE`으로 설정되었는지 확인할 수 있습니다.

```
SELECT
    usename AS user_name,
    useconfig AS user_specific_settings
FROM
    pg_user
WHERE
    usename = 'myuser';
```

출력은 다음과 같습니다.

```
 user_name | user_specific_settings
```

```
-----------+------------------------
 myuser    | {pgaudit.log=NONE}
(1 row)
```

다음 명령을 사용하여 데이터베이스 세션 중에 특정 사용자에 대한 로깅을 해제할 수 있습니다.

```
ALTER USER myuser IN DATABASE mydatabase SET pgaudit.log TO 'none';
```

다음 쿼리를 사용하여 pgaudit.log의 설정 열에서 특정 사용자 및 데이터베이스 조합을 확인할 수 있습니다.

```
SELECT
    usename AS "user_name",
    datname AS "database_name",
    pg_catalog.array_to_string(setconfig, E'\n') AS "settings"
FROM
    pg_catalog.pg_db_role_setting s
    LEFT JOIN pg_catalog.pg_database d ON d.oid = setdatabase
    LEFT JOIN pg_catalog.pg_user r ON r.usesysid = setrole
WHERE
    usename = 'myuser'
    AND datname = 'mydatabase'
ORDER BY
    1,
    2;
```

출력은 다음과 비슷합니다.

```
  user_name | database_name |     settings
-----------+---------------+-----------------
 myuser    | mydatabase    | pgaudit.log=none
(1 row)
```

myuser에 대한 감사를 해제한 후 mydatabase에 대한 변경 내용을 추적하지 않기로 결정한다면 다음 명령을 사용하여 특정 데이터베이스에 대한 감사를 해제합니다.

```
ALTER DATABASE mydatabase SET pgaudit.log to 'NONE';
```

그런 다음, 다음 쿼리를 사용하여 database_specific_settings 열에서 pgaudit.log가 NONE으로 설정되어 있는지 확인합니다.

```
SELECT
a.datname AS database_name,
b.setconfig AS database_specific_settings
FROM
pg_database a
FULL JOIN pg_db_role_setting b ON a.oid = b.setdatabase
WHERE
a.datname = 'mydatabase';
```

출력은 다음과 같습니다.

```
 database_name | database_specific_settings
---------------+----------------------------
 mydatabase    | {pgaudit.log=NONE}
(1 row)
```

myuser의 설정을 기본 설정으로 되돌리려면 다음 명령을 사용합니다.

```
ALTER USER myuser RESET pgaudit.log;
```

데이터베이스의 설정을 기본 설정으로 되돌리려면 다음 명령을 사용합니다.

```
ALTER DATABASE mydatabase RESET pgaudit.log;
```

사용자와 데이터베이스를 기본 설정으로 초기화하려면 다음 명령을 사용합니다.

```
ALTER USER myuser IN DATABASE mydatabase RESET pgaudit.log;
```

pgaudit.log를 pgaudit.log 파라미터에 허용되는 다른 값 중 하나로 설정하여 특정 이벤트를 로그에 캡처할 수도 있습니다. 자세한 내용은 pgaudit.log 파라미터에 허용되는 설정 목록 단원을 참조하십시오.

```
ALTER USER myuser SET pgaudit.log TO 'read';
ALTER DATABASE mydatabase SET pgaudit.log TO 'function';
ALTER USER myuser IN DATABASE mydatabase SET pgaudit.log TO 'read,function'
```

pgAudit 확장 프로그램에 대한 참조

이 섹션에 나열된 파라미터를 하나 이상 변경하여 감사 로그의 세부 정보 수준을 지정할 수 있습니다.

pgAudit 동작 제어

다음 표에 나열된 파라미터 중 하나 이상을 변경하여 감사 로깅을 제어할 수 있습니다.

파라미터	설명
pgaudit.log	세션 감사 로깅으로 로깅될 문의 클래스를 지정합니다. 허용되는 값에는 ddl, 함수, 기타, 읽기, 역할, 쓰기, 없음, 모두 등이 포함됩니다. 자세한 내용은 pgaudit.log 파라미터에 허용되는 설정 목록 단원을 참조하십시오.
pgaudit.log_catalog	활성화(1로 설정)하면 문의 모든 관계가 pg_catalog에 있는 경우 감사 추적에 문을 추가합니다.
pgaudit.log_level	로그 항목에 사용할 로그 수준을 지정합니다. 허용되는 값은 ebug5, debug4, debug3, debug2, debug1, 정보, 알림, 경고, 로그입니다.
pgaudit.log_parameter	활성화(1로 설정)하면 문과 함께 전달된 파라미터가 감사 로그에 캡처됩니다.
pgaudit.log_relation	활성화(1로 설정)하면 세션의 감사 로그에서 SELECT 또는 DML 문에서 참조되는 각 관계(TABLE, VIEW 등) 에 대해 별도의 로그 항목을 생성합니다.
pgaudit.log_statement_once	로깅에 문/하위 문 조합에 대한 첫 번째 로그 항목이 있는 문 텍스트 및 파라미터를 포함할지 아니면 모든 항목이 있는 문 텍스트 및 파라미터를 포함할지를 지정합니다.
pgaudit.role	객체 감사 로깅에 사용할 마스터 역할을 지정합니다. 유일하게 허용되는 항목은 rds_pgaudit 입니다.

pgaudit.log 파라미터에 허용되는 설정 목록

값	설명
없음	이 값이 기본값입니다. 데이터베이스 변경 사항이 로깅되지 않습니다.

값	설명
모두	모든 항목(읽기, 쓰기, 함수, 역할, ddl, 기타)을 로깅합니다.
ddl	ROLE 클래스에 포함되지 않은 모든 데이터 정의 언어(DDL) 문을 로깅합니다.
함수	함수 호출 및 DO 블록을 로깅합니다.
기타	기타 명령을 로깅합니다(예: DISCARD, FETCH, CHECKPOINT , VACUUM, SET).
읽기	원본이 관계(예: 테이블) 또는 쿼리인 경우 SELECT 및 COPY를 로깅합니다.
역할	역할 및 권한과 관련된 문을 로깅합니다(예: GRANT, REVOKE, CREATE ROLE, ALTER ROLE, DROP ROLE).
write	대상이 관계(테이블)인 경우 INSERT, UPDATE, DELETE, TRUNCATE, COPY를 로깅합니다.

세션 감사를 사용하여 여러 이벤트 유형을 기록하려면 쉼표로 구분된 목록을 사용합니다. 모든 이벤트 유형을 기록하려면 pgaudit.log를 ALL로 설정합니다. DB 인스턴스를 재부팅하여 변경 사항을 적용합니다.

객체 감사를 사용하면 특정 관계를 사용하도록 감사 로깅을 구체화할 수 있습니다. 예를 들어 하나 이상의 테이블에서 READ 작업에 대한 감사 로깅을 지정할 수 있습니다.

PostgreSQL pg_cron 확장을 사용하여 유지 관리 예약

PostgreSQL pg_cron 확장을 사용하여 PostgreSQL 데이터베이스 내에서 유지 관리 명령을 예약할 수 있습니다. 확장에 대한 자세한 내용은 pg_cron 설명서의 pg_cron이란 무엇입니까? 단원을 참조하십시오.

pg_cron 확장은 RDS for PostgreSQL 엔진 버전 12.5 이상에서 지원됩니다.

pg_cron 사용에 대한 자세한 내용은 RDS for PostgreSQL 또는 Aurora PostgreSQL 호환 에디션 데이터베이스에서 pg_cron을 사용하여 작업 예약을 참조하세요.

주제

- pg_cron 확장 설정
- 데이터베이스 사용자에게 pg_cron 사용 권한 부여
- pg_cron 작업 예약
- pg_cron 확장에 대한 참조

pg_cron 확장 설정

다음과 같이 pg_cron 확장을 설정합니다.

1. shared_preload_libraries 파라미터 값에 pg_cron 을 추가하여 PostgreSQL DB 인스턴스와 연결된 사용자 지정 파라미터 그룹을 수정합니다.
   - RDS for PostgreSQL DB 인스턴스가 rds.allowed_extensions 파라미터를 사용하여 설치할 수 있는 확장을 명시적으로 나열하는 경우 목록에 pg_cron 확장을 추가해야 합니다. RDS for PostgreSQL의 특정 버전에서만 rds.allowed_extensions 파라미터를 지원합니다. 기본적으로 사용 가능한 모든 확장이 허용됩니다. 자세한 내용은 PostgreSQL 확장의 설치 제한 단원을 참조하십시오.

   파라미터 그룹에 대한 변경 사항을 적용하려면 PostgreSQL DB 인스턴스를 다시 시작합니다. 파라미터 그룹 사용에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정. 단원을 참조하십시오.
2. PostgreSQL DB 인스턴스가 다시 시작된 후 rds_superuser 권한이 있는 계정을 사용하여 다음 명령을 수행합니다. 예를 들어 RDS for PostgreSQL DB 인스턴스를 생성할 때 기본 설정을 사용한 경우 사용자 postgres로 연결하고 확장을 생성합니다.

```
CREATE EXTENSION pg_cron;
```

pg_cron 스케줄러는 postgres라는 기본 PostgreSQL 데이터베이스에 설정됩니다. pg_cron 객체가 이 postgres 데이터베이스에 생성되고 모든 예약 작업이 이 데이터베이스에서 실행됩니다.

3. 기본 설정을 사용하거나 PostgreSQL DB 인스턴스 내의 다른 데이터베이스에서 실행되도록 작업을 예약할 수 있습니다. PostgreSQL DB 인스턴스 내의 다른 데이터베이스에 대한 작업을 예약하려면 기본 데이터베이스 이외의 데이터베이스에 대한 cron 작업 예약의 예제를 참조하세요.

데이터베이스 사용자에게 pg_cron 사용 권한 부여

pg_cron 확장을 설치하려면 rds_superuser 권한이 필요합니다. 그러나 pg_cron 사용 권한은 (rds_superuser 그룹/역할의 구성원에 의해) 다른 데이터베이스 사용자에게 부여되어 자신의 작업을 예약할 수 있습니다. 프로덕션 환경에서 작업을 개선하는 경우 필요한 경우에만 cron 스키마에 권한을 부여하는 것이 좋습니다.

cron 스키마에서 데이터베이스 사용자 권한을 부여하려면 다음 명령을 실행합니다.

```
postgres=> GRANT USAGE ON SCHEMA cron TO db-user;
```

이렇게 하면 액세스 권한이 있는 개체에 대해 cron 작업을 예약하기 위해 cron 스키마에 액세스할 수 있는 권한이 *db-user*에게 부여됩니다. 데이터베이스 사용자에게 권한이 없으면 다음과 같이 postgresql.log 파일에 오류 메시지를 게시한 후 작업이 실패합니다.

```
2020-12-08 16:41:00 UTC::@:[30647]:ERROR: permission denied for table table-name
2020-12-08 16:41:00 UTC::@:[27071]:LOG: background worker "pg_cron" (PID 30647) exited
 with exit code 1
```

즉, cron 스키마에 대한 사용 권한이 부여된 데이터베이스 사용자는 예약하려는 개체(테이블, 스키마 등)에 대한 사용 권한도 있어야 합니다.

cron 작업 및 성공 또는 실패에 대한 세부 정보도 cron.job_run_details 테이블에 캡처됩니다. 자세한 내용은 작업 예약 및 상태 캡처를 위한 테이블 섹션을 참조하세요.

pg_cron 작업 예약

다음 단원에서는 pg_cron 작업을 사용하여 다양한 관리 태스크를 예약하는 방법을 보여 줍니다.

> **Note**
>
> pg_cron 작업을 생성할 때 max_worker_processes 설정이 cron.max_running_jobs 개수보다 큰지 확인하십시오. 백그라운드 작업자 프로세스가 부족하면 pg_cron 작업은 실패

합니다. 기본 pg_cron 작업 수는 5입니다. 자세한 내용은 pg_cron 확장을 관리하기 위한 파라미터 단원을 참조하십시오.

주제

- 테이블 베큠
- pg_cron 기록 테이블 지우기
- postgresql.log 파일에만 오류 로깅
- 기본 데이터베이스 이외의 데이터베이스에 대한 cron 작업 예약

테이블 베큠

Autovacuum은 대부분의 경우 정리 유지 관리를 처리합니다. 하지만 선택한 시간에 특정 테이블을 베큠하도록 예약해야 하는 경우도 있습니다.

또한 Amazon RDS for PostgreSQL에서 PostgreSQL 자동 정리 사용 섹션도 참조하세요.

다음은 매일 22:00(GMT)에 특정 테이블에 cron.schedule를 사용하는 작업을 설정하는 VACUUM FREEZE 함수의 사용 예입니다.

```
SELECT cron.schedule('manual vacuum', '0 22 * * *', 'VACUUM FREEZE pgbench_accounts');
 schedule
----------
1
(1 row)
```

앞의 예제를 실행한 후, 다음과 같이 cron.job_run_details 테이블에서 기록을 확인할 수 있습니다.

```
postgres=> SELECT * FROM cron.job_run_details;
jobid  | runid | job_pid | database | username | command                          |
 status    | return_message | start_time                    | end_time
-------+-------+---------+----------+----------+-------------------------------
+-----------+----------------+-------------------------------
+-------------------------------
 1     | 1     | 3395    | postgres | adminuser| vacuum freeze pgbench_accounts
 | succeeded | VACUUM         | 2020-12-04 21:10:00.050386+00 | 2020-12-04
 21:10:00.072028+00
```

```
(1 row)
```

다음은 실패한 작업을 확인하기 위한 cron.job_run_details 테이블의 쿼리입니다.

```
postgres=> SELECT * FROM cron.job_run_details WHERE status = 'failed';
jobid | runid | job_pid | database | username | command                          | status
 | return_message                                  | start_time                    |
 end_time
------+-------+---------+----------+----------+----------------------------------+--------
+-------------------------------------------------+-------------------------------
+-------------------------------
 5    | 4     | 30339   | postgres | adminuser| vacuum freeze pgbench_account | failed
 | ERROR: relation "pgbench_account" does not exist | 2020-12-04 21:48:00.015145+00 |
 2020-12-04 21:48:00.029567+00
(1 row)
```

자세한 내용은 작업 예약 및 상태 캡처를 위한 테이블 섹션을 참조하세요.

pg_cron 기록 테이블 지우기

cron.job_run_details 테이블에는 시간이 지남에 따라 매우 커질 수 있는 cron 작업 기록이 포함되어 있습니다. 이 테이블을 지우는 작업을 예약하는 것이 좋습니다. 예를 들어, 1 주일 분량의 항목을 보관하면 문제 해결을 위해 충분할 수 있습니다.

다음 예에서는 cron.schedule 함수를 사용하여, 매일 자정에 실행되어 cron.job_run_details 테이블을 지우는 작업을 예약합니다. 이 작업은 지난 7 일 동안의 항목만 유지합니다. rds_superuser 계정을 사용하여 다음과 같은 작업을 예약합니다.

```
SELECT cron.schedule('0 0 * * *', $$DELETE
    FROM cron.job_run_details
    WHERE end_time < now() - interval '7 days'$$);
```

자세한 내용은 작업 예약 및 상태 캡처를 위한 테이블 섹션을 참조하세요.

postgresql.log 파일에만 오류 로깅

cron.job_run_details 테이블에 대한 쓰기 작업을 완전히 차단하려면 PostgreSQL DB 인스턴스와 연결된 파라미터 그룹을 수정하고 cron.log_run 파라미터를 '꺼짐'으로 설정합니다. pg_cron 확장이 더 이상 테이블에 기록하지 않고 postgresql.log 파일에만 오류를 캡처합니다. 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 섹션을 참조하세요.

다음 명령을 사용하여 cron.log_run 파라미터 값을 확인합니다.

```
postgres=> SHOW cron.log_run;
```

자세한 내용은 pg_cron 확장을 관리하기 위한 파라미터 섹션을 참조하세요.

기본 데이터베이스 이외의 데이터베이스에 대한 cron 작업 예약

pg_cron의 메타데이터는 모두 postgres라는 PostgreSQL 기본 데이터베이스에 보관됩니다. 백그라운드 작업자는 유지 관리 cron 작업을 실행하는 데 사용되므로 PostgreSQL DB 인스턴스 내의 모든 데이터베이스에서 작업을 예약할 수 있습니다.

> **Note**
>
> rds_superuser 역할 또는 rds_superuser 권한이 있는 사용자만 데이터베이스의 모든 cron 작업을 나열할 수 있습니다. 다른 사용자는 cron.job 테이블에서 자신의 작업만 볼 수 있습니다.

1. cron 데이터베이스에서 정상적으로 cron.schedule을(를) 사용하는 것처럼 작업을 예약합니다.

   ```
   postgres=> SELECT cron.schedule('database1 manual vacuum', '29 03 * * *', 'vacuum
    freeze test_table');
   ```

2. rds_superuser 역할을 가진 사용자는 방금 생성한 작업에 대한 데이터베이스 열을 업데이트하여 PostgreSQL DB 인스턴스 내의 다른 데이터베이스에서 실행되도록 합니다.

   ```
   postgres=> UPDATE cron.job SET database = 'database1' WHERE jobid = 106;
   ```

3. cron.job 테이블을 쿼리하여 확인합니다.

   ```
   postgres=> SELECT * FROM cron.job;
   jobid | schedule    | command                          | nodename  | nodeport |
    database | username  | active | jobname
   ------+-------------+----------------------------------+-----------+----------
   +----------+----------+--------+-----------------------
   106   | 29 03 * * * | vacuum freeze test_table         | localhost | 8192     |
    database1| adminuser | t      | database1 manual vacuum
     1   | 59 23 * * * | vacuum freeze pgbench_accounts | localhost | 8192     |
    postgres | adminuser | t      | manual vacuum
   (2 rows)
   ```

> **Note**
>
> 경우에 따라, 다른 데이터베이스에서 실행하고자 하는 cron 작업을 추가할 수 있습니다. 이 경우, 올바른 데이터베이스 열을 업데이트하기 전에 작업이 기본 데이터베이스(postgres)에서 실행을 시도할 수 있습니다. 사용자 이름에 권한이 있으면 작업이 기본 데이터베이스에서 성공적으로 실행됩니다.

pg_cron 확장에 대한 참조

pg_cron 확장과 함께 다음 파라미터, 함수 및 테이블을 사용할 수 있습니다. 자세한 내용은 pg_cron 설명서의 pg_cron이란 무엇입니까? 단원을 참조하세요.

주제

- pg_cron 확장을 관리하기 위한 파라미터
- 함수 참조: cron.schedule
- 함수 참조: cron.unschedule
- 작업 예약 및 상태 캡처를 위한 테이블

pg_cron 확장을 관리하기 위한 파라미터

다음은 pg_cron 확장 동작을 제어하는 파라미터의 목록입니다.

파라미터	설명
cron.database_name	pg_cron 메타데이터가 보관되는 데이터베이스입니다.
cron.host	PostgreSQL에 연결할 호스트 이름입니다. 이 값은 수정할 수 없습니다.
cron.log_run	job_run_details 테이블에서 실행되는 모든 작업을 로깅합니다. 유효한 값은 on 또는 off입니다. 자세한 내용은 작업 예약 및 상태 캡처를 위한 테이블 섹션을 참조하세요.

파라미터	설명
cron.log_statement	모든 cron 문을 실행하기 전에 기록합니다. 유효한 값은 on 또는 off입니다.
cron.max_running_jobs	동시에 실행할 수 있는 최대 작업 수입니다.
cron.use_background_workers	클라이언트 세션 대신 백그라운드 작업자를 사용합니다. 이 값은 수정할 수 없습니다.

다음 SQL 명령을 사용하여 이러한 파라미터와 해당 값을 표시합니다.

```
postgres=> SELECT name, setting, short_desc FROM pg_settings WHERE name LIKE 'cron.%'
 ORDER BY name;
```

함수 참조: cron.schedule

이 함수는 cron 작업을 예약합니다. 작업은 처음에 기본 postgres 데이터베이스에서 예약됩니다. 이 함수는 작업 식별자를 나타내는 bigint 값을 반환합니다. PostgreSQL DB 인스턴스 내의 다른 데이터베이스에서 작업이 실행되도록 예약하려면 기본 데이터베이스 이외의 데이터베이스에 대한 cron 작업 예약의 예제를 참조하세요.

이 함수에는 두 가지 구문 형식이 있습니다.

구문

```
cron.schedule (job_name,
    schedule,
    command
);

cron.schedule (schedule,
    command
);
```

파라미터

파라미터	설명
job_name	cron 작업의 이름입니다.
schedule	cron 작업의 일정을 나타내는 텍스트입니다. 형식은 표준 cron 형식입니다.
command	실행할 명령의 텍스트입니다.

예시

```
postgres=> SELECT cron.schedule ('test','0 10 * * *', 'VACUUM pgbench_history');
 schedule
----------
      145
(1 row)

postgres=> SELECT cron.schedule ('0 15 * * *', 'VACUUM pgbench_accounts');
 schedule
----------
      146
(1 row)
```

함수 참조: cron.unschedule

이 함수는 cron 작업을 삭제합니다. job_name 또는 job_id를 지정할 수 있습니다. 정책은 사용자가 작업 일정을 제거할 수 있는 소유자인지를 확인합니다. 이 함수는 성공 또는 실패를 나타내는 부울 값을 반환합니다.

함수의 구문 형식은 다음과 같습니다.

구문

```
cron.unschedule (job_id);

cron.unschedule (job_name);
```

파라미터

파라미터	설명
job_id	cron 작업이 예약된 경우 cron.schedule 함수에서 반환된 작업 식별자입니다.
job_name	cron.schedule 함수로 예약된 cron 작업의 이름입니다.

예시

```
postgres=> SELECT cron.unschedule(108);
 unschedule
------------
 t
(1 row)

postgres=> SELECT cron.unschedule('test');
 unschedule
------------
 t
(1 row)
```

작업 예약 및 상태 캡처를 위한 테이블

다음 표는 cron 작업을 예약하고 작업 완료 방법을 기록하는 데 사용됩니다.

표	설명
cron.job	예약된 각 작업에 대한 메타데이터를 포함합니다. 이 테이블과의 대부분의 상호 작용은 cron.schedule 및 cron.unschedule 함수를 사용하여 수행해야 합니다. **Important** 이 테이블에 직접 업데이트 또는 삽입 권한을 부여하지 않는 것이 좋습니다. 이렇게 하면 사용자가

표	설명
	username(으)로 실행되도록 rds-superuser 열을 업데이트할 수 있습니다.
cron.job_run_details	이전에 예약된 작업 실행에 대한 기록 정보를 포함합니다. 이는 실행한 작업에서 상태, 반환 메시지, 시작 및 종료 시간을 조사하는 데 유용합니다. **Note** 이 테이블이 무한정 증가하지 않게 하려면 정기적으로 삭제하세요. 관련 예제는 pg_cron 기록 테이블 지우기 섹션을 참조하세요

pglogical을 사용하여 인스턴스 간 데이터 동기화

현재 사용 가능한 모든 RDS for PostgreSQL 버전은 `pglogical` 확장을 지원합니다. pglogical 확장은 PostgreSQL 버전 10에서 도입된 기능적으로 유사한 논리적 복제 기능보다 먼저 출시되었습니다. 자세한 내용은 Amazon RDS for PostgreSQL에 대한 논리적 복제 수행 섹션을 참조하세요.

`pglogical` 확장은 둘 이상의 간의 논리적 복제를 지원합니다. RDS for PostgreSQL DB 인스턴스. 서로 다른 PostgreSQL 버전 간의 복제와 PostgreSQL DB 인스턴스용 RDS 및 Aurora PostgreSQL DB 클러스터에서 실행되는 데이터베이스 간의 복제도 지원합니다. `pglogical` 확장은 게시-구독 모델을 사용하여 게시자의 테이블 및 기타 객체(예: 시퀀스)의 변경 사항을 구독자에 복제합니다. 이 확장은 복제 슬롯을 사용하여, 다음과 같이 게시자 노드의 변경 사항이 구독자 노드로 동기화되게 합니다.

- 게시자 노드는 다른 노드에 복제할 데이터의 소스인 RDS for PostgreSQL DB 인스턴스입니다. 게시자 노드는 게시 세트에서 복제될 테이블을 정의합니다.
- 구독자 노드는 게시자로부터 WAL 업데이트를 받는 RDS for PostgreSQL DB 인스턴스입니다. 구독자는 구독을 생성하여 게시자에 연결하고 디코딩된 WAL 데이터를 얻습니다. 구독자가 구독을 생성하면 게시자 노드에서 복제 슬롯이 생성됩니다.

아래에서 `pglogical` 확장 설정 관련 정보를 확인할 수 있습니다.

주제

- plogical 확장에 대한 요구 사항 및 제한
- pglogical 확장 설정
- RDS for PostgreSQL DB 인스턴스용 논리적 복제 설정
- 메이저 업그레이드 후 논리적 복제 재설정
- RDS for PostgreSQL용 논리적 복제 슬롯 관리
- plogical 확장용 파라미터 참조

plogical 확장에 대한 요구 사항 및 제한

현재 사용 가능한 모든 RDS for PostgreSQL 릴리스는 `pglogical` 확장을 지원합니다.

게시자 노드와 구독자 노드 모두가 논리적 복제를 할 수 있도록 설정되어야 합니다.

게시자에서 구독자로 복제할 테이블은 이름과 스키마가 동일해야 합니다. 또한 이러한 테이블은 동일한 열을 포함해야 하며, 각 열은 동일한 데이터 유형을 사용해야 합니다. 게시자와 구독자 테이블 모두 프라이머리 키가 동일해야 합니다. PRIMARY KEY만 고유 제약 조건으로 사용하는 것이 좋습니다.

구독자 노드의 테이블에는 CHECK 제약 조건 및 NOT NULL 제약 조건에 대해 게시자 노드의 테이블에 있는 것보다 더 많은 허용 제약 조건이 존재할 수 있습니다.

pglogical 확장은 PostgreSQL(버전 10 이상)에 내장된 논리적 복제 기능에서는 지원하지 않는 양방향 복제 같은 기능을 제공합니다. 자세한 내용은 pglogical을 사용한 PostgreSQL 양방향 복제를 참조하십시오.

pglogical 확장 설정

RDS for PostgreSQL DB 인스턴스 에 pglogical 확장을 설정하려면 PostgreSQL DB 인스턴스용 사용자 지정 DB 파라미터 그룹 의 공유 라이브러리에 pglogical을 추가해야 합니다. 논리적 디코딩을 켜려면 rds.logical_replication 파라미터의 값을 1로 설정해야 합니다. 마지막으로, 데이터베이스에서 확장을 만듭니다. 이러한 작업에는 AWS Management Console 또는 AWS CLI를 사용할 수 있습니다.

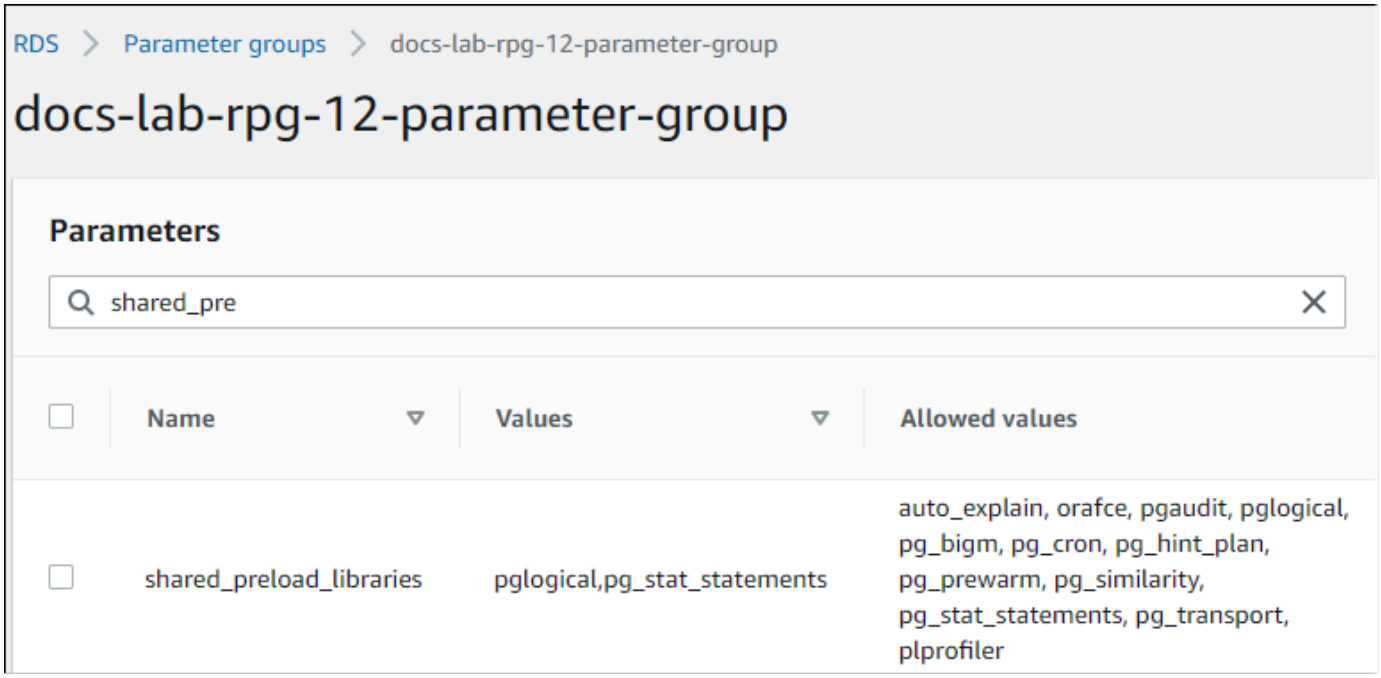
이러한 작업을 수행하려면 rds_superuser 역할의 권한이 있어야 합니다.

다음 단계에서는 사용자의 RDS for PostgreSQL DB 인스턴스가 사용자 지정 DB에 연결되어 있다고 가정합니다. 사용자 지정 DB 클러스터 파라미터 그룹을 만드는 방법에 관한 자세한 내용은 Amazon RDS의 파라미터 그룹 섹션을 참조하세요.

콘솔

pglogical 확장 설정 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 RDS for PostgreSQL DB 인스턴스를 선택합니다.
3. 의 구성 탭을 엽니다. RDS for PostgreSQL DB 인스턴스 인스턴스 세부 정보 중에서 파라미터 그룹 링크를 찾습니다.
4. 링크를 선택하여 와 연결된 사용자 지정 파라미터를 엽니다. RDS for PostgreSQL DB 인스턴스
5. 파라미터 검색 필드에 shared_pre를 입력하여 shared_preload_libraries 파라미터를 찾습니다.
6. 파라미터 편집을 선택하여 속성 값에 액세스합니다.
7. 값 필드의 목록에 pglogical를 추가합니다. 쉼표를 사용하여 값 목록에서 항목을 구분합니다.



8. `rds.logical_replication` 파라미터를 찾아 1로 설정하여 논리적 복제를 켭니다.
9. RDS for PostgreSQL DB 인스턴스를 재부팅하여 파라미터 변경 사항이 적용되게 합니다.
10. 인스턴스를 사용할 수 있다면 `psql`(또는 pgAdmin)을 사용하여 RDS for PostgreSQL DB 인스턴스에 연결할 수 있습니다.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

11. plogical이 초기화되었는지 확인하려면 다음 명령을 실행합니다.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pglogical
(1 row)
```

12. 논리적 디코딩을 활성화하는 설정을 다음과 같이 확인합니다.

```
SHOW wal_level;
wal_level
-----------
 logical
(1 row)
```

13. 다음과 같이 확장을 생성합니다.

```
CREATE EXTENSION pglogical;
EXTENSION CREATED
```

14. Save changes(변경 사항 저장)를 선택합니다.
15. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
16. 데이터베이스 목록에서 RDS for PostgreSQL DB 인스턴스를 선택한 다음, 작업 메뉴에서 재부팅을 선택합니다.

AWS CLI

plogical 확장 설정 방법

AWS CLI를 사용하여 pglogical을 설정하려면 다음 절차와 같이 modify-db-parameter-group 작업을 호출하여 사용자 지정 파라미터 그룹의 특정 파라미터를 수정합니다.

1. 다음 AWS CLI 명령을 사용하여 `shared_preload_libraries` 파라미터에 `pglogical`를 추가합니다.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pglogical,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. 다음 AWS CLI 명령을 사용하여 `rds.logical_replication`을 1로 설정하여 용 논리적 디코딩 기능을 켭니다. RDS for PostgreSQL DB 인스턴스

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=rds.logical_replication,ParameterValue=1,ApplyMethod=pending-
reboot" \
   --region aws-region
```

3. 다음 AWS CLI 명령으로 RDS for PostgreSQL DB 인스턴스를 재부팅하여 pglogical 라이브러리가 초기화되도록 합니다.

```
aws rds reboot-db-instance \
    --db-instance-identifier your-instance \
    --region aws-region
```

4. 인스턴스를 사용할 수 있다면 psql을 사용하여 RDS for PostgreSQL DB 인스턴스에 연결합니다.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

5. 다음과 같이 확장을 생성합니다.

```
CREATE EXTENSION pglogical;
EXTENSION CREATED
```

6. 다음 AWS CLI 명령으로 RDS for PostgreSQL DB 인스턴스를 재부팅합니다.

```
aws rds reboot-db-instance \
    --db-instance-identifier your-instance \
    --region aws-region
```

RDS for PostgreSQL DB 인스턴스용 논리적 복제 설정

다음 절차는 RDS for PostgreSQL DB 인스턴스 간에 논리적 복제를 시작하는 방법을 보여줍니다. 다음 단계에서는 원본(게시자)과 대상(구독자) 모두에 pglogical 확장 설정에서 설명하는 방법에 따라 pglogical 확장이 설정되어 있다고 가정합니다.

> **Note**
>
> 구독자 노드의 node_name은 rds로 시작할 수 없습니다.

게시자 노드를 생성하고 복제할 테이블을 정의하는 방법

이 단계에서는 RDS for PostgreSQL DB 인스턴스에 데이터베이스 하나가 있고 이러한 데이터베이스에는 다른 노드에 복제할 하나 이상의 테이블이 있다고 가정합니다. 구독자의 테이블 구조를 게시자에서 다시 만들어야 하므로, 필요한 경우 먼저 테이블 구조를 가져와야 합니다. 이렇게 하려면 psql 메타 명령을 사용한 다음 구독자 인스턴스에서 동일한 테이블을 생성해야 합니다. 다음 절차에서는 시연을 위해 게시자(원본)에서 예제 테이블을 만듭니다.

1. psql을 사용하여 구독자용 소스로 사용할 테이블이 있는 인스턴스에 연결합니다.

```
psql --host=source-instance.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

복제하려는 기존 테이블이 없는 경우 다음과 같이 샘플 테이블을 생성할 수 있습니다.

a. 다음 SQL 문을 사용하여 예제 테이블을 생성합니다.

```
CREATE TABLE docs_lab_table (a int PRIMARY KEY);
```

b. 다음 SQL 문을 사용하여 테이블에 생성된 데이터를 입력합니다.

```
INSERT INTO docs_lab_table VALUES (generate_series(1,5000));
INSERT 0 5000
```

c. 다음 SQL 문을 사용하여 테이블에 데이터가 있는지 확인합니다.

```
SELECT count(*) FROM docs_lab_table;
```

2. 다음과 같이 이 RDS for PostgreSQL DB 인스턴스를 게시자 노드로 식별합니다.

```
SELECT pglogical.create_node(
    node_name := 'docs_lab_provider',
    dsn := 'host=source-instance.aws-region.rds.amazonaws.com port=5432
 dbname=labdb');
 create_node
-------------
   3410995529
(1 row)
```

3. 복제할 테이블을 기본 복제 세트에 추가합니다. 복제 세트에 대한 자세한 내용은 pglogical 설명서의 복제 세트를 참조하십시오.

```
SELECT pglogical.replication_set_add_table('default', 'docs_lab_table', 'true',
 NULL, NULL);
 replication_set_add_table
  --------------------------
  t
  (1 row)
```

게시자 노드 설정이 완료되었습니다. 이제 게시자로부터 업데이트를 수신하도록 구독자 노드를 설정할 수 있습니다.

구독자 노드를 설정하고, 업데이트를 수신할 구독을 만드는 방법

이 단계에서는 RDS for PostgreSQL DB 인스턴스가 pglogical 확장을 이용해 설정되었다고 가정합니다. 자세한 내용은 pglogical 확장 설정 섹션을 참조하세요.

1. psql을 사용하여 게시자로부터 업데이트를 수신할 인스턴스에 연결합니다.

```
psql --host=target-instance.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

2. 구독자 RDS for PostgreSQL DB 인스턴스에서 게시자에 존재하는 것과 동일한 테이블을 만듭니다. 이 예제에서 테이블은 docs_lab_table입니다. 다음과 같이 테이블을 만들 수 있습니다.

```
CREATE TABLE docs_lab_table (a int PRIMARY KEY);
```

3. 이 테이블이 비어 있는지 확인합니다.

```
SELECT count(*) FROM docs_lab_table;
 count
-------
  0
(1 row)
```

4. 다음과 같이 이 RDS for PostgreSQL DB 인스턴스를 구독자 노드로 식별합니다.

```
SELECT pglogical.create_node(
    node_name := 'docs_lab_target',
    dsn := 'host=target-instance.aws-region.rds.amazonaws.com port=5432
 sslmode=require dbname=labdb user=postgres password=********');
 create_node
-------------
   2182738256
(1 row)
```

5. 구독을 생성합니다.

```
SELECT pglogical.create_subscription(
   subscription_name := 'docs_lab_subscription',
```

```
   provider_dsn := 'host=source-instance.aws-region.rds.amazonaws.com port=5432
 sslmode=require dbname=labdb user=postgres password=*******',
   replication_sets := ARRAY['default'],
   synchronize_data := true,
   forward_origins := '{}' );
 create_subscription
---------------------
1038357190
(1 row)
```

이 단계를 완료하면 게시자 테이블의 데이터가 구독자 테이블에서 생성됩니다. 다음 SQL 쿼리를 사용하면 이 문제가 발생했는지 확인할 수 있습니다.

```
SELECT count(*) FROM docs_lab_table;
 count
-------
  5000
(1 row)
```

이 시점 이후로는 게시자의 테이블에 적용한 변경 사항이 구독자의 테이블에 복제됩니다.

메이저 업그레이드 후 논리적 복제 재설정

논리적 복제용 게시자 노드로 설정된 RDS for PostgreSQL DB 인스턴스의 메이저 버전 업그레이드를 수행하려면, 먼저 활성화되지 않는 슬롯을 포함한 모든 복제 슬롯을 삭제해야 합니다. 게시자 노드에서 데이터베이스 트랜잭션을 일시적으로 전환하고, 복제 슬롯을 삭제하고, RDS for PostgreSQL DB인스턴스를 업그레이드한 다음 복제를 다시 설정하고 재시작하는 것이 좋습니다.

복제 슬롯은 게시자 노드에서만 호스팅됩니다. 논리적 복제 시나리오에서 RDS for PostgreSQL 구독자 노드는 삭제할 슬롯이 없지만, 게시자에 대한 구독이 있는 구독자 노드로 지정된 동안에는 메이저 버전으로 업그레이드할 수 없습니다. RDS for PostgreSQL를 구독자 노드로 업그레이드하기 전에 구독과 노드를 삭제합니다. 자세한 내용은 RDS for PostgreSQL용 논리적 복제 슬롯 관리 섹션을 참조하세요.

논리적 복제가 중단되었는지 확인

다음과 같이 게시자 노드 또는 구독자 노드를 쿼리하면 복제 프로세스 중단 여부를 확인할 수 있습니다.

게시자 노드를 확인하는 방법

- psql을 사용하여 게시자 노드에 연결한 다음 pg_replication_slots 함수를 쿼리합니다. 활성 열의 값을 기록해 둡니다. 일반적으로 이 값은 t(true)를 반환하며, 복제가 활성 상태라는 뜻입니다. 쿼리가 f(false)를 반환한다면 구독자로의 복제가 중단되었다는 뜻입니다.

```
SELECT slot_name,plugin,slot_type,active FROM pg_replication_slots;
                    slot_name              |      plugin      | slot_type | active
-------------------------------------------+------------------+-----------+--------
 pgl_labdb_docs_labcb4fa94_docs_lab3de412c | pglogical_output | logical   | f
(1 row)
```

구독자 노드를 확인하는 방법

구독자 노드에서는 세 가지 방법으로 복제 상태를 확인할 수 있습니다.

- 구독자 노드의 PostgreSQL 로그를 확인하여 실패 메시지를 찾습니다. 로그는 다음과 같이 종료 코드 1을 포함하는 메시지를 이용해 실패를 식별합니다.

```
2022-07-06 16:17:03 UTC::@:[7361]:LOG: background worker "pglogical apply
 16404:2880255011" (PID 14610) exited with exit code 1
2022-07-06 16:19:44 UTC::@:[7361]:LOG: background worker "pglogical apply
 16404:2880255011" (PID 21783) exited with exit code 1
```

- pg_replication_origin 함수를 쿼리합니다. 다음과 같이 psql을 사용하여 구독자 노드의 데이터베이스에 연결하고 pg_replication_origin 함수를 쿼리합니다.

```
SELECT * FROM pg_replication_origin;
 roident | roname
---------+--------
(0 rows)
```

  결과 집합이 비어 있다면 복제가 중단되었다는 뜻입니다. 일반적인 출력은 다음과 같습니다.

```
  roident |                       roname
 ---------+-----------------------------------------------------
        1 | pgl_labdb_docs_labcb4fa94_docs_lab3de412c
 (1 row)
```

- 다음 예제와 같이 pglogical.show_subscription_status 함수를 쿼리합니다.

```
SELECT subscription_name,status,slot_name FROM pglogical.show_subscription_status();
     subscription_name | status |              slot_name
-------+----------------+--------+-----------------------------------
 docs_lab_subscription | down   | pgl_labdb_docs_labcb4fa94_docs_lab3de412c
(1 row)
```

이 출력은 복제가 중단되었음을 보여줍니다. 상태는 down입니다. 일반적으로 출력에서는 상태가 replicating으로 표시됩니다.

논리적 복제 프로세스가 중단된 경우 다음 단계에 따라 복제를 재설정할 수 있습니다.

게시자와 구독자 노드 간의 논리적 복제를 재설정하는 방법

복제를 다시 설정하려면 먼저 게시자 노드에서 구독자의 연결을 끊은 다음, 다음 단계의 설명에 따라 구독을 다시 설정합니다.

1. 다음과 같이 psql을 사용하여 구독자 노드에 연결합니다.

```
psql --host=222222222222.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

2. pglogical.alter_subscription_disable 함수를 사용하여 구독을 비활성화합니다.

```
SELECT pglogical.alter_subscription_disable('docs_lab_subscription',true);
 alter_subscription_disable
----------------------------
 t
(1 row)
```

3. 다음과 같이 pg_replication_origin을 쿼리하여 게시자 노드의 식별자를 가져옵니다.

```
SELECT * FROM pg_replication_origin;
 roident |               roname
---------+-----------------------------------
       1 | pgl_labdb_docs_labcb4fa94_docs_lab3de412c
(1 row)
```

4. 이전 단계의 응답을 pg_replication_origin_create 명령과 함께 사용하여, 구독을 다시 설정할 때 사용할 수 있는 식별자를 할당합니다.

```
SELECT pg_replication_origin_create('pgl_labdb_docs_labcb4fa94_docs_lab3de412c');
  pg_replication_origin_create
------------------------------
                             1
(1 row)
```

5. 다음 예제와 같이 구독의 이름을 true 상태로 전달하여 구독을 활성화합니다.

```
SELECT pglogical.alter_subscription_enable('docs_lab_subscription',true);
  alter_subscription_enable
---------------------------
 t
(1 row)
```

노드의 상태를 확인합니다. 노드의 상태는 이 예제에서처럼 replicating이어야 합니다.

```
SELECT subscription_name,status,slot_name
  FROM pglogical.show_subscription_status();
            subscription_name |   status    |              slot_name
------------------------------+-------------+-------------------------------------
 docs_lab_subscription         | replicating |
 pgl_labdb_docs_lab98f517b_docs_lab3de412c
(1 row)
```

게시자 노드에서 구독자 복제 슬롯의 상태를 확인합니다. 슬롯의 active 열은 t(true)를 반환해야 하며, 복제가 다시 설정되었다는 뜻입니다.

```
SELECT slot_name,plugin,slot_type,active
  FROM pg_replication_slots;
                   slot_name              |      plugin      | slot_type | active
------------------------------------------+------------------+-----------+--------
 pgl_labdb_docs_lab98f517b_docs_lab3de412c | pglogical_output | logical   | t
(1 row)
```

RDS for PostgreSQL용 논리적 복제 슬롯 관리

논리적 복제 시나리오에서 게시자 노드 역할을 하는 RDS for PostgreSQL DB 인스턴스의 메이저 버전 업그레이드를 수행하려면, 먼저 인스턴스에서 복제 슬롯을 삭제해야 합니다. 메이저 버전 업그레이드 사전 점검 프로세스에서는 업그레이드를 진행하려면 슬롯을 삭제해야 한다는 메시지가 표시됩니다.

RDS for PostgreSQL DB 인스턴스에서 슬롯을 삭제하려면 먼저 구독을 삭제한 다음 슬롯을 삭제합니다.

pglogical 확장을 사용하여 만든 복제 슬롯을 식별하려면 각 데이터베이스에 로그인하여 노드 이름을 확인하십시오. 구독자 노드를 쿼리하면 이 예제에서처럼 출력에 게시자와 구독자 노드가 모두 표시됩니다.

```
SELECT * FROM pglogical.node;
node_id   |     node_name
------------+------------------
 2182738256 | docs_lab_target
 3410995529 | docs_lab_provider
(2 rows)
```

다음 쿼리를 사용하면 구독 세부 정보를 확인할 수 있습니다.

```
SELECT sub_name,sub_slot_name,sub_target
  FROM pglogical.subscription;
 sub_name |         sub_slot_name          | sub_target
----------+--------------------------------+------------
  docs_lab_subscription     | pgl_labdb_docs_labcb4fa94_docs_lab3de412c | 2182738256
(1 row)
```

이제 다음과 같이 구독을 삭제할 수 있습니다.

```
SELECT pglogical.drop_subscription(subscription_name := 'docs_lab_subscription');
 drop_subscription
-------------------
                 1
(1 row)
```

구독을 삭제한 후에는 노드를 삭제해도 됩니다.

```
SELECT pglogical.drop_node(node_name := 'docs-lab-subscriber');
 drop_node
-----------
 t
(1 row)
```

다음과 같은 방법을 이용해 노드가 더 이상 존재하지 않는지 확인할 수 있습니다.

```
SELECT * FROM pglogical.node;
 node_id | node_name
---------+-----------
(0 rows)
```

plogical 확장용 파라미터 참조

표에서는 pglogical 확장과 관련된 파라미터를 확인할 수 있습니다. pglogical.conflict_log_level 및 pglogical.conflict_resolution 같은 파라미터는 업데이트 충돌을 처리하는 용도로 사용합니다. 게시자의 변경 사항을 구독한 테이블과을 로컬로 변경하면 충돌이 발생할 수 있습니다. 양방향 복제 또는 동일한 게시자로부터 여러 구독자를 복제하는 경우를 비롯한 다양한 시나리오에서도 충돌이 발생할 수 있습니다. 자세한 내용은 pglogical을 사용한 PostgreSQL 양방향 복제를 참조하십시오.

파라미터	설명
pglogical.batch_inserts	가능한 경우 배치를 삽입합니다. 기본적으로는 설정되지 않습니다. 켜려면 '1'로 변경하고 해제하려면 '0'으로 변경합니다.
pglogical.conflict_log_level	해결된 충돌을 로깅하는 데 사용할 로그 수준을 설정합니다. 지원되는 문자열 값은 debug5, debug4, debug3, debug2, debug1, info, notice, warning, error, log, fatal 및 panic입니다.
pglogical.conflict_resolution	해결할 수 있는 충돌인 경우 충돌을 해결하는 데 사용할 메서드를 설정합니다. 지원되는 문자열 값은 error, apply_remote, keep_local, last_update_wins 및 first_update_wins입니다.
pglogical.extra_connection_options	모든 피어 노드 연결에 추가할 연결 옵션입니다.
pglogical.synchronous_commit	pglogical 특정 동기 커밋 값입니다.
pglogical.use_spi	하위 수준 API 대신 SPI(서버 프로그래밍 인터페이스)를 사용하여 변경 사항을 적용합니다. 켜려면 '1'로 설정하고 해제하려면 '0'으로 설정합니다. SPI에 대한 자세한 내용은 PostgreSQL 설명서의 서버 프로그래밍 인터페이스를 참조하십시오.

pgactive를 사용하여 액티브-액티브 복제 지원

pgactive 확장은 액티브-액티브 복제를 사용하여 여러 RDS for PostgreSQL 데이터베이스에서 쓰기 작업을 지원하고 조정합니다. Amazon RDS for PostgreSQL은 다음 버전에서 pgactive 확장을 지원합니다.

- RDS for PostgreSQL 17.0 이상의 모든 버전
- RDS for PostgreSQL 16.1 이상의 16 버전
- RDS for PostgreSQL 15.4-R2 이상의 15 버전
- RDS for PostgreSQL 14.10 이상의 14 버전
- RDS for PostgreSQL 13.13 이상의 13 버전
- RDS for PostgreSQL 12.17 이상의 12 버전
- RDS for PostgreSQL 11.22

> **Note**
>
> 복제 구성에서 둘 이상의 데이터베이스에 쓰기 작업이 있는 경우 충돌이 발생할 수 있습니다. 자세한 내용은 액티브-액티브 복제의 충돌 처리 섹션을 참조하세요.

주제

- pgactive 확장에 대한 제한 사항
- pgactive 확장 기능 초기화
- RDS for PostgreSQL DB 인스턴스용 액티브-액티브 복제 설정
- pgactive 멤버 간의 복제 지연 측정
- pgactive 확장에 대한 파라미터 설정 구성
- 액티브-액티브 충돌 이해
- pgactive 스키마 이해
- pgactive 함수 참조
- 액티브-액티브 복제의 충돌 처리
- 액티브-액티브 복제의 시퀀스 처리

pgactive 확장에 대한 제한 사항

- 모든 테이블에는 프라이머리 키가 필요합니다. 없는 경우 업데이트 및 삭제가 허용되지 않습니다. 프라이머리 키 열의 값은 업데이트해서는 안 됩니다.
- 시퀀스는 간격이 있을 수 있으며 경우에 따라 순서를 따르지 않을 수도 있습니다. 시퀀스는 복제되지 않습니다. 자세한 내용은 액티브-액티브 복제의 시퀀스 처리 섹션을 참조하세요.
- DDL 및 대형 객체는 복제되지 않습니다.
- 보조 고유 인덱스로 인해 데이터 차이가 발생할 수 있습니다.
- 그룹 내 모든 노드에서 데이터 정렬이 동일해야 합니다.
- 노드 간 로드 밸런싱은 안티 패턴입니다.
- 대규모 트랜잭션으로 인해 복제 지연이 발생할 수 있습니다.

pgactive 확장 기능 초기화

RDS for PostgreSQL DB 인스턴스에서 `pgactive` 확장 기능을 초기화하려면 `rds.enable_pgactive` 파라미터 값을 1로 설정한 다음 데이터베이스에 확장을 생성합니다. 이렇게 하면 파라미터 `rds.logical_replication` 및 `track_commit_timestamp`가 자동으로 활성화되고 `wal_level` 값이 `logical`로 설정됩니다.

이러한 작업을 수행하려면 `rds_superuser` 역할의 권한이 있어야 합니다.

AWS Management Console 또는 AWS CLI를 사용하여 PostgreSQL DB 인스턴스에 필요한 RDS를 생성할 수 있습니다. 다음 단계에서는 RDS for PostgreSQL DB 인스턴스가 사용자 지정 DB 파라미터 그룹에 연결되어 있다고 가정합니다. 사용자 지정 DB 파라미터 그룹 생성에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하세요.

콘솔

pgactive 확장 기능을 초기화하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 RDS for PostgreSQL DB 인스턴스를 선택합니다.
3. RDS for PostgreSQL DB 인스턴스의 구성 탭을 엽니다. 인스턴스 세부 정보에서 DB 인스턴스 파라미터 그룹 링크를 찾습니다.
4. 링크를 선택하여 RDS for PostgreSQL DB 인스턴스와 연결된 사용자 지정 파라미터를 엽니다.
5. `rds.enable_pgactive` 파라미터를 찾아 1로 설정하여 `pgactive` 기능을 초기화합니다.

6. 변경 사항 저장을 선택합니다.
7. Amazon RDS 콘솔의 탐색 창에서 데이터베이스를 선택합니다.
8. RDS for PostgreSQL DB 인스턴스를 선택한 다음 작업 메뉴에서 재부팅을 선택합니다.
9. DB 인스턴스 재부팅을 확인하여 변경 사항을 적용합니다.
10. DB 인스턴스를 사용할 수 있게 되면 `psql` 또는 다른 PostgreSQL 클라이언트를 사용하여 RDS for PostgreSQL DB 인스턴스에 연결합니다.

    다음 예시에서는 RDS for PostgreSQL DB 인스턴스에 *`postgres`*라는 기본 데이터베이스가 있다고 가정합니다.

    ```
    psql --host=mydb.111122223333.aws-region.rds.amazonaws.com --port=5432 --
    username=postgres --password=PASSWORD --dbname=postgres
    ```

11. pgactive가 초기화되었는지 확인하려면 다음 명령을 실행합니다.

    ```
    postgres=>SELECT setting ~ 'pgactive'
    FROM pg_catalog.pg_settings
    WHERE name = 'shared_preload_libraries';
    ```

    pgactive가 `shared_preload_libraries`에 들어 있는 경우 앞의 명령은 다음을 반환합니다.

    ```
    ?column?
    ----------
     t
    ```

AWS CLI

pgactive 확장 기능을 초기화하려면

AWS CLI를 사용하여 `pgactive`를 초기화하려면 다음 절차와 같이 modify-db-parameter-group 작업을 호출하여 사용자 지정 파라미터 그룹의 특정 파라미터를 수정합니다.

1. 다음 AWS CLI 명령으로 `rds.enable_pgactive`를 1로 설정하여 RDS for PostgreSQL DB 인스턴스의 `pgactive` 기능을 초기화합니다.

    ```
    postgres=>aws rds modify-db-parameter-group \
       --db-parameter-group-name custom-param-group-name \
    ```

```
  --parameters
 "ParameterName=rds.enable_pgactive,ParameterValue=1,ApplyMethod=pending-reboot" \
  --region aws-region
```

2. 다음 AWS CLI 명령으로 RDS for PostgreSQL DB 인스턴스를 재부팅하여 pgactive 라이브러리가 초기화되도록 합니다.

```
aws rds reboot-db-instance \
    --db-instance-identifier your-instance \
    --region aws-region
```

3. 인스턴스를 사용할 수 있다면 psql을 사용하여 RDS for PostgreSQL DB 인스턴스에 연결합니다.

```
psql --host=mydb.111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=master user --password=PASSWORD --dbname=postgres
```

4. pgactive가 초기화되었는지 확인하려면 다음 명령을 실행합니다.

```
postgres=>SELECT setting ~ 'pgactive'
FROM pg_catalog.pg_settings
WHERE name = 'shared_preload_libraries';
```

pgactive가 shared_preload_libraries에 들어 있는 경우 앞의 명령은 다음을 반환합니다.

```
?column?
----------
 t
```

RDS for PostgreSQL DB 인스턴스용 액티브-액티브 복제 설정

다음 절차는 pgactive를 사용할 수 있을 때 RDS for PostgreSQL DB 인스턴스 간에 액티브-액티브 복제를 시작하는 방법을 보여 줍니다. 다중 리전 고가용성 예제를 실행하려면 Amazon RDS for PostgreSQL 인스턴스를 서로 다른 두 리전에 배포하고 VPC 피어링을 설정해야 합니다. 자세한 내용은 VPC 피어링을 참조하세요.

> **Note**
>
> 여러 리전 간에 트래픽을 전송하면 추가 비용이 발생할 수 있습니다.

이 단계에서는 RDS for PostgreSQL DB 인스턴스가 pgactive 확장을 이용해 활성화되었다고 가정합니다. 자세한 내용은 pgactive 확장 기능 초기화 섹션을 참조하세요.

pgactive 확장을 사용하여 첫 번째 RDS for PostgreSQL DB 인스턴스를 구성하려면

다음 예제는 pgactive 그룹을 생성하는 방법과 RDS for PostgreSQL에서 pgactive 확장을 생성하는 데 필요한 기타 단계를 보여줍니다.

1. psql 또는 다른 클라이언트 도구를 사용하여 첫 번째 RDS for PostgreSQL DB 인스턴스에 연결할 수 있습니다.

```
psql --host=firstinstance.111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password=PASSWORD --dbname=postgres
```

2. 다음 명령을 사용하여 RDS for PostgreSQL 인스턴스에 데이터베이스를 생성합니다.

```
postgres=> CREATE DATABASE app;
```

3. 다음 명령을 사용하여 새 데이터베이스로 연결을 전환합니다.

```
\c app
```

4. 다음 SQL 문을 사용하여 샘플 테이블을 생성하고 채웁니다.

   a. 다음 SQL 문을 사용하여 예제 테이블을 생성합니다.

```
app=> CREATE SCHEMA inventory;
CREATE TABLE inventory.products (
id int PRIMARY KEY, product_name text NOT NULL,
created_at timestamptz NOT NULL DEFAULT CURRENT_TIMESTAMP);
```

   b. 다음 SQL 문을 사용하여 테이블에 일부 샘플 데이터를 입력합니다.

```
app=> INSERT INTO inventory.products (id, product_name)
VALUES (1, 'soap'), (2, 'shampoo'), (3, 'conditioner');
```

   c. 다음 SQL 문을 사용하여 테이블에 데이터가 있는지 확인합니다.

```
 app=>SELECT count(*) FROM inventory.products;

 count
-------
```

```
3
```

5. 기존 데이터베이스에 pgactive 확장을 생성합니다.

```
app=> CREATE EXTENSION pgactive;
```

6. 다음 명령을 사용하여 pgactive 그룹을 만들고 초기화합니다.

```
app=>
-- connection info for endpoint1
CREATE SERVER pgactive_server_endpoint1
    FOREIGN DATA WRAPPER pgactive_fdw
    OPTIONS (host '<endpoint1>', dbname 'app');
CREATE USER MAPPING FOR postgres
    SERVER pgactive_server_endpoint1
    OPTIONS (user 'postgres', password '<password>');
      -- connection info for endpoint2
CREATE SERVER pgactive_server_endpoint2
    FOREIGN DATA WRAPPER pgactive_fdw
    OPTIONS (host '<endpoint2>', dbname 'app');
CREATE USER MAPPING FOR postgres
    SERVER pgactive_server_endpoint2
    OPTIONS (user 'postgres', password '<password>');
```

이제 복제 그룹을 초기화하고 이 첫 번째 인스턴스를 추가할 수 있습니다.

```
SELECT pgactive.pgactive_create_group(
    node_name := 'endpoint1-app',
    node_dsn := 'user_mapping=postgres
 pgactive_foreign_server=pgactive_server_endpoint1'

);
```

다음 명령을 대체 방법이지만 덜 안전한 방법으로 사용하여 pgactive 그룹을 만들고 초기화합니다.

```
app=> SELECT pgactive.pgactive_create_group(
    node_name := 'node1-app',
```

```
    node_dsn := 'dbname=app host=firstinstance.111122223333.aws-
region.rds.amazonaws.com user=postgres password=PASSWORD');
```

node1-app은 pgactive 그룹 내 노드를 고유하게 식별하기 위해 할당하는 이름입니다.

> **Note**
>
> 공개적으로 액세스할 수 있는 DB 인스턴스에서 이 단계를 성공적으로 수행하려면 rds.custom_dns_resolution 파라미터를 1로 설정하여 활성화해야 합니다.

7. DB 인스턴스가 준비되었는지 확인하려면 다음 명령을 사용합니다.

```
app=> SELECT pgactive.pgactive_wait_for_node_ready();
```

이 명령이 제대로 실행되면 다음과 같은 출력이 표시됩니다.

```
pgactive_wait_for_node_ready
-----------------------------
(1 row)
```

두 번째 RDS for PostgreSQL 인스턴스를 구성하고 **pgactive** 그룹에 연결하려면

다음 예제는 RDS for PostgreSQL DB 인스턴스를 pgactive 그룹에 조인하는 방법과 DB 인스턴스에서 pgactive 확장을 생성하는 데 필요한 기타 단계를 보여줍니다.

이 단계에서는 다른 RDS for PostgreSQL DB 인스턴스가 pgactive 확장을 이용해 설정되었다고 가정합니다. 자세한 내용은 pgactive 확장 기능 초기화 섹션을 참조하세요.

1. psql을 사용하여 게시자로부터 업데이트를 수신할 인스턴스에 연결합니다.

```
psql --host=secondinstance.111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password=PASSWORD --dbname=postgres
```

2. 다음 명령을 사용하여 두 번째 RDS for PostgreSQL DB 인스턴스에 데이터베이스를 생성합니다.

```
postgres=> CREATE DATABASE app;
```

3. 다음 명령을 사용하여 새 데이터베이스로 연결을 전환합니다.

```
\c app
```

4. 기존 데이터베이스에 pgactive 확장을 생성합니다.

```
app=> CREATE EXTENSION pgactive;
```

5. 다음 명령을 사용하여 RDS for PostgreSQL 두 번째 DB 인스턴스를 더 안전한 방법으로 pgactive 그룹에 조인합니다.

```
-- connection info for endpoint1
CREATE SERVER pgactive_server_endpoint1
    FOREIGN DATA WRAPPER pgactive_fdw
    OPTIONS (host '<endpoint1>', dbname 'app');
CREATE USER MAPPING FOR postgres
    SERVER pgactive_server_endpoint1
    OPTIONS (user 'postgres', password '<password>');

-- connection info for endpoint2
CREATE SERVER pgactive_server_endpoint2
    FOREIGN DATA WRAPPER pgactive_fdw
    OPTIONS (host '<endpoint2>', dbname 'app');
CREATE USER MAPPING FOR postgres
    SERVER pgactive_server_endpoint2
    OPTIONS (user 'postgres', password '<password>');
```

```
SELECT pgactive.pgactive_join_group(
    node_name := 'endpoint2-app',
    node_dsn := 'user_mapping=postgres
 pgactive_foreign_server=pgactive_server_endpoint2',
    join_using_dsn := 'user_mapping=postgres
 pgactive_foreign_server=pgactive_server_endpoint1'
);
```

다음 명령을 대체 방법이지만 덜 안전한 방법으로 사용하여 RDS for PostgreSQL 두 번째 DB 인스턴스를 pgactive 그룹에 조인합니다.

```
app=> SELECT pgactive.pgactive_join_group(
node_name := 'node2-app',
node_dsn := 'dbname=app host=secondinstance.111122223333.aws-
region.rds.amazonaws.com user=postgres password=PASSWORD',
join_using_dsn := 'dbname=app host=firstinstance.111122223333.aws-
region.rds.amazonaws.com user=postgres password=PASSWORD');
```

node2-app은 pgactive 그룹 내 노드를 고유하게 식별하기 위해 할당하는 이름입니다.

6. DB 인스턴스가 준비되었는지 확인하려면 다음 명령을 사용합니다.

```
app=> SELECT pgactive.pgactive_wait_for_node_ready();
```

이 명령이 제대로 실행되면 다음과 같은 출력이 표시됩니다.

```

pgactive_wait_for_node_ready
-----------------------------
(1 row)
```

첫 번째 RDS for PostgreSQL 데이터베이스가 비교적 큰 경우 pgactive.pgactive_wait_for_node_ready()에서 복원 작업의 진행 보고서가 출력되는 것을 볼 수 있습니다. 출력 결과는 다음과 비슷합니다:

```
NOTICE:  restoring database 'app', 6% of 7483 MB complete
NOTICE:  restoring database 'app', 42% of 7483 MB complete
NOTICE:  restoring database 'app', 77% of 7483 MB complete
NOTICE:  restoring database 'app', 98% of 7483 MB complete
NOTICE:  successfully restored database 'app' from node node1-app in
 00:04:12.274956
 pgactive_wait_for_node_ready
-----------------------------
(1 row)
```

이 시점부터 pgactive는 두 DB 인스턴스 간에 데이터를 동기화합니다.

7. 다음 명령을 사용하여 두 번째 DB 인스턴스의 데이터베이스에 데이터가 있는지 확인할 수 있습니다.

```
app=> SELECT count(*) FROM inventory.products;
```

데이터가 성공적으로 동기화되면 다음과 같은 출력이 표시됩니다.

```
 count
-------
 3
```

8. 다음 명령을 실행하여 새 값을 삽입합니다.

```
app=> INSERT INTO inventory.products (id, product_name) VALUES (4, 'lotion');
```

9. 첫 번째 DB 인스턴스의 데이터베이스에 연결하고 다음 쿼리를 실행합니다.

```
app=> SELECT count(*) FROM inventory.products;
```

액티브-액티브 복제가 초기화된 경우 출력은 다음과 비슷합니다.

```
count
-------
 4
```

pgactive 그룹에서 DB 인스턴스를 분리하고 제거하려면

다음 단계를 사용하여 pgactive 그룹에서 DB 인스턴스를 분리하고 제거할 수 있습니다.

1. 다음 명령을 사용하여 첫 번째 DB 인스턴스에서 두 번째 DB 인스턴스를 분리할 수 있습니다.

```
app=> SELECT * FROM pgactive.pgactive_detach_nodes(ARRAY['node2-app']);
```

2. 다음 명령을 사용하여 두 번째 DB 인스턴스에서 pgactive 확장을 제거합니다.

```
app=> SELECT * FROM pgactive.pgactive_remove();
```

확장을 강제로 제거하려면:

```
app=> SELECT * FROM pgactive.pgactive_remove(true);
```

3. 다음 명령을 사용하여 확장을 제거합니다.

```
app=> DROP EXTENSION pgactive;
```

pgactive 멤버 간의 복제 지연 측정

다음 쿼리를 사용하여 pgactive 멤버 간의 복제 지연을 볼 수 있습니다. 모든 pgactive 노드에서 이 쿼리를 실행하면 전체 상황을 파악할 수 있습니다.

```
app=> SELECT * FROM pgactive.pgactive_get_replication_lag_info();
#-[ RECORD 1 ]--------+-----------------------------------------------
#node_name            | node2-app
#node_sysid           | 7481018224801653637
#application_name     | pgactive:7481018224801653637:send
#slot_name            | pgactive_16385_7481018224801653637_0_16385__
#active               | t
#active_pid           | 783486
#pending_wal_decoding | 0
#pending_wal_to_apply | 0
#restart_lsn          | 0/2108150
#confirmed_flush_lsn  | 0/2154690
#sent_lsn             | 0/2154690
#write_lsn            | 0/2154690
#flush_lsn            | 0/2154690
#replay_lsn           | 0/2154690
#-[ RECORD 2 ]--------+-----------------------------------------------
#node_name            | node1-app
#node_sysid           | 7481018033434600853
#application_name     | pgactive:7481018033434600853:send
#slot_name            | pgactive_16385_7481018033434600853_0_16385__
#active               | t
#active_pid           | 783488
#pending_wal_decoding | 0
#pending_wal_to_apply | 0
#restart_lsn          | 0/20F5AD0
#confirmed_flush_lsn  | 0/214EF68
#sent_lsn             | 0/214EF68
#write_lsn            | 0/214EF68
#flush_lsn            | 0/214EF68
#replay_lsn           | 0/214EF68
```

최소한 다음 진단을 모니터링합니다.

활성화

활성이 false이면 알림을 설정합니다. 이는 슬롯이 현재 사용 중이 아님을 나타냅니다(구독자 인스턴스가 게시자와 연결 해제됨).

pending_wal_decoding

PostgreSQL의 논리적 복제에서 WAL 파일은 이진 형식으로 저장됩니다. 게시자는 이러한 WAL 변경 사항을 디코딩하고 논리적 변경(예: 삽입, 업데이트 또는 삭제 작업)으로 변환해야 합니다.

지표 pending_wal_decoding은 게시자 측에서 디코딩 대기 중인 WAL 파일 수를 보여 줍니다.

이 수는 다음과 같은 요인으로 인해 증가할 수 있습니다.

- 구독자가 연결되지 않은 경우 활성 상태는 false이고 pending_wal_decoding은 증가합니다.
- 슬롯이 활성 상태이지만 게시자가 WAL 변경 볼륨을 따라갈 수 없습니다.

pending_wal_to_apply

지표 pending_wal_apply는 구독자 측에서 적용 대기 중인 WAL 파일 수를 나타냅니다.

몇 가지 요인으로 인해 구독자가 변경 사항을 적용하지 못하고 다음과 같은 디스크 전체 시나리오가 발생할 수 있습니다.

- 스키마 차이 - 예를 들어 샘플이라는 테이블의 WAL 스트림이 변경되었지만 해당 테이블이 구독자 측에 없는 경우
- 프라이머리 키 열의 값이 업데이트됨
- 보조 고유 인덱스로 인해 데이터 분산이 발생할 수 있음

pgactive 확장에 대한 파라미터 설정 구성

다음 쿼리를 사용하여 pgactive 확장과 관련된 모든 파라미터를 볼 수 있습니다.

```
app=> SELECT * FROM pg_settings WHERE name LIKE 'pgactive.%';
```

다양한 파라미터를 사용하여 pgactive 확장을 구성할 수 있습니다. 이러한 파라미터는 AWS Management Console 또는 AWS CLI 인터페이스를 통해 설정할 수 있습니다.

기본 pgactive 확장 파라미터

다음 표에서는 pgactive 확장의 기본 파라미터에 대한 참조를 제공합니다.

파라미터	단위	Default	설명
pgactive.conflict_logging_include_tuples	boolean	–	pgactive 확장에 대한 전체 튜플 정보를 로깅합니다. Note 변경 사항을 적용하려면 서버를 재시작해야 합니다.
pgactive.log_conflicts_to_table	boolean	–	pgactive 확장이 감지된 충돌을 pgactive.pgactive_conflict_history 테이블에 로깅할지 여부를 결정합니다. 자세한 내용은 충돌 로깅을 참조하세요. Note 변경 사항을 적용하려면 서버를 재시작해

파라미터	단위	Default	설명
			야 합니다.
pgactive.log_conflicts_to_logfile	boolean	–	pgactive 확장이 감지된 충돌을 PostgreSQL 로그 파일에 로깅할지 여부를 결정합니다. 자세한 내용은 충돌 로깅을 참조하세요. Note 변경 사항을 적용하려면 서버를 재시작해야 합니다.

파라미터	단위	Default	설명
pgactive.synchronous_commit	boolean	끄기	pgactive 적용 작업자의 커밋 동작을 결정합니다. 비활성화(꺼짐)되면 적용 작업자가 비동기 커밋을 수행하여 적용 작업 중에 PostgreSQL 처리량이 개선되지만, 업스트림에 대한 재생 확인은 지연됩니다. off로 설정하면 항상 안전하고 트랜잭션 손실이나 건너뛰기가 발생하지 않습니다. 이 설정은 다운스트림 노드의 디스크 플러시 타이밍 및 확인이 업스트림으로 전송되는 시점에만 영향을 미칩니다. 체크포인트 또는 주기적 작업과 같은 관련 없는 작업을 통해 커밋이 디스크로 플러시될 때까지 시스템에서 재생 플러시 확인 전송이 지연됩니다. 그러나 업스트림이 synchrono

파라미터	단위	Default	설명
			us_standby_names 에 나열된 다운스트림을 포함하는 경우 이를 off로 설정하면 업스트림에서 동기 커밋이 클라이언트에 성공을 보고하는 데 더 오랜 시간이 걸립니다. 이 경우 파라미터를 on으로 설정하세요. **Note** synchronous_standby_names 에 나열된 노드를 사용하여 이 파라미터를 on으로 설정하더라도 액티브-액티브 구성에서 복제 충돌이 발생할 수 있습니

파라미터	단위	Default	설명
			다. 이는 시스템에 노드 간 잠금 및 글로벌 스냅샷 관리가 부족하여 서로 다른 노드의 동시 트랜잭션이 동일한 튜플을 수정할 수 있기 때문입니다. 또한 트랜잭션은 업스트림 노드에서 커밋한 후에만 복제를 시작합니다. 동기 커밋을 활성화해도 pgactive 확장이 항상 일관된 시스템으로

파라미터	단위	Default	설명
			변환되지 않습니다.
pgactive.temp_dump_directory	string	–	초기 설정 중 데이터베이스 복제 작업에 필요한 임시 스토리지 경로를 정의합니다. 이 디렉터리는 postgres 사용자가 쓸 수 있어야 하며 전체 데이터베이스 덤프를 포함할 만큼 충분한 스토리지 공간이 있어야 합니다. 시스템은 논리적 복사 작업이 포함된 초기 데이터베이스 설정 중에만 이 위치를 사용합니다. 이 파라미터는 pgactive_init_copy command에서 사용되지 않습니다.

파라미터	단위	Default	설명
pgactive.max_ddl_lock_delay	milliseconds	-1	동시 쓰기 트랜잭션을 강제로 중단하기 전에 DDL 잠금의 최대 대기 시간을 지정합니다. 기본값은 -1이며, 이는 max_standby_streaming_delay 에 설정된 값을 채택합니다. 이 파라미터는 시간 단위를 허용합니다. 예를 들어 10초를 나타내는 10s로 설정할 수 있습니다. 이 대기 기간 동안 시스템은 지속적 쓰기 트랜잭션이 커밋되거나 롤백될 때까지 기다리는 동안 DDL 잠금 획득을 시도합니다. 자세한 내용은 DDL 잠금을 참조하세요.

파라미터	단위	Default	설명
pgactive.ddl_lock_timeout	milliseconds	-1	DDL 잠금 시도가 잠금을 얻기 위해 대기하는 시간을 지정합니다. 기본값은 lock_timeout에 지정된 값을 사용하는 -1입니다. 10초를 나타내는 10s와 같은 시간 단위를 사용하여 이 파라미터를 설정할 수 있습니다. 이 타이머는 DDL 잠금을 얻기 위한 대기 기간만 제어합니다. 시스템이 잠금을 얻고 DDL 작업을 시작하면 타이머가 중지됩니다. 이 파라미터는 DDL 잠금을 유지할 수 있는 총기간 또는 전체 DDL 작업 시간을 제한하지 않습니다. 작업의 총기간을 제어하려면 statement_timeout 을 대신 사용합니다. 자세한 내용은 DDL 잠금을 참조하세요.

파라미터	단위	Default	설명
pgactive.debug_trace_ddl_locks_level	boolean	–	pgactive 확장의 DDL 잠금 작업에 대한 기본 디버그 로그 수준을 재정의합니다. 이 설정을 구성하면 기본 수준 대신 LOG 디버그 수준에서 DDL 잠금 관련 메시지가 내보내집니다. 전체 서버에서 상세 DEBUG1 또는 DEBUG2 로그 수준을 활성화하지 않고 DDL 잠금 활동을 모니터링하려면 이 파라미터를 사용합니다. 사용 가능한 로그 수준(상세 수준이 낮은 순): • none - DDL 잠금 메시지가 DEBUG1 이하의 서버 로그 수준으로만 표시됩니다. • statement - DDL 잠금 획득 시도에 대한 LOG 출력을 추가합니다.

파라미터	단위	Default	설명
			• acquire_release - 잠금 획득, 해제, 거부 이벤트 및 원격 DDL 잠금의 피어 노드 애플리케이션을 기록합니다. • peers - 피어 노드 간 DDL 잠금 협상에 대한 추가 세부 정보를 제공합니다. • debug - 모든 DDL 잠금 관련 활동을 LOG 수준에서 로깅합니다. 모니터링 옵션에 대한 자세한 내용은 글로벌 DDL 잠금 모니터링을 참조하세요. Note 이 설정에 대한 변경 사항은 구성을 다시 로드할 때 적

파라미터	단위	Default	설명
			용됩니다. 서버를 다시 시작할 필요가 없습니다.

추가 pgactive 확장 파라미터

다음 표에는 pgactive 확장에 사용할 수 있는 덜 자주 사용되는 구성 옵션과 내부 구성 옵션이 나와 있습니다.

파라미터	단위	Default	설명
pgactive.debug_apply_delay	integer	–	pgactive.pgactive_connections 항목에 명시적 적용 지연이 없는 구성된 연결에 적용 지연(밀리초)을 설정합니다. 이 지연은 노드 만들기 또는 조인 시간 중에 설정되며, pgactive는 커밋된 이후 지정된 밀리초 이상이 경과할 때까지 피어 노드에서 트랜잭션을 재생하지 않습니다.

파라미터	단위	Default	설명
			주로 테스트 환경에서 지연 시간이 긴 네트워크를 시뮬레이션하여 충돌을 더 쉽게 발생시키는 데 사용됩니다. 예를 들어 노드 A와 B의 지연 시간이 500ms인 경우 노드 A에 값을 삽입한 후 노드 B에 충돌하는 삽입을 수행할 수 있는 시간이 500ms 이상입니다. **Note** 서버를 다시 로드하거나 적용 작업자를 다시 시작해야 적용됩니다.

파라미터	단위	Default	설명
pgactive.connectability_check_duration	integer	–	실패한 시도 중에 데이터베이스 작업자가 연결을 설정하려고 시도하는 기간(초)을 지정합니다. 작업자는 성공하거나 이 제한 시간 값에 도달할 때까지 초당 1회 연결을 시도합니다. 이 설정은 작업자가 연결을 설정할 준비가 되기 전에 데이터베이스 엔진이 시작될 때 유용합니다.

파라미터	단위	Default	설명
pgactive.skip_ddl_replication	boolean	on	pgactive가 활성화된 Amazon RDS에서 DDL 변경 사항이 복제되거나 처리되는 방식을 제어합니다. on으로 설정하면 노드가 비 pgcctive 노드와 같은 DDL 변경 사항을 처리합니다. 이 파라미터를 사용할 때는 다음 요구 사항이 적용됩니다. • skip_ddl_replication 값이 업스트림 노드와 다른 경우 새 노드는 pgactive 그룹에 조인할 수 없습니다. • 파라미터 값이 업스트림 노드와 일치하지 않으면 기존 노드가 pgactive 작업자를 시작할 수 없습니다. • 모든 pgactive 멤버는 동일한 파라미터 값을

파라미터	단위	Default	설명
			사용해야 합니다. 최고 사용자 권한을 사용하여 글로벌, 로컬(세션 수준)의 두 가지 방법으로 이 파라미터를 수정할 수 있습니다. **Note** 이 파라미터를 잘못 변경하면 복제 설정이 손상될 수 있습니다.

파라미터	단위	Default	설명
pgactive.do_not_replicate	boolean	–	이 파라미터는 내부 전용입니다. 트랜잭션에서 이 파라미터를 설정하면 변경 사항이 DB 클러스터의 다른 노드에 복제되지 않습니다. **Note** 이 파라미터를 잘못 변경하면 복제 설정이 손상될 수 있습니다.

파라미터	단위	Default	설명
pgactive.discard_mismatched_row_attributes	boolean	–	이 파라미터는 전문가 전용입니다. 특정 복제 문제를 해결할 때만 이 파라미터를 사용하는 것이 좋습니다. 다음과 같은 경우 이 파라미터를 사용합니다. • 수신 복제 스트림에 로컬 테이블보다 더 많은 열이 있는 행이 포함되어 있습니다. • 이러한 원격 행에는 Null이 아닌 값이 포함됩니다. 이 설정은 다음 오류 메시지를 재정의하고 데이터 분산이 발생하여 복제가 계속되도록 합니다. cannot right-pad mismatched attributes; attno %u is missing in local table

파라미터	단위	Default	설명
			and remote row has non-null, non-dropped value for this attribute Note 이 파라미터를 잘못 변경하면 복제 설정이 손상될 수 있습니다.

파라미터	단위	Default	설명
pgactive.debug_trace_replay	boolean	–	on으로 설정하면 다운스트림 적용 작업자가 처리하는 각 원격 작업에 대해 로그 메시지를 내보냅니다. 로그에는 다음이 포함됩니다. • 유형 변경 • 영향을 받는 테이블 이름 • 트랜잭션 시작 이후 변경 사항 횟수 • 트랜잭션 커밋 LSN • 커밋 타임스탬프 • 업스트림 노드 식별자 • 전달 노드 식별자(해당하는 경우) 로그는 대기 중인 DDL 명령과 테이블 삭제도 캡처합니다. para> 기본적으로 로그에는 행 필드 콘텐츠가 포함되지

파라미터	단위	Default	설명
			않습니다. 로그에 행 값을 포함하려면 다음 플래그가 활성화된 상태로 다시 컴파일해야 합니다. • VERBOSE_INSERT • VERBOSE_UPDATE • VERBOSE_DELETE **Note** 이 로깅 설정을 활성화하면 성능에 영향이 미칠 수 있습니다. 문제 해결을 위해 필요한 경우에만 활성화하는 것이 좋습니다. 이 설정에 대한 변경

파라미터	단위	Default	설명
			사항은 구성을 다시 로드할 때 적용됩니다. 서버를 다시 시작할 필요가 없습니다.

파라미터	단위	Default	설명
pgactive.extra_apply_connection_options		–	pgactive 노드와의 모든 피어 노드 연결에 대해 연결 파라미터를 구성할 수 있습니다. 이러한 파라미터는 keepalive 및 SSL 모드와 같은 설정을 제어합니다. 기본적으로 pgactive는 다음 연결 파라미터를 사용합니다. • connect_timeout=30 • keepalives=1 • keepalives_idle=20 • keepalives_interval=20 • keepalives_count=5 기본 파라미터를 재정의하려면 다음과 유사한 명령을 사용합니다. pgactive.extra_apply_connection_options =

파라미터	단위	Default	설명
			`'keepalives=0'` 개별 노드 연결 문자열은 이러한 설정과 pgactive의 기본 제공 연결 옵션보다 우선합니다. 연결 문자열 형식에 대한 자세한 내용은 libpq connection strings를 참조하세요. 기본 keepalive 설정을 활성화된 상태로 유지하는 것이 좋습니다. 신뢰할 수 없는 네트워크를 통해 대규모 트랜잭션을 완료하는 데 문제가 있는 경우에만 keepalive를 비활성화합니다. **Note** 기본 keepalive 설정을 활성화된 상태로 유지하는 것

파라미터	단위	Default	설명
			이 좋습니다. 신뢰할 수 없는 네트워크를 통해 대규모 트랜잭션을 완료하는 데 문제가 있는 경우에만 keepalive를 비활성화합니다. 이 설정에 대한 변경 사항은 구성을 다시 로드할 때 적용됩니다. 서버를 다시 시작할 필요가 없습니다.

파라미터	단위	Default	설명
pgactive.init_node_parallel_jobs (int)		–	논리적 노드가 pgactive.pgactive_join_group 함수와 조인하는 동안 pg_dump 및 pg_restore 가 사용할 수 있는 병렬 작업 수를 지정합니다. 이 설정에 대한 변경 사항은 구성을 다시 로드할 때 적용됩니다. 서버를 다시 시작할 필요가 없습니다.

파라미터	단위	Default	설명
pgactive.max_nodes	int	4	pgactive 확장 그룹에 허용되는 최대 노드 수를 지정합니다. 기본값은 노드 4개입니다. 이 파라미터의 값을 설정할 때 다음 사항을 고려해야 합니다. • pgactive 확장 그룹의 모든 노드는 동일한 파라미터 값을 사용해야 합니다. • 파라미터 값이 업스트림 노드와 다른 경우 새 노드는 조인할 수 없습니다. • 파라미터 값이 업스트림 노드와 다른 경우 기존 노드는 pgactive 확장 작업자를 시작할 수 없습니다. • 그룹이 크면 추가 모니터링 및 유지 관리 작업이 필요하므로 이 파라미터의

파라미터	단위	Default	설명
			값을 신중하게 지정합니다. 이 파라미터는 구성 파일에서 ALTER SYSTEM SET 명령을 사용하여 두 가지 방법으로 설정할 수 있습니다. 이 파라미터의 기본값은 4입니다. 즉, 어느 시점에든 pgactive 확장 그룹에 최대 4개의 노드가 허용됩니다. Note 변경 사항은 서버를 다시 시작한 후에 적용됩니다.

파라미터	단위	Default	설명
pgactive.permit_node_identifier_getter_function_creation	boolean	–	이 파라미터는 내부 전용입니다. 활성화하면 pgactive 확장을 통해 pgactive 노드 식별자 getter 함수를 만들 수 있습니다.

액티브-액티브 충돌 이해

액티브-액티브 모드에서 pgactive를 사용하는 경우 여러 노드에서 동일한 테이블에 쓰면 데이터 충돌이 발생할 수 있습니다. 일부 클러스터링 시스템은 분산 잠금을 사용하여 동시 액세스를 방지하지만 pgactive는 지리적으로 분산된 애플리케이션에 더 적합한 낙관적 접근 방식을 취합니다.

일부 데이터베이스 클러스터링 시스템은 분산 잠금을 사용하여 동시 데이터 액세스를 방지합니다. 이 접근 방식은 서버가 가까이에 있을 때 작동하지만 우수한 성능을 위해 매우 짧은 지연 시간을 요구하기 때문에 지리적으로 분산된 애플리케이션을 지원하지 않습니다. pgactive 확장은 분산 잠금(비관적 접근 방식)을 사용하는 대신 낙관적 접근 방식을 사용합니다. 이는 다음을 의미합니다.

- 가능하면 충돌을 방지하는 데 도움이 됩니다.
- 특정 유형의 충돌이 발생하도록 허용합니다.
- 충돌이 발생할 때 충돌을 해결합니다.

이 접근 방식은 분산 애플리케이션을 구축할 때 더 많은 유연성을 제공합니다.

충돌이 발생하는 방법

노드 간 충돌은 관련된 모든 트랜잭션이 동일한 노드에서 동시에 발생한 경우 발생할 수 없는 이벤트 시퀀스에서 발생합니다. 노드는 트랜잭션 커밋 후에만 변경 사항을 교환하기 때문에 각 트랜잭션은 커밋된 노드에서 개별적으로 유효하지만 그동안 다른 작업을 수행한 다른 노드에서 실행되는 경우에는 유효하지 않습니다. pgactive 적용은 기본적으로 다른 노드에서 트랜잭션을 재생하므로 적용 중인 트랜잭션과 수신 노드에서 커밋된 트랜잭션 간에 충돌이 있는 경우 재생 작업이 실패할 수 있습니다.

모든 트랜잭션이 단일 노드에서 실행될 때 대부분의 충돌이 발생할 수 없는 이유는 PostgreSQL에 이를 방지하기 위한 다음과 같은 트랜잭션 간 통신 메커니즘이 있기 때문입니다.

- UNIQUE 인덱스
- SEQUENCE
- 행 및 관계 잠금
- SERIALIZABLE 종속성 추적

이러한 모든 메커니즘은 원치 않는 동시성 문제를 방지하기 위해 트랜잭션 간에 통신하는 방법입니다.

pgactive는 분산 트랜잭션 관리자 또는 잠금 관리자를 사용하지 않으므로 지연 시간을 줄이고 네트워크 파티션을 잘 처리합니다. 그러나 이는 서로 다른 노드의 트랜잭션이 서로 완전히 격리되어 실행됨을 의미합니다. 격리는 일반적으로 데이터베이스 일관성을 개선하지만 이 경우 충돌을 방지하려면 격리를 줄여야 합니다.

충돌 유형

발생할 수 있는 충돌은 다음과 같습니다.

주제

- PRIMARY KEY 또는 UNIQUE 충돌
- INSERT/INSERT 충돌
- 여러 UNIQUE 제약 조건을 위반하는 INSERT
- UPDATE/UPDATE 충돌
- PRIMARY KEY의 UPDATE 충돌
- 여러 UNIQUE 제약 조건을 위반하는 UPDATE
- UPDATE/DELETE 충돌
- INSERT/UPDATE 충돌
- DELETE/DELETE 충돌
- 외래 키 제약 조건 충돌
- 제외 제약 조건 충돌
- 글로벌 데이터 충돌
- 잠금 충돌 및 교착 상태 중단
- 분산 충돌

PRIMARY KEY 또는 UNIQUE 충돌

행 충돌은 여러 작업이 단일 노드에서 불가능한 방식으로 동일한 행 키를 수정하려고 할 때 발생합니다. 이러한 충돌은 가장 일반적인 유형의 데이터 충돌을 나타냅니다.

pgactive는 마지막 업데이트 우선 처리 또는 사용자 지정 충돌 핸들러를 통해 감지된 충돌을 해결합니다.

행 충돌에는 다음이 포함됩니다.

- INSERT 및 INSERT
- INSERT 및 UPDATE
- UPDATE 및 DELETE
- INSERT 및 DELETE
- DELETE 및 DELETE
- INSERT 및 DELETE

INSERT/INSERT 충돌

이 가장 일반적인 충돌은 서로 다른 두 노드의 INSERT가 동일한 PRIMARY KEY 값(또는 PRIMARY KEY가 없는 경우 동일한 UNIQUE 제약 조건 값)으로 튜플을 만들 때 발생합니다.

pgactivelink는 최신 튜플을 유지하기 위해 원본 호스트의 타임스탬프를 사용하여 INSERT 충돌을 해결합니다. 사용자 지정 충돌 핸들러를 사용하여 이 기본 동작을 재정의할 수 있습니다. 이 프로세스에는 특별한 관리자 작업이 필요하지 않지만 pgactivelink는 모든 노드에서 INSERT 작업 중 하나를 삭제합니다. 사용자 지정 핸들러가 구현하지 않는 한 자동 데이터 병합이 수행되지 않습니다.

pgactivelink는 단일 제약 조건 위반과 관련된 충돌만 해결할 수 있습니다. INSERT가 여러 UNIQUE 제약 조건을 위반하는 경우 추가 충돌 해결 전략을 구현해야 합니다.

여러 UNIQUE 제약 조건을 위반하는 INSERT

INSERT/INSERT 충돌은 PRIMARY KEY를 포함하여 여러 UNIQUE 제약 조건을 위반할 수 있습니다. pgactivelink는 단일 UNIQUE 제약 조건과 관련된 충돌만 처리할 수 있습니다. 충돌이 여러 UNIQUE 제약 조건을 위반하면 적용 작업자가 실패하고 다음 오류를 반환합니다.

```
multiple unique constraints violated by remotely INSERTed tuple.
```

이전 버전에서는 이 상황으로 인해 '분산된 고유성 충돌' 오류가 대신 발생했습니다.

이러한 충돌을 해결하려면 수동 조치를 취해야 합니다. 충돌하는 로컬 튜플을 DELETE하거나 UPDATE하여 새 원격 튜플과의 충돌을 제거합니다. 여러 충돌하는 튜플을 해결해야 할 수 있습니다. 현재 pgactivelink는 여러 고유 제약 조건을 위반하는 튜플을 무시, 삭제 또는 병합하는 기능을 기본적으로 제공하지 않습니다.

> **Note**
>
> 자세한 내용은 여러 UNIQUE 제약 조건을 위반하는 UPDATE를 참조하세요.

UPDATE/UPDATE 충돌

이 충돌은 PRIMARY KEY를 변경하지 않고 두 노드가 동시에 동일한 튜플을 수정할 때 발생합니다. pgactivelink는 마지막 업데이트 우선 로직 또는 사용자 지정 충돌 핸들러(정의된 경우)를 사용하여 이러한 충돌을 해결합니다. PRIMARY KEY는 튜플 매칭 및 충돌 해결에 필수적입니다. PRIMARY KEY가 없는 테이블의 경우 pgactivelink는 다음 오류와 함께 UPDATE 작업을 거부합니다.

```
Cannot run UPDATE or DELETE on table (tablename) because it does not have a primary key.
```

PRIMARY KEY의 UPDATE 충돌

pgactive에는 PRIMARY KEY 업데이트를 처리할 때 제한 사항이 있습니다. PRIMARY KEY에서 UPDATE 작업을 수행할 수 있지만 pgactive는 이러한 작업에 대해 마지막 업데이트 우선 로직을 사용하여 충돌을 자동으로 해결할 수 없습니다. PRIMARY KEY 업데이트가 기존 값과 충돌하지 않도록 해야 합니다. PRIMARY KEY 업데이트 중에 충돌이 발생하면 수동 개입이 필요한 분산 충돌이 됩니다. 이러한 상황 처리에 대한 자세한 내용은 분산 충돌 섹션을 참조하세요.

여러 UNIQUE 제약 조건을 위반하는 UPDATE

수신 UPDATE가 여러 UNIQUE 제약 조건 또는 PRIMARY KEY 값을 위반하는 경우 pgactivelink는 마지막 업데이트 우선 충돌 해결을 적용할 수 없습니다. 이 동작은 여러 제약 조건 위반이 있는 INSERT 작업과 유사합니다. 이러한 상황에서는 수동 개입이 필요한 분산 충돌이 발생합니다. 자세한 내용은 분산 충돌 섹션을 참조하세요.

UPDATE/DELETE 충돌

이 충돌은 한 노드가 행을 UPDATE하는 동시에 다른 노드가 해당 행을 DELETE할 때 발생합니다. 이 경우 재생 시 UPDATE/DELETE 충돌이 발생합니다. 해결 방법은 사용자 지정 충돌 핸들러에서 달리 지정하지 않는 한 DELETE 이후에 도착하는 모든 UPDATE를 삭제하는 것입니다.

pgactivelink는 튜플을 매칭하고 충돌을 해결하기 위해 PRIMARY KEY가 필요합니다. PRIMARY KEY가 없는 테이블의 경우 다음 오류와 함께 DELETE 작업을 거부합니다.

```
Cannot run UPDATE or DELETE on table (tablename) because it does not have a primary key.
```

> **Note**
>
> pgactivelink는 UPDATE/DELETE와 INSERT/UPDATE 충돌을 구분할 수 없습니다. 두 경우 모두 UPDATE는 존재하지 않는 행에 영향을 미칩니다. 비동기식 복제와 노드 간 재생 순서의 부재로 인해 pgactivelink는 UPDATE가 새 행(INSERT가 아직 수신되지 않음)인지 아니면 삭제된 행인지 확인할 수 없습니다. 두 시나리오 모두에서 pgactivelink는 UPDATE를 삭제합니다.

INSERT/UPDATE 충돌

이 충돌은 다중 노드 환경에서 발생할 수 있습니다. 이는 한 노드가 행을 INSERT, 두 번째 노드를 UPDATE하고, 세 번째 노드가 원래 INSERT보다 먼저 UPDATE를 수신할 때 발생합니다. 사용자 지정 충돌 트리거에서 달리 지정하지 않는 한 기본적으로 pgactivelink는 UPDATE를 삭제하여 이러한 충돌을 해결합니다. 이 해결 방법을 사용하면 노드 간에 데이터 불일치가 발생할 수 있습니다. 유사한 시나리오 및 처리에 대한 자세한 내용은 UPDATE/DELETE 충돌 섹션을 참조하세요.

DELETE/DELETE 충돌

이 충돌은 서로 다른 두 노드가 동시에 동일한 튜플을 삭제할 때 발생합니다. pgactivelink는 두 DELETE 작업의 최종 결과가 동일하기 때문에 이러한 충돌을 무해하게 간주합니다. 이 시나리오에서 pgactivelink는 데이터 일관성에 영향을 주지 않고 DELETE 작업 중 하나를 무시합니다.

외래 키 제약 조건 충돌

FOREIGN KEY 제약 조건은 기존 로컬 데이터에 원격 트랜잭션을 적용할 때 충돌을 일으킬 수 있습니다. 이러한 충돌은 일반적으로 트랜잭션이 원본 노드의 논리적 순서와 다른 순서로 적용될 때 발생합니다.

기본적으로 pgactive는 session_replication_role이 `replica`인 상태로 변경 사항을 적용하여 복제 중에 외래 키 검사를 우회합니다. 액티브-액티브 구성에서는 외래 키 위반이 발생할 수 있습니다. 대부분의 위반은 일시적이며 복제가 동기화되면 해결됩니다. 그러나 pgactive는 노드 간 행 잠금을 지원하지 않기 때문에 참조 대상이 없는 외래 키가 발생할 수 있습니다.

이 동작은 파티션 내성 비동기식 액티브-액티브 시스템에 고유합니다. 예를 들어 노드 A는 새 하위 행을 삽입하는 반면 노드 B는 상위 행을 동시에 삭제합니다. 시스템은 노드 간에 이러한 유형의 동시 수정을 방지할 수 없습니다.

외래 키 충돌을 최소화하려면 다음을 수행하는 것이 좋습니다.

- 외래 키 관계를 밀접하게 관련된 엔터티로 제한합니다.
- 가능하면 단일 노드에서 관련 엔터티를 수정합니다.
- 수정이 거의 필요하지 않은 개체를 선택합니다.
- 수정을 위한 애플리케이션 수준 동시성 제어를 구현합니다.

제외 제약 조건 충돌

pgactive 링크는 제외 제약 조건을 지원하지 않으며 이러한 조건이 만들어지지 못하게 합니다.

> **Note**
>
> 기존 독립 실행형 데이터베이스를 pgactivelink 데이터베이스로 변환하는 경우 모든 제외 제약 조건을 수동으로 삭제합니다.

분산 비동기 시스템에서는 제약 조건을 위반하는 행 집합이 없음을 보장할 수 없습니다. 이는 서로 다른 노드의 모든 트랜잭션이 완전히 격리되기 때문입니다. 제외 제약 조건으로 인해 노드에서 다른 노드로 재생을 진행할 수 없는 재생 교착 상태가 발생할 수 있습니다.

pgactive Link가 제외 제약 조건을 만들도록 강제하거나 독립 실행형 데이터베이스를 pgactive Link로 변환할 때 기존 제외 제약 조건을 삭제하지 않으면 복제가 중단될 수 있습니다. 복제 진행 상황을 복원하려면 원격 트랜잭션을 적용할 수 있도록 수신되는 원격 튜플과 충돌하는 로컬 튜플을 제거하거나 변경합니다.

글로벌 데이터 충돌

pgactivelink를 사용하는 경우 노드에 역할과 같은 서로 다른 글로벌 PostgreSQL 시스템 전체 데이터가 있을 때 충돌이 발생할 수 있습니다. 이러한 충돌로 인해 주로 DDL과 같은 작업이 성공하고 한 노드에서 커밋되지만 다른 노드에는 적용되지 않을 수 있습니다.

사용자가 한 노드에 있지만 다른 노드에는 없는 경우 다음 복제 문제가 발생할 수 있습니다.

- Node1에는 이름이 `fred`인 사용자가 있지만 이 사용자는 Node2에 존재하지 않습니다.

- `fred`가 Node1에서 테이블을 만들면 `fred`를 소유자로 사용하여 테이블이 복제됩니다.
- 이 DDL 명령이 Node2에 적용되면 `fred` 사용자가 없기 때문에 실패합니다.
- 이 실패는 Node2의 PostgreSQL 로그에서 ERROR를 생성하고 `pgactive.pgactive_stats.nr_rollbacks` 카운터를 증가시킵니다.

해결 방법: Node2에서 사용자 `fred`를 생성합니다. 사용자는 동일한 권한이 필요하지 않지만 두 노드 모두에 있어야 합니다.

한 노드에는 테이블이 있지만 다른 노드에는 없는 경우 데이터 수정 작업이 실패합니다.

- Node1에는 Node2에 존재하지 않는 `foo`라는 테이블이 있습니다.
- Node1의 `foo` 테이블에 대한 모든 DML 작업은 Node2에 복제되었을 때 실패합니다.

해결 방법: 동일한 구조로 Node2에 `foo` 테이블을 생성합니다.

> **Note**
>
> pgactivelink는 현재 CREATE USER 명령 또는 DDL 작업을 복제하지 않습니다. DDL 복제는 향후 릴리스에 계획되어 있습니다.

잠금 충돌 및 교착 상태 중단

pgactive 적용 프로세스는 일반 사용자 세션처럼 작동하므로 표준 행 및 테이블 잠금 규칙을 따릅니다. 이로 인해 pgactivelink 적용 프로세스가 사용자 트랜잭션 또는 다른 적용 프로세스가 보유한 잠금을 대기할 수 있습니다.

다음 유형의 잠금은 적용 프로세스에 영향을 미칠 수 있습니다.

- 사용자 세션별 명시적 테이블 수준 잠금(LOCK TABLE ...)
- 사용자 세션에 의한 명시적 행 수준 잠금(SELECT ... FOR UPDATE/FOR SHARE)
- 외래 키로 인한 잠금
- 로컬 활동이나 다른 서버의 적용으로 인해 발생하는 행의 UPDATE, INSERT, DELETE에 따른 암시적 잠금

다음 사이에 교착 상태가 발생할 수 있습니다.

- pgactivelink 적용 프로세스 및 사용자 트랜잭션
- 두 개의 적용 프로세스

교착 상태가 발생하면 PostgreSQL의 교착 상태 감지기는 문제 트랜잭션 중 하나를 종료합니다. pgactivelink 적용 작업자의 프로세스가 종료되면 자동으로 재시도되고 일반적으로 성공합니다.

> **Note**
>
> - 이러한 문제는 일시적이며 일반적으로 관리자의 개입이 필요하지 않습니다. 유휴 사용자 세션의 잠금으로 인해 적용 프로세스가 장기간 차단된 경우 사용자 세션을 종료하여 복제를 재개할 수 있습니다. 이 상황은 사용자가 다른 사용자 세션에 영향을 미치는 긴 잠금을 유지하는 경우와 유사합니다.
> - 잠금 관련 재생 지연을 식별하려면 PostgreSQL에서 `log_lock_waits` 기능을 활성화합니다.

분산 충돌

노드 간에 동일해야 하는 데이터가 예기치 않게 다를 때 분산 충돌이 발생합니다. 이러한 충돌은 발생하지 않아야 하지만 현재 구현에서 모든 충돌을 안정적으로 방지할 수 있는 것은 아닙니다.

> **Note**
>
> 모든 노드가 변경을 처리하기 전에 다른 노드가 동일한 행의 키를 변경할 경우 행의 PRIMARY KEY를 수정하면 분산 충돌이 발생할 수 있습니다. 프라이머리 키를 변경하지 않거나 지정된 노드 하나로 변경을 제한합니다. 자세한 내용은 the section called “PRIMARY KEY의 UPDATE 충돌 ” 섹션을 참조하세요.

행 데이터와 관련된 분산 충돌에는 일반적으로 관리자의 개입이 필요합니다. 이러한 충돌을 해결하려면 `pgactive.pgactive_do_not_replicate`를 사용하여 복제를 일시적으로 비활성화하면서 한 노드의 데이터를 다른 노드와 일치하도록 수동으로 조정해야 합니다. pgactive를 문서화된 대로 사용하고 안전하지 않은 것으로 표시된 설정이나 함수를 사용하지 않는 경우 이러한 충돌이 발생하지 않습니다.

관리자는 이러한 충돌을 수동으로 해결해야 합니다. 충돌 유형에 따라와 `pgactive.pgactive_do_not_replicate` 같은 고급 옵션을 사용해야 합니다. 이러한 옵션을 잘못

사용하면 상황이 악화될 수 있으므로 주의해서 사용하세요. 가능한 충돌이 다양하기 때문에 범용 해결 지침을 제공할 수 없습니다.

서로 다른 노드에서 동일해야 하는 데이터가 예기치 않게 다를 때 분산 충돌이 발생합니다. 이러한 충돌은 발생하지 않아야 하지만 현재 구현에서 이러한 모든 충돌을 안정적으로 방지할 수 있는 것은 아닙니다.

충돌 방지 또는 허용

대부분의 경우 적절한 애플리케이션 설계를 사용하여 충돌을 방지하거나 애플리케이션이 충돌을 허용하도록 할 수 있습니다.

충돌은 여러 노드에서 동시 작업이 수행되는 경우에만 발생합니다. 충돌을 방지하려면:

- 하나의 노드에만 쓰기
- 각 노드의 독립 데이터베이스 하위 집합에 쓰기(예: 각 노드에 별도의 스키마 할당)

INSERT 및 INSERT 충돌의 경우 글로벌 시퀀스를 사용하여 충돌을 완전히 방지합니다.

사용 사례에서 충돌이 허용되지 않는 경우 애플리케이션 수준에서 분산 잠금을 구현하는 것이 좋습니다. 가장 좋은 방법은 모든 충돌을 방지하려고 하지 않고 pgactive의 충돌 해결 메커니즘과 연계되도록 애플리케이션을 설계하는 것입니다. 자세한 내용은 the section called “충돌 유형” 섹션을 참조하세요.

충돌 로깅

pgactivelink는 `pgactive.pgactive_conflict_history` 테이블에 충돌 인시던트를 로깅하여 액티브-액티브 충돌을 진단하고 처리하는 데 도움이 됩니다. 이 테이블에 대한 충돌 로깅은 `pgactive.log_conflicts_to_table`을 true로 설정한 경우에만 발생합니다. 또한 pgactive 확장은 `pgactive.log_conflicts_to_table` 설정에 관계없이 log_min_messages가 `LOG` 또는 `lower`로 설정된 경우 PostgreSQL 로그 파일에 충돌을 기록합니다.

충돌 기록 테이블을 사용하여 다음을 수행할 수 있습니다.

- 애플리케이션이 충돌을 생성하는 빈도 측정
- 충돌이 발생하는 위치 식별
- 애플리케이션을 개선하여 충돌률 감소
- 충돌 해결로 원하는 결과가 나오지 않는 경우 감지
- 사용자 정의 충돌 트리거 또는 애플리케이션 설계 변경이 필요한 위치 확인

행 충돌의 경우 선택적으로 행 값을 로깅할 수 있습니다. 이는 `pgactive.log_conflicts_to_table` 설정에 의해 제어됩니다. 참고:

- 이는 글로벌 데이터베이스 전체 옵션입니다.
- 행 값 로깅에 대한 테이블별 제어는 없습니다.
- 필드 번호, 배열 요소 또는 필드 길이에는 제한이 적용되지 않습니다.
- 충돌을 트리거할 수 있는 수 메가바이트 크기의 행으로 작업하는 경우 이 기능을 활성화하지 않는 것이 좋습니다.

충돌 기록 테이블에는 데이터베이스의 모든 테이블(각각 스키마가 다를 수 있음)의 데이터가 포함되어 있으므로 로깅된 행 값은 JSON 필드로 저장됩니다. JSON은 SQL에서 직접적으로 호출하는 것과 마찬가지로 `row_to_json`을 사용하여 만들어집니다. PostgreSQL은 `json_to_row` 함수를 제공하지 않으므로 로깅된 JSON에서 복합 형식 튜플을 재구성하려면 테이블별 코드(PL/pgSQL, PL/Python, PL/Perl 등)가 필요합니다.

> **Note**
>
> 사용자 정의 충돌에 대한 지원은 향후 확장 기능으로 계획되어 있습니다.

pgactive 스키마 이해

pgactive 스키마는 RDS for PostgreSQL에서 액티브-액티브 복제를 관리합니다. 이 스키마에는 복제 구성 및 상태 정보를 저장하는 테이블이 포함되어 있습니다.

> **Note**
>
> pgactive 스키마는 발전 중이며 변경될 수 있습니다. 이러한 테이블의 데이터를 직접 수정하지 마세요.

pgactive 스키마의 키 테이블에는 다음이 포함됩니다.

- `pgactive_nodes` - 액티브-액티브 복제 그룹의 노드에 대한 정보를 저장합니다.
- `pgactive_connections` - 각 노드의 연결 세부 정보를 저장합니다.

pgactive_nodes

pgactive_nodes는 액티브-액티브 복제 그룹에 참여하는 노드에 대한 정보를 저장합니다.

열	유형	콜레이션	Nullable	Default
node_sysid	텍스트	–	Null이 아님	–
node_timeline	oid	–	Null이 아님	–
node_dboid	oid	–	Null이 아님	–
node_status	char	–	Null이 아님	–
node_name	텍스트	–	Null이 아님	–
node_dsn	텍스트	–	Null이 아님	–
node_init_from_dsn	텍스트	–	Null이 아님	–
node_read_only	부울	–	–	false
node_seq_id	smallint	–	Null이 아님	–

node_sysid

pgactive_create_group 또는 pgactive_join_group 중에 생성된 노드의 고유 ID

node_status

노드 준비 상태:

- b - 설정 시작
- i - 초기화 중
- c - 캐치업
- o - 아웃바운드 슬롯을 만드는 중
- r - 준비됨
- k - 종료됨

> **Note**
>
> 이 열은 노드가 연결 또는 연결 해제되었는지 여부를 나타내지 않습니다.

node_name

사용자가 제공한 고유 노드 이름입니다.

node_dsn

연결 문자열 또는 사용자 매핑 이름입니다.

node_init_from_dsn

이 노드가 생성된 DSN입니다.

pgactive_connection

pgactive_connections는 각 노드에 대한 연결 세부 정보를 저장합니다.

열	유형	콜레이션	Nullable	Default
conn_sysid	텍스트	없음	Null이 아님	없음
conn_timeline	oid	없음	Null이 아님	없음
conn_dboid	oid	없음	Null이 아님	없음
conn_dsn	텍스트	없음	Null이 아님	없음
conn_apply_delay	정수	없음	없음	없음
conn_replication_sets	텍스트	없음	없음	없음

conn_sysid

이 항목이 참조하는 노드의 노드 식별자입니다.

conn_dsn

pgactive.pgactive_nodes node_dsn과 동일합니다.

conn_apply_delay

설정된 경우 원격 노드에서 각 트랜잭션을 적용하기 전에 기다릴 시간(밀리초)입니다. 주로 디버깅용입니다. null인 경우 글로벌 기본값이 적용됩니다.

복제 세트 작업

복제 세트는 복제 작업에서 포함하거나 제외할 테이블을 결정합니다. 기본적으로 달리 지정하지 않는 한 다음 함수를 사용하여 모든 테이블이 복제됩니다.

- `pgactive_exclude_table_replication_set()` - 복제에서 지정된 테이블을 제외합니다.
- `pgactive_include_table_replication_set()` - 복제에 지정된 테이블을 포함합니다.

> **Note**
>
> 복제 세트를 구성하기 전에 다음 사항을 고려하세요.
>
> - `pgactive_create_group()`을 실행한 후 그리고 `pgactive_join_group()` 전에만 테이블 포함 또는 제외를 구성할 수 있습니다.
> - `pgactive_exclude_table_replication_set()`를 사용한 후에는 `pgactive_include_table_replication_set()`를 사용할 수 없습니다.
> - `pgactive_include_table_replication_set()`를 사용한 후에는 `pgactive_exclude_table_replication_set()`를 사용할 수 없습니다.

시스템은 초기 구성에 따라 새로 만들어진 테이블을 다르게 처리합니다.

- 테이블을 제외한 경우: `pgactive_join_group()` 이후에 생성된 모든 새 테이블은 복제에 자동으로 포함됩니다.
- 테이블을 포함한 경우: `pgactive_join_group()` 이후에 생성된 모든 새 테이블은 복제에서 자동으로 제외됩니다.

특정 테이블에 대한 복제 세트 구성을 보려면 `pgactive.pgactive_get_table_replication_sets()` 함수를 사용합니다.

pgactive 함수 참조

다음은 효과적으로 사용하기 위한 파라미터, 반환 값 및 실제 사용 정보가 포함된 pgactive 함수의 목록입니다.

get_last_applied_xact_info

지정된 노드에 대해 마지막으로 적용된 트랜잭션 정보를 검색합니다.

인수

- sysid(텍스트) - 타임라인 OID
- dboid(OID)

반환 타입

다음을 기록합니다.

- last_applied_xact_id(OID)
- last_applied_xact_committs(타임스탬프와 시간대)
- last_applied_xact_at(타임스탬프와 시간대)

사용 노트

이 함수를 사용하여 지정된 노드에 마지막으로 적용된 트랜잭션 정보를 검색합니다.

pgactive_apply_pause

복제 적용 프로세스를 일시 중지합니다.

인수

없음

반환 타입

boolean

사용 노트

복제 적용 프로세스를 일시 중지하려면 이 함수를 직접적으로 호출합니다.

pgactive_apply_resume

복제 적용 프로세스를 재개합니다.

인수

없음

반환 타입

void

사용 노트

복제 적용 프로세스를 재개하려면 이 함수를 직접적으로 호출합니다.

pgactive_is_apply_paused

복제 적용이 현재 일시 중지되었는지 확인합니다.

인수

없음

반환 타입

boolean

사용 노트

이 함수를 사용하여 복제 적용이 현재 일시 중지되었는지 확인합니다.

pgactive_create_group

독립 실행형 데이터베이스를 초기 노드로 변환하여 pgactive 그룹을 만듭니다.

인수

- node_name(텍스트)
- node_dsn(텍스트)
- apply_delay 정수 DEFAULT NULL::정수 - replication_sets 텍스트[] DEFAULT ARRAY['default'::텍스트]

반환 타입

void

사용 노트

독립 실행형 데이터베이스를 초기 노드로 변환하여 pgactive 그룹을 만듭니다. 이 함수는 노드를 pgactive 노드로 변환하기 전에 무결성 검사를 수행합니다. 이 함수를 사용하기 전에 PostgreSQL 클러스터에 pgactive 백그라운드 작업자를 지원할 수 있는 충분한 `max_worker_processes`가 있는지 확인합니다.

pgactive_detach_nodes

pgactive 그룹에서 지정된 노드를 제거합니다.

인수

- p_nodes(텍스트[])

반환 타입

void

사용 노트

이 함수를 사용하여 pgactive 그룹에서 지정된 노드를 제거합니다.

pgactive_exclude_table_replication_set

복제에서 특정 테이블을 제외합니다.

인수

- p_relation(regclass)

반환 타입

void

사용 노트

이 함수를 사용하여 복제에서 특정 테이블을 제외합니다.

pgactive_get_replication_lag_info

노드 세부 정보, WAL 상태 및 LSN 값을 포함한 자세한 복제 지연 정보를 검색합니다.

인수

없음

반환 타입

SETOF 레코드 - node_name 텍스트 - node_sysid 텍스트 - application_name 텍스트 - slot_name 텍스트 - active 부울 - active_pid 정수 - pending_wal_decoding bigint - 전송자 노드에서 디코딩할 WAL의 대략적인 크기(바이트) - pending_wal_to_apply bigint - 수신 노드에서 적용할 WAL의 대략적인 크기(바이트) - restart_lsn pg_lsn - confirmed_flush_lsn pg_lsn - sent_lsn pg_lsn - write_lsn pg_lsn - flush_lsn pg_lsn - replay_lsn pg_lsn

사용 노트

이 함수를 직접적으로 호출하여 노드 세부 정보, WAL 상태 및 LSN 값을 포함한 복제 지연 정보를 검색합니다.

pgactive_get_stats

pgactive 복제 통계를 검색합니다.

인수

없음

반환 타입

SETOF 레코드 - rep_node_id oid - rilocalid oid - riremoteid text - nr_commit bigint - nr_rollback bigint - nr_insert bigint - nr_insert_conflict bigint - nr_update bigint - nr_update_conflict bigint - nr_delete bigint - nr_delete_conflict bigint - nr_disconnect bigint

사용 노트

이 함수를 사용하여 pgactive 복제 통계를 검색합니다.

pgactive_get_table_replication_sets

특정 관계에 대한 복제 세트 구성을 가져옵니다.

인수

- 관계(regclass)

반환 타입

SETOF 레코드

사용 노트

이 함수를 직접적으로 호출하여 특정 관계에 대한 복제 세트 구성을 가져옵니다.

pgactive_include_table_replication_set

복제에 특정 테이블을 포함합니다.

인수

- p_relation(regclass)

반환 타입

void

사용 노트

복제에 특정 테이블을 포함하려면 이 함수를 사용합니다.

pgactive_join_group

기존 pgactive 그룹에 노드를 추가합니다.

인수

- node_name(텍스트)
- node_dsn(텍스트)
- join_using_dsn(텍스트)
- apply_delay(정수, 선택 사항)
- replication_sets(텍스트[], 기본값: ['default'])
- bypass_collation_check(부울, 기본값: false)
- bypass_node_identifier_creation(부울, 기본값: false)
- bypass_user_tables_check(부울, 기본값: false)

반환 타입

void

사용 노트

이 함수를 직접적으로 호출하여 기존 pgactive 그룹에 노드를 추가합니다. PostgreSQL 클러스터에 pgactive 백그라운드 작업자를 위한 충분한 max_worker_processes가 있는지 확인합니다.

pgactive_remove

로컬 노드에서 모든 pgactive 구성 요소를 제거합니다.

인수

- force(부울, 기본값: false)

반환 타입

void

사용 노트

로컬 노드에서 모든 pgactive 구성 요소를 제거하려면 이 함수를 직접적으로 호출합니다.

pgactive_snowflake_id_nextval

노드별 고유 시퀀스 값을 생성합니다.

인수

- regclass

반환 타입

bigint

사용 노트

이 함수를 사용하여 노드별 고유 시퀀스 값을 생성합니다.

pgactive_update_node_conninfo

pgactive 노드의 연결 정보를 업데이트합니다.

인수

- node_name_to_update(텍스트)

- node_dsn_to_update(텍스트)

반환 타입

void

사용 노트

이 함수를 사용하여 pgactive 노드의 연결 정보를 업데이트합니다.

pgactive_wait_for_node_ready

그룹 생성 또는 조인 작업의 진행 상황을 모니터링합니다.

인수

- timeout(정수, 기본값: 0)
- progress_interval(정수, 기본값: 60)

반환 타입

void

사용 노트

이 함수를 직접적으로 호출하여 그룹 만들기 또는 조인 작업의 진행 상황을 모니터링합니다.

액티브-액티브 복제의 충돌 처리

pgactive 확장은 클러스터별이 아니라 데이터베이스별로 작동합니다. pgactive를 사용하는 각 DB 인스턴스는 독립 인스턴스이며 모든 소스의 데이터 변경을 수락할 수 있습니다. 변경 사항이 DB 인스턴스로 전송되면 PostgreSQL은 변경 사항을 로컬에서 커밋한 다음 pgactive를 사용하여 변경 사항을 다른 DB 인스턴스에 비동기적으로 복제합니다. 두 PostgreSQL DB 인스턴스가 거의 동시에 같은 레코드를 업데이트하는 경우 충돌이 발생할 수 있습니다.

pgactive 확장은 충돌 감지 및 자동 해결을 위한 메커니즘을 제공합니다. 두 DB 인스턴스 모두에서 트랜잭션이 커밋된 시점의 타임스탬프를 추적하고 최신 타임스탬프와 함께 변경 사항을 자동으로 적용합니다. 또한 pgactive 확장은 pgactive.pgactive_conflict_history 테이블에서 충돌이 발생하는 경우에 이를 로깅합니다.

pgactive.pgactive_conflict_history는 계속 규모가 늘어납니다. 제거 정책을 정의할 수 있습니다. 정기적으로 일부 레코드를 삭제하거나 이 관계에 대한 파티션 스키마를 정의하여 수행할 수 있으며, 나중에 원하는 파티션을 분리, 삭제, 잘라낼 수 있습니다. 정기적으로 제거 정책을 구현하기 위한 한

가지 옵션은 pg_cron 확장을 사용하는 것입니다. PostgreSQL pg_cron 확장을 사용하여 유지 관리 일정 예약 pg_cron 기록 테이블에 대한 다음 예제 정보를 참조하세요.

액티브-액티브 복제의 시퀀스 처리

pgactive 확장이 포함된 RDS for PostgreSQL DB 인스턴스는 서로 다른 두 개의 시퀀스 메커니즘을 사용하여 고유한 값을 생성합니다.

글로벌 시퀀스

글로벌 시퀀스를 사용하려면 CREATE SEQUENCE 명령문을 사용하여 로컬 시퀀스를 생성합니다. 시퀀스의 다음 고유 값을 가져오려면 usingnextval(seqname) 대신 pgactive.pgactive_snowflake_id_nextval(seqname)을 사용합니다.

다음 예제에서는 글로벌 시퀀스를 생성합니다.

```
app=> CREATE TABLE gstest (
      id bigint primary key,
      parrot text
    );
```

```
app=>CREATE SEQUENCE gstest_id_seq OWNED BY gstest.id;
```

```
app=> ALTER TABLE gstest \
      ALTER COLUMN id SET DEFAULT \
      pgactive.pgactive_snowflake_id_nextval('gstest_id_seq');
```

분할된 시퀀스

분할 단계 또는 분할된 시퀀스에서는 각 노드에 일반 PostgreSQL 시퀀스가 사용됩니다. 각 시퀀스는 같은 양만큼 증가하고 다른 오프셋에서 시작합니다. 예를 들어, 100단계에서 노드 1은 101, 201, 301 등의 시퀀스를 생성하고 노드 2는 102, 202, 302 등의 시퀀스를 생성합니다. 이 스키마는 노드가 장기간 통신할 수 없는 경우에도 잘 작동하지만 설계자가 스키마를 설정할 때 최대 노드 수를 지정해야 하며 노드별 구성이 필요합니다. 실수로 인해 시퀀스가 겹치기 쉽습니다.

다음과 같이 노드에 원하는 시퀀스를 생성하여 pgactive를 사용해 이 접근 방식을 비교적 간단히 구성할 수 있습니다.

```
CREATE TABLE some_table (generated_value bigint primary key);
```

```
app=> CREATE SEQUENCE some_seq INCREMENT 100 OWNED BY some_table.generated_value;
```

```
app=> ALTER TABLE some_table ALTER COLUMN generated_value SET DEFAULT
 nextval('some_seq');
```

그런 다음 각 노드에서 `setval`을 호출하여 다음과 같이 다른 오프셋 시작 값을 지정합니다.

```
app=>
-- On node 1
SELECT setval('some_seq', 1);

-- On node 2
SELECT setval('some_seq', 2);
```

pg_repack 확장을 사용하여 테이블 및 인덱스에서 부풀림을 줄입니다.

pg_repack 확장을 사용하여 VACUUM FULL의 대안으로 표와 인덱스에서 팽창을 제거할 수 있습니다. 이 확장은 RDS for PostgreSQL 버전 9.6.3 이상에서 지원됩니다. pg_repack 확장 및 전체 표 재압축에 대한 자세한 내용은 GitHub 프로젝트 설명서를 참조하세요.

VACUUM FULL과 달리 pg_repack 확장에는 다음과 같은 경우 표 재구축 작업 중 짧은 기간 동안만 배타적 잠금(AccessExclusiveLock)이 필요합니다.

- 로그 표 초기 생성 - 다음 예와 같이 데이터의 초기 복사 중에 발생하는 변경 사항을 기록하기 위해 로그 표가 생성됩니다.

```
postgres=>\dt+ repack.log_*
List of relations
-[ RECORD 1 ]-+----------
Schema        | repack
Name          | log_16490
Type          | table
Owner         | postgres
Persistence   | permanent
Access method | heap
Size          | 65 MB
Description   |
```

- 최종 교체 및 삭제 단계.

나머지 재구축 작업의 경우 원래 표 행을 새 표로 복사하려면 원래 표에 ACCESS SHARE 잠금만 있으면 됩니다. 이렇게 하면 INSERT, UPDATE, DELETE 작업을 평소처럼 진행할 수 있습니다.

추천

pg_repack 확장을 사용하여 표와 색인에서 팽창을 제거할 때는 다음 권장 사항이 적용됩니다.

- 업무 외 시간이나 유지 관리 기간 중에 재압축을 수행하여 다른 데이터베이스 활동의 성능에 미치는 영향을 최소화합니다.
- 재구축 작업 중 차단 세션을 면밀히 모니터링하고 원래 표에 특히 배타적 잠금이 필요한 최종 교체 및 삭제 단계 중 pg_repack을 차단할 수 있는 작업이 없는지 확인합니다. 자세한 내용은 쿼리를 차단하는 요소 식별을 참조하세요.

차단 세션이 표시되면 신중하게 고려한 후 다음 명령을 사용하여 세션을 종료할 수 있습니다. 이를 통해 pg_repack을 계속하여 재구축을 완료하는 데 도움이 됩니다.

```
SELECT pg_terminate_backend(pid);
```

- 트랜잭션 비율이 매우 높은 시스템에서 pg_repack's 로그 표의 누적된 변경 사항을 적용하는 동안 적용 프로세스가 변경 속도를 따라가지 못할 수 있습니다. 이 경우 pg_repack에서 적용 프로세스를 완료할 수 없습니다. 자세한 내용은 재압축 중 새 표 모니터링 단원을 참조하십시오. 인덱스가 심하게 팽창된 경우 다른 해결 방법은 인덱스 전용 재압축을 수행하는 것입니다. 이는 또한 VACUUM의 인덱스 정리 주기를 더 빠르게 완료하는 데도 도움이 됩니다.

  PostgreSQL 버전 12의 수동 VACUUM을 사용하면 인덱스 정리 단계를 건너뛸 수 있으며, PostgreSQL 버전 14의 긴급 autovacuum 중에는 인덱스 정리 단계를 자동으로 건너뛸 수 있습니다. 이렇게 하면 인덱스 팽창을 제거하지 않고도 VACUUM을 더 빠르게 완료할 수 있습니다. 이는 랩어라운드 VACUUM 방지와 같은 긴급 상황에만 사용 가능합니다. 자세한 내용은 Amazon Aurora 사용 설명서의 인덱스 팽창 방지를 참조하세요.

필수 조건

- 표에는 PRIMARY KEY 또는 null이 아닌 UNIQUE 제약 조건이 있어야 합니다.
- 확장 버전은 클라이언트와 서버 모두 동일해야 합니다.
- RDS 인스턴스에 팽창이 없는 표의 전체 크기보다 더 많은 FreeStorageSpace가 있는지 확인합니다. 예를 들어, TOAST와 인덱스를 포함한 표의 총 크기를 2TB로, 표의 총 팽창을 1TB로 가정해 보겠습니다. 필요한 FreeStorageSpace는 다음 계산에서 반환된 값보다 커야 합니다.

  2TB (Table size) - 1TB (Table bloat) = 1TB

  다음 쿼리를 사용하여 표의 전체 크기를 확인하고 pgstattuple을 사용하여 팽창을 도출할 수 있습니다. 자세한 내용은 Amazon Aurora 사용 설명서의 표 및 인덱스 팽창 진단을 참조하세요.

  ```
  SELECT pg_size_pretty(pg_total_relation_size('table_name')) AS total_table_size;
  ```

  이 스페이스는 활동 완료 후 회수됩니다.
- RDS 인스턴스에 재압축 작업을 처리할 수 있는 충분한 컴퓨팅 및 IO 용량이 있는지 확인합니다. 최적의 성능 균형을 보장하려면 인스턴스 클래스를 확장하는 방안을 고려할 수 있습니다.

pg_repack 확장을 사용하려면

1. 다음 명령을 실행하여 RDS for PostgreSQL 인스턴스에 pg_repack 확장을 설치합니다.

```
CREATE EXTENSION pg_repack;
```

2. 다음 명령을 실행하여 pg_repack에서 생성한 임시 로그 표에 대한 쓰기 액세스 권한을 부여합니다.

```
ALTER DEFAULT PRIVILEGES IN SCHEMA repack GRANT INSERT ON TABLES TO PUBLIC;
ALTER DEFAULT PRIVILEGES IN SCHEMA repack GRANT USAGE, SELECT ON SEQUENCES TO
 PUBLIC;
```

3. pg_repack 클라이언트 유틸리티를 사용하여 데이터베이스에 연결합니다. rds_superuser 권한이 있는 계정을 사용합니다. 예를 들어 rds_test 역할이 rds_superuser 권한을 가지고 있다고 가정하겠습니다. 다음 구문은 postgres 데이터베이스의 모든 표 인덱스를 포함하는 전체 표에 대해 pg_repack을 수행합니다.

```
pg_repack -h db-instance-name.111122223333.aws-region.rds.amazonaws.com -U rds_test
 -k postgres
```

> **Note**
>
> -k 옵션을 사용하여 연결해야 합니다. -a 옵션은 지원되지 않습니다.

pg_repack 클라이언트의 응답은 재압축된 DB 인스턴스상의 표에 대한 정보를 제공합니다.

```
INFO: repacking table "pgbench_tellers"
INFO: repacking table "pgbench_accounts"
INFO: repacking table "pgbench_branches"
```

4. 다음 구문은 postgres 데이터베이스의 인덱스를 포함한 단일 표 orders를 재압축합니다.

```
pg_repack -h db-instance-name.111122223333.aws-region.rds.amazonaws.com -U rds_test
 --table orders -k postgres
```

다음 구문은 postgres 데이터베이스의 orders 표에 대한 인덱스만 재압축합니다.

```
pg_repack -h db-instance-name.111122223333.aws-region.rds.amazonaws.com -U rds_test
 --table orders --only-indexes -k postgres
```

재압축 중 새 표 모니터링

- 데이터베이스 크기는 교체 및 삭제 재압축 단계까지 표의 전체 크기에서 팽창을 뺀 값만큼 증가합니다. 데이터베이스 크기의 증가율을 모니터링하고, 재압축 속도를 계산하고, 초기 데이터 전송을 완료하는 데 걸리는 시간을 대략적으로 예측할 수 있습니다.

  예를 들어, 표의 총 크기를 2TB로, 데이터베이스의 크기를 4TB로, 표의 총 팽창을 1TB로 가정해 보겠습니다. 재압축 작업 종료 시 계산의 결과로 반환되는 데이터베이스 총 크기 값은 다음과 같습니다.

  ```
  2TB (Table size) + 4 TB (Database size) - 1TB (Table bloat) = 5TB
  ```

  두 시점 사이의 증가율(바이트)을 샘플링하여 재압축 작업의 속도를 대략적으로 추정할 수 있습니다. 증가율이 분당 1GB인 경우 초기 표 구축 작업을 완료하는 데 약 1,000분 또는 약 16.6시간이 걸릴 수 있습니다. 초기 표 구축 외에도 pg_repack에서는 누적된 변경 사항을 적용해야 합니다. 소요 시간은 진행 중인 변경 사항과 누적 변경 사항을 적용하는 속도에 따라 달라집니다.

  > **Note**
  >
  > pgstattuple 확장을 사용하여 표의 팽창을 계산할 수 있습니다. 자세한 내용은 pgstattuple을 참조하세요.

- 재압축 스키마 아래에 있는 pg_repack's 로그 표의 행 수는 초기 로드 후 새 표에 적용하기 위해 보류 중인 변경 사항의 양을 나타냅니다.

  pg_stat_all_tables에서 pg_repack's 로그 표를 확인하여 새 표에 적용된 변경 사항을 모니터링할 수 있습니다. pg_stat_all_tables.n_live_tup은 새 표에 적용할 보류 중인 레코드 수를 나타냅니다. 자세한 내용은 pg_stat_all_tables를 참조하세요.

  ```
  postgres=>SELECT relname,n_live_tup FROM pg_stat_all_tables WHERE schemaname =
   'repack' AND relname ILIKE '%log%';

  -[ RECORD 1 ]---------
  relname    | log_16490
  n_live_tup | 2000000
  ```

- pg_stat_statements 확장을 사용하여 재압축 작업의 각 단계에 소요된 시간을 확인할 수 있습니다. 이는 프로덕션 환경에서 동일한 재압축 작업을 적용할 준비를 하는 데 유용합니다. LIMIT 절을 조정하여 출력을 더 확장할 수 있습니다.

```
postgres=>SELECT
     SUBSTR(query, 1, 100) query,
     round((round(total_exec_time::numeric, 6) / 1000 / 60),4)
 total_exec_time_in_minutes
 FROM
     pg_stat_statements
 WHERE
     query ILIKE '%repack%'
 ORDER BY
     total_exec_time DESC LIMIT 5;

 query                                                                    |
 total_exec_time_in_minutes
--------------------------------------------------------------------------
+---------------------------
 CREATE UNIQUE INDEX index_16493 ON repack.table_16490 USING btree (a) |
        6.8627
 INSERT INTO repack.table_16490 SELECT a FROM ONLY public.t1           |
        6.4150
 SELECT repack.repack_apply($1, $2, $3, $4, $5, $6)                    |
        0.5395
 SELECT repack.repack_drop($1, $2)                                     |
        0.0004
 SELECT repack.repack_swap($1)                                         |
        0.0004
(5 rows)
```

재압축은 부적절한 작업이므로, 원본 표는 영향을 받지 않으며 원본 표를 복구해야 하는 예상치 못한 문제가 발생하지 않습니다. 재압축이 예기치 않게 실패할 경우 오류의 원인을 검사하여 해결해야 합니다.

문제가 해결되면 표가 있는 데이터베이스에서 pg_repack 확장을 삭제하고 다시 만든 다음 pg_repack 단계를 다시 시도하세요. 또한 컴퓨팅 리소스의 가용성과 표의 동시 접근성은 재압축 작업을 적시에 완료하는 데 중요한 역할을 합니다.

PLV8 확장 업그레이드 및 사용

PLV8은 PostgreSQL을 위한 신뢰할 수 있는 Javascript 언어 확장입니다. 저장 프로시저, 트리거 및 SQL에서 호출할 수 있는 기타 절차 코드에 사용할 수 있습니다. 이 언어 확장은 모든 최신 PostgreSQL 릴리스에서 지원됩니다.

PLV8을 사용하고 PostgreSQL을 새 PLV8 버전으로 업그레이드하는 경우 즉시 새로운 확장을 사용하세요. 다음 단계를 수행하여 카탈로그 메타데이터를 새 PLV8 버전과 동기화합니다. 이 단계는 선택 사항이지만 메타데이터 불일치 경고를 방지하려면 완료하는 것이 좋습니다.

업그레이드 프로세스는 기존 PLV8 기능을 모두 삭제합니다. 따라서 업그레이드하기 전에 RDS for PostgreSQL DB 인스턴스의 스냅샷을 생성하는 것이 좋습니다. 자세한 내용은 Amazon RDS의 단일 AZ DB 인스턴스에 대한 DB 스냅샷 생성 단원을 참조하십시오.

> **Important**
>
> PostgreSQL 버전 18부터 Amazon RDS for PostgreSQL은 `plcoffee` 및 `plls` PostgreSQL 확장을 더 이상 사용하지 않습니다. 향후 엔진 버전 업그레이드를 위한 업그레이드 경로를 확보하려면 애플리케이션에서 CoffeeScript 및 LiveScript 사용을 중지하는 것이 좋습니다.

새 PLV8 버전과 카탈로그 메타데이터 동기화

1. 업데이트해야 하는 것을 확인합니다. 이렇게 하려면 인스턴스에 연결된 동안 다음 명령을 실행합니다.

   ```
   SELECT * FROM pg_available_extensions WHERE name IN ('plv8','plls','plcoffee');
   ```

   결과에 기본 버전보다 낮은 설치 버전에 대한 값이 포함된 경우 이 절차를 계속 진행하여 확장을 업데이트합니다. 예를 들어, 다음 결과 집합은 업데이트해야 함을 나타냅니다.

   ```
   name    | default_version | installed_version |                    comment
   --------+-----------------+-------------------
   +-------------------------------------------------
   plls    | 2.1.0           | 1.5.3             | PL/LiveScript (v8) trusted
    procedural language
   plcoffee| 2.1.0           | 1.5.3             | PL/CoffeeScript (v8) trusted
    procedural language
   plv8    | 2.1.0           | 1.5.3             | PL/JavaScript (v8) trusted
    procedural language
   ```

```
(3 rows)
```

2. RDS for PostgreSQL DB 인스턴스의 스냅샷을 생성합니다. 이미 해당 스냅샷을 생성한 경우 이 단계를 건너뜁니다. 스냅샷이 생성되는 동안 다음 단계를 따라 계속 진행할 수 있습니다.
3. DB 인스턴스에 있는 PLV8 함수의 개수를 파악해야 업그레이드 이후 모두 존재하는지 확인할 수 있습니다. 예를 들어 다음 SQL 쿼리는 plv8, plcoffee 및 plls로 작성된 함수의 수를 반환합니다.

```
SELECT proname, nspname, lanname
FROM pg_proc p, pg_language l, pg_namespace n
WHERE p.prolang = l.oid
AND n.oid = p.pronamespace
AND lanname IN ('plv8','plcoffee','plls');
```

4. pg_dump를 사용하여 스키마 전용 덤프 파일을 생성합니다. 예를 들어, 클라이언트 시스템의 /tmp 디렉터리에 파일을 생성합니다.

```
./pg_dump -Fc --schema-only -U master postgres >/tmp/test.dmp
```

이 예에서는 다음 옵션을 사용합니다.

- -Fc - 사용자 지정 형식
- --schema-only - 스키마(이 경우에서는 함수)를 생성하는 데 필요한 명령만 덤프
- -U – RDS 기본 사용자 이름
- database – DB 인스턴스의 데이터베이스 이름

pg_dump에 대한 자세한 내용은 PostgreSQL 설명서의 pg_dump를 참조하세요.

5. 덤프 파일에 있는 'CREATE FUNCTION' DDL 문을 추출하십시오. 다음 예에서는 grep 명령을 사용하여 함수를 생성하는 DDL 문을 추출하여 파일에 저장합니다. 후속 단계에서 이를 사용하여 함수를 다시 생성합니다.

```
./pg_restore -l /tmp/test.dmp | grep FUNCTION > /tmp/function_list
```

pg_restore에 대한 자세한 내용은 PostgreSQL 설명서의 pg_restore를 참조하세요.

6. 함수 및 확장 기능을 중단합니다. 다음 예는 모든 PLV8 기반 객체를 중단합니다. 캐스케이드 옵션은 모든 종속이 중단되도록 합니다.

```
DROP EXTENSION plv8 CASCADE;
```

PostgreSQL 인스턴스에 plcoffee 또는 plls 기반 객체가 포함된 경우 이러한 확장 기능에 대해 이 단계를 반복합니다.

7. 확장 기능을 생성합니다. 다음 예에서는 plv8, plcoffee 및 plls 확장을 생성합니다.

```
CREATE EXTENSION plv8;
CREATE EXTENSION plcoffee;
CREATE EXTENSION plls;
```

8. 덤프 파일과 "드라이버" 파일을 사용하여 함수를 생성합니다.

   다음 예에서는 이전에 추출한 함수를 다시 생성합니다.

```
./pg_restore -U master -d postgres -Fc -L /tmp/function_list /tmp/test.dmp
```

9. 다음 쿼리를 사용하여 모든 함수가 다시 생성되었는지 확인합니다.

```
SELECT * FROM pg_available_extensions WHERE name IN ('plv8','plls','plcoffee');
```

   PLV8 버전 2는 결과 세트에 다음 추가 행을 추가합니다.

```
    proname    |  nspname   | lanname
---------------+------------+----------
 plv8_version  | pg_catalog | plv8
```

PL/Rust를 사용하여 Rust 언어로 PostgreSQL 함수 작성

PL/Rust는 PostgreSQL에 대한 신뢰할 수 있는 Rust 언어 확장입니다. 저장 프로시저, 함수 및 SQL에서 호출할 수 있는 기타 절차 코드에 사용할 수 있습니다. PL/Rust 언어 확장은 다음 버전에서 사용할 수 있습니다.

- RDS for PostgreSQL 17.1 이상의 17 버전
- RDS for PostgreSQL 16.1 이상의 16 버전
- RDS for PostgreSQL 15.2-R2 이상의 15 버전
- RDS for PostgreSQL 14.9 이상의 14 버전
- RDS for PostgreSQL 13.12 이상의 13 버전

자세한 내용은 GitHub의 PL/Rust를 참조하세요.

주제

- PL/Rust 설정
- PL/Rust를 사용하여 함수 생성
- PL/Rust가 있는 상자 사용
- PL/Rust 제한 사항

PL/Rust 설정

DB 인스턴스에 plrust 확장을 설치하려면 DB 인스턴스와 연결된 DB 파라미터 그룹의 `shared_preload_libraries` 파라미터에 plrust를 추가합니다. plrust 확장이 설치되어 있으면 함수를 만들 수 있습니다.

`shared_preload_libraries` 파라미터를 수정하려면 DB 인스턴스가 사용자 지정 파라미터 그룹과 연결되어 있어야 합니다. 사용자 지정 DB 파라미터 그룹 생성에 대한 자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하세요.

AWS Management Console 또는 AWS CLI를 사용하여 plrust 확장을 설치할 수 있습니다.

다음 단계에서는 DB 인스턴스가 사용자 지정 DB 파라미터 그룹에 연결되어 있다고 가정합니다.

콘솔

`shared_preload_libraries` 파라미터에 plrust 확장 설치

`rds_superuser` 그룹(역할)의 멤버인 계정을 사용하여 다음 단계를 완료합니다.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
3. DB 인스턴스의 이름을 선택하여 세부 정보를 표시합니다.
4. DB 인스턴스의 구성 탭을 열고 DB 인스턴스 파라미터 그룹 링크를 찾습니다.
5. 링크를 선택하여 DB 인스턴스와 연결된 사용자 지정 파라미터를 엽니다.
6. 파라미터 검색 필드에 `shared_pre`를 입력하여 **`shared_preload_libraries`** 파라미터를 찾습니다.

7. 파라미터 편집을 선택하여 속성 값에 액세스합니다.
8. 값 필드의 목록에 plrust를 추가합니다. 쉼표를 사용하여 값 목록에서 항목을 구분합니다.
9. shared_preload_libraries 파라미터 변경 사항을 적용하려면 DB 인스턴스를 재부팅합니다. 초기 재부팅을 완료하려면 추가 시간이 필요할 수 있습니다.
10. 인스턴스를 사용할 수 있게 되면 plrust가 초기화되었는지 확인합니다. psql을 사용하여 DB 인스턴스에 연결하고 다음 명령을 실행합니다.

```
SHOW shared_preload_libraries;
```

출력은 다음과 비슷한 형태가 됩니다.

```
shared_preload_libraries
--------------------------
rdsutils,plrust
(1 row)
```

AWS CLI

shared_preload_libraries 파라미터에 plrust 확장 설치

rds_superuser 그룹(역할)의 멤버인 계정을 사용하여 다음 단계를 완료합니다.

1. modify-db-parameter-group AWS CLI 명령을 사용하여 plrust를 shared_preload_libraries 파라미터에 추가합니다.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=plrust,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. reboot-db-instance AWS CLI 명령을 사용하여 DB 인스턴스를 재부팅하고 plrust 라이브러리를 초기화합니다. 초기 재부팅을 완료하려면 추가 시간이 필요할 수 있습니다.

```
aws rds reboot-db-instance \
    --db-instance-identifier your-instance \
    --region aws-region
```

3. 인스턴스를 사용할 수 있게 되면 plrust가 초기화되었는지 확인합니다. psql을 사용하여 DB 인스턴스에 연결하고 다음 명령을 실행합니다.

```
SHOW shared_preload_libraries;
```

출력은 다음과 비슷한 형태가 됩니다.

```
shared_preload_libraries
--------------------------
rdsutils,plrust
(1 row)
```

PL/Rust를 사용하여 함수 생성

PL/Rust는 함수를 동적 라이브러리로 컴파일하고 로드한 다음 실행합니다.

다음 Rust 함수는 배열에서 배수를 필터링합니다.

```
postgres=> CREATE LANGUAGE plrust;
CREATE EXTENSION
```

```
CREATE OR REPLACE FUNCTION filter_multiples(a BIGINT[], multiple BIGINT) RETURNS
 BIGINT[]
    IMMUTABLE STRICT
    LANGUAGE PLRUST AS
$$
    Ok(Some(a.into_iter().filter(|x| x.unwrap() % multiple != 0).collect()))
$$;

WITH gen_values AS (
SELECT ARRAY(SELECT * FROM generate_series(1,100)) as arr)
SELECT filter_multiples(arr, 3)
from gen_values;
```

PL/Rust가 있는 상자 사용

RDS for PostgreSQL 16.3-R2 이상 버전, 15.7-R2 이상 15 버전, 14.12-R2 이상 14 버전, 13.15-R2 이상 13 버전에서, PL/Rust는 추가 상자를 지원합니다.

- url

- `regex`
- `serde`
- `serde_json`

RDS for PostgreSQL 15.5-R2 이상 버전, 14.10-R2 이상 14 버전, 13.13-R2 이상 13 버전에서 RDS for PostgreSQL은 PL/Rust에서 두 개의 추가 크레이트를 지원합니다.

- `croaring-rs`
- `num-bigint`

Amazon RDS for PostgreSQL 버전 15.4, 14.9, 13.12부터 PL/Rust는 다음 상자를 지원합니다.

- `aes`
- `ctr`
- `rand`

이 상자에서는 기본 기능만 지원됩니다. 새 RDS for PostgreSQL 버전에는 업데이트된 버전의 상자가 포함될 수 있으며 기존 버전의 상자는 더 이상 지원되지 않을 수 있습니다.

PL/Rust 함수가 새 메이저 버전과 호환되는지 테스트하려면 메이저 버전 업그레이드 수행에 대한 모범 사례를 따르세요. 자세한 내용은 블로그 Amazon RDS를 PostgreSQL의 메이저 및 마이너 버전으로 업그레이드하기 위한 모범 사례 및 Amazon RDS 사용 설명서에 나온 Amazon RDS용 PostgreSQL DB 엔진 업그레이드를 참조하세요.

PL/Rust 함수를 생성할 때 종속성을 사용하는 예제는 종속성 사용에서 확인하세요.

PL/Rust 제한 사항

기본적으로 데이터베이스 사용자는 PL/Rust를 사용할 수 없습니다. PL/Rust에 대한 액세스를 제공하려면 rds_superuser 권한이 있는 사용자로 연결하고 다음 명령을 실행합니다.

```
postgres=> GRANT USAGE ON LANGUAGE PLRUST TO user;
```

PostGIS 확장을 사용하여 공간 데이터 관리

PostGIS는 공간 정보를 저장하고 관리하기 위해 PostgreSQL을 확장한 것입니다. PostGIS에 대한 자세한 내용은 PostGIS.net을 참조하세요.

버전 10.5부터 PostgreSQL은 맵 상자 벡터 타일 데이터 작업을 위해 PostGIS에서 사용하는 libprotobuf 1.3.0 라이브러리를 지원합니다.

PostGIS 확장을 설정하려면 `rds_superuser` 권한이 필요합니다. PostGIS 확장 프로그램 및 공간 데이터를 관리할 사용자(역할)를 생성하는 것이 좋습니다. PostGIS 확장 프로그램 및 관련 구성 요소는 PostgreSQL에 수천 개의 함수를 추가합니다. 사용 사례에 적합한 경우 자체 스키마에서 PostGIS 확장 프로그램을 만드는 것이 좋습니다. 다음 예제에서는 자체 데이터베이스에 확장 프로그램을 설치하는 방법을 보여주지만, 필수 사항은 아닙니다.

주제

- 1단계: PostGIS 확장을 관리할 사용자(역할) 생성
- 2단계: PostGIS 확장 모듈을 로드합니다.
- 3단계: 확장 스키마 소유권 이전
- 4단계: PostGIS 테이블 소유권 이전
- 5단계: 확장 모듈을 테스트합니다.
- 6단계: PostGIS 확장 업그레이드
- PostGIS 확장 버전
- PostGIS 2를 PostGIS 3으로 업그레이드

1단계: PostGIS 확장을 관리할 사용자(역할) 생성

먼저 `rds_superuser` 권한이 있는 사용자로 RDS for PostgreSQL DB 인스턴스에 연결합니다. 인스턴스를 설정할 때 기본 이름을 유지했다면 `postgres`로 연결합니다.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --username=postgres
 --password
```

PostGIS 확장을 관리하기 위해 별도의 역할(사용자)을 생성합니다.

```
postgres=>  CREATE ROLE gis_admin LOGIN PASSWORD 'change_me';
CREATE ROLE
```

역할이 확장을 설치하도록 허용하려면 이 역할 rds_superuser 권한을 부여합니다.

```
postgres=> GRANT rds_superuser TO gis_admin;
GRANT
```

PostGIS 아티팩트에 사용할 데이터베이스를 만듭니다. 이 단계는 선택 사항입니다. 또는 PostGIS 확장을 위해 사용자 데이터베이스에 스키마를 생성할 수 있지만 이 역시 필수 사항은 아닙니다.

```
postgres=> CREATE DATABASE lab_gis;
CREATE DATABASE
```

gis_admin에게 lab_gis 데이터베이스에 대한 모든 권한을 부여합니다.

```
postgres=> GRANT ALL PRIVILEGES ON DATABASE lab_gis TO gis_admin;
GRANT
```

세션을 종료하고 다음과 같이 RDS for PostgreSQL DB 인스턴스에 gis_admin으로 다시 연결합니다.

```
postgres=> psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=gis_admin --password --dbname=lab_gis
Password for user gis_admin:...
lab_gis=>
```

다음 단계에 설명된 대로 확장 프로그램을 계속 설정합니다.

2단계: PostGIS 확장 모듈을 로드합니다.

PostGIS 확장에는 지리 공간 기능을 제공하기 위해 함께 작동하는 여러 관련 확장이 포함되어 있습니다. 사용 사례에 따라 이 단계에서 만든 확장이 모두 필요하지 않을 수 있습니다.

CREATE EXTENSION 문을 사용하여 PostGIS 확장 모듈을 로드합니다.

```
CREATE EXTENSION postgis;
CREATE EXTENSION
CREATE EXTENSION postgis_raster;
CREATE EXTENSION
CREATE EXTENSION fuzzystrmatch;
CREATE EXTENSION
CREATE EXTENSION postgis_tiger_geocoder;
CREATE EXTENSION
```

```
CREATE EXTENSION postgis_topology;
CREATE EXTENSION
CREATE EXTENSION address_standardizer_data_us;
CREATE EXTENSION
```

다음 예에 표시된 확장 및 해당 소유자가 나열된 SQL 쿼리를 실행하여 결과를 확인할 수 있습니다.

```
SELECT n.nspname AS "Name",
  pg_catalog.pg_get_userbyid(n.nspowner) AS "Owner"
  FROM pg_catalog.pg_namespace n
  WHERE n.nspname !~ '^pg_' AND n.nspname <> 'information_schema'
  ORDER BY 1;
List of schemas
     Name     |   Owner
--------------+-----------
 public       | postgres
 tiger        | rdsadmin
 tiger_data   | rdsadmin
 topology     | rdsadmin
(4 rows)
```

3단계: 확장 스키마 소유권 이전

ALTER SCHEMA 문을 사용하여 스키마 소유권을 `gis_admin` 역할로 이전합니다.

```
ALTER SCHEMA tiger OWNER TO gis_admin;
ALTER SCHEMA
ALTER SCHEMA tiger_data OWNER TO gis_admin;
ALTER SCHEMA
ALTER SCHEMA topology OWNER TO gis_admin;
ALTER SCHEMA
```

다음 SQL 쿼리를 실행하여 소유권 변경을 확인할 수 있습니다. 또는 psql 명령줄에서 \dn 메타 명령을 사용할 수 있습니다.

```
SELECT n.nspname AS "Name",
  pg_catalog.pg_get_userbyid(n.nspowner) AS "Owner"
  FROM pg_catalog.pg_namespace n
  WHERE n.nspname !~ '^pg_' AND n.nspname <> 'information_schema'
  ORDER BY 1;
```

```
       List of schemas
     Name     |     Owner
--------------+---------------
 public       | postgres
 tiger        | gis_admin
 tiger_data   | gis_admin
 topology     | gis_admin
(4 rows)
```

4단계: PostGIS 테이블 소유권 이전

> **Note**
>
> PostGIS 함수의 소유권을 변경하지 마세요. PostGIS를 올바르게 운영하고 향후 업그레이드하려면 이러한 함수가 원래 소유권을 유지해야 합니다. PostGIS 권한에 대한 자세한 내용은 PostgreSQL 보안을 참조하세요.

다음 함수를 사용하여 PostGIS 테이블 소유권을 `gis_admin` 역할로 이전합니다. psql 프롬프트에서 다음 문을 실행하여 함수를 생성합니다.

```
CREATE FUNCTION exec(text) returns text language plpgsql volatile AS $f$ BEGIN EXECUTE
 $1; RETURN $1; END; $f$;
CREATE FUNCTION
```

그런 다음 쿼리로 `exec` 함수를 실행하면 함수가 해당하는 문을 실행하여 권한을 변경합니다.

```
SELECT exec('ALTER TABLE ' || quote_ident(s.nspname) || '.' || quote_ident(s.relname)
 || ' OWNER TO gis_admin;')
  FROM (
    SELECT nspname, relname
    FROM pg_class c JOIN pg_namespace n ON (c.relnamespace = n.oid)
    WHERE nspname in ('tiger','topology') AND
    relkind IN ('r','S','v') ORDER BY relkind = 'S')
s;
```

5단계: 확장 모듈을 테스트합니다.

스키마 이름을 지정할 필요가 없도록 하려면 다음 명령을 사용하여 검색 경로에 `tiger` 스키마를 추가하세요.

```
SET search_path=public,tiger;
SET
```

다음 SELECT 문을 사용하여 tiger 스키마를 테스트합니다.

```
SELECT address, streetname, streettypeabbrev, zip
 FROM normalize_address('1 Devonshire Place, Boston, MA 02109') AS na;
address | streetname | streettypeabbrev |  zip
---------+------------+------------------+-------
       1 | Devonshire | Pl               | 02109
(1 row)
```

이 확장에 대한 자세한 내용은 PostGIS 문서에서 Tiger Geocoder를 참조하세요.

다음 SELECT 문을 사용하여 topology 스키마를 테스트합니다. 이렇게 하면 지정된 공간 참조 식별자(26986) 및 기본 허용 오차(0.5)를 사용하여 새 토폴로지 개체(my_new_topo)를 등록하는 createtopology 함수를 호출합니다. 자세한 내용은 PostgreSQL의 CreateTopology 문서를 참조하세요.

```
SELECT topology.createtopology('my_new_topo',26986,0.5);
 createtopology
----------------
              1
(1 row)
```

6단계: PostGIS 확장 업그레이드

PostgreSQL의 각 새 릴리스는 해당 릴리스와 호환되는 하나 이상의 PostGIS 확장 버전을 지원합니다. PostgreSQL 엔진을 새 버전으로 업그레이드해도 PostGIS 확장은 자동으로 업그레이드되지 않습니다. PostgreSQL 엔진을 업그레이드하려면 일반적으로 PostGIS를 현재 PostgreSQL 버전에서 사용 가능한 최신 버전으로 업그레이드를 먼저 해야 합니다. 자세한 내용은 PostGIS 확장 버전 단원을 참조하세요.

PostgreSQL 엔진 업그레이드 후 PostGIS 확장을 새로 업그레이드된 PostgreSQL 엔진 버전에 대해 지원되는 버전으로 다시 업그레이드합니다. PostgreSQL 엔진 업그레이드에 대한 자세한 내용은 RDS for PostgreSQL에 대한 메이저 버전 업그레이드를 수행하는 방법 단원을 따라가세요.

RDS for PostgreSQL DB 인스턴스에서 사용 가능한 PostGIS 확장 버전 업데이트를 언제든지 확인할 수 있습니다. 이렇게 하려면 다음 명령을 실행합니다. 이 기능은 PostGIS 2.5.0 이상 버전에서 사용할 수 있습니다.

```
SELECT postGIS_extensions_upgrade();
```

애플리케이션이 최신 PostGIS 버전을 지원하지 않는 경우에도 다음과 같이 사용 중인 메이저 버전에서 사용할 수 있는 이전 버전의 PostGIS를 설치할 수 있습니다.

```
CREATE EXTENSION postgis VERSION "2.5.5";
```

이전 버전에서 특정 PostGIS 버전으로 업그레이드하려는 경우 다음 명령을 사용할 수도 있습니다.

```
ALTER EXTENSION postgis UPDATE TO "2.5.5";
```

업그레이드하는 버전에 따라 이 기능을 다시 사용해야 할 수도 있습니다. 기능의 첫 번째 실행 결과에 따라 추가 업그레이드 기능이 필요한지 여부가 결정됩니다. PostGIS 2에서 PostGIS 3으로 업그레이드하는 경우를 예로 들 수 있습니다. 자세한 내용은 PostGIS 2를 PostGIS 3으로 업그레이드을 참조하세요.

PostgreSQL 엔진의 메이저 버전 업그레이드를 준비하기 위해 이 확장을 업그레이드했다면 다른 예비 작업을 계속할 수 있습니다. 자세한 내용은 RDS for PostgreSQL에 대한 메이저 버전 업그레이드를 수행하는 방법 단원을 참조하세요.

PostGIS 확장 버전

Aurora PostgreSQL 릴리스 정보의 Amazon RDS for PostgreSQL 릴리스 정보의 Extension versions for Amazon RDS for PostgreSQL의 확장 버전 릴리스에서 사용 가능한 버전 목록을 얻으려면 다음 명령을 사용하세요.

```
SELECT * FROM pg_available_extension_versions WHERE name='postgis';
```

Amazon RDS for PostgreSQL 릴리스 정보의 다음 단원에서 버전 정보를 확인할 수 있습니다.

- Amazon RDS에서 PostgreSQL 버전 16 확장 지원
- Amazon RDS에서 지원되는 PostgreSQL 버전 15 확장
- Amazon RDS에서 지원되는 PostgreSQL 버전 14 확장
- Amazon RDS에서 지원되는 PostgreSQL 버전 13 확장
- Amazon RDS에서 지원되는 PostgreSQL 버전 12 확장
- Amazon RDS에서 지원되는 PostgreSQL 버전 11 확장
- Amazon RDS에서 지원되는 PostgreSQL 버전 10 확장

- Amazon RDS에서 지원되는 PostgreSQL 버전 9.6.x 확장

PostGIS 2를 PostGIS 3으로 업그레이드

버전 3.0부터 PostGIS 래스터 기능이 이제 별도의 확장인 postgis_raster입니다. 이 확장에는 자체 설치 및 업그레이드 경로가 있습니다. 이러한 경로를 통해 코어에서 핵심 postgis 확장에서 래스터 이미지 처리에 필요한 수십 가지 기능, 데이터 유형 및 기타 아티팩트를 제거합니다. 즉, 사용 사례에 래스터 처리가 필요하지 않은 경우 postgis_raster 확장을 설치할 필요가 없습니다.

다음 업그레이드 예에서 첫 번째 업그레이드 명령은 래스터 기능을 postgis_raster 확장으로 추출합니다. 그런 다음 postgis_raster를 새 버전으로 업그레이드하려면 두 번째 업그레이드 명령이 필요합니다.

PostGIS 2에서 PostGIS 3으로 업그레이드하려면

1. 의 PostgreSQL 버전에 사용할 수 있는 PostGIS의 기본 버전을 식별합니다. RDS for PostgreSQL DB 인스턴스 이렇게 하려면 다음 쿼리를 실행합니다.

```
SELECT * FROM pg_available_extensions
    WHERE default_version > installed_version;
  name   | default_version | installed_version |                                comment
---------+-----------------+-------------------
+-------------------------------------------------------------
 postgis | 3.1.4           | 2.3.7             | PostGIS geometry and geography
 spatial types and functions
(1 row)
```

2. 에서 각 데이터베이스에 설치된 PostGIS 버전을 식별합니다. 사용자의 RDS for PostgreSQL DB 인스턴트. 즉, 다음과 같이 각 사용자 데이터베이스를 쿼리합니다.

```
SELECT
    e.extname AS "Name",
    e.extversion AS "Version",
    n.nspname AS "Schema",
    c.description AS "Description"
FROM
    pg_catalog.pg_extension e
    LEFT JOIN pg_catalog.pg_namespace n ON n.oid = e.extnamespace
    LEFT JOIN pg_catalog.pg_description c ON c.objoid = e.oid
    AND c.classoid = 'pg_catalog.pg_extension'::pg_catalog.regclass
WHERE
```

```
    e.extname LIKE '%postgis%'
ORDER BY
    1;
  Name   | Version | Schema |                                Description
---------+---------+--------
+------------------------------------------------------------------------
 postgis | 2.3.7   | public | PostGIS geometry, geography, and raster spatial types
 and functions
(1 row)
```

기본 버전(PostGIS 3.1.4)과 설치된 버전(PostGIS 2.3.7) 간의 이러한 불일치는 PostGIS 확장을 업그레이드해야 함을 의미합니다.

```
ALTER EXTENSION postgis UPDATE;
ALTER EXTENSION
WARNING: unpackaging raster
WARNING: PostGIS Raster functionality has been unpackaged
```

3. 다음 쿼리를 실행하여 래스터 기능이 이제 자체 패키지에 포함되어 있는지 확인합니다.

```
SELECT
    probin,
    count(*)
FROM
    pg_proc
WHERE
    probin LIKE '%postgis%'
GROUP BY
    probin;
          probin          | count
--------------------------+-------
 $libdir/rtpostgis-2.3    | 107
 $libdir/postgis-3        | 487
(2 rows)
```

출력 결과를 통해 여전히 버전 간에 여전히 차이가 있음을 알 수 있습니다. PostGIS 함수는 버전 3(postgis-3)이고 래스터 함수(rtpostgis)는 버전 2(rtpostgis-2.3)입니다. 업그레이드를 완료하려면 다음과 같이 upgrade 명령을 다시 실행합니다.

```
postgres=> SELECT postgis_extensions_upgrade();
```

경고 메시지는 무시해도 됩니다. 다음 쿼리를 다시 실행하여 업그레이드가 완료되었는지 확인합니다. PostGIS 및 모든 관련 확장 프로그램이 업그레이드 필요 항목으로 표시되지 않으면 업그레이드가 완료된 것입니다.

```
SELECT postgis_full_version();
```

4. 다음 쿼리를 사용하여 완료된 업그레이드 프로세스와 별도로 패키징된 확장을 확인하고 해당 버전이 일치하는지 확인합니다.

```
SELECT
    e.extname AS "Name",
    e.extversion AS "Version",
    n.nspname AS "Schema",
    c.description AS "Description"
FROM
    pg_catalog.pg_extension e
    LEFT JOIN pg_catalog.pg_namespace n ON n.oid = e.extnamespace
    LEFT JOIN pg_catalog.pg_description c ON c.objoid = e.oid
        AND c.classoid = 'pg_catalog.pg_extension'::pg_catalog.regclass
WHERE
    e.extname LIKE '%postgis%'
ORDER BY
    1;
      Name      | Version | Schema |                            Description
----------------+---------+--------
+------------------------------------------------------------------------
 postgis        | 3.1.5   | public | PostGIS geometry, geography, and raster
 spatial types and functions
 postgis_raster | 3.1.5   | public | PostGIS raster types and functions
(2 rows)
```

출력은 PostGIS 2 확장이 PostGIS 3으로 업그레이드되었으며 `postgis`와 현재 별도의 `postgis_raster` 확장이 모두 버전 3.1.5임을 보여줍니다.

이 업그레이드가 완료된 후 래스터 기능을 사용하지 않으려는 경우 다음과 같이 확장을 삭제할 수 있습니다.

```
DROP EXTENSION postgis_raster;
```

Amazon RDS for PostgreSQL용 지원되는 외부 데이터 래퍼 작업

외부 데이터 래퍼(FDW)는 외부 데이터에 대한 액세스를 제공하는 특정 유형의 확장입니다. 예를 들어 `oracle_fdw` 확장을 사용하면 RDS for PostgreSQL DB 클러스터가 Oracle 데이터베이스와 함께 작동할 수 있습니다. 또 다른 예로, PostgreSQL 네이티브 `postgres_fdw` 확장을 사용하면 RDS for PostgreSQL DB 인스턴스 외부에서 PostgreSQL DB 인스턴스에 저장된 데이터에 액세스할 수 있습니다.

아래에서는 여러 지원되는 PostgreSQL 외부 데이터 래퍼에 관한 정보를 확인할 수 있습니다.

주제

- log_fdw 확장으로 SQL을 사용하여 DB 로그에 액세스
- postgres_fdw 확장을 사용하여 외부 데이터 액세스
- mysql_fdw 확장을 사용하여 MySQL 데이터베이스 작업
- oracle_fdw 확장을 사용하여 Oracle 데이터베이스 작업
- tds_fdw 확장을 사용하여 SQL Server 데이터베이스 작업

log_fdw 확장으로 SQL을 사용하여 DB 로그에 액세스

RDS for PostgreSQL DB 인스턴스는 SQL 인터페이스를 사용하여 데이터베이스 엔진 로그에 액세스할 수 있는 `log_fdw` 확장을 지원합니다. `log_fdw` 확장은 데이터베이스 로그용 외부 테이블을 간편하게 생성할 수 있게 해주는 2가지 함수를 제공합니다.

- `list_postgres_log_files` – 데이터베이스 로그 디렉터리의 파일과 파일 크기(단위: 바이트)를 나열합니다.
- `create_foreign_table_for_log_file(table_name text, server_name text, log_file_name text)` – 현재 데이터베이스에서 지정된 파일에 대해 외부 테이블을 빌드합니다.

`log_fdw`가 생성하는 모든 함수는 `rds_superuser`가 소유합니다. `rds_superuser` 역할의 구성원은 다른 데이터베이스 사용자에게 이러한 함수에 대한 액세스 권한을 부여할 수 있습니다.

기본적으로 로그 파일은 `log_destination` 파라미터에 명시된 대로 Amazon RDS에 의해 `stderr`(표준 오류) 형식으로 생성됩니다. 이 파라미터에는 두 가지 옵션만 있습니다. `stderr` 및 `csvlog`(쉼표로 구분된 값, CSV)입니다. 파라미터에 `csvlog` 옵션을 추가하는 경우 Amazon RDS가 `stderr` 및 `csvlog` 로그를 모두 생성합니다. 이는 DB 클러스터의 스토리지 용량에 영향을 줄 수 있

으므로 로그 처리에 영향을 주는 다른 파라미터를 알고 있어야 합니다. 자세한 내용은 로그 대상 설정 (stderr, csvlog) 단원을 참조하십시오.

csvlog 로그를 생성할 때의 한 가지 이점은 log_fdw 확장을 사용하여 여러 열로 깔끔하게 분할된 데이터로 외부 테이블을 작성할 수 있다는 점입니다. 그러려면 인스턴스를 사용자 지정 DB 파라미터 그룹과 연결해야 log_destination에 대한 설정을 변경할 수 있습니다. 이에 관한 정보는 RDS for PostgreSQL DB 인스턴스에 파라미터로 작업 단원을 참조하세요.

다음 예제에서는 log_destination 파라미터에 cvslog가 포함된 것으로 간주합니다.

log_fdw 확장을 사용하려면

1. log_fdw 확장을 설치합니다.

```
postgres=> CREATE EXTENSION log_fdw;
CREATE EXTENSION
```

2. 로그 서버를 외부 데이터 래퍼로 생성합니다.

```
postgres=> CREATE SERVER log_server FOREIGN DATA WRAPPER log_fdw;
CREATE SERVER
```

3. 로그 파일 목록에서 모든 파일을 선택합니다.

```
postgres=> SELECT * FROM list_postgres_log_files() ORDER BY 1;
```

   샘플 응답은 다음과 같습니다.

```
          file_name           | file_size_bytes
------------------------------+-----------------
 postgresql.log.2023-08-09-22.csv |            1111
 postgresql.log.2023-08-09-23.csv |            1172
 postgresql.log.2023-08-10-00.csv |            1744
 postgresql.log.2023-08-10-01.csv |            1102
(4 rows)
```

4. 선택한 파일에 대해 단일 'log_entry' 열이 있는 테이블을 생성합니다.

```
postgres=> SELECT create_foreign_table_for_log_file('my_postgres_error_log',
     'log_server', 'postgresql.log.2023-08-09-22.csv');
```

응답은 테이블이 현재 존재한다는 것 외에는 세부 정보를 제공하지 않습니다.

```
-----------------------------------
(1 row)
```

5. 로그 파일의 샘플을 선택합니다. 다음 코드는 로그 시간 및 오류 메시지 설명을 검색합니다.

```
postgres=> SELECT log_time, message FROM my_postgres_error_log ORDER BY 1;
```

샘플 응답은 다음과 같습니다.

```
            log_time             |                                 message
---------------------------------
+------------------------------------------------------------------------------
Tue Aug 09 15:45:18.172 2023 PDT | ending log output to stderr
Tue Aug 09 15:45:18.175 2023 PDT | database system was interrupted; last known up
 at 2023-08-09 22:43:34 UTC
Tue Aug 09 15:45:18.223 2023 PDT | checkpoint record is at 0/90002E0
Tue Aug 09 15:45:18.223 2023 PDT | redo record is at 0/90002A8; shutdown FALSE
Tue Aug 09 15:45:18.223 2023 PDT | next transaction ID: 0/1879; next OID: 24578
Tue Aug 09 15:45:18.223 2023 PDT | next MultiXactId: 1; next MultiXactOffset: 0
Tue Aug 09 15:45:18.223 2023 PDT | oldest unfrozen transaction ID: 1822, in
 database 1
(7 rows)
```

postgres_fdw 확장을 사용하여 외부 데이터 액세스

postgres_fdw 확장으로 원격 데이터베이스에 있는 테이블의 데이터에 액세스할 수 있습니다. PostgreSQL DB 인스턴스에서 원격 연결을 설정하는 경우 읽기 전용 복제본에도 액세스할 수 있습니다.

postgres_fdw로 원격 데이터베이스 서버에 액세스하려면

1. postgres_fdw 확장을 설치합니다.

```
CREATE EXTENSION postgres_fdw;
```

2. CREATE SERVER로 외부 데이터 서버를 생성합니다.

```
CREATE SERVER foreign_server
FOREIGN DATA WRAPPER postgres_fdw
OPTIONS (host 'xxx.xx.xxx.xx', port '5432', dbname 'foreign_db');
```

3. 원격 서버에 사용할 역할 식별을 위하여 사용자 매핑을 생성합니다.

> **Important**
>
> 로그에 표시되지 않도록 암호를 수정하려면 세션 수준에서 `log_statement=none`을 설정합니다. 파라미터 수준에서 설정하면 암호가 수정되지 않습니다.

```
CREATE USER MAPPING FOR local_user
SERVER foreign_server
OPTIONS (user 'foreign_user', password 'password');
```

4. 원격 서버에서 테이블을 매핑할 테이블을 생성합니다.

```
CREATE FOREIGN TABLE foreign_table (
        id integer NOT NULL,
        data text)
SERVER foreign_server
OPTIONS (schema_name 'some_schema', table_name 'some_table');
```

mysql_fdw 확장을 사용하여 MySQL 데이터베이스 작업

RDS for PostgreSQL DB 인스턴스에서 MySQL 호환 데이터베이스에 액세스하려면 `mysql_fdw` 확장을 설치하고 사용하면 됩니다. 이 외부 데이터 래퍼를 사용하면 RDS for MySQL, Aurora MySQL, MariaDB 및 기타 MySQL 호환 데이터베이스로 작업할 수 있습니다. RDS for PostgreSQL DB 인스턴스에서 MySQL 데이터베이스로의 연결은 클라이언트 및 서버 구성에 따라 최선의 방식으로 암호화됩니다. 그러나 원하는 경우 암호화를 적용할 수 있습니다. 자세한 내용은 확장에서 전송 중 암호화 사용 단원을 참조하십시오.

`mysql_fdw` 확장은 Amazon RDS for PostgreSQL 버전 14.2, 13.6 이상에서 지원됩니다. RDS for PostgreSQL DB에서 MySQL 호환 데이터베이스 인스턴스의 테이블에 대한 선택, 삽입, 업데이트 및 삭제를 지원합니다.

주제

- mysql_fdw 확장을 사용하도록 RDS for PostgreSQL DB 설정
- 예: RDS for PostgreSQL에서 RDS for MySQL 데이터베이스로 작업
- 확장에서 전송 중 암호화 사용

mysql_fdw 확장을 사용하도록 RDS for PostgreSQL DB 설정

RDS for PostgreSQL DB 인스턴스에서 `mysql_fdw` 확장을 설정하려면 DB 인스턴스에서 확장을 로드한 다음 MySQL DB 인스턴스에 대한 연결 지점을 생성해야 합니다. 이 작업을 수행하려면 MySQL DB 인스턴스에 대한 다음과 같은 세부 정보가 필요합니다.

- 호스트 이름 또는 엔드포인트. RDS for MySQL DB 인스턴스의 경우 콘솔을 사용하여 엔드포인트를 찾을 수 있습니다. 연결 및 보안(Connectivity & security) 탭을 선택하고 '엔드포인트 및 포트(Endpoint and port)' 섹션을 살펴봅니다.
- 포트 번호. MySQL의 기본 포트 번호는 3306입니다.
- 데이터베이스 이름 DB 식별자입니다.

또한 MySQL 포트 3306의 액세스 제어 목록(ACL) 또는 보안 그룹에 대한 액세스를 제공해야 합니다. RDS for PostgreSQL DB 인스턴스와 RDS for MySQL DB 인스턴스에 포트 3306에 대한 액세스 권한이 있어야 합니다. 액세스가 올바르게 구성되지 않은 경우 MySQL 호환 테이블에 연결하려고 하면 다음과 비슷한 오류 메시지가 표시됩니다.

```
ERROR: failed to connect to MySQL: Can't connect to MySQL server on 'hostname.aws-
region.rds.amazonaws.com:3306' (110)
```

다음 절차에서는 `rds_superuser` 계정으로 외부 서버를 생성합니다. 그런 다음 특정 사용자에게 외부 서버에 대한 액세스 권한을 부여합니다. 그러면 이러한 사용자가 MySQL DB 인스턴스로 작업하기 위해 적절한 MySQL 사용자 계정에 대한 자체 매핑을 생성합니다.

mysql_fdw로 MySQL 데이터베이스 서버에 액세스하려면

1. `rds_superuser` 역할이 있는 계정을 사용하여 PostgreSQL DB 인스턴스에 연결합니다. RDS for PostgreSQL DB 인스턴스를 생성할 때 기본값을 수락한 경우 사용자 이름은 `postgres`이고 다음과 같이 `psql` 명령줄 도구를 사용하여 연결할 수 있습니다.

   ```
   psql --host=your-DB-instance.aws-region.rds.amazonaws.com --port=5432 --
   username=postgres --password
   ```

2. 다음과 같이 mysql_fdw 확장을 설치합니다.

```
postgres=> CREATE EXTENSION mysql_fdw;
CREATE EXTENSION
```

RDS for PostgreSQL DB 인스턴스에 확장을 설치한 후 MySQL 데이터베이스에 대한 연결을 제공하는 외부 서버를 설정합니다.

외부 서버를 생성하려면

RDS for PostgreSQL DB 인스턴스에서 작업을 수행합니다. 이러한 단계에서는 postgres와 같은 rds_superuser 권한이 있는 사용자로 연결되어 있다고 가정합니다.

1. RDS for PostgreSQL DB 인스턴스에서 외부 서버를 생성합니다.

```
postgres=> CREATE SERVER mysql-db FOREIGN DATA WRAPPER mysql_fdw OPTIONS (host 'db-
name.111122223333.aws-region.rds.amazonaws.com', port '3306');
CREATE SERVER
```

2. 적절한 사용자에게 외부 서버에 대한 액세스 권한을 부여합니다. 관리자가 아닌 사용자, 즉 rds_superuser 역할이 없는 사용자여야 합니다.

```
postgres=> GRANT USAGE ON FOREIGN SERVER mysql-db to user1;
GRANT
```

PostgreSQL 사용자는 외부 서버를 통해 MySQL 데이터베이스에 대한 자체 연결을 생성하고 관리합니다.

예: RDS for PostgreSQL에서 RDS for MySQL 데이터베이스로 작업

RDS for PostgreSQL DB 인스턴스에 간단한 테이블이 있다고 가정합니다. RDS for PostgreSQL 사용자는 해당 테이블에 대해 (SELECT), INSERT, UPDATE, DELETE 항목을 쿼리하려고 합니다. 앞의 절차에서 설명한 대로 RDS for PostgreSQL DB 인스턴스에 mysql_fdw 확장이 생성되었다고 가정합니다. rds_superuser 권한이 있는 사용자로 RDS for PostgreSQL DB 인스턴스에 연결한 후 다음 단계를 진행할 수 있습니다.

1. RDS for PostgreSQL DB 인스턴스에서 외부 서버를 생성합니다.

```
test=> CREATE SERVER mysqldb FOREIGN DATA WRAPPER mysql_fdw OPTIONS (host 'your-
DB.aws-region.rds.amazonaws.com', port '3306');
CREATE SERVER
```

2. rds_superuser 권한이 없는 사용자(예: user1)에게 사용 권한을 부여합니다.

```
test=> GRANT USAGE ON FOREIGN SERVER mysqldb TO user1;
GRANT
```

3. *user1*으로 연결한 다음 MySQL 사용자에 대한 매핑을 생성합니다.

```
test=> CREATE USER MAPPING FOR user1 SERVER mysqldb OPTIONS (username 'myuser',
 password 'mypassword');
CREATE USER MAPPING
```

4. MySQL 테이블에 연결된 외부 테이블을 만듭니다.

```
test=> CREATE FOREIGN TABLE mytab (a int, b text) SERVER mysqldb OPTIONS (dbname
 'test', table_name '');
CREATE FOREIGN TABLE
```

5. 외부 테이블에 대한 간단한 쿼리를 실행합니다.

```
test=> SELECT * FROM mytab;
a |   b
---+-------
1 | apple
(1 row)
```

6. MySQL 테이블에서 데이터를 추가, 변경 및 제거할 수 있습니다. 예를 들면 다음과 같습니다.

```
test=> INSERT INTO mytab values (2, 'mango');
INSERT 0 1
```

SELECT 쿼리를 다시 실행하여 결과를 확인합니다.

```
test=> SELECT * FROM mytab ORDER BY 1;
 a |   b
---+-------
1 | apple
2 | mango
```

```
(2 rows)
```

확장에서 전송 중 암호화 사용

RDS for PostgreSQL에서 MySQL에 대한 연결은 기본적으로 전송 중 암호화(TLS/SSL)를 사용합니다. 그러나 클라이언트와 서버 구성이 다를 경우 연결이 암호화되지 않은 상태로 폴백됩니다. RDS for MySQL 사용자 계정에서 `REQUIRE SSL` 옵션을 지정하여 나가는 모든 연결에 대해 암호화를 적용할 수 있습니다. 이와 동일한 접근 방식이 MariaDB 및 Aurora MySQL 사용자 계정에서도 작동합니다.

`REQUIRE SSL`로 구성된 MySQL 사용자 계정의 경우 보안 연결을 설정할 수 없으면 연결 시도가 실패합니다.

기존 MySQL 데이터베이스 사용자 계정에 암호화를 적용하려면 `ALTER USER` 명령을 사용할 수 있습니다. 구문은 다음 표에 나온 대로 MySQL 버전에 따라 다릅니다. 자세한 내용은 MySQL 참조 매뉴얼의 ALTER USER를 참조하세요.

MySQL 5.7, MySQL 8.0	MySQL 5.6
`ALTER USER 'user'@'%' REQUIRE SSL;`	`GRANT USAGE ON *.* to 'user'@'%' REQUIRE SSL;`

`mysql_fdw` 확장에 대한 자세한 내용은 mysql_fdw 설명서를 참조하세요.

oracle_fdw 확장을 사용하여 Oracle 데이터베이스 작업

RDS for PostgreSQL DB 인스턴스에서 Oracle 데이터베이스에 액세스하려면 `oracle_fdw` 확장을 설치하고 사용할 수 있습니다. 이 확장은 Oracle 데이터베이스의 외부 데이터 래퍼입니다. 이 확장 프로그램에 대해 자세히 알아보려면 oracle_fdw 문서를 참조하세요.

`oracle_fdw` 확장은 RDS for PostgreSQL 버전 12.7 및 13.3 이상에서 지원됩니다.

주제

- oracle_fdw 확장 켜기
- 예제: Amazon RDS for Oracle Database에 연결된 외부 서버 사용
- 전송 중 암호화 작업
- pg_user_mappings 보기 및 권한 이해

oracle_fdw 확장 켜기

oracle_fdw 확장을 사용하려면 다음 절차를 수행하십시오.

oracle_fdw 확장을 켜려면

- `rds_superuser` 권한이 있는 계정을 사용하여 다음 명령을 실행합니다.

```
CREATE EXTENSION oracle_fdw;
```

예제: Amazon RDS for Oracle Database에 연결된 외부 서버 사용

다음 예에서는 Amazon RDS for Oracle Database에 연결된 외부 서버를 사용하는 방법을 보여줍니다.

RDS for Oracle Database에 연결된 외부 서버를 만들려면

1. RDS for Oracle DB 인스턴스에서는 다음 사항에 유의하세요.
   - Endpoint
   - 포트
   - 데이터베이스 이름
2. 외부 서버를 만듭니다.

```
test=> CREATE SERVER oradb FOREIGN DATA WRAPPER oracle_fdw OPTIONS (dbserver
 '//endpoint:port/DB_name');
CREATE SERVER
```

3. `rds_superuser` 권한이 없는 사용자(예: `user1`)에게 사용 권한을 부여합니다.

```
test=> GRANT USAGE ON FOREIGN SERVER oradb TO user1;
GRANT
```

4. `user1`로 연결하고 Oracle 사용자에 대한 매핑을 생성합니다.

```
test=> CREATE USER MAPPING FOR user1 SERVER oradb OPTIONS (user 'oracleuser',
 password 'mypassword');
CREATE USER MAPPING
```

5. Oracle 테이블에 연결된 외부 테이블을 만듭니다.

```
test=> CREATE FOREIGN TABLE mytab (a int) SERVER oradb OPTIONS (table 'MYTABLE');
CREATE FOREIGN TABLE
```

6. 외부 테이블을 쿼리합니다.

```
test=>  SELECT * FROM mytab;
a
---
1
(1 row)
```

쿼리에서 다음 오류를 보고하면 보안 그룹 및 액세스 제어 목록(ACL)을 확인하여 두 인스턴스가 모두 통신할 수 있는지 확인합니다.

```
ERROR: connection for foreign table "mytab" cannot be established
DETAIL: ORA-12170: TNS:Connect timeout occurred
```

전송 중 암호화 작업

PostgreSQL에서 Oracle 간의 전송 중인 암호화는 클라이언트와 서버 구성 파라미터의 조합을 기반으로 합니다. Oracle 21c를 사용하는 예제는 Oracle 문서의 About the Values for Negotiating Encryption and Integrity를 참조하세요. Amazon RDS에서 oracle_fdw에 사용되는 클라이언트는 `ACCEPTED`로 구성됩니다. 즉, Oracle 데이터베이스 서버 구성에 따라 암호화가 달라지며, 암호화에 Oracle Security Library(libnnz)를 사용한다는 의미입니다.

데이터베이스가 RDS for Oracle에 있는 경우 암호화를 구성하려면 Oracle 기본 네트워크 암호화를 참조하세요.

pg_user_mappings 보기 및 권한 이해

PostgreSQL 카탈로그 `pg_user_mapping`에서 RDS for PostgreSQL 사용자의 매핑을 외부 데이터(원격) 서버의 사용자에게 저장합니다. 카탈로그에 대한 액세스는 제한되어 있지만 `pg_user_mappings` 보기를 사용하여 매핑을 확인합니다. 다음에서는 예제 Oracle 데이터베이스를 통해 권한이 적용되는 방법을 보여 주는 예제를 확인할 수 있지만, 이 정보는 다른 외부 데이터 래퍼에도 일반적으로 적용됩니다.

다음 결과에서는 세 가지 예제 사용자에 매핑된 역할 및 권한을 찾을 수 있습니다. 사용자 `rdssu1`과 `rdssu2`는 `rds_superuser` 역할이며 `user1`은 아닙니다. 다음 예제에서는 `psql` 메타 명령 `\du`를 사용하여 기존 역할을 나열합니다.

```
test=>  \du
                                                                List of roles
    Role name    |                          Attributes                          |
                  Member of
-----------------+-------------------------------------------------------------
+-------------------------------------------------------------
 rdssu1          |                                                              |
 {rds_superuser}
 rdssu2          |                                                              |
 {rds_superuser}
 user1           |                                                              | {}
```

rds_superuser 권한이 있는 사용자를 포함한 모든 사용자는 pg_user_mappings 테이블에서 자신의 사용자 매핑(umoptions)을 볼 수 있습니다. 다음 예제와 같이 rdssu1이 모든 사용자 매핑을 얻으려고 하면 rdssu1rds_superuser 권한에도 불구하고 오류가 발생합니다.

```
test=> SELECT * FROM pg_user_mapping;
ERROR: permission denied for table pg_user_mapping
```

다음은 일부 예입니다.

```
test=> SET SESSION AUTHORIZATION rdssu1;
SET
test=> SELECT * FROM pg_user_mappings;
 umid  | srvid | srvname | umuser | usename    |            umoptions
-------+-------+---------+--------+------------+----------------------------------
 16414 | 16411 | oradb   |  16412 | user1      |
 16423 | 16411 | oradb   |  16421 | rdssu1     | {user=oracleuser,password=mypwd}
 16424 | 16411 | oradb   |  16422 | rdssu2     |
 (3 rows)

test=> SET SESSION AUTHORIZATION rdssu2;
SET
test=> SELECT * FROM pg_user_mappings;
 umid  | srvid | srvname | umuser | usename    |            umoptions
-------+-------+---------+--------+------------+----------------------------------
 16414 | 16411 | oradb   |  16412 | user1      |
 16423 | 16411 | oradb   |  16421 | rdssu1     |
 16424 | 16411 | oradb   |  16422 | rdssu2     | {user=oracleuser,password=mypwd}
 (3 rows)

test=> SET SESSION AUTHORIZATION user1;
```

```
SET
test=> SELECT * FROM pg_user_mappings;
 umid  | srvid | srvname | umuser | usename    |           umoptions
-------+-------+---------+--------+------------+-------------------------------
 16414 | 16411 | oradb   |  16412 | user1      | {user=oracleuser,password=mypwd}
 16423 | 16411 | oradb   |  16421 | rdssu1     |
 16424 | 16411 | oradb   |  16422 | rdssu2     |
 (3 rows)
```

information_schema._pg_user_mappings 및 pg_catalog.pg_user_mappings의 구현 차이점 때문에 수동으로 생성된 rds_superuser는 pg_catalog.pg_user_mappings에서 암호를 보려면 추가 권한이 필요합니다.

information_schema._pg_user_mappings에서 암호를 보려면 rds_superuser에 대한 추가 권한은 필요하지 않습니다.

rds_superuser 역할이 없는 사용자는 다음 조건에서만 pg_user_mappings에서 암호를 볼 수 있습니다.

- 현재 사용자는 매핑되는 사용자이며 서버를 소유하거나 서버에 대한 USAGE 권한을 보유하고 있습니다.
- 현재 사용자는 서버 소유자이고 매핑은 PUBLIC에 대한 것입니다.

tds_fdw 확장을 사용하여 SQL Server 데이터베이스 작업

PostgreSQL tds_fdw 확장을 사용하여 Sybase 및 Microsoft SQL Server 데이터베이스와 같은 테이블 형식 데이터 스트림(TDS) 프로토콜을 지원하는 데이터베이스에 액세스할 수 있습니다. 이 외부 데이터 래퍼를 사용하면 RDS for PostgreSQL DB 인스턴스 에서 Amazon RDS for Microsoft SQL Server를 포함하여 TDS 프로토콜을 사용하는 데이터베이스에 연결할 수 있습니다. 자세한 내용은 GitHub에서 tds-fdw/tds_fdw 설명서를 참조하세요.

tds_fdw 확장은 Amazon RDS for PostgreSQL 버전 14.2 및 13.6 이상 릴리스에서 지원됩니다.

tds_fdw 확장을 사용하도록 Aurora PostgreSQL DB 설정

다음 절차에서는 RDS for PostgreSQL DB 인스턴스와 함께 tds_fdw를 설정하고 사용하는 예를 확인할 수 있습니다. tds_fdw를 사용하여 SQL Server 데이터베이스에 연결하기 전에 인스턴스에 대한 다음 세부 정보를 확인해야 합니다.

- 호스트 이름 또는 엔드포인트. RDS for SQL Server DB 인스턴스의 경우 콘솔을 사용하여 엔드포인트를 찾을 수 있습니다. 연결 및 보안(Connectivity & security) 탭을 선택하고 '엔드포인트 및 포트(Endpoint and port)' 섹션을 살펴봅니다.
- 포트 번호. Microsoft SQL Server의 기본 포트 번호는 포트 1433입니다.
- 데이터베이스 이름 DB 식별자입니다.

또한 SQL Server 포트 1433의 액세스 제어 목록(ACL) 또는 보안 그룹에 대한 액세스를 제공해야 합니다. RDS for PostgreSQL DB 인스턴스와 RDS for MySQL DB 인스턴스가 모두 포트 1433에 액세스할 수 있어야 합니다. 액세스가 올바르게 구성되지 않은 경우 Microsoft SQL Server를 쿼리하려고 할 때 다음 오류 메시지가 나타납니다.

```
ERROR: DB-Library error: DB #: 20009, DB Msg: Unable to connect:
Adaptive Server is unavailable or does not exist (mssql2019.aws-
region.rds.amazonaws.com), OS #: 0, OS Msg: Success, Level: 9
```

tds_fdw를 사용하여 SQL Server 데이터베이스에 연결하려면

1. `rds_superuser` 역할이 있는 계정을 사용하여 PostgreSQL DB 인스턴스에 연결합니다.

   ```
   psql --host=your-DB-instance.aws-region.rds.amazonaws.com --port=5432 --
   username=test --password
   ```

2. `tds_fdw` 확장을 설치합니다.

   ```
   test=> CREATE EXTENSION tds_fdw;
   CREATE EXTENSION
   ```

RDS for PostgreSQL DB 인스턴스에 확장을 설치한 후 외부 서버를 설정합니다.

외부 서버를 생성하려면

`rds_superuser` 권한이 있는 계정을 사용하여 RDS for PostgreSQL DB 인스턴스에서 다음과 같은 작업을 수행합니다.

1. RDS for PostgreSQL DB 인스턴스에서 외부 서버를 생성합니다.

```
test=> CREATE SERVER sqlserverdb FOREIGN DATA WRAPPER tds_fdw OPTIONS
 (servername 'mssql2019.aws-region.rds.amazonaws.com', port '1433', database
 'tds_fdw_testing');
CREATE SERVER
```

SQLServer 측에서 ASCII가 아닌 데이터에 액세스하려면 RDS for PostgreSQL DB 인스턴스S에서 character_set 옵션을 사용하여 서버 링크를 생성합니다.

```
test=> CREATE SERVER sqlserverdb FOREIGN DATA WRAPPER tds_fdw OPTIONS (servername
 'mssql2019.aws-region.rds.amazonaws.com', port '1433', database 'tds_fdw_testing',
 character_set 'UTF-8');
CREATE SERVER
```

2. `rds_superuser` 역할 권한이 없는 사용자(예: `user1`)에게 권한을 부여합니다.

```
test=> GRANT USAGE ON FOREIGN SERVER sqlserverdb TO user1;
```

3. user1으로 연결한 다음 SQL Server 사용자에 대한 매핑을 생성합니다.

```
test=> CREATE USER MAPPING FOR user1 SERVER sqlserverdb OPTIONS (username
 'sqlserveruser', password 'password');
CREATE USER MAPPING
```

4. SQL Server 테이블에 연결된 외부 테이블을 만듭니다.

```
test=> CREATE FOREIGN TABLE mytab (a int) SERVER sqlserverdb OPTIONS (table
 'MYTABLE');
CREATE FOREIGN TABLE
```

5. 외부 테이블을 쿼리합니다.

```
test=> SELECT * FROM mytab;
 a
---
 1
(1 row)
```

연결에 전송 중 암호화 사용

RDS for PostgreSQL에서 SQL Server로의 연결은 SQL Server 데이터베이스 구성에 따라 전송 중 암호화(TLS/SSL)를 사용합니다. SQL Server에 암호화가 구성되지 않은 경우 SQL Server 데이터베이스에 대한 요청을 수행하는 RDS for PostgreSQL 클라이언트가 암호화되지 않은 상태로 폴백됩니다.

`rds.force_ssl` 파라미터를 설정하여 RDS for SQL Server DB 인스턴스에 대한 연결에 암호화를 적용할 수 있습니다. 자세한 방법은 DB 인스턴스 연결이 SSL을 사용하도록 지정을 참조하세요. RDS for SQL Server에 대한 SSL/TLS 구성과 관련된 자세한 내용은 Microsoft SQL Server DB 인스턴스와 함께 SSL 사용을 참조하세요.

PostgreSQL용 신뢰할 수 있는 언어 확장 작업

PostgreSQL용 신뢰할 수 있는 언어 확장은 PostgreSQL 확장을 구축하기 위한 오픈 소스 개발 키트입니다. 이를 통해 고성능 PostgreSQL 확장을 구축하고 RDS for PostgreSQL DB 인스턴스에서 안전하게 실행할 수 있습니다. PostgreSQL용 신뢰할 수 있는 언어 확장(TLE)을 사용하면 PostgreSQL 기능 확장을 위해 문서화된 접근 방식을 따르는 PostgreSQL 확장을 생성할 수 있습니다. 자세한 내용은 PostgreSQL 설명서에서 Packaging Related Objects into an Extension을 참조하세요.

TLE의 주요 이점 중 하나는 PostgreSQL 인스턴스의 기반이 되는 파일 시스템에 대한 액세스를 제공하지 않는 환경에서 사용할 수 있다는 것입니다. 이전에는 새 확장을 설치하려면 파일 시스템에 액세스해야 했습니다. TLE는 이러한 제약을 제거합니다. TLE는 RDS for PostgreSQL DB 인스턴스에서 실행되는 데이터베이스를 포함하여 모든 PostgreSQL 데이터베이스를 위한 새 확장을 생성할 수 있는 개발 환경을 제공합니다.

TLE는 TLE를 사용하여 만든 확장의 안전하지 않은 리소스에 대한 액세스를 방지하도록 설계되었습니다. 런타임 환경은 확장 결함이 미치는 영향을 단일 데이터베이스 연결로 제한합니다. 또한 TLE는 데이터베이스 관리자에게 확장을 설치할 수 있는 사용자를 세밀하게 제어할 수 있도록 하며, 확장을 실행하기 위한 권한 모델을 제공합니다.

TLE는 다음 RDS for PostgreSQL 버전에서 지원됩니다.

- 버전 17.1 이상 17 버전
- 버전 16.1 이상 16 버전
- 버전 15.2 이상 15 버전
- 버전 14.5 이상 14 버전
- 버전 13.12 이상 13 버전

신뢰할 수 있는 언어 확장 개발 환경 및 런타임은 `pg_tle` PostgreSQL 확장 버전 1.0.1로 패키징됩니다. JavaScript, Perl, Tcl, PL/pgSQL 및 SQL에서 확장 생성을 지원합니다. 다른 PostgreSQL 확장을 설치하는 것과 동일한 방식으로 RDS for PostgreSQL DB 인스턴스에 `pg_tle` 확장을 설치합니다. `pg_tle`가 설정되면 개발자는 이를 사용하여 TLE 확장이라는 새 PostgreSQL 확장을 생성할 수 있습니다.

다음 주제에는 신뢰할 수 있는 언어 확장 설정 방법 및 자체 TLE 확장 생성 시작 방법에 대한 정보가 나와 있습니다.

주제

- 용어
- PostgreSQL용 신뢰할 수 있는 언어 확장을 사용하기 위한 요구 사항
- RDS for PostgreSQL DB 인스턴스에서 신뢰할 수 있는 언어 확장 설정
- PostgreSQL용 신뢰할 수 있는 언어 확장 개요
- RDS for PostgreSQL용 TLE 확장 생성
- 데이터베이스에서 TLE 확장 삭제
- PostgreSQL용 신뢰할 수 있는 언어 확장 제거
- TLE 확장과 함께 PostgreSQL 후크 사용
- TLE에서 사용자 지정 데이터 유형 사용
- PostgreSQL용 신뢰할 수 있는 언어 확장에 대한 함수 참조
- PostgreSQL용 신뢰할 수 있는 언어 확장에 대한 후크 참조

용어

신뢰할 수 있는 언어 확장을 더 잘 이해하려면 이 항목에서 사용되는 용어에 대한 다음 용어집을 참조하세요.

PostgreSQL용 신뢰할 수 있는 언어 확장

: PostgreSQL용 신뢰할 수 있는 언어 확장은 `pg_tle` 확장으로 패키징된 오픈 소스 개발 키트의 공식 이름입니다. 모든 PostgreSQL 시스템에서 사용할 수 있습니다. 자세한 내용은 GitHub의 aws/pg_tle를 참조하세요.

신뢰할 수 있는 언어 확장

: 신뢰할 수 있는 언어 확장은 PostgreSQL용 신뢰할 수 있는 언어 확장의 약칭입니다. 이 설명서에서는 이 약칭과 약어(TLE)도 사용됩니다.

신뢰할 수 있는 언어

: 신뢰할 수 있는 언어는 특정 보안 속성을 가진 프로그래밍 또는 스크립팅 언어입니다. 예를 들어 신뢰할 수 있는 언어는 일반적으로 파일 시스템에 대한 액세스를 제한하며, 지정된 네트워킹 속성의 사용을 제한합니다. TLE 개발 키트는 신뢰할 수 있는 언어를 지원하도록 설계되었습니다. PostgreSQL은 신뢰할 수 있거나 신뢰할 수 없는 확장 생성에 사용되는 여러 언어를 지원합니다. 예제는 PostgreSQL 설명서의 Trusted and Untrusted PL/Perl을 참조하세요. 신뢰할 수 있는 언어 확장을 사용하여 생성한 확장은 기본적으로 신뢰할 수 있는 언어 메커니즘을 사용합니다.

TLE 확장

TLE 확장은 신뢰할 수 있는 언어 확장(TLE) 개발 키트를 사용하여 생성된 PostgreSQL 확장입니다.

PostgreSQL용 신뢰할 수 있는 언어 확장을 사용하기 위한 요구 사항

TLE 개발 키트를 설정하고 사용하기 위한 요구 사항은 다음과 같습니다.

- RDS for PostgreSQL 버전 – 신뢰할 수 있는 언어 확장은 RDS for PostgreSQL 버전 13.12 이상 13 버전, 14.5 이상 14 버전 및 15.2 이상 버전에서만 지원됩니다.
  - RDS for PostgreSQL 인스턴스를 업그레이드해야 하는 경우, RDS for PostgreSQL DB 엔진 업그레이드를 참조하세요.
  - PostgreSQL을 실행하는 Amazon RDS DB 인스턴스가 아직 없는 경우 새로 생성할 수 있습니다. 자세한 내용은 RDS for PostgreSQL DB 인스턴스의 경우 PostgreSQL DB 인스턴스 생성 및 해당 인스턴스에 연결을 참조하세요.
- **`rds_superuser`** 권한 필요 - `pg_tle` 확장을 설정하고 구성하려면 데이터베이스 사용자 역할에 `rds_superuser` 역할의 권한이 있어야 합니다. 기본적으로 이 역할은 를 생성한 `postgres` 사용자에게 부여됩니다. RDS for PostgreSQL DB 인스턴스
- 사용자 지정 DB 파라미터 그룹 필요 - RDS for PostgreSQL DB 인스턴스는 사용자 지정 DB 파라미터 그룹을 사용하여 구성해야 합니다.
  - RDS for PostgreSQL DB 인스턴스가 사용자 지정 DB 파라미터 그룹을 사용하여 구성되지 않은 경우, 파라미터 그룹을 하나 생성하여 RDS for PostgreSQL DB 인스턴스에 연결해야 합니다. 단계에 대한 간략한 요약은 사용자 지정 DB 파라미터 그룹 생성 및 적용 섹션을 참조하세요.
  - RDS for PostgreSQL DB 인스턴스가 이미 사용자 지정 DB 파라미터 그룹을 사용하여 구성되어 있는 경우 신뢰할 수 있는 언어 확장을 설정할 수 있습니다. 세부 정보는 RDS for PostgreSQL DB 인스턴스에서 신뢰할 수 있는 언어 확장 설정을 참조하세요.

사용자 지정 DB 파라미터 그룹 생성 및 적용

다음 단계를 사용하여 사용자 지정 DB 파라미터 그룹을 생성하고 이를 사용하도록 RDS for PostgreSQL DB 인스턴스를 구성합니다.

콘솔

사용자 지정 DB 파라미터 그룹을 생성하고 RDS for PostgreSQL DB 인스턴스에 사용하는 방법

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. Amazon RDS 메뉴에서 Parameter groups(파라미터 그룹)를 선택합니다.
3. Create parameter group(파라미터 그룹 생성)을 선택합니다.
4. Parameter group details(파라미터 그룹 세부 정보) 페이지에서 다음 정보를 입력합니다.
   - Parameter group family(파라미터 그룹 패밀리)에서 postgres14를 선택합니다.
   - Type(유형)에서 DB Parameter Group을 선택합니다.
   - Group name(그룹 이름)에서 작업 컨텍스트에서 의미 있는 파라미터 그룹 이름을 지정합니다.
   - Description(설명)에 다른 팀원이 쉽게 찾을 수 있도록 유용한 설명을 입력합니다.
5. Create(생성)를 선택합니다. 사용자 지정 DB 파라미터 그룹이 AWS 리전에 생성됩니다. 이제 다음 단계에 따라 파라미터 그룹을 사용하도록 RDS for PostgreSQL DB 인스턴스를 수정할 수 있습니다.
6. Amazon RDS 메뉴에서 Databases(데이터베이스)를 선택합니다.
7. 나열된 항목 중에서 TLE와 함께 사용할 RDS for PostgreSQL DB 인스턴스를 선택한 다음 Modify(수정)를 선택합니다.
8. DB 인스턴스 설정 수정 페이지의 추가 구성 섹션에서 데이터베이스 옵션을 찾고 선택기에서 사용자 지정 DB 파라미터 그룹을 선택합니다.
9. Continue(계속)를 선택하여 변경 내용을 저장합니다.
10. Apply immediately(즉시 적용)를 선택하면 TLE를 사용하도록 RDS for PostgreSQL DB 인스턴스를 계속 설정할 수 있습니다.

신뢰할 수 있는 언어 확장을 사용하도록 시스템을 계속 설정하려면 RDS for PostgreSQL DB 인스턴스에서 신뢰할 수 있는 언어 확장 설정 섹션을 참조하세요.

DB 파라미터 그룹은 Amazon RDS DB 인스턴스용 DB 파라미터 그룹 섹션을 참조하세요.

AWS CLI

기본 AWS 리전으로 AWS CLI를 구성하면 CLI 명령을 사용할 때 `--region` 인수를 지정하지 않아도 됩니다. 자세한 내용은 AWS Command Line Interface 사용 설명서의 구성 기초 섹션을 참조하세요.

사용자 지정 DB 파라미터 그룹을 생성하고 RDS for PostgreSQL DB 인스턴스에 사용하는 방법

1. create-db-parameter-group AWS CLI 명령을 사용하여 사용자 AWS 리전의 postgres14를 기반으로 사용자 지정 DB 파라미터 그룹을 만들 수 있습니다.

   대상 LinuxmacOS, 또는Unix:

   ```
   aws rds create-db-parameter-group \
     --region aws-region \
     --db-parameter-group-name custom-params-for-pg-tle \
     --db-parameter-group-family postgres14 \
     --description "My custom DB parameter group for Trusted Language Extensions"
   ```

   Windows의 경우:

   ```
   aws rds create-db-parameter-group ^
     --region aws-region ^
     --db-parameter-group-name custom-params-for-pg-tle ^
     --db-parameter-group-family postgres14 ^
     --description "My custom DB parameter group for Trusted Language Extensions"
   ```

   사용자 지정 DB 파라미터 그룹은 AWS 리전에서 사용할 수 있으므로 파라미터 그룹을 사용하도록 RDS for PostgreSQL DB 인스턴스를 수정할 수 있습니다.

2. modify-db-instance AWS CLI 명령을 사용하여 사용자 지정 DB 파라미터 그룹을 RDS for PostgreSQL DB 인스턴스에 적용합니다. 이 명령은 활성 인스턴스를 즉시 재부팅합니다.

   대상 LinuxmacOS, 또는Unix:

   ```
   aws rds modify-db-instance \
     --region aws-region \
     --db-instance-identifier your-instance-name \
     --db-parameter-group-name custom-params-for-pg-tle \
     --apply-immediately
   ```

   Windows의 경우:

   ```
   aws rds modify-db-instance ^
     --region aws-region ^
     --db-instance-identifier your-instance-name ^
     --db-parameter-group-name custom-params-for-pg-tle ^
   ```

```
  --apply-immediately
```

신뢰할 수 있는 언어 확장을 사용하도록 시스템을 계속 설정하려면 RDS for PostgreSQL DB 인스턴스에서 신뢰할 수 있는 언어 확장 설정 섹션을 참조하세요.

자세한 내용은 Amazon RDS의 파라미터 그룹 단원을 참조하세요.

RDS for PostgreSQL DB 인스턴스에서 신뢰할 수 있는 언어 확장 설정

다음 단계에서는 사용자의 RDS for PostgreSQL DB 인스턴스가 사용자 지정 DB 파라미터 그룹에 연결되어 있다고 가정합니다. 이러한 단계에 AWS Management Console 또는 AWS CLI를 사용할 수 있습니다.

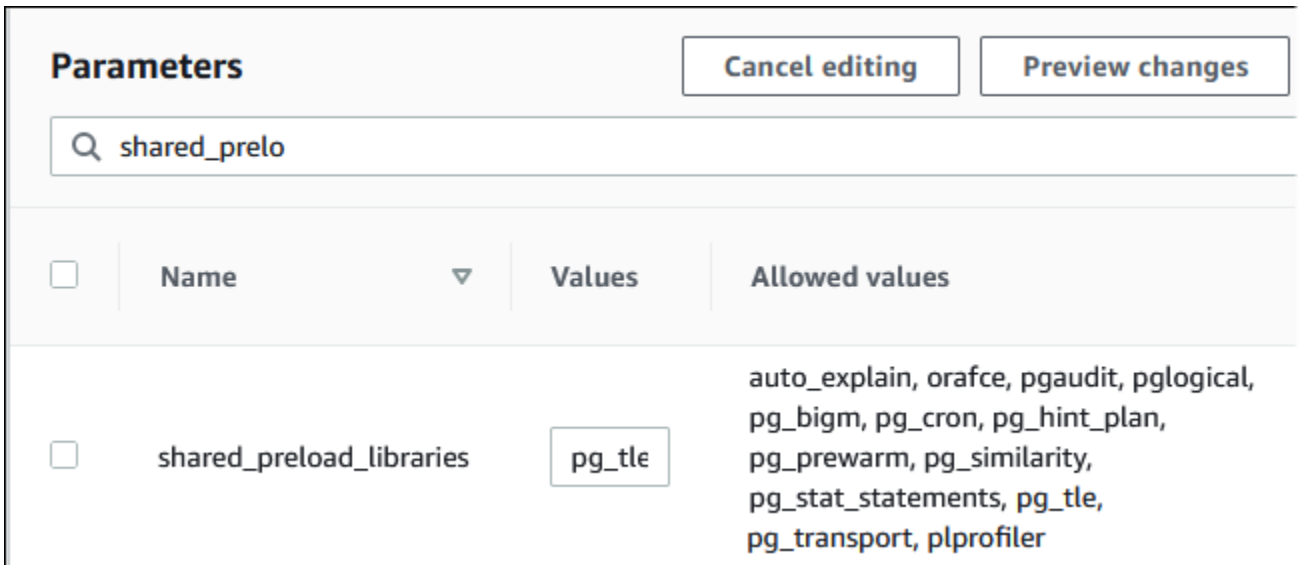
RDS for PostgreSQL DB 인스턴스에 신뢰할 수 있는 언어 확장을 설정하는 경우 특정 데이터베이스 권한이 있는 데이터베이스 사용자가 사용할 수 있도록 특정 데이터베이스에 설치합니다.

콘솔

신뢰할 수 있는 언어 확장 설정 방법

`rds_superuser` 그룹(역할)의 구성원인 계정을 사용하여 다음 단계를 수행합니다.

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 RDS for PostgreSQL DB 인스턴스를 선택합니다.
3. 의 구성 탭을 엽니다. RDS for PostgreSQL DB 인스턴스 인스턴스 세부 정보 중에서 파라미터 그룹 링크를 찾습니다.
4. 링크를 선택하여 와 연결된 사용자 지정 파라미터를 엽니다. RDS for PostgreSQL DB 인스턴스
5. 파라미터 검색 필드에 `shared_pre`를 입력하여 `shared_preload_libraries` 파라미터를 찾습니다.
6. 파라미터 편집을 선택하여 속성 값에 액세스합니다.
7. 값 필드의 목록에 `pg_tle`를 추가합니다. 쉼표를 사용하여 값 목록에서 항목을 구분합니다.



8. RDS for PostgreSQL DB 인스턴스를 재부팅하여 `shared_preload_libraries` 파라미터 변경 사항이 적용되도록 합니다.
9. 인스턴스를 사용할 수 있게 되면 `pg_tle`가 초기화되었는지 확인합니다. `psql`을 사용하여 RDS for PostgreSQL DB 인스턴스에 연결하고 다음 명령을 실행합니다.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pg_tle
(1 row)
```

10. `pg_tle` 확장이 초기화되었으므로 이제 확장을 생성할 수 있습니다.

```
CREATE EXTENSION pg_tle;
```

다음 `psql` 메타 명령을 사용하여 확장이 설치되었는지 확인할 수 있습니다.

```
labdb=> \dx
                          List of installed extensions
  Name   | Version |   Schema   |                Description
---------+---------+------------+-------------------------------------------
 pg_tle  | 1.0.1   | pgtle      | Trusted-Language Extensions for PostgreSQL
 plpgsql | 1.0     | pg_catalog | PL/pgSQL procedural language
```

11. PostgreSQL DB 인스턴스를 설정할 때 RDS for PostgreSQL DB 인스턴스를 위해 만든 기본 사용자 이름에 `pgtle_admin` 역할을 부여합니다. 기본값을 수락했다면 `postgres`입니다.

```
labdb=> GRANT pgtle_admin TO postgres;
GRANT ROLE
```

다음 예제와 같이 psql 메타 명령을 사용하여 역할이 부여되었는지 확인할 수 있습니다. pgtle_admin 및postgres 역할만 출력에 표시됩니다. 자세한 내용은 rds_superuser 역할 이해. 섹션을 참조하세요.

```
labdb=> \du
                          List of roles
    Role name    |           Attributes            |               Member of
-----------------+---------------------------------
+-----------------------------------
pgtle_admin     | Cannot login                     | {}
postgres        | Create role, Create DB          +| {rds_superuser,pgtle_admin}
                | Password valid until infinity    |...
```

12. \q 메타 명령을 사용하여psql 세션을 닫습니다.

```
\q
```

TLE 확장 생성을 시작하려면 예: SQL을 사용하여 신뢰할 수 있는 언어 확장 생성 섹션을 참조하세요.

AWS CLI

기본 AWS 리전으로 AWS CLI를 구성하면 CLI 명령을 사용할 때 --region 인수를 지정하지 않아도 됩니다. 자세한 내용은 AWS Command Line Interface 사용 설명서의 구성 기초 섹션을 참조하세요.

신뢰할 수 있는 언어 확장 설정 방법

1. modify-db-parameter-group AWS CLI 명령을 사용하여 pg_tle를 shared_preload_libraries 파라미터에 추가합니다.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pg_tle,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. reboot-db-instance AWS CLI 명령을 사용하여 RDS for PostgreSQL DB 인스턴스를 재부팅하고 pg_tle 라이브러리를 초기화합니다.

```
aws rds reboot-db-instance \
    --db-instance-identifier your-instance \
    --region aws-region
```

3. 인스턴스를 사용할 수 있게 되면 pg_tle가 초기화되었는지 확인할 수 있습니다. psql을 사용하여 RDS for PostgreSQL DB 인스턴스에 연결하고 다음 명령을 실행합니다.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pg_tle
(1 row)
```

pg_tle이 초기화되었으므로 이제 확장을 생성할 수 있습니다.

```
CREATE EXTENSION pg_tle;
```

4. PostgreSQL DB 인스턴스를 설정할 때 RDS for PostgreSQL DB 인스턴스를 위해 만든 기본 사용자 이름에 pgtle_admin 역할을 부여합니다. 기본값을 수락했다면 postgres입니다.

```
GRANT pgtle_admin TO postgres;
GRANT ROLE
```

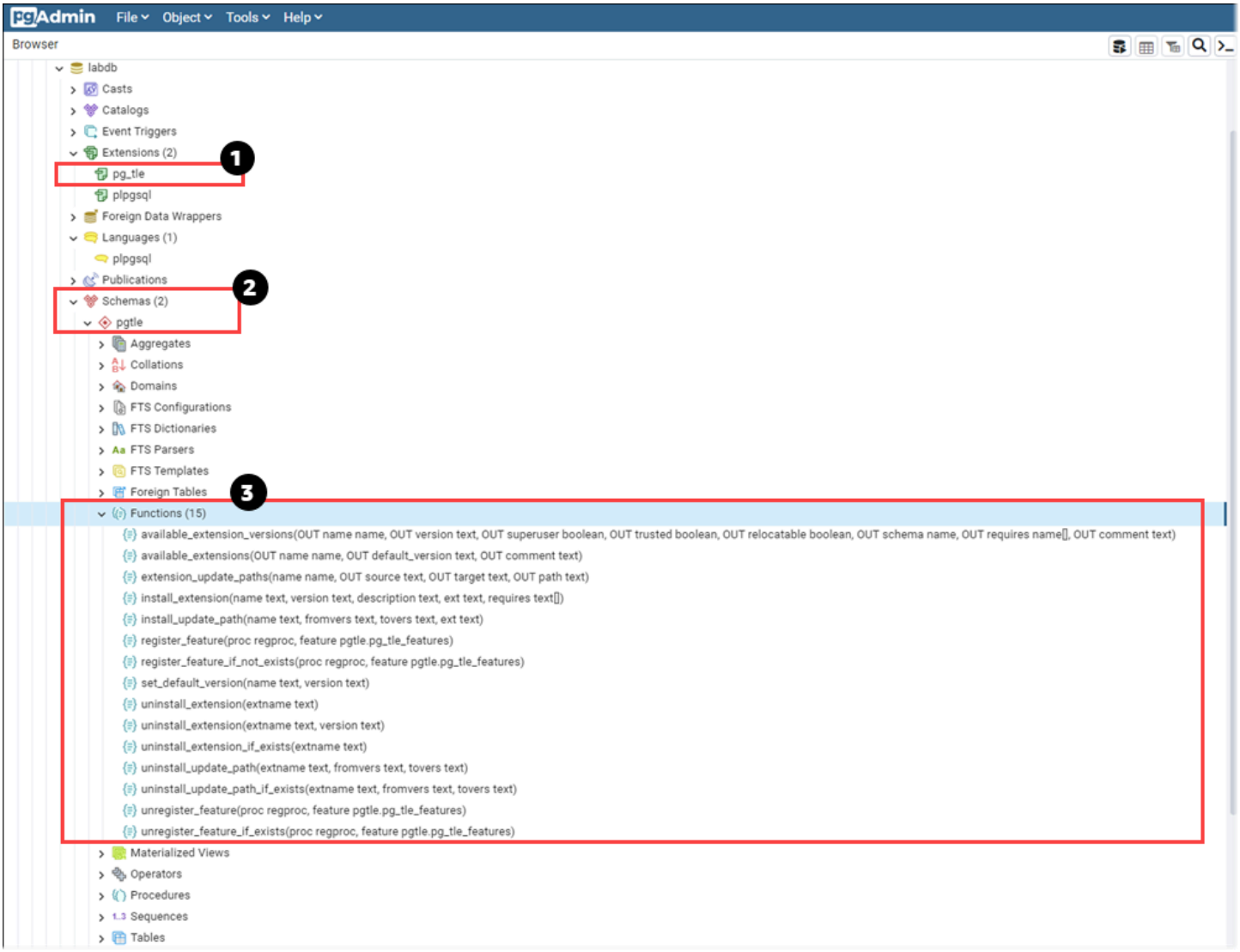
5. 다음과 같이 psql 세션을 닫습니다.

```
labdb=> \q
```

TLE 확장 생성을 시작하려면 예: SQL을 사용하여 신뢰할 수 있는 언어 확장 생성 섹션을 참조하세요.

PostgreSQL용 신뢰할 수 있는 언어 확장 개요

PostgreSQL용 신뢰할 수 있는 언어 확장은 다른 PostgreSQL 확장을 설정하는 것과 동일한 방식으로 RDS for PostgreSQL DB 인스턴스에 설치하는 PostgreSQL 확장입니다. PgAdmin 클라이언트 도구에 있는 예제 데이터베이스의 다음 이미지에서 pg_tle 확장을 구성하는 일부 구성 요소를 볼 수 있습니다.



다음 세부 정보를 볼 수 있습니다.

1. PostgreSQL용 신뢰할 수 있는 언어 확장(TLE)은 `pg_tle` 확장으로 패키징된 개발 키트입니다. 따라서 `pg_tle`는 설치되는 데이터베이스의 사용 가능한 확장에 추가됩니다.
2. TLE에는 자체 스키마 `pgtle`이 있습니다. 이 스키마에는 생성한 확장을 설치하고 관리하는 도우미 함수(3)가 포함되어 있습니다.
3. TLE는 확장 설치, 등록, 관리를 위한 12개 이상의 도우미 함수를 제공합니다. 이러한 함수에 대한 자세한 내용은 PostgreSQL용 신뢰할 수 있는 언어 확장에 대한 함수 참조 섹션을 참조하세요.

`pg_tle` 확장의 기타 구성 요소에는 다음이 포함됩니다.

- **pgtle_admin** 역할 - pg_tle 확장이 설치될 때 pgtle_admin 역할이 생성됩니다. 이 역할은 권한이 부여되므로 그에 따라 취급되어야 합니다. 데이터베이스 사용자에게 pgtle_admin 역할을 부여할 때는 최소 권한 원칙을 따르는 것이 좋습니다. 즉, 새 TLE 확장을 생성, 설치, 관리할 수 있는 데이터베이스 사용자에게만 pgtle_admin 역할을 부여합니다(예: postgres).
- **pgtle.feature_info** 테이블 - pgtle.feature_info 테이블은 TLE, 후크, TLE와 후크가 사용하는 사용자 지정 저장 프로시저 및 함수에 대한 정보가 포함된 보호된 테이블입니다. pgtle_admin 권한이 있는 경우 다음과 같은 신뢰할 수 있는 언어 확장 함수를 사용하여 테이블에서 해당 정보를 추가하고 업데이트할 수 있습니다.
  - pgtle.register_feature
  - pgtle.register_feature_if_not_exists
  - pgtle.unregister_feature
  - pgtle.unregister_feature_if_exists

RDS for PostgreSQL용 TLE 확장 생성

pg_tle 확장이 설치된 RDS for PostgreSQL DB 인스턴스에서 TLE를 사용하여 생성한 확장을 설치할 수 있습니다. pg_tle 확장은 설치된 PostgreSQL 데이터베이스로 범위가 지정됩니다. TLE를 사용하여 생성한 확장은 동일한 데이터베이스로 범위가 지정됩니다.

다양한 pgtle 함수를 사용하여 TLE 확장을 구성하는 코드를 설치하세요. 다음과 같은 신뢰할 수 있는 언어 확장 함수는 모두 pgtle_admin 역할이 필요합니다.

- pgtle.install_extension
- pgtle.install_update_path
- pgtle.register_feature
- pgtle.register_feature_if_not_exists
- pgtle.set_default_version
- pgtle.uninstall_extension(name)
- pgtle.uninstall_extension(name, version)
- pgtle.uninstall_extension_if_exists
- pgtle.uninstall_update_path
- pgtle.uninstall_update_path_if_exists
- pgtle.unregister_feature

- pgtle.unregister_feature_if_exists

예: SQL을 사용하여 신뢰할 수 있는 언어 확장 생성

다음 예제는 다양한 공식을 사용하여 거리를 계산하는 몇 가지 SQL 함수가 포함된 pg_distance라는 TLE 확장을 생성하는 방법을 보여 줍니다. 목록에서 맨해튼 거리를 계산하는 함수와 유클리드 거리를 계산하는 함수를 찾을 수 있습니다. 이러한 공식의 차이에 대한 자세한 내용은 Wikipedia의 Taxicab geometry와 Euclidean geometry를 참조하세요.

RDS for PostgreSQL DB 인스턴스에서 신뢰할 수 있는 언어 확장 설정에 설명된 대로 pg_tle 확장을 설정한 경우 자체 RDS for PostgreSQL DB 인스턴스에서 이 예제를 사용할 수 있습니다.

> **Note**
>
> 이 절차를 수행하려면 pgtle_admin 역할의 권한이 있어야 합니다.

예제 TLE 확장을 생성하는 방법

다음 단계에서는 labdb라는 예제 데이터베이스를 사용합니다. 이 데이터베이스는 postgres 기본 사용자가 소유합니다. postgres 역할에는 pgtle_admin 역할의 권한도 있습니다.

1. psql을 사용하여 에 연결합니다. RDS for PostgreSQL DB 인스턴스

```
psql --host=db-instance-123456789012.aws-region.rds.amazonaws.com
--port=5432 --username=postgres --password --dbname=labdb
```

2. 다음 코드를 복사하고 psql 세션 콘솔에 붙여넣어 이름이 pg_distance인 TLE 확장을 생성합니다.

```
SELECT pgtle.install_extension
(
 'pg_distance',
 '0.1',
  'Distance functions for two points',
$_pg_tle_$
    CREATE FUNCTION dist(x1 float8, y1 float8, x2 float8, y2 float8, norm int)
    RETURNS float8
    AS $$
      SELECT (abs(x2 - x1) ^ norm + abs(y2 - y1) ^ norm) ^ (1::float8 / norm);
    $$ LANGUAGE SQL;
```

```
    CREATE FUNCTION manhattan_dist(x1 float8, y1 float8, x2 float8, y2 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 1);
    $$ LANGUAGE SQL;

    CREATE FUNCTION euclidean_dist(x1 float8, y1 float8, x2 float8, y2 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 2);
    $$ LANGUAGE SQL;
$_pg_tle_$
);
```

출력은 다음과 같습니다.

```
install_extension
---------------
 t
(1 row)
```

pg_distance 확장을 구성하는 아티팩트가 이제 데이터베이스에 설치되었습니다. 이러한 아티팩트에는 제어 파일과 확장 코드가 포함되며, 이러한 항목은 CREATE EXTENSION 명령을 사용하여 확장을 생성하려면 있어야 하는 항목입니다. 즉, 데이터베이스 사용자가 확장의 함수를 사용할 수 있도록 하려면 확장을 생성해야 합니다.

3. 확장을 생성하려면 다른 확장과 마찬가지로 CREATE EXTENSION 명령을 사용하세요. 다른 확장과 마찬가지로 데이터베이스 사용자는 데이터베이스에서 CREATE 권한이 있어야 합니다.

```
CREATE EXTENSION pg_distance;
```

4. pg_distance TLE 확장을 테스트하려면 TLE 확장을 사용하여 네 점 사이의 맨해튼 거리를 계산하면 됩니다.

```
labdb=> SELECT manhattan_dist(1, 1, 5, 5);
8
```

동일한 점 집합 간의 유클리드 거리를 계산하려면 다음을 사용하면 됩니다.

```
labdb=> SELECT euclidean_dist(1, 1, 5, 5);
```

```
5.656854249492381
```

pg_distance 확장은 데이터베이스에 함수를 로드하여 데이터베이스에서 권한이 있는 모든 사용자가 사용할 수 있도록 합니다.

TLE 확장 수정

이 TLE 확장에 패키징된 함수의 쿼리 성능을 향상시키려면 다음 두 PostgreSQL 속성을 해당 사양에 추가합니다.

- IMMUTABLE - IMMUTABLE 속성은 쿼리 최적화 프로그램이 최적화를 통해 쿼리 응답 시간을 개선할 수 있도록 합니다. 자세한 내용은 PostgreSQL 설명서에서 Function Volatility Categories를 참조하세요.
- PARALLEL SAFE - PARALLEL SAFE 속성은 PostgreSQL이 병렬 모드에서 함수를 실행할 수 있도록 하는 또 다른 속성입니다. 자세한 내용은 PostgreSQL 설명서에서 CREATE FUNCTION을 참조하십시오.

다음 예제에서는 pgtle.install_update_path 함수를 사용하여 각 함수에 이러한 속성을 추가하여 pg_distance TLE 확장의 0.2 버전을 생성하는 방법을 확인할 수 있습니다. 이 함수에 대한 자세한 내용은 pgtle.install_update_path 단원을 참조하세요. 이 작업을 수행하려면 pgtle_admin 역할이 있어야 합니다.

기존 TLE 확장을 업데이트하고 기본 버전을 지정하는 방법

1. psql 또는 pgAdmin 같은 다른 클라이언트 도구를 사용하여 RDS for PostgreSQL DB 인스턴스에 연결합니다.

```
psql --host=db-instance-123456789012.aws-region.rds.amazonaws.com
--port=5432 --username=postgres --password --dbname=labdb
```

2. 다음 코드를 복사하고 psql 세션 콘솔에 붙여넣어 기존 TLE 확장을 수정합니다.

```
SELECT pgtle.install_update_path
(
 'pg_distance',
 '0.1',
 '0.2',
$_pg_tle_$
```

```
    CREATE OR REPLACE FUNCTION dist(x1 float8, y1 float8, x2 float8, y2 float8,
 norm int)
    RETURNS float8
    AS $$
      SELECT (abs(x2 - x1) ^ norm + abs(y2 - y1) ^ norm) ^ (1::float8 / norm);
    $$ LANGUAGE SQL IMMUTABLE PARALLEL SAFE;

    CREATE OR REPLACE FUNCTION manhattan_dist(x1 float8, y1 float8, x2 float8, y2
 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 1);
    $$ LANGUAGE SQL IMMUTABLE PARALLEL SAFE;

    CREATE OR REPLACE FUNCTION euclidean_dist(x1 float8, y1 float8, x2 float8, y2
 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 2);
    $$ LANGUAGE SQL IMMUTABLE PARALLEL SAFE;
$_pg_tle_$
);
```

다음과 비슷한 응답이 나타납니다.

```
install_update_path
--------------------
 t
(1 row)
```

이 확장 버전을 기본 버전으로 지정하면 데이터베이스 사용자가 데이터베이스에서 확장을 생성 또는 업데이트할 때 버전을 지정하지 않아도 됩니다.

3. TLE 확장의 수정된 버전(버전 0.2)을 기본 버전으로 지정하려면 다음 예제와 같이 `pgtle.set_default_version` 함수를 사용하세요.

```
SELECT pgtle.set_default_version('pg_distance', '0.2');
```

이 함수에 대한 자세한 내용은 pgtle.set_default_version 단원을 참조하세요.

4. 코드가 준비되면 다음과 같이 `ALTER EXTENSION ... UPDATE` 명령을 사용하여 일반적인 방법으로 설치된 TLE 확장을 업데이트할 수 있습니다.

```
ALTER EXTENSION pg_distance UPDATE;
```

데이터베이스에서 TLE 확장 삭제

다른 PostgreSQL 확장과 동일한 방식으로 DROP EXTENSION 명령을 사용하여 TLE 확장을 삭제할 수 있습니다. 확장을 삭제해도 확장을 구성하는 설치 파일은 제거되지 않으므로 사용자가 확장을 다시 생성할 수 있습니다. 확장 및 해당 설치 파일을 제거하려면 다음 2단계 프로세스를 수행하세요.

TLE 확장을 삭제하고 해당 설치 파일을 제거하는 방법

1. psql 또는 다른 클라이언트 도구를 사용하여 RDS for PostgreSQL DB 인스턴스에 연결합니다.

   ```
   psql --host=.111122223333.aws-region.rds.amazonaws.com --port=5432 --
   username=postgres --password --dbname=dbname
   ```

2. 다른 PostgreSQL 확장과 같은 방법으로 확장을 삭제합니다.

   ```
   DROP EXTENSION your-TLE-extension
   ```

   예를 들어 예: SQL을 사용하여 신뢰할 수 있는 언어 확장 생성에 설명된 대로 pg_distance 확장을 생성한 경우 다음과 같이 확장을 삭제할 수 있습니다.

   ```
   DROP EXTENSION pg_distance;
   ```

   다음과 같이 확장이 삭제되었음을 확인하는 출력이 표시됩니다.

   ```
   DROP EXTENSION
   ```

   이 시점에서 확장은 더 이상 데이터베이스에서 활성 상태가 아닙니다. 그러나 설치 파일과 제어 파일은 여전히 데이터베이스에서 사용할 수 있으므로 데이터베이스 사용자는 원하는 경우 확장을 다시 생성할 수 있습니다.

   - 데이터베이스 사용자가 TLE 확장자를 생성할 수 있도록 확장 파일을 그대로 두려면 여기서 멈추면 됩니다.
   - 확장을 구성하는 모든 파일을 제거하려면 다음 단계를 계속합니다.

3. 확장의 모든 설치 파일을 제거하려면 pgtle.uninstall_extension 함수를 사용하세요. 이 함수는 확장의 모든 코드와 제어 파일을 제거합니다.

```
SELECT pgtle.uninstall_extension('your-tle-extension-name');
```

예를 들어 모든 pg_distance 설치 파일을 제거하려면 다음 명령을 사용합니다.

```
SELECT pgtle.uninstall_extension('pg_distance');
 uninstall_extension
---------------------
 t
(1 row)
```

PostgreSQL용 신뢰할 수 있는 언어 확장 제거

더 이상 TLE를 사용하여 TLE 확장을 직접 생성하지 않으려면 pg_tle 확장을 삭제하고 모든 아티팩트를 제거하면 됩니다. 이 작업에는 데이터베이스의 모든 TLE 확장 삭제 및 pgtle 스키마 삭제가 포함됩니다.

데이터베이스에서 **pg_tle** 확장 및 해당 스키마를 삭제하는 방법

1. psql 또는 다른 클라이언트 도구를 사용하여 RDS for PostgreSQL DB 인스턴스에 연결합니다.

```
psql --host=.111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=dbname
```

2. 데이터베이스에서 pg_tle 확장을 삭제합니다. 데이터베이스에서 자체 TLE 확장이 아직 실행 중인 경우 해당 확장도 삭제해야 합니다. 이를 위해 다음과 같이 CASCADE 키워드를 사용할 수 있습니다.

```
DROP EXTENSION pg_tle CASCADE;
```

pg_tle 확장이 데이터베이스에서 아직 활성 상태가 아닌 경우 CASCADE 키워드를 사용할 필요가 없습니다.

3. pgtle 스키마를 삭제합니다. 이 작업을 수행하면 데이터베이스에서 모든 관리 함수가 제거됩니다.

```
DROP SCHEMA pgtle CASCADE;
```

이 명령은 프로세스가 완료되면 다음을 반환합니다.

```
DROP SCHEMA
```

pg_tle 확장, 해당 스키마와 함수, 모든 아티팩트가 제거됩니다. TLE를 사용하여 새 확장을 생성하려면 설정 프로세스를 다시 진행하세요. 자세한 내용은 RDS for PostgreSQL DB 인스턴스에서 신뢰할 수 있는 언어 확장 설정 단원을 참조하십시오.

TLE 확장과 함께 PostgreSQL 후크 사용

후크는 PostgreSQL에서 사용할 수 있는 콜백 메커니즘으로, 개발자가 일반 데이터베이스 작업 중에 사용자 지정 함수 또는 기타 루틴을 호출할 수 있습니다. TLE 개발 키트는 PostgreSQL 후크를 지원하므로 런타임에 사용자 지정 함수를 PostgreSQL 동작과 통합할 수 있습니다. 예를 들어 후크를 사용하여 인증 프로세스를 사용자 지정 코드와 연결하거나 특정 요구 사항에 맞게 쿼리 계획 및 실행 프로세스를 수정할 수 있습니다.

TLE 확장은 후크를 사용할 수 있습니다. 후크의 범위가 전역적이면 모든 데이터베이스에 적용됩니다. 따라서 TLE 확장이 전역 후크를 사용하는 경우 사용자가 액세스할 수 있는 모든 데이터베이스에 TLE 확장을 생성해야 합니다.

pg_tle 확장을 사용하여 신뢰할 수 있는 언어 확장을 직접 구축하는 경우 SQL API에서 사용 가능한 후크를 사용하여 확장의 함수를 구축할 수 있습니다. 모든 후크는 pg_tle에 등록해야 합니다. 일부 후크의 경우 다양한 구성 파라미터를 설정해야 할 수도 있습니다. 예를 들어 passcode 확인 후크는 켜기, 해제 또는 필수로 설정할 수 있습니다. 사용 가능한 pg_tle 후크의 특정 요구 사항에 대한 자세한 내용은 PostgreSQL용 신뢰할 수 있는 언어 확장에 대한 후크 참조 섹션을 참조하세요.

예: PostgreSQL 후크를 사용하는 확장 생성

이 섹션에서 설명하는 예제에서는 PostgreSQL 후크를 사용하여 특정 SQL 작업 중에 제공된 암호를 확인하고 데이터베이스 사용자가 password_check.bad_passwords 테이블에 포함된 암호로 암호를 설정하지 못하도록 합니다. 이 테이블에는 가장 일반적으로 사용되지만 쉽게 깰 수 있는 상위 10개 암호가 나와 있습니다.

이 예제를 RDS for PostgreSQL DB 인스턴스에서 설정하려면 신뢰할 수 있는 언어 확장이 이미 설치되어 있어야 합니다. 자세한 내용은 RDS for PostgreSQL DB 인스턴스에서 신뢰할 수 있는 언어 확장 설정을 참조하세요.

암호 확인 후크 예제를 설정하는 방법

1. psql을 사용하여 에 연결합니다. RDS for PostgreSQL DB 인스턴스

```
psql --host=db-instance-123456789012.aws-region.rds.amazonaws.com
--port=5432 --username=postgres --password --dbname=labdb
```

2. 암호 확인 후크 코드 목록에서 코드를 복사하여 데이터베이스에 붙여넣습니다.

```
SELECT pgtle.install_extension (
  'my_password_check_rules',
  '1.0',
  'Do not let users use the 10 most commonly used passwords',
$_pgtle_$
  CREATE SCHEMA password_check;
  REVOKE ALL ON SCHEMA password_check FROM PUBLIC;
  GRANT USAGE ON SCHEMA password_check TO PUBLIC;

  CREATE TABLE password_check.bad_passwords (plaintext) AS
  VALUES
    ('123456'),
    ('password'),
    ('12345678'),
    ('qwerty'),
    ('123456789'),
    ('12345'),
    ('1234'),
    ('111111'),
    ('1234567'),
    ('dragon');
  CREATE UNIQUE INDEX ON password_check.bad_passwords (plaintext);

  CREATE FUNCTION password_check.passcheck_hook(username text, password text,
 password_type pgtle.password_types, valid_until timestamptz, valid_null boolean)
  RETURNS void AS $$
    DECLARE
      invalid bool := false;
    BEGIN
      IF password_type = 'PASSWORD_TYPE_MD5' THEN
```

```
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE ('md5' || md5(bp.plaintext || username)) = password
        ) INTO invalid;
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password
 dictionary';
        END IF;
      ELSIF password_type = 'PASSWORD_TYPE_PLAINTEXT' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE bp.plaintext = password
        ) INTO invalid;
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password
 dictionary';
        END IF;
      END IF;
    END
  $$ LANGUAGE plpgsql SECURITY DEFINER;

  GRANT EXECUTE ON FUNCTION password_check.passcheck_hook TO PUBLIC;

  SELECT pgtle.register_feature('password_check.passcheck_hook', 'passcheck');
$_pgtle_$
);
```

확장이 데이터베이스에 로드되면 다음과 같은 출력이 표시됩니다.

```
 install_extension
-------------------
 t
(1 row)
```

3. 데이터베이스에 연결되어 있는 동안 확장을 생성할 수 있습니다.

```
CREATE EXTENSION my_password_check_rules;
```

4. 다음 psql 메타 명령을 사용하여 데이터베이스에 확장이 생성되었는지 확인할 수 있습니다.

```
\dx
```

```
                          List of installed extensions
          Name           | Version |   Schema   |
 Description
------------------------+---------+------------
+-------------------------------------------------------------
 my_password_check_rules | 1.0     | public     | Prevent use of any of the top-ten
 most common bad passwords
 pg_tle                  | 1.0.1   | pgtle      | Trusted-Language Extensions for
 PostgreSQL
 plpgsql                 | 1.0     | pg_catalog | PL/pgSQL procedural language
(3 rows)
```

5. AWS CLI로 작업할 다른 터미널 세션을 엽니다. 암호 확인 후크를 켜려면 사용자 지정 DB 파라미터 그룹을 수정해야 합니다. 이렇게 하려면 다음 예제에서와 같이 modify-db-parameter-group CLI 명령을 사용합니다.

```
aws rds modify-db-parameter-group \
    --region aws-region \
    --db-parameter-group-name your-custom-parameter-group \
    --parameters
 "ParameterName=pgtle.enable_password_check,ParameterValue=on,ApplyMethod=immediate"
```

파라미터가 성공적으로 켜지면 다음과 같은 출력이 표시됩니다.

```
(
    "DBParameterGroupName": "docs-lab-parameters-for-tle"
}
```

파라미터 그룹 설정 변경 사항이 적용되기까지 몇 분 정도 걸릴 수 있습니다. 하지만 이 파라미터는 동적이므로 설정을 적용하기 위해 RDS for PostgreSQL 인스턴스를 다시 시작할 필요는 없습니다.

6. psql 세션을 열고 데이터베이스를 쿼리하여 password_check 후크가 켜졌는지 확인합니다.

```
labdb=> SHOW pgtle.enable_password_check;
pgtle.enable_password_check
-----------------------------
on
(1 row)
```

이제 password-check 후크가 활성 상태입니다. 다음 예제와 같이 새 역할을 생성하고 잘못된 암호 중 하나를 사용하여 이를 테스트할 수 있습니다.

```
CREATE ROLE test_role PASSWORD 'password';
ERROR:  Cannot use passwords from the common password dictionary
CONTEXT:  PL/pgSQL function
 password_check.passcheck_hook(text,text,pgtle.password_types,timestamp with time
 zone,boolean) line 21 at RAISE
SQL statement "SELECT password_check.passcheck_hook(
    $1::pg_catalog.text,
    $2::pg_catalog.text,
    $3::pgtle.password_types,
    $4::pg_catalog.timestamptz,
    $5::pg_catalog.bool)"
```

출력은 가독성을 위해 형식이 지정되었습니다.

다음 예제는 pgsql 대화형 메타 명령 \password 동작도 password_check 후크의 영향을 받는다는 것을 보여 줍니다.

```
postgres=> SET password_encryption TO 'md5';
SET
postgres=> \password
Enter new password for user "postgres":*****
Enter it again:*****
ERROR:  Cannot use passwords from the common password dictionary
CONTEXT:  PL/pgSQL function
 password_check.passcheck_hook(text,text,pgtle.password_types,timestamp with time
 zone,boolean) line 12 at RAISE
SQL statement "SELECT password_check.passcheck_hook($1::pg_catalog.text,
 $2::pg_catalog.text, $3::pgtle.password_types, $4::pg_catalog.timestamptz,
 $5::pg_catalog.bool)"
```

원하는 경우 이 TLE 확장을 삭제하고 소스 파일을 제거할 수 있습니다. 자세한 내용은 데이터베이스에서 TLE 확장 삭제 섹션을 참조하세요.

암호 확인 후크 코드 목록

여기에 표시된 예제 코드는 my_password_check_rules TLE 확장의 사양을 정의합니다. 이 코드를 복사하여 데이터베이스에 붙여넣으면 my_password_check_rules 확장 코드가 데이터베이스에 로드되고 확장에서 사용할 수 있도록 password_check 후크가 등록됩니다.

```
SELECT pgtle.install_extension (
  'my_password_check_rules',
  '1.0',
  'Do not let users use the 10 most commonly used passwords',
$_pgtle_$
  CREATE SCHEMA password_check;
  REVOKE ALL ON SCHEMA password_check FROM PUBLIC;
  GRANT USAGE ON SCHEMA password_check TO PUBLIC;

  CREATE TABLE password_check.bad_passwords (plaintext) AS
  VALUES
    ('123456'),
    ('password'),
    ('12345678'),
    ('qwerty'),
    ('123456789'),
    ('12345'),
    ('1234'),
    ('111111'),
    ('1234567'),
    ('dragon');
  CREATE UNIQUE INDEX ON password_check.bad_passwords (plaintext);

  CREATE FUNCTION password_check.passcheck_hook(username text, password text,
 password_type pgtle.password_types, valid_until timestamptz, valid_null boolean)
  RETURNS void AS $$
    DECLARE
      invalid bool := false;
    BEGIN
      IF password_type = 'PASSWORD_TYPE_MD5' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE ('md5' || md5(bp.plaintext || username)) = password
        ) INTO invalid;
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password dictionary';
        END IF;
      ELSIF password_type = 'PASSWORD_TYPE_PLAINTEXT' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE bp.plaintext = password
```

```
        ) INTO invalid;
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password dictionary';
        END IF;
      END IF;
    END
  $$ LANGUAGE plpgsql SECURITY DEFINER;

  GRANT EXECUTE ON FUNCTION password_check.passcheck_hook TO PUBLIC;

  SELECT pgtle.register_feature('password_check.passcheck_hook', 'passcheck');
$_pgtle_$
);
```

TLE에서 사용자 지정 데이터 유형 사용

PostgreSQL은 데이터베이스의 복잡한 데이터 구조를 효율적으로 처리하기 위해 새로운 기본 유형(다른 명칭: 스칼라 유형)을 등록하는 명령을 지원합니다. 기본 유형을 사용하면 데이터를 내부적으로 저장하는 방법과 데이터를 외부 텍스트 표현으로 변환 및 외부 텍스트 표현에서 변환하는 방법을 사용자 정의할 수 있습니다. 이러한 사용자 지정 데이터 유형은 숫자 또는 텍스트 등의 기본 제공 유형으로는 충분한 검색 의미를 제공할 수 없는 기능적 도메인을 지원하도록 PostgreSQL을 확장할 때 유용합니다.

RDS for PostgreSQL을 사용하면 신뢰할 수 있는 언어 확장에서 사용자 지정 데이터 유형을 만들고, 이러한 새 데이터 유형에서 SQL 및 인덱스 작업을 지원하는 함수를 정의할 수 있습니다. 사용자 지정 데이터 유형은 다음 버전에서 사용할 수 있습니다.

- RDS for PostgreSQL 15.4 이상의 15 버전
- RDS for PostgreSQL 14.9 이상의 14 버전
- RDS for PostgreSQL 13.12 이상의 13 버전

자세한 내용을 알아보려면 신뢰할 수 있는 언어 기반 유형을 참조하세요.

PostgreSQL용 신뢰할 수 있는 언어 확장에 대한 함수 참조

PostgreSQL용 신뢰할 수 있는 언어 확장에서 사용할 수 있는 함수에 대한 다음 참조 설명서를 참조하세요. 이러한 함수를 사용하면 TLE 확장, 즉 신뢰할 수 있는 언어 확장 개발 키트를 사용하여 개발한 PostgreSQL 확장을 설치, 등록, 업데이트 및 관리할 수 있습니다.

함수

- pgtle.available_extensions
- pgtle.available_extension_versions
- pgtle.extension_update_paths
- pgtle.install_extension
- pgtle.install_update_path
- pgtle.register_feature
- pgtle.register_feature_if_not_exists
- pgtle.set_default_version
- pgtle.uninstall_extension(name)
- pgtle.uninstall_extension(name, version)
- pgtle.uninstall_extension_if_exists
- pgtle.uninstall_update_path
- pgtle.uninstall_update_path_if_exists
- pgtle.unregister_feature
- pgtle.unregister_feature_if_exists

pgtle.available_extensions

`pgtle.available_extensions` 함수는 집합을 반환하는 함수입니다. 이 함수는 데이터베이스에서 사용 가능한 모든 TLE 확장을 반환합니다. 반환된 각 행에는 단일 TLE 확장에 대한 정보가 포함되어 있습니다.

함수 프로토타입

```
pgtle.available_extensions()
```

역할

없음.

인수

없음.

출력

- name - TLE 확장의 이름입니다.
- default_version - 버전을 지정하지 않고 CREATE EXTENSION을 호출할 때 사용할 TLE 확장의 버전입니다.
- description - TLE 확장에 대한 자세한 설명입니다.

사용 예

```
SELECT * FROM pgtle.available_extensions();
```

pgtle.available_extension_versions

available_extension_versions 함수는 집합을 반환하는 함수입니다. 이 함수는 사용 가능한 모든 TLE 확장 및 버전의 목록을 반환합니다. 각 행에는 특정 역할이 필요한지 여부를 포함하여 지정된 TLE 확장의 특정 버전에 대한 정보가 포함되어 있습니다.

함수 프로토타입

```
pgtle.available_extension_versions()
```

역할

없음.

인수

없음.

출력

- name - TLE 확장의 이름입니다.
- version - TLE 확장의 버전입니다.
- superuser - TLE 확장의 경우 이 값은 항상 false입니다. TLE 확장을 생성하거나 업데이트하는 데 필요한 권한은 지정된 데이터베이스에서 다른 객체를 만드는 데 필요한 권한과 동일합니다.
- trusted - TLE 확장의 경우 이 값은 항상 false입니다.
- relocatable - TLE 확장의 경우 이 값은 항상 false입니다.
- schema - TLE 확장이 설치된 스키마의 이름을 지정합니다.

- `requires` - 이 TLE 확장에 필요한 다른 확장의 이름을 포함하는 배열입니다.
- `description` - TLE 확장에 대한 자세한 설명입니다.

출력 값에 대한 자세한 내용은 PostgreSQL 설명서에서 Packaging Related Objects into an Extension > Extension Files를 참조하세요.

사용 예

```
SELECT * FROM pgtle.available_extension_versions();
```

pgtle.extension_update_paths

`extension_update_paths` 함수는 집합을 반환하는 함수입니다. 이 함수는 TLE 확장에 가능한 모든 업데이트 경로 목록을 반환합니다. 각 행에는 해당 TLE 확장에 사용할 수 있는 업그레이드 또는 다운그레이드가 포함됩니다.

함수 프로토타입

```
pgtle.extension_update_paths(name)
```

역할

없음.

인수

`name` - 업그레이드 경로를 가져올 TLE 확장의 이름입니다.

출력

- `source` - 업데이트의 소스 버전입니다.
- `target` - 업데이트의 대상 버전입니다.
- `path` - TLE 확장을 `source` 버전에서 `target` 버전으로 업데이트하는 데 사용되는 업그레이드 경로입니다(예: `0.1--0.2`).

사용 예

```
SELECT * FROM pgtle.extension_update_paths('your-TLE');
```

pgtle.install_extension

install_extension 함수를 사용하면 데이터베이스에 TLE 확장을 구성하는 아티팩트를 설치한 다음 CREATE EXTENSION 명령을 사용하여 TLE 확장을 생성할 수 있습니다.

함수 프로토타입

```
pgtle.install_extension(name text, version text, description text, ext text, requires
 text[] DEFAULT NULL::text[])
```

역할

없음.

인수

- name - TLE 확장의 이름입니다. 이 값은 CREATE EXTENSION 호출에 사용됩니다.
- version - TLE 확장의 버전입니다.
- description - TLE 확장에 대한 자세한 설명입니다. 이 설명은 pgtle.available_extensions()의 comment 필드에 표시됩니다.
- ext - TLE 확장의 콘텐츠입니다. 이 값에는 함수와 같은 객체가 포함됩니다.
- requires - 이 TLE 확장의 종속성을 지정하는 선택적 파라미터입니다. pg_tle 확장은 종속성으로 자동 추가됩니다.

이러한 인수 중 다수는 PostgreSQL 인스턴스의 파일 시스템에 PostgreSQL 확장을 설치하기 위한 확장 제어 파일에 포함된 인수와 동일합니다. PostgreSQL 확장에 대한 자세한 내용은 PostgreSQL 설명서에서 Packaging Related Objects into an Extension의 Extension Files를 참조하세요.

출력

이 함수는 성공 시 OK를, 오류 시 NULL을 반환합니다.

- OK - TLE 확장이 데이터베이스에 성공적으로 설치되었습니다.
- NULL - TLE 확장이 데이터베이스에 성공적으로 설치되지 않았습니다.

사용 예

```
SELECT pgtle.install_extension(
```

```
 'pg_tle_test',
 '0.1',
 'My first pg_tle extension',
$_pgtle_$
  CREATE FUNCTION my_test()
  RETURNS INT
  AS $$
    SELECT 42;
  $$ LANGUAGE SQL IMMUTABLE;
$_pgtle_$
);
```

pgtle.install_update_path

`install_update_path` 함수는 서로 다른 두 버전의 TLE 확장 간의 업데이트 경로를 제공합니다. 이 함수를 사용하면 TLE 확장의 사용자가 `ALTER EXTENSION ... UPDATE` 구문을 사용하여 버전을 업데이트할 수 있습니다.

함수 프로토타입

```
pgtle.install_update_path(name text, fromvers text, tovers text, ext text)
```

역할

`pgtle_admin`

인수

- `name` - TLE 확장의 이름입니다. 이 값은 `CREATE EXTENSION` 호출에 사용됩니다.
- `fromvers` - 업그레이드를 위한 TLE 확장의 소스 버전입니다.
- `tovers` - 업그레이드를 위한 TLE 확장의 대상 버전입니다.
- `ext` - 업데이트의 콘텐츠입니다. 이 값에는 함수와 같은 객체가 포함됩니다.

출력

없음.

사용 예

```
SELECT pgtle.install_update_path('pg_tle_test', '0.1', '0.2',
```

```
  $_pgtle_$
    CREATE OR REPLACE FUNCTION my_test()
    RETURNS INT
    AS $$
      SELECT 21;
    $$ LANGUAGE SQL IMMUTABLE;
  $_pgtle_$
);
```

pgtle.register_feature

`register_feature` 함수는 지정된 내부 PostgreSQL 기능을 `pgtle.feature_info` 테이블에 추가합니다. PostgreSQL 후크는 내부 PostgreSQL 기능의 예입니다. 신뢰할 수 있는 언어 확장 개발 키트는 PostgreSQL 후크 사용을 지원합니다. 현재 이 함수는 다음 기능을 지원합니다.

- `passcheck` - PostgreSQL의 암호 확인 동작을 사용자 지정하는 프로시저 또는 함수에 암호 확인 후크를 등록합니다.

함수 프로토타입

```
pgtle.register_feature(proc regproc, feature pg_tle_feature)
```

역할

`pgtle_admin`

인수

- `proc` - 기능에 사용할 저장 프로시저 또는 함수의 이름입니다.
- `feature` - 함수에 등록할 `pg_tle` 기능(예: `passcheck`)의 이름입니다.

출력

없음.

사용 예

```
SELECT pgtle.register_feature('pw_hook', 'passcheck');
```

pgtle.register_feature_if_not_exists

`pgtle.register_feature_if_not_exists` 함수는 지정된 PostgreSQL 기능을 `pgtle.feature_info` 테이블에 추가하고 해당 기능을 사용하는 TLE 확장 또는 기타 프로시저나 함수를 식별합니다. 후크 및 신뢰할 수 있는 언어 확장에 대한 자세한 내용은 TLE 확장과 함께 PostgreSQL 후크 사용 섹션을 참조하세요.

함수 프로토타입

```
pgtle.register_feature_if_not_exists(proc regproc, feature pg_tle_feature)
```

역할

`pgtle_admin`

인수

- `proc` - TLE 확장을 위한 기능으로 사용할 로직(코드)이 포함된 저장 프로시저 또는 함수의 이름입니다. `pw_hook` 코드가 그 예입니다.
- `feature` - TLE 함수에 등록할 PostgreSQL 기능의 이름입니다. 현재 사용 가능한 기능은 `passcheck` 후크뿐입니다. 자세한 내용은 암호-확인 후크(passcheck) 단원을 참조하십시오.

출력

지정된 확장에 대한 기능을 등록한 후 `true`를 반환합니다. 기능이 이미 등록된 경우 `false`를 반환합니다.

사용 예

```
SELECT pgtle.register_feature_if_not_exists('pw_hook', 'passcheck');
```

pgtle.set_default_version

`set_default_version` 함수를 사용하면 TLE 확장의 `default_version`을 지정할 수 있습니다. 이 함수를 사용하여 업그레이드 경로를 정의하고 해당 버전을 TLE 확장의 기본값으로 지정할 수 있습니다. 데이터베이스 사용자가 `CREATE EXTENSION` 및 `ALTER EXTENSION ... UPDATE` 명령에서 TLE 확장을 지정하면 해당 버전의 TLE 확장이 해당 사용자의 데이터베이스에 생성됩니다.

이 함수는 성공 시 `true`를 반환합니다. `name` 인수에 지정된 TLE 확장자가 없는 경우에는 함수가 오류를 반환합니다. 마찬가지로 TLE 확장의 `version`이 존재하지 않으면 오류가 반환됩니다.

함수 프로토타입

```
pgtle.set_default_version(name text, version text)
```

역할

pgtle_admin

인수

- name - TLE 확장의 이름입니다. 이 값은 CREATE EXTENSION 호출에 사용됩니다.
- version - 기본값을 설정할 TLE 확장의 버전입니다.

출력

- true - 기본 버전 설정에 성공하면 함수가 true를 반환합니다.
- ERROR - 지정된 이름 또는 버전의 TLE 확장이 존재하지 않는 경우 오류 메시지를 반환합니다.

사용 예

```
SELECT * FROM pgtle.set_default_version('my-extension', '1.1');
```

pgtle.uninstall_extension(name)

uninstall_extension 함수는 데이터베이스에서 TLE 확장의 모든 버전을 제거합니다. 이 함수는 CREATE EXTENSION의 이후 호출이 TLE 확장을 설치하는것을 방지합니다. 데이터베이스에 TLE 확장자가 없으면 오류가 발생합니다.

uninstall_extension 함수는 현재 데이터베이스에서 활성 상태인 TLE 확장은 제거하지 않습니다. 현재 활성 상태인 TLE 확장을 제거하려면 명시적으로 DROP EXTENSION을 호출하여 제거해야 합니다.

함수 프로토타입

```
pgtle.uninstall_extension(extname text)
```

역할

pgtle_admin

인수

- extname - 제거할 TLE 확장의 이름입니다. 이 이름은 지정된 데이터베이스에서 사용할 TLE 확장을 로드하기 위해 CREATE EXTENSION과 함께 사용되는 이름과 같습니다.

출력

없음.

사용 예

```
SELECT * FROM pgtle.uninstall_extension('pg_tle_test');
```

pgtle.uninstall_extension(name, version)

uninstall_extension(name, version) 함수는 지정된 버전의 TLE 확장을 데이터베이스에서 제거합니다. 이 기능은 CREATE EXTENSION 및 ALTER EXTENSION이 TLE 확장을 설치하거나 지정된 버전으로 업데이트하는 것을 방지합니다. 이 함수는 TLE 확장의 모든 업데이트 경로도 제공합니다. 이 함수는 현재 데이터베이스에서 활성 상태인 TLE 확장은 제거하지 않습니다. TLE 확장을 제거하려면 명시적으로 DROP EXTENSION을 호출해야 합니다. TLE 확장의 모든 버전을 제거하려면 pgtle.uninstall_extension(name)을 참조하세요.

함수 프로토타입

```
pgtle.uninstall_extension(extname text, version text)
```

역할

pgtle_admin

인수

- extname - TLE 확장의 이름입니다. 이 값은 CREATE EXTENSION 호출에 사용됩니다.
- version - 데이터베이스에서 제거할 TLE 확장의 버전입니다.

출력

없음.

사용 예

```
SELECT * FROM pgtle.uninstall_extension('pg_tle_test', '0.2');
```

pgtle.uninstall_extension_if_exists

uninstall_extension_if_exists 함수는 지정된 데이터베이스에서 TLE 확장의 모든 버전을 제거합니다. TLE 확장자가 존재하지 않는 경우 함수는 아무것도 반환하지 않습니다(오류 메시지가 표시되지 않음). 지정된 확장이 현재 데이터베이스 내에서 활성 상태인 경우 이 함수는 해당 확장을 삭제하지 않습니다. 이 함수를 사용하여 아티팩트를 제거하려면 먼저 명시적으로 DROP EXTENSION을 호출하여 TLE 확장을 제거해야 합니다.

함수 프로토타입

```
pgtle.uninstall_extension_if_exists(extname text)
```

역할

pgtle_admin

인수

- extname - TLE 확장의 이름입니다. 이 값은 CREATE EXTENSION 호출에 사용됩니다.

출력

uninstall_extension_if_exists 함수는 지정된 확장을 제거한 true를 반환합니다. 지정된 확장이 없는 경우 함수는 false를 반환합니다.

- true - TLE 확장을 제거한 후 true를 반환합니다.
- false - 데이터베이스에 TLE 확장자가 없으면 false를 반환합니다.

사용 예

```
SELECT * FROM pgtle.uninstall_extension_if_exists('pg_tle_test');
```

pgtle.uninstall_update_path

uninstall_update_path 함수는 TLE 확장에서 특정 업데이트 경로를 제거합니다. 이렇게 하면 ALTER EXTENSION ... UPDATE TO가 이 경로를 업데이트 경로로 사용할 수 없습니다.

이 업데이트 경로의 버전 중 하나가 현재 사용 중인 TLE 확장은 데이터베이스에 남아 있습니다.

지정한 업데이트 경로가 존재하지 않는 경우 이 함수는 오류를 반환합니다.

함수 프로토타입

```
pgtle.uninstall_update_path(extname text, fromvers text, tovers text)
```

역할

pgtle_admin

인수

- extname - TLE 확장의 이름입니다. 이 값은 CREATE EXTENSION 호출에 사용됩니다.
- fromvers - 업데이트 경로에서 사용할 TLE 확장의 소스 버전입니다.
- tovers - 업데이트 경로에서 사용할 TLE 확장의 대상 버전입니다.

출력

없음.

사용 예

```
SELECT * FROM pgtle.uninstall_update_path('pg_tle_test', '0.1', '0.2');
```

pgtle.uninstall_update_path_if_exists

uninstall_update_path_if_exists 함수는 TLE 확장에서 지정된 업데이트 경로를 제거한다는 점에서 uninstall_update_path와 비슷합니다. 하지만 업데이트 경로가 존재하지 않는 경우 이 함수는 오류 메시지를 표시하지 않습니다. 대신 함수는 false를 반환합니다.

함수 프로토타입

```
pgtle.uninstall_update_path_if_exists(extname text, fromvers text, tovers text)
```

역할

pgtle_admin

인수

- extname - TLE 확장의 이름입니다. 이 값은 CREATE EXTENSION 호출에 사용됩니다.
- fromvers - 업데이트 경로에서 사용할 TLE 확장의 소스 버전입니다.
- tovers - 업데이트 경로에서 사용할 TLE 확장의 대상 버전입니다.

출력

- true - 함수가 TLE 확장 경로를 성공적으로 업데이트했습니다.
- false - 함수가 TLE 확장의 경로를 업데이트하지 못했습니다.

사용 예

```
SELECT * FROM pgtle.uninstall_update_path_if_exists('pg_tle_test', '0.1', '0.2');
```

pgtle.unregister_feature

unregister_feature 함수는 후크 등 pg_tle 기능을 사용하도록 등록된 함수를 제거하는 방법을 제공합니다. 기능 등록에 대한 자세한 내용은 pgtle.register_feature 섹션을 참조하세요.

함수 프로토타입

```
pgtle.unregister_feature(proc regproc, feature pg_tle_features)
```

역할

pgtle_admin

인수

- proc - pg_tle 기능에 등록할 저장된 함수의 이름입니다.
- feature - 함수에 등록할 pg_tle 기능의 이름입니다. 예를 들어 passcheck는 개발하는 신뢰할 수 있는 언어 확장에서 사용하도록 등록할 수 있는 기능입니다. 자세한 내용은 암호-확인 후크 (passcheck) 단원을 참조하십시오.

출력

없음.

사용 예

```
SELECT * FROM pgtle.unregister_feature('pw_hook', 'passcheck');
```

pgtle.unregister_feature_if_exists

`unregister_feature` 함수는 후크 등 `pg_tle` 기능을 사용하도록 등록된 함수를 제거하는 방법을 제공합니다. 자세한 내용은 TLE 확장과 함께 PostgreSQL 후크 사용 단원을 참조하십시오. 기능을 성공적으로 등록 취소한 후 `true`를 반환합니다. 기능이 등록되지 않은 경우 `false`를 반환합니다.

TLE 확장의 `pg_tle` 기능 등록에 대한 자세한 내용은 pgtle.register_feature 섹션을 참조하세요.

함수 프로토타입

```
pgtle.unregister_feature_if_exists('proc regproc', 'feature pg_tle_features')
```

역할

`pgtle_admin`

인수

- `proc` - `pg_tle` 기능을 포함하도록 등록된 저장된 함수의 이름입니다.
- `feature` - 신뢰할 수 있는 언어 확장에 등록된 `pg_tle` 기능의 이름입니다.

출력

다음과 같이 `true` 또는`false`를 반환합니다.

- `true` - 함수가 확장에서 성공적으로 기능 등록을 취소했습니다.
- `false` - 함수가 TLE 확장에서 기능을 등록 취소하지 못했습니다.

사용 예

```
SELECT * FROM pgtle.unregister_feature_if_exists('pw_hook', 'passcheck');
```

PostgreSQL용 신뢰할 수 있는 언어 확장에 대한 후크 참조

PostgreSQL용 신뢰할 수 있는 언어 확장은 PostgreSQL 후크를 지원합니다. 후크는 PostgreSQL의 핵심 기능을 확장하기 위해 개발자가 사용할 수 있는 내부 콜백 메커니즘입니다. 개발자는 후크를 사용하여 다양한 데이터베이스 작업 중에 사용할 자체 함수 또는 프로시저를 구현하여 PostgreSQL의 동작을 일정 방식으로 수정할 수 있습니다. 예를 들어 passcheck 후크를 사용하면 사용자(역할)의 암호를 생성하거나 변경할 때 제공된 암호를 PostgreSQL이 처리하는 방법을 사용자 지정할 수 있습니다.

TLE 확장에 사용할 수 있는 암호 확인 후크에 대해 알아보려면 다음 설명서를 참조하세요. 클라이언트 인증 후크를 포함하여 사용 가능한 후크에 대한 자세한 내용은 Trusted Language Extensions hooks를 참조하세요.

암호-확인 후크(passcheck)

passcheck 후크는 다음 SQL 명령 및 psql 메타 명령에 대한 암호 확인 프로세스 도중 PostgreSQL 동작을 사용자 지정하는 데 사용됩니다.

- `CREATE ROLE username ...PASSWORD` - 자세한 내용은 PostgreSQL 설명서의 CREATE ROLE을 참조하세요.
- `ALTER ROLE username...PASSWORD` - 자세한 내용은 PostgreSQL 설명서의 ALTER ROLE을 참조하세요.
- `\password username` - 이 대화형 psql 메타 명령은 `ALTER ROLE ... PASSWORD` 구문을 투명하게 사용하기 전에 암호를 해시하여 지정된 사용자의 암호를 안전하게 변경합니다. 메타 명령은 `ALTER ROLE ... PASSWORD` 명령의 보안 래퍼이므로 후크는 psql 메타 명령의 동작에 적용됩니다.

문제 해결 예는 암호 확인 후크 코드 목록을(를) 참조하세요.

목차

- 함수 프로토타입
- 인수
- 구성
- 사용 노트

함수 프로토타입

```
passcheck_hook(username text, password text, password_type pgtle.password_types,
 valid_until timestamptz, valid_null boolean)
```

인수

passcheck 후크 함수는 다음 인수를 사용합니다.

- username - 암호를 설정하는 역할(사용자 이름)의 이름(텍스트)입니다.
- password - 일반 텍스트 또는 해시된 암호입니다. 입력한 암호는 password_type에서 지정한 유형과 일치해야 합니다.
- password_type - 암호의 pgtle.password_type 형식을 지정합니다. 이 형식은 다음 옵션 중 하나일 수 있습니다.
  - PASSWORD_TYPE_PLAINTEXT - 일반 텍스트 암호입니다.
  - PASSWORD_TYPE_MD5 - MD5(message digest 5) 알고리즘을 사용하여 해시된 암호입니다.
  - PASSWORD_TYPE_SCRAM_SHA_256 - SCRAM-SHA-256 알고리즘을 사용하여 해시된 암호입니다.
- valid_until - 암호가 무효가 되는 시간을 지정합니다. 이 인수는 선택 사항입니다. 이 인수를 사용하는 경우 시간을 timestamptz 값으로 지정하세요.
- valid_null - 이 부울이 true로 설정된 경우 valid_until 옵션은 NULL로 설정됩니다.

구성

함수 pgtle.enable_password_check는 암호 확인 후크가 활성 상태인지 여부를 제어합니다. 암호 확인 후크에서 세 가지 설정이 가능합니다.

- off - passcheck 암호 확인 후크를 해제합니다. 이것이 기본값입니다.
- on - 테이블과 비교해 암호를 검사할 수 있도록 passcode 암호 확인 후크를 켭니다.
- require - 암호 확인 후크를 정의해야 합니다.

사용 노트

passcheck 후크를 켜거나 해제하려면 RDS for PostgreSQL DB 인스턴스에 대한 사용자 지정 DB 파라미터 그룹을 수정해야 합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
    --region aws-region \
    --db-parameter-group-name your-custom-parameter-group \
    --parameters
 "ParameterName=pgtle.enable_password_check,ParameterValue=on,ApplyMethod=immediate"
```

Windows의 경우:

```
aws rds modify-db-parameter-group ^
    --region aws-region ^
    --db-parameter-group-name your-custom-parameter-group ^
    --parameters
 "ParameterName=pgtle.enable_password_check,ParameterValue=on,ApplyMethod=immediate"
```

AWS SDK를 사용한 Amazon RDS용 코드 예제

다음 코드 예제에서는 Amazon RDS를 AWS 소프트웨어 개발 키트(SDK)와 함께 사용하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

교차 서비스 예시는 여러 AWS 서비스 전반에서 작동하는 샘플 애플리케이션입니다.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 AWS SDK와 함께 이 서비스 사용을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

시작하기

Hello Amazon RDS

다음 코드 예제에서는 Amazon RDS를 사용하여 시작하는 방법을 보여줍니다.

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.Threading.Tasks;
using Amazon.RDS;
using Amazon.RDS.Model;

namespace RDSActions;

public static class HelloRds
```

```
{
    static async Task Main(string[] args)
    {
        var rdsClient = new AmazonRDSClient();

        Console.WriteLine($"Hello Amazon RDS! Following are some of your DB
 instances:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get the first twenty DB instances.
        var response = await rdsClient.DescribeDBInstancesAsync(
            new DescribeDBInstancesRequest()
            {
                MaxRecords = 20 // Must be between 20 and 100.
            });

        foreach (var instance in response.DBInstances)
        {
            Console.WriteLine($"\tDB name: {instance.DBName}");
            Console.WriteLine($"\tArn: {instance.DBInstanceArn}");
            Console.WriteLine($"\tIdentifier: {instance.DBInstanceIdentifier}");
            Console.WriteLine();
        }
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DescribeDBInstances를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rds)

# Set this project's name.
project("hello_rds")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_rds.cpp)

target_link_libraries(${PROJECT_NAME}
```

```
        ${AWSSDK_LINK_LIBRARIES})
```

hello_rds.cpp 소스 파일의 코드입니다.

```
#include <aws/core/Aws.h>
#include <aws/rds/RDSClient.h>
#include <aws/rds/model/DescribeDBInstancesRequest.h>
#include <iostream>

/*
 *  A "Hello Rds" starter application which initializes an Amazon Relational
 Database Service (Amazon RDS) client and
 *  describes the Amazon RDS instances.
 *
 *  main function
 *
 *  Usage: 'hello_rds'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::RDS::RDSClient rdsClient(clientConfig);
        Aws::String marker;
        std::vector<Aws::String> instanceDBIDs;

        do {
            Aws::RDS::Model::DescribeDBInstancesRequest request;

            if (!marker.empty()) {
                request.SetMarker(marker);
            }
```

```
            Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
                    rdsClient.DescribeDBInstances(request);

            if (outcome.IsSuccess()) {
                for (auto &instance: outcome.GetResult().GetDBInstances()) {
                    instanceDBIDs.push_back(instance.GetDBInstanceIdentifier());
                }
                marker = outcome.GetResult().GetMarker();
            } else {
                result = 1;
                std::cerr << "Error with RDS::DescribeDBInstances. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!marker.empty());

        std::cout << instanceDBIDs.size() << " RDS instances found." <<
 std::endl;
        for (auto &instanceDBID: instanceDBIDs) {
            std::cout << "   Instance: " << instanceDBID << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DescribeDBInstances를 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/rds"
)

// main uses the AWS SDK for Go V2 to create an Amazon Relational Database
 Service (Amazon RDS)
// client and list up to 20 DB instances in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 rdsClient := rds.NewFromConfig(sdkConfig)
 const maxInstances = 20
 fmt.Printf("Let's list up to %v DB instances.\n", maxInstances)
 output, err := rdsClient.DescribeDBInstances(context.TODO(),
  &rds.DescribeDBInstancesInput{MaxRecords: aws.Int32(maxInstances)})
 if err != nil {
  fmt.Printf("Couldn't list DB instances: %v\n", err)
  return
 }
 if len(output.DBInstances) == 0 {
  fmt.Println("No DB instances found.")
 } else {
  for _, instance := range output.DBInstances {
   fmt.Printf("DB instance %v has database %v.\n",
 *instance.DBInstanceIdentifier,
    *instance.DBName)
  }
 }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 DescribeDBInstances를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }
```

```
    public static void describeInstances(RdsClient rdsClient) {
        try {
            DescribeDbInstancesResponse response =
 rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                System.out.println("Instance ARN is: " +
 instance.dbInstanceArn());
                System.out.println("The Engine is " + instance.engine());
                System.out.println("Connection endpoint is" +
 instance.endpoint().address());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeDBInstances를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
"""
Purpose

Shows how to use the AWS SDK for Python (Boto3) with the Amazon Relational
 Database Service
(Amazon RDS) to list the databases in your account.
"""

import boto3
```

```
from botocore.exceptions import ClientError

# Create an RDS client
rds_client = boto3.client("rds")

# Create a paginator for the describe_db_instances operation
paginator = rds_client.get_paginator("describe_db_instances")

try:
    # Use the paginator to get a list of DB instances
    response_iterator = paginator.paginate(
        PaginationConfig={
            "MaxItems": 123,
            "PageSize": 50,  # Adjust PageSize as needed
            "StartingToken": None,
        }
    )

    # Iterate through the pages of the response
    instances_found = False
    for page in response_iterator:
        if "DBInstances" in page and page["DBInstances"]:
            instances_found = True
            print("Your RDS instances are:")
            for db in page["DBInstances"]:
                print(db["DBInstanceIdentifier"])

    if not instances_found:
        print("No RDS instances found!")

except ClientError as e:
    print(f"Couldn't list RDS instances. Here's why: {e.response['Error']
['Message']}")
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 DescribeDBInstances를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-rds'
require 'logger'

# RDSManager is a class responsible for managing RDS operations
# such as listing all RDS DB instances in the current AWS account.
class RDSManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all RDS DB instances in the current AWS account.
  def list_db_instances
    @logger.info('Listing RDS DB instances')

    paginator = @client.describe_db_instances
    instances = []

    paginator.each_page do |page|
      instances.concat(page.db_instances)
    end

    if instances.empty?
      @logger.info('No instances found.')
    else
      @logger.info("Found #{instances.count} instance(s):")
      instances.each do |instance|
        @logger.info(" * #{instance.db_instance_identifier}
 (#{instance.db_instance_status})")
      end
    end
```

```
  end
end

if $PROGRAM_NAME == __FILE__
  rds_client = Aws::RDS::Client.new(region: 'us-west-2')
  manager = RDSManager.new(rds_client)
  manager.list_db_instances
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DescribeDBInstances를 참조하십시오.

코드 예시

- AWS SDK를 사용한 Amazon RDS에 대한 작업
  - AWS SDK 또는 CLI와 함께 CreateDBInstance 사용
  - AWS SDK 또는 CLI와 함께 CreateDBParameterGroup 사용
  - AWS SDK 또는 CLI와 함께 CreateDBSnapshot 사용
  - AWS SDK 또는 CLI와 함께 DeleteDBInstance 사용
  - AWS SDK 또는 CLI와 함께 DeleteDBParameterGroup 사용
  - AWS SDK 또는 CLI와 함께 DescribeAccountAttributes 사용
  - AWS SDK 또는 CLI와 함께 DescribeDBEngineVersions 사용
  - AWS SDK 또는 CLI와 함께 DescribeDBInstances 사용
  - AWS SDK 또는 CLI와 함께 DescribeDBParameterGroups 사용
  - AWS SDK 또는 CLI와 함께 DescribeDBParameters 사용
  - AWS SDK 또는 CLI와 함께 DescribeDBSnapshots 사용
  - AWS SDK 또는 CLI와 함께 DescribeOrderableDBInstanceOptions 사용
  - AWS SDK 또는 CLI와 함께 GenerateRDSAuthToken 사용
  - AWS SDK 또는 CLI와 함께 ModifyDBInstance 사용
  - AWS SDK 또는 CLI와 함께 ModifyDBParameterGroup 사용
  - AWS SDK 또는 CLI와 함께 RebootDBInstance 사용
- AWS SDK를 사용한 Amazon RDS에 대한 시나리오
  - AWS SDK를 사용하여 Amazon RDS DB 인스턴스 시작하기
- AWS SDK를 사용하는 Amazon RDS의 서버리스 예제

  - Lambda 함수를 사용하여 Amazon RDS 데이터베이스에 연결
- AWS SDK를 사용한 Amazon RDS용 교차 서비스 예제
  - Aurora 서버리스 작업 항목 트래커 만들기

AWS SDK를 사용한 Amazon RDS에 대한 작업

다음 코드 예제에서는 AWS SDK를 통해 개별 Amazon RDS 작업을 수행하는 방법을 보여줍니다. 이들 발췌문은 Amazon RDS API를 호출하며, 컨텍스트에서 실행되어야 하는 더 큰 프로그램에서 발췌한 코드입니다. 각 예제에는 GitHub에 대한 링크가 포함되어 있습니다. 여기에서 코드 설정 및 실행에 대한 지침을 찾을 수 있습니다.

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록을 보려면 Amazon Relational Database Service(RDS) API 참조 참조하세요.

예시

- AWS SDK 또는 CLI와 함께 CreateDBInstance 사용
- AWS SDK 또는 CLI와 함께 CreateDBParameterGroup 사용
- AWS SDK 또는 CLI와 함께 CreateDBSnapshot 사용
- AWS SDK 또는 CLI와 함께 DeleteDBInstance 사용
- AWS SDK 또는 CLI와 함께 DeleteDBParameterGroup 사용
- AWS SDK 또는 CLI와 함께 DescribeAccountAttributes 사용
- AWS SDK 또는 CLI와 함께 DescribeDBEngineVersions 사용
- AWS SDK 또는 CLI와 함께 DescribeDBInstances 사용
- AWS SDK 또는 CLI와 함께 DescribeDBParameterGroups 사용
- AWS SDK 또는 CLI와 함께 DescribeDBParameters 사용
- AWS SDK 또는 CLI와 함께 DescribeDBSnapshots 사용
- AWS SDK 또는 CLI와 함께 DescribeOrderableDBInstanceOptions 사용
- AWS SDK 또는 CLI와 함께 GenerateRDSAuthToken 사용
- AWS SDK 또는 CLI와 함께 ModifyDBInstance 사용
- AWS SDK 또는 CLI와 함께 ModifyDBParameterGroup 사용
- AWS SDK 또는 CLI와 함께 RebootDBInstance 사용

AWS SDK 또는 CLI와 함께 **CreateDBInstance** 사용

다음 코드 예제는 CreateDBInstance의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- DB 인스턴스 시작하기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /// <summary>
   /// Create an RDS DB instance with a particular set of properties. Use the
action DescribeDBInstancesAsync
   /// to determine when the DB instance is ready to use.
   /// </summary>
   /// <param name="dbName">Name for the DB instance.</param>
   /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
   /// <param name="parameterGroupName">DB parameter group to associate with the
instance.</param>
   /// <param name="dbEngine">The engine for the DB instance.</param>
   /// <param name="dbEngineVersion">Version for the DB instance.</param>
   /// <param name="instanceClass">Class for the DB instance.</param>
   /// <param name="allocatedStorage">The amount of storage in gibibytes (GiB)
to allocate to the DB instance.</param>
   /// <param name="adminName">Admin user name.</param>
   /// <param name="adminPassword">Admin user password.</param>
   /// <returns>DB instance object.</returns>
   public async Task<DBInstance> CreateDBInstance(string dbName, string
dbInstanceIdentifier,
       string parameterGroupName, string dbEngine, string dbEngineVersion,
```

```
        string instanceClass, int allocatedStorage, string adminName, string
adminPassword)
    {
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBName = dbName,
                DBInstanceIdentifier = dbInstanceIdentifier,
                DBParameterGroupName = parameterGroupName,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass,
                AllocatedStorage = allocatedStorage,
                MasterUsername = adminName,
                MasterUserPassword = adminPassword
            });

        return response.DBInstance;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 CreateDBInstance를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBName(DB_NAME);
```

```
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetEngine(engineVersion.GetEngine());
        request.SetEngineVersion(engineVersion.GetEngineVersion());
        request.SetDBInstanceClass(dbInstanceClass);
        request.SetStorageType(DB_STORAGE_TYPE);
        request.SetAllocatedStorage(DB_ALLOCATED_STORAGE);
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 CreateDBInstance를 참조하십시오.

CLI

AWS CLI

DB 인스턴스를 생성하려면

다음 `create-db-instance` 예제에서는 필수 옵션을 사용하여 새 DB 인스턴스를 시작합니다.

```
aws rds create-db-instance \
    --db-instance-identifier test-mysql-instance \
    --db-instance-class db.t3.micro \
    --engine mysql \
    --master-username admin \
    --master-user-password secret99 \
```

```
    --allocated-storage 20
```

출력:

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "test-mysql-instance",
        "DBInstanceClass": "db.t3.micro",
        "Engine": "mysql",
        "DBInstanceStatus": "creating",
        "MasterUsername": "admin",
        "AllocatedStorage": 20,
        "PreferredBackupWindow": "12:55-13:25",
        "BackupRetentionPeriod": 1,
        "DBSecurityGroups": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-12345abc",
                "Status": "active"
            }
        ],
        "DBParameterGroups": [
            {
                "DBParameterGroupName": "default.mysql5.7",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-2ff2ff2f",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
                {
                    "SubnetIdentifier": "subnet-########",
                    "SubnetAvailabilityZone": {
                        "Name": "us-west-2c"
                    },
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-########",
                    "SubnetAvailabilityZone": {
```

```
                    "Name": "us-west-2d"
                },
                "SubnetStatus": "Active"
            },
            {
                "SubnetIdentifier": "subnet-########",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2a"
                },
                "SubnetStatus": "Active"
            },
            {
                "SubnetIdentifier": "subnet-########",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2b"
                },
                "SubnetStatus": "Active"
            }
        ]
    },
    "PreferredMaintenanceWindow": "sun:08:07-sun:08:37",
    "PendingModifiedValues": {
        "MasterUserPassword": "****"
    },
    "MultiAZ": false,
    "EngineVersion": "5.7.22",
    "AutoMinorVersionUpgrade": true,
    "ReadReplicaDBInstanceIdentifiers": [],
    "LicenseModel": "general-public-license",
    "OptionGroupMemberships": [
        {
            "OptionGroupName": "default:mysql-5-7",
            "Status": "in-sync"
        }
    ],
    "PubliclyAccessible": true,
    "StorageType": "gp2",
    "DbInstancePort": 0,
    "StorageEncrypted": false,
    "DbiResourceId": "db-5555EXAMPLE44444444EXAMPLE",
    "CACertificateIdentifier": "rds-ca-2019",
    "DomainMemberships": [],
    "CopyTagsToSnapshot": false,
    "MonitoringInterval": 0,
```

```
        "DBInstanceArn": "arn:aws:rds:us-west-2:123456789012:db:test-mysql-
instance",
        "IAMDatabaseAuthenticationEnabled": false,
        "PerformanceInsightsEnabled": false,
        "DeletionProtection": false,
        "AssociatedRoles": []
    }
}
```

자세한 내용은 Amazon RDS 사용 설명서의 Amazon RDS DB 인스턴스 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateDBInstance를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
type DbInstances struct {
 RdsClient *rds.Client
}



// CreateInstance creates a DB instance.
func (instances *DbInstances) CreateInstance(instanceName string, dbName string,
 dbEngine string, dbEngineVersion string, parameterGroupName string,
 dbInstanceClass string,
 storageType string, allocatedStorage int32, adminName string, adminPassword
 string) (
 *types.DBInstance, error) {
 output, err := instances.RdsClient.CreateDBInstance(context.TODO(),
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBName:               aws.String(dbName),
  DBParameterGroupName: aws.String(parameterGroupName),
```

```
  Engine:               aws.String(dbEngine),
  EngineVersion:        aws.String(dbEngineVersion),
  DBInstanceClass:      aws.String(dbInstanceClass),
  StorageType:          aws.String(storageType),
  AllocatedStorage:     aws.Int32(allocatedStorage),
  MasterUsername:       aws.String(adminName),
  MasterUserPassword:   aws.String(adminPassword),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 CreateDBInstance를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import com.google.gson.Gson;
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
```

```
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For more details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 *
 */

public class CreateDBInstance {
    public static long sleepTime = 20;

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier> <dbName> <secretName>

                Where:
                    dbInstanceIdentifier - The database instance identifier.\s
                    dbName - The database name.\s
                    secretName - The name of the AWS Secrets Manager secret that
 contains the database credentials."
                """;

        if (args.length != 3) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String dbName = args[1];
        String secretName = args[2];
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        createDatabaseInstance(rdsClient, dbInstanceIdentifier, dbName,
user.getUsername(), user.getPassword());
        waitForInstanceReady(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
                .region(region)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                .build();
    }

    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                .secretId(secretName)
                .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    public static void createDatabaseInstance(RdsClient rdsClient,
            String dbInstanceIdentifier,
            String dbName,
            String userName,
```

```
            String userPassword) {

        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .allocatedStorage(100)
                    .dbName(dbName)
                    .engine("mysql")
                    .dbInstanceClass("db.m4.large")
                    .engineVersion("8.0")
                    .storageType("standard")
                    .masterUsername(userName)
                    .masterUserPassword(userPassword)
                    .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            // Loop until the cluster is ready.
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
```

```
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available"))
                        instanceReady = true;
                    else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available!");

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateDBInstance를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createDatabaseInstance(
    dbInstanceIdentifierVal: String?,
    dbNamedbVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNamedbVal
```

```
            engine = "mysql"
            dbInstanceClass = "db.m4.large"
            engineVersion = "8.0"
            storageType = "standard"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
    }
}

// Waits until the database instance is available.
suspend fun waitForInstanceReady(dbInstanceIdentifierVal: String?) {
    val sleepTime: Long = 20
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus.toString()
                    if (instanceReadyStr.contains("available")) {
                        instanceReady = true
                    } else {
                        println("...$instanceReadyStr")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
        println("Database instance is available!")
    }
```

```
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateDBInstance를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;



$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

$dbIdentifier = '<<{{db-identifier}}>>';
$dbClass = 'db.t2.micro';
$storage = 5;
$engine = 'MySQL';
$username = 'MyUser';
$password = 'MyPassword';

try {
    $result = $rdsClient->createDBInstance([
        'DBInstanceIdentifier' => $dbIdentifier,
        'DBInstanceClass' => $dbClass,
        'AllocatedStorage' => $storage,
        'Engine' => $engine,
        'MasterUsername' => $username,
        'MasterUserPassword' => $password,
```

```
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 CreateDBInstance를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_db_instance(
        self,
        db_name,
```

```
        instance_id,
        parameter_group_name,
        db_engine,
        db_engine_version,
        instance_class,
        storage_type,
        allocated_storage,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB instance.

        :param db_name: The name of the database that is created in the DB
instance.
        :param instance_id: The ID to give the newly created DB instance.
        :param parameter_group_name: A parameter group to associate with the DB
instance.
        :param db_engine: The database engine of a database to create in the DB
instance.
        :param db_engine_version: The engine version for the created database.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :param storage_type: The storage type of the DB instance.
        :param allocated_storage: The amount of storage allocated on the DB
instance, in GiBs.
        :param admin_name: The name of the admin user for the created database.
        :param admin_password: The admin password for the created database.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBName=db_name,
                DBInstanceIdentifier=instance_id,
                DBParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
                DBInstanceClass=instance_class,
                StorageType=storage_type,
                AllocatedStorage=allocated_storage,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            db_inst = response["DBInstance"]
```

```
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- API 세부 정보는 Python용 AWS SDK(Boto3) API 참조의 CreateDBInstance를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **CreateDBParameterGroup** 사용

다음 코드 예제는 CreateDBParameterGroup의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- DB 인스턴스 시작하기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
```

```
    /// Create a new DB parameter group. Use the action
DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="family">Family of the DB parameter group.</param>
    /// <param name="description">Description of the DB parameter group.</param>
    /// <returns>The new DB parameter group.</returns>
    public async Task<DBParameterGroup> CreateDBParameterGroup(
        string name, string family, string description)
    {
        var response = await _amazonRDS.CreateDBParameterGroupAsync(
            new CreateDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                DBParameterGroupFamily = family,
                Description = description
            });
        return response.DBParameterGroup;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 CreateDBParameterGroup을 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBParameterGroupRequest request;
```

```
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example parameter group.");

        Aws::RDS::Model::CreateDBParameterGroupOutcome outcome =
                client.CreateDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 CreateDBParameterGroup을 참조하십시오.

CLI

AWS CLI

DB 파라미터 그룹을 생성하려면

다음 `create-db-parameter-group` 예제에서는 DB 파라미터 그룹을 생성합니다.

```
aws rds create-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --db-parameter-group-family MySQL5.6 \
    --description "My new parameter group"
```

출력:

```
{
    "DBParameterGroup": {
        "DBParameterGroupName": "mydbparametergroup",
        "DBParameterGroupFamily": "mysql5.6",
        "Description": "My new parameter group",
        "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:mydbparametergroup"
```

```
    }
}
```

자세한 내용은 Amazon RDS 사용 설명서의 DB 파라미터 그룹 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateDBParameterGroup을 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
type DbInstances struct {
 RdsClient *rds.Client
}



// CreateParameterGroup creates a DB parameter group that is based on the
 specified
// parameter group family.
func (instances *DbInstances) CreateParameterGroup(
 parameterGroupName string, parameterGroupFamily string, description string) (
 *types.DBParameterGroup, error) {

 output, err := instances.RdsClient.CreateDBParameterGroup(context.TODO(),
  &rds.CreateDBParameterGroupInput{
   DBParameterGroupName:   aws.String(parameterGroupName),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
   Description:            aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBParameterGroup, err
```

```
 }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 CreateDBParameterGroup을 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
       try {
           CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
                   .dbParameterGroupName(dbGroupName)
                   .dbParameterGroupFamily(dbParameterGroupFamily)
                   .description("Created by using the AWS SDK for Java")
                   .build();

           CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
           System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateDBParameterGroup을 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB parameter group that is based on the specified parameter
 group
        family.

        :param parameter_group_name: The name of the newly created parameter
 group.
        :param parameter_group_family: The family that is used as the basis of
 the new
                                       parameter group.
        :param description: A description given to the parameter group.
```

```
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- API 세부 정보는 Python용 AWS SDK(Boto3) API 참조의 CreateDBParameterGroup을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **CreateDBSnapshot** 사용

다음 코드 예제는 CreateDBSnapshot의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- DB 인스턴스 시작하기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Create a snapshot of a DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBSnapshot> CreateDBSnapshot(string dbInstanceIdentifier,
string snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBSnapshotAsync(
            new CreateDBSnapshotRequest()
            {
                DBSnapshotIdentifier = snapshotIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        return response.DBSnapshot;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 CreateDBSnapshot을 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::CreateDBSnapshotRequest request;
            request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBSnapshotOutcome outcome =
                    client.CreateDBSnapshot(request);

            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with RDS::CreateDBSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 CreateDBSnapshot을 참조하십시오.

CLI

AWS CLI

DB 스냅샷을 생성하려면

다음 예제에서는 DB 스냅샷을 생성합니다.

```
aws rds create-db-snapshot \
    --db-instance-identifier database-mysql \
    --db-snapshot-identifier mydbsnapshot
```

출력:

```
{
    "DBSnapshot": {
        "DBSnapshotIdentifier": "mydbsnapshot",
        "DBInstanceIdentifier": "database-mysql",
        "Engine": "mysql",
        "AllocatedStorage": 100,
        "Status": "creating",
        "Port": 3306,
        "AvailabilityZone": "us-east-1b",
        "VpcId": "vpc-6594f31c",
        "InstanceCreateTime": "2019-04-30T15:45:53.663Z",
        "MasterUsername": "admin",
        "EngineVersion": "5.6.40",
        "LicenseModel": "general-public-license",
        "SnapshotType": "manual",
        "Iops": 1000,
        "OptionGroupName": "default:mysql-5-6",
        "PercentProgress": 0,
        "StorageType": "io1",
        "Encrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
AKIAIOSFODNN7EXAMPLE",
        "DBSnapshotArn": "arn:aws:rds:us-
east-1:123456789012:snapshot:mydbsnapshot",
        "IAMDatabaseAuthenticationEnabled": false,
        "ProcessorFeatures": [],
        "DbiResourceId": "db-AKIAIOSFODNN7EXAMPLE"
    }
}
```

자세한 내용은 Amazon RDS 사용 설명서의 DB 스냅샷 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateDBSnapshot을 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
type DbInstances struct {
 RdsClient *rds.Client
}



// CreateSnapshot creates a snapshot of a DB instance.
func (instances *DbInstances) CreateSnapshot(instanceName string, snapshotName
 string) (
 *types.DBSnapshot, error) {
 output, err := instances.RdsClient.CreateDBSnapshot(context.TODO(),
 &rds.CreateDBSnapshotInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBSnapshot, nil
 }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 CreateDBSnapshot을 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   // Create an Amazon RDS snapshot.
   public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
       try {
           CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .dbSnapshotIdentifier(dbSnapshotIdentifier)
                   .build();

           CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
           System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateDBSnapshot을 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;



$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

$dbIdentifier = '<<{{db-identifier}}>>';
$snapshotName = '<<{{backup_2018_12_25}}>>';

try {
    $result = $rdsClient->createDBSnapshot([
        'DBInstanceIdentifier' => $dbIdentifier,
        'DBSnapshotIdentifier' => $snapshotName,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 CreateDBSnapshot을 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_snapshot(self, snapshot_id, instance_id):
        """
        Creates a snapshot of a DB instance.

        :param snapshot_id: The ID to give the created snapshot.
        :param instance_id: The ID of the DB instance to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_snapshot(
                DBSnapshotIdentifier=snapshot_id,
 DBInstanceIdentifier=instance_id
            )
            snapshot = response["DBSnapshot"]
```

```
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 CreateDBSnapshot를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require "aws-sdk-rds"  # v2: require 'aws-sdk'

# Create a snapshot for an Amazon Relational Database Service (Amazon RDS)
# DB instance.
#
# @param rds_resource [Aws::RDS::Resource] The resource containing SDK logic.
# @param db_instance_name [String] The name of the Amazon RDS DB instance.
# @return [Aws::RDS::DBSnapshot, nil] The snapshot created, or nil if error.
def create_snapshot(rds_resource, db_instance_name)
  id = "snapshot-#{rand(10**6)}"
  db_instance = rds_resource.db_instance(db_instance_name)
  db_instance.create_snapshot({
                                db_snapshot_identifier: id
                              })
rescue Aws::Errors::ServiceError => e
```

```
  puts "Couldn't create DB instance snapshot #{id}:\n #{e.message}"
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateDBSnapshot을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteDBInstance** 사용

다음 코드 예제는 DeleteDBInstance의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- DB 인스턴스 시작하기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Delete a particular DB instance.
/// </summary>
/// <param name="dbInstanceIdentifier">DB instance identifier.</param>
/// <returns>DB instance object.</returns>
public async Task<DBInstance> DeleteDBInstance(string dbInstanceIdentifier)
{
    var response = await _amazonRDS.DeleteDBInstanceAsync(
        new DeleteDBInstanceRequest()
        {
            DBInstanceIdentifier = dbInstanceIdentifier,
```

```
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });

        return response.DBInstance;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DeleteDBInstance를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with RDS::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
```

```
                        << std::endl;
                result = false;
            }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DeleteDBInstance를 참조하십시오.

CLI

AWS CLI

DB 인스턴스를 삭제하려면

다음 delete-db-instance 예제에서는 test-instance-final-snap이라는 최종 DB 스냅샷을 만든 후 지정된 DB 인스턴스를 삭제합니다.

```
aws rds delete-db-instance \
    --db-instance-identifier test-instance \
    --final-db-snapshot-identifier test-instance-final-snap
```

출력:

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "test-instance",
        "DBInstanceStatus": "deleting",
        ...some output truncated...
    }
}
```

- API 세부 정보는 AWS CLI 명령 참조의 DeleteDBInstance를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
type DbInstances struct {
 RdsClient *rds.Client
}



// DeleteInstance deletes a DB instance.
func (instances *DbInstances) DeleteInstance(instanceName string) error {
 _, err := instances.RdsClient.DeleteDBInstance(context.TODO(),
 &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      true,
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 DeleteDBInstance를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
```

```
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteDBInstance {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier>\s

                Where:
                    dbInstanceIdentifier - The database instance identifier\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    public static void deleteDatabaseInstance(RdsClient rdsClient, String
 dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
 DeleteDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .deleteAutomatedBackups(true)
```

```
                    .skipFinalSnapshot(true)
                    .build();

            DeleteDbInstanceResponse response =
 rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.print("The status of the database is " +
 response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteDBInstance를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteDatabaseInstance(dbInstanceIdentifierVal: String?) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteDBInstance를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;


//Create an RDSClient
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-1'
]);

$dbIdentifier = '<<{{db-identifier}}>>';

try {
    $result = $rdsClient->deleteDBInstance([
        'DBInstanceIdentifier' => $dbIdentifier,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 DeleteDBInstance를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
```

```
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- API 세부 정보는 Python용 AWS SDK(Boto3) API 참조의 DeleteDBInstance를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteDBParameterGroup** 사용

다음 코드 예제는 DeleteDBParameterGroup의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- DB 인스턴스 시작하기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
```

```
    /// Delete a DB parameter group. The group cannot be a default DB parameter
group
    /// or be associated with any DB instances.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDBParameterGroup(string name)
    {
        var response = await _amazonRDS.DeleteDBParameterGroupAsync(
            new DeleteDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DeleteDBParameterGroup을 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DeleteDBParameterGroupRequest request;
        request.SetDBParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBParameterGroupOutcome outcome =
                client.DeleteDBParameterGroup(request);
```

```
        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::DeleteDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DeleteDBParameterGroup을 참조하십시오.

CLI

AWS CLI

DB 파라미터 그룹을 삭제하려면

다음 command 예제에서는 DB 파라미터 그룹을 삭제합니다.

```
aws rds delete-db-parameter-group \
    --db-parameter-group-name mydbparametergroup
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용을 알아보려면 Amazon RDS 사용 설명서의 DB 파라미터 그룹 작업을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteDBParameterGroup을 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
type DbInstances struct {
 RdsClient *rds.Client
}



// DeleteParameterGroup deletes the named DB parameter group.
func (instances *DbInstances) DeleteParameterGroup(parameterGroupName string)
 error {
 _, err := instances.RdsClient.DeleteDBParameterGroup(context.TODO(),
  &rds.DeleteDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 DeleteDBParameterGroup을 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    // Delete the parameter group after database has been deleted.
    // An exception is thrown if you attempt to delete the para group while
 database
    // exists.
```

```
   public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
           throws InterruptedException {
       try {
           boolean isDataDel = false;
           boolean didFind;
           String instanceARN;

           // Make sure that the database has been deleted.
           while (!isDataDel) {
               DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
               List<DBInstance> instanceList = response.dbInstances();
               int listSize = instanceList.size();
               didFind = false;
               int index = 1;
               for (DBInstance instance : instanceList) {
                   instanceARN = instance.dbInstanceArn();
                   if (instanceARN.compareTo(dbARN) == 0) {
                       System.out.println(dbARN + " still exists");
                       didFind = true;
                   }
                   if ((index == listSize) && (!didFind)) {
                       // Went through the entire list and did not find the
database ARN.
                       isDataDel = true;
                   }
                   Thread.sleep(sleepTime * 1000);
                   index++;
               }
           }

           // Delete the para group.
           DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
                   .dbParameterGroupName(dbGroupName)
                   .build();

           rdsClient.deleteDBParameterGroup(parameterGroupRequest);
           System.out.println(dbGroupName + " was deleted.");

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
```

```
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteDBParameterGroup을 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
```

```
        """
        try:
            self.rds_client.delete_db_parameter_group(
                DBParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API 세부 정보는 Python용 AWS SDK(Boto3) API 참조의 DeleteDBParameterGroup을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DescribeAccountAttributes** 사용

다음 코드 예제는 DescribeAccountAttributes의 사용 방법을 보여 줍니다.

CLI

AWS CLI

계정 속성을 설명하려면

다음 describe-account-attributes 예제에서는 현재 AWS 계정의 속성을 검색합니다.

```
aws rds describe-account-attributes
```

출력:

```
{
    "AccountQuotas": [
        {
            "Max": 40,
```

```
            "Used": 4,
            "AccountQuotaName": "DBInstances"
        },
        {
            "Max": 40,
            "Used": 0,
            "AccountQuotaName": "ReservedDBInstances"
        },
        {
            "Max": 100000,
            "Used": 40,
            "AccountQuotaName": "AllocatedStorage"
        },
        {
            "Max": 25,
            "Used": 0,
            "AccountQuotaName": "DBSecurityGroups"
        },
        {
            "Max": 20,
            "Used": 0,
            "AccountQuotaName": "AuthorizationsPerDBSecurityGroup"
        },
        {
            "Max": 50,
            "Used": 1,
            "AccountQuotaName": "DBParameterGroups"
        },
        {
            "Max": 100,
            "Used": 3,
            "AccountQuotaName": "ManualSnapshots"
        },
        {
            "Max": 20,
            "Used": 0,
            "AccountQuotaName": "EventSubscriptions"
        },
        {
            "Max": 50,
            "Used": 1,
            "AccountQuotaName": "DBSubnetGroups"
        },
        {
```

```
            "Max": 20,
            "Used": 1,
            "AccountQuotaName": "OptionGroups"
        },
        {
            "Max": 20,
            "Used": 6,
            "AccountQuotaName": "SubnetsPerDBSubnetGroup"
        },
        {
            "Max": 5,
            "Used": 0,
            "AccountQuotaName": "ReadReplicasPerMaster"
        },
        {
            "Max": 40,
            "Used": 1,
            "AccountQuotaName": "DBClusters"
        },
        {
            "Max": 50,
            "Used": 0,
            "AccountQuotaName": "DBClusterParameterGroups"
        },
        {
            "Max": 5,
            "Used": 0,
            "AccountQuotaName": "DBClusterRoles"
        }
    ]
}
```

- API 세부 정보는 AWS CLI 명령 참조의 DescribeAccountAttributes를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.AccountQuota;
import software.amazon.awssdk.services.rds.model.RdsException;
import
 software.amazon.awssdk.services.rds.model.DescribeAccountAttributesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeAccountAttributes {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        getAccountAttributes(rdsClient);
        rdsClient.close();
    }

    public static void getAccountAttributes(RdsClient rdsClient) {
        try {
            DescribeAccountAttributesResponse response =
 rdsClient.describeAccountAttributes();
            List<AccountQuota> quotasList = response.accountQuotas();
            for (AccountQuota quotas : quotasList) {
                System.out.println("Name is: " + quotas.accountQuotaName());
                System.out.println("Max value is " + quotas.max());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

```
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeAccountAttributes를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getAccountAttributes() {
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeAccountAttributes(DescribeAccountAttributesRequest {})
        response.accountQuotas?.forEach { quotas ->
            val response = response.accountQuotas
            println("Name is: ${quotas.accountQuotaName}")
            println("Max value is ${quotas.max}")
        }
    }
}
```

- API 세부 정보는 Kotlin용 AWS SDK API 참조의 DescribeAccountAttributes를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DescribeDBEngineVersions** 사용

다음 코드 예제는 DescribeDBEngineVersions의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- DB 인스턴스 시작하기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="dbParameterGroupFamily">Optional parameter group family
name.</param>
    /// <returns>List of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>> DescribeDBEngineVersions(string
engine,
        string dbParameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = dbParameterGroupFamily
            });
        return response.DBEngineVersions;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DescribeDBEngineVersions을 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBEngineVersions(const Aws::String &engineName,
                                      const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                      const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // Used for pagination.
```

```
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }


        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            auto &engineVersions = outcome.GetResult().GetDBEngineVersions();
            engineVersionsResult.insert(engineVersionsResult.end(),
 engineVersions.begin(),
                                        engineVersions.end());
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }

    } while (!marker.empty());


    return true;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DescribeDBEngineVersions을 참조하십시오.

CLI

AWS CLI

MySQL DB 엔진의 DB 엔진 버전을 설명하려면

다음 `describe-db-engine-versions` 예제는 지정된 DB 엔진의 각 DB 엔진 버전에 대한 세부 정보를 표시합니다.

```
aws rds describe-db-engine-versions \
```

```
    --engine mysql
```

출력:

```
{
    "DBEngineVersions": [
        {
            "Engine": "mysql",
            "EngineVersion": "5.5.46",
            "DBParameterGroupFamily": "mysql5.5",
            "DBEngineDescription": "MySQL Community Edition",
            "DBEngineVersionDescription": "MySQL 5.5.46",
            "ValidUpgradeTarget": [
                {
                    "Engine": "mysql",
                    "EngineVersion": "5.5.53",
                    "Description": "MySQL 5.5.53",
                    "AutoUpgrade": false,
                    "IsMajorVersionUpgrade": false
                },
                {
                    "Engine": "mysql",
                    "EngineVersion": "5.5.54",
                    "Description": "MySQL 5.5.54",
                    "AutoUpgrade": false,
                    "IsMajorVersionUpgrade": false
                },
                {
                    "Engine": "mysql",
                    "EngineVersion": "5.5.57",
                    "Description": "MySQL 5.5.57",
                    "AutoUpgrade": false,
                    "IsMajorVersionUpgrade": false
                },
                ...some output truncated...
            ]
        }
```

자세한 내용은 Amazon RDS 사용 설명서의 Amazon Relational Database Service(RDS)란 무엇인가요?를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DescribeDBEngineVersions를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
type DbInstances struct {
 RdsClient *rds.Client
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (instances *DbInstances) GetEngineVersions(engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := instances.RdsClient.DescribeDBEngineVersions(context.TODO(),
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 DescribeDBEngineVersions을 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public static void describeDBEngines(RdsClient rdsClient) {
       try {
           DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                   .defaultOnly(true)
                   .engine("mysql")
                   .maxRecords(20)
                   .build();

           DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
           List<DBEngineVersion> engines = response.dbEngineVersions();

           // Get all DBEngineVersion objects.
           for (DBEngineVersion engineOb : engines) {
               System.out.println("The name of the DB parameter group family for
the database engine is "
                       + engineOb.dbParameterGroupFamily());
               System.out.println("The name of the database engine " +
engineOb.engine());
               System.out.println("The version number of the database engine " +
engineOb.engineVersion());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeDBEngineVersions을 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned
 list of
                                       engine versions to those that are
 compatible with
```

```
                                    this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions
```

- API 세부 정보는 Python(Boto3)용 AWS SDK API 참조의 DescribeDBEngineVersions를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DescribeDBInstances** 사용

다음 코드 예제는 DescribeDBInstances의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- DB 인스턴스 시작하기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Returns a list of DB instances.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
    /// <returns>List of DB instances.</returns>
    public async Task<List<DBInstance>> DescribeDBInstances(string
dbInstanceIdentifier = null)
    {
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });
        // Get the entire list using the paginator.
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DescribeDBInstances를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets a DB instance description.
/*!
 \sa describeDBInstance()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                     Aws::RDS::Model::DBInstance &instanceResult,
                                     const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with RDS::DescribeDBInstances. "
```

```
                    << outcome.GetError().GetMessage()
                    << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DescribeDBInstances를 참조하십시오.

CLI

AWS CLI

DB 인스턴스를 설명하려면

다음 `describe-db-instances` 예제에서는 지정된 DB 인스턴스에 대한 세부 정보를 검색합니다.

```
aws rds describe-db-instances \
    --db-instance-identifier mydbinstancecf
```

출력:

```
{
    "DBInstances": [
        {
            "DBInstanceIdentifier": "mydbinstancecf",
            "DBInstanceClass": "db.t3.small",
            "Engine": "mysql",
            "DBInstanceStatus": "available",
            "MasterUsername": "masterawsuser",
            "Endpoint": {
                "Address": "mydbinstancecf.abcexample.us-
east-1.rds.amazonaws.com",
                "Port": 3306,
                "HostedZoneId": "Z2R2ITUGPM61AM"
```

```
            },
            ...some output truncated...
        }
    ]
}
```

- API 세부 정보는 AWS CLI 명령 참조의 DescribeDBInstances를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
type DbInstances struct {
 RdsClient *rds.Client
}



// GetInstance gets data about a DB instance.
func (instances *DbInstances) GetInstance(instanceName string) (
 *types.DBInstance, error) {
 output, err := instances.RdsClient.DescribeDBInstances(context.TODO(),
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
```

```
  return &output.DBInstances[0], nil
 }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 DescribeDBInstances를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();
```

```
        describeInstances(rdsClient);
        rdsClient.close();
    }

    public static void describeInstances(RdsClient rdsClient) {
        try {
            DescribeDbInstancesResponse response =
 rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                System.out.println("Instance ARN is: " +
 instance.dbInstanceArn());
                System.out.println("The Engine is " + instance.engine());
                System.out.println("Connection endpoint is" +
 instance.endpoint().address());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeDBInstances를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun describeInstances() {
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbInstances(DescribeDbInstancesRequest
 {})
        response.dbInstances?.forEach { instance ->
```

```
            println("Instance Identifier is ${instance.dbInstanceIdentifier}")
            println("The Engine is ${instance.engine}")
            println("Connection endpoint is ${instance.endpoint?.address}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeDBInstances를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;


//Create an RDSClient
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

try {
    $result = $rdsClient->describeDBInstances();
    foreach ($result['DBInstances'] as $instance) {
        print('<p>DB Identifier: ' . $instance['DBInstanceIdentifier']);
        print('<br />Endpoint: ' . $instance['Endpoint']["Address"]
            . ':' . $instance['Endpoint']["Port"]);
        print('<br />Current Status: ' . $instance["DBInstanceStatus"]);
        print('</p>');
    }
    print(" Raw Result ");
    var_dump($result);
```

```
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 DescribeDBInstances를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
```

```
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 DescribeDBInstances를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require "aws-sdk-rds"  # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) DB instances.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
```

```
# @return [Array, nil] List of all DB instances, or nil if error.
def list_instances(rds_resource)
  db_instances = []
  rds_resource.db_instances.each do |i|
    db_instances.append({
                          "name": i.id,
                          "status": i.db_instance_status
                        })
  end
  db_instances
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list instances:\n#{e.message}"
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DescribeDBInstances를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DescribeDBParameterGroups** 사용

다음 코드 예제는 DescribeDBParameterGroups의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- DB 인스턴스 시작하기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Get descriptions of DB parameter groups.
    /// </summary>
    /// <param name="name">Optional name of the DB parameter group to describe.</
param>
    /// <returns>The list of DB parameter group descriptions.</returns>
    public async Task<List<DBParameterGroup>> DescribeDBParameterGroups(string
 name = null)
    {
        var response = await _amazonRDS.DescribeDBParameterGroupsAsync(
            new DescribeDBParameterGroupsRequest()
            {
                DBParameterGroupName = name
            });
        return response.DBParameterGroups;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DescribeDBParameterGroups를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DescribeDBParameterGroupsRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
```

```
        Aws::RDS::Model::DescribeDBParameterGroupsOutcome outcome =
                client.DescribeDBParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB parameter group named '" <<
                      PARAMETER_GROUP_NAME << "' already exists." << std::endl;
            dbParameterGroupFamily = outcome.GetResult().GetDBParameterGroups()
[0].GetDBParameterGroupFamily();
        }

        else {
            std::cerr << "Error with RDS::DescribeDBParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DescribeDBParameterGroups를 참조하십시오.

CLI

AWS CLI

DB 파라미터 그룹을 설명하려면

다음 `describe-db-parameter-groups` 예제에서는 DB 파라미터 그룹에 대한 세부 정보를 검색합니다.

```
aws rds describe-db-parameter-groups
```

출력:

```
{
    "DBParameterGroups": [
        {
            "DBParameterGroupName": "default.aurora-mysql5.7",
            "DBParameterGroupFamily": "aurora-mysql5.7",
            "Description": "Default parameter group for aurora-mysql5.7",
            "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:default.aurora-mysql5.7"
```

```
        },
        {
            "DBParameterGroupName": "default.aurora-postgresql9.6",
            "DBParameterGroupFamily": "aurora-postgresql9.6",
            "Description": "Default parameter group for aurora-postgresql9.6",
            "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:default.aurora-postgresql9.6"
        },
        {
            "DBParameterGroupName": "default.aurora5.6",
            "DBParameterGroupFamily": "aurora5.6",
            "Description": "Default parameter group for aurora5.6",
            "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:default.aurora5.6"
        },
        {
            "DBParameterGroupName": "default.mariadb10.1",
            "DBParameterGroupFamily": "mariadb10.1",
            "Description": "Default parameter group for mariadb10.1",
            "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:default.mariadb10.1"
        },
        ...some output truncated...
    ]
}
```

자세한 내용을 알아보려면 Amazon RDS 사용 설명서의 DB 파라미터 그룹 작업을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DescribeDBParameterGroups를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
type DbInstances struct {
```

```
 RdsClient *rds.Client
}



// GetParameterGroup gets a DB parameter group by name.
func (instances *DbInstances) GetParameterGroup(parameterGroupName string) (
 *types.DBParameterGroup, error) {
 output, err := instances.RdsClient.DescribeDBParameterGroups(
  context.TODO(), &rds.DescribeDBParameterGroupsInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBParameterGroups[0], err
 }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 DescribeDBParameterGroups를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
       try {
           DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
                   .dbParameterGroupName(dbGroupName)
                   .maxRecords(20)
                   .build();

           DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
           List<DBParameterGroup> groups = response.dbParameterGroups();
           for (DBParameterGroup group : groups) {
               System.out.println("The group name is " +
group.dbParameterGroupName());
               System.out.println("The group description is " +
group.description());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeDBParameterGroups를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
```

```
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The parameter group.
        """
        try:
            response = self.rds_client.describe_db_parameter_groups(
                DBParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 DescribeDBParameterGroups를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require "aws-sdk-rds"  # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) parameter groups.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all parameter groups, or nil if error.
def list_parameter_groups(rds_resource)
  parameter_groups = []
  rds_resource.db_parameter_groups.each do |p|
    parameter_groups.append({
                              "name": p.db_parameter_group_name,
                              "description": p.description
                            })
  end
  parameter_groups
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list parameter groups:\n #{e.message}"
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DescribeDBParameterGroups를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DescribeDBParameters** 사용

다음 코드 예제는 DescribeDBParameters의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- DB 인스턴스 시작하기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Get a list of DB parameters from a specific parameter group.
    /// </summary>
    /// <param name="dbParameterGroupName">Name of a specific DB parameter
group.</param>
    /// <param name="source">Optional source for selecting parameters.</param>
    /// <returns>List of parameter values.</returns>
    public async Task<List<Parameter>> DescribeDBParameters(string
dbParameterGroupName, string source = null)
    {
        var results = new List<Parameter>();
        var paginateParameters = _amazonRDS.Paginators.DescribeDBParameters(
            new DescribeDBParametersRequest()
            {
                DBParameterGroupName = dbParameterGroupName,
                Source = source
            });
        // Get the entire list using the paginator.
        await foreach (var parameters in paginateParameters.Parameters)
        {
            results.Add(parameters);
```

```
        }
        return results;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DescribeDBParameters 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets DB parameters using the 'DescribeDBParameters' api.
/*!
 \sa getDBParameters()
 \param parameterGroupName: The name of the parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBParameters(const Aws::String &parameterGroupName,
                                  const Aws::String &namePrefix,
                                  const Aws::String &source,
                                  Aws::Vector<Aws::RDS::Model::Parameter>
 &parametersResult,
                                  const Aws::RDS::RDSClient &client) {
    Aws::String marker;
```

```
    do {
        Aws::RDS::Model::DescribeDBParametersRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBParametersOutcome outcome =
                client.DescribeDBParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DescribeDBParameters 참조하십시오.

CLI

AWS CLI

DB 파라미터 그룹의 파라미터를 설명하려면

다음 `describe-db-parameters` 예제에서는 지정된 DB 파라미터 그룹의 세부 정보를 검색합니다.

```
aws rds describe-db-parameters \
    --db-parameter-group-name mydbpg
```

출력:

```
{
    "Parameters": [
        {
            "ParameterName": "allow-suspicious-udfs",
            "Description": "Controls whether user-defined functions that have
 only an xxx symbol for the main function can be loaded",
            "Source": "engine-default",
            "ApplyType": "static",
            "DataType": "boolean",
            "AllowedValues": "0,1",
            "IsModifiable": false,
            "ApplyMethod": "pending-reboot"
        },
        {
            "ParameterName": "auto_generate_certs",
            "Description": "Controls whether the server autogenerates SSL key and
 certificate files in the data directory, if they do not already exist.",
            "Source": "engine-default",
            "ApplyType": "static",
            "DataType": "boolean",
            "AllowedValues": "0,1",
            "IsModifiable": false,
            "ApplyMethod": "pending-reboot"
        },
        ...some output truncated...
    ]
}
```

자세한 내용을 알아보려면 Amazon RDS 사용 설명서의 DB 파라미터 그룹 작업을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DescribeDBParameters를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
type DbInstances struct {
 RdsClient *rds.Client
}



// GetParameters gets the parameters that are contained in a DB parameter group.
func (instances *DbInstances) GetParameters(parameterGroupName string, source
 string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator := rds.NewDescribeDBParametersPaginator(instances.RdsClient,
  &rds.DescribeDBParametersInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Source:               aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(context.TODO())
  if err != nil {
   log.Printf("Couldn't get parameters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
```

```
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 DescribeDBParameters를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   // Retrieve parameters in the group.
   public static void describeDbParameters(RdsClient rdsClient, String
dbGroupName, int flag) {
       try {
           DescribeDbParametersRequest dbParameterGroupsRequest;
           if (flag == 0) {
               dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                       .dbParameterGroupName(dbGroupName)
                       .build();
           } else {
               dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                       .dbParameterGroupName(dbGroupName)
                       .source("user")
                       .build();
           }

           DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
           List<Parameter> dbParameters = response.parameters();
           String paraName;
           for (Parameter para : dbParameters) {
               // Only print out information about either auto_increment_offset
or
               // auto_increment_increment.
               paraName = para.parameterName();
               if ((paraName.compareTo("auto_increment_offset") == 0)
```

```
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeDBParameters를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client
```

```
    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
filtered
                            to contain only parameters that start with this
prefix.
        :param source: When specified, only parameters from this source are
retrieved.
                       For example, a source of 'user' retrieves only parameters
that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator = self.rds_client.get_paginator("describe_db_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
        else:
            return parameters
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 DescribeDBParameters를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require "aws-sdk-rds"  # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) parameter groups.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all parameter groups, or nil if error.
def list_parameter_groups(rds_resource)
  parameter_groups = []
  rds_resource.db_parameter_groups.each do |p|
    parameter_groups.append({
                              "name": p.db_parameter_group_name,
                              "description": p.description
                            })
  end
  parameter_groups
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list parameter groups:\n #{e.message}"
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DescribeDBParameters를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DescribeDBSnapshots** 사용

다음 코드 예제는 DescribeDBSnapshots의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- DB 인스턴스 시작하기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Return a list of DB snapshots for a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBSnapshot>> DescribeDBSnapshots(string
dbInstanceIdentifier)
    {
        var results = new List<DBSnapshot>();
        var snapshotsPaginator = _amazonRDS.Paginators.DescribeDBSnapshots(
            new DescribeDBSnapshotsRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        // Get the entire list using the paginator.
        await foreach (var snapshots in snapshotsPaginator.DBSnapshots)
        {
```

```
            results.Add(snapshots);
        }
        return results;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DescribeDBSnapshots를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DescribeDBSnapshotsRequest request;
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBSnapshotsOutcome outcome =
                    client.DescribeDBSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBSnapshots()[0];
            }
            else {
                std::cerr << "Error with RDS::DescribeDBSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DescribeDBSnapshots를 참조하십시오.

CLI

AWS CLI

예 1: DB 인스턴스의 DB 스냅샷을 설명하려면

다음 describe-db-snapshots 예제에서는 DB 인스턴스의 DB 스냅샷 세부 정보를 검색합니다.

```
aws rds describe-db-snapshots \
    --db-snapshot-identifier mydbsnapshot
```

출력:

```
{
    "DBSnapshots": [
        {
            "DBSnapshotIdentifier": "mydbsnapshot",
            "DBInstanceIdentifier": "mysqldb",
            "SnapshotCreateTime": "2018-02-08T22:28:08.598Z",
            "Engine": "mysql",
            "AllocatedStorage": 20,
            "Status": "available",
            "Port": 3306,
            "AvailabilityZone": "us-east-1f",
            "VpcId": "vpc-6594f31c",
            "InstanceCreateTime": "2018-02-08T22:24:55.973Z",
            "MasterUsername": "mysqladmin",
            "EngineVersion": "5.6.37",
            "LicenseModel": "general-public-license",
            "SnapshotType": "manual",
            "OptionGroupName": "default:mysql-5-6",
            "PercentProgress": 100,
            "StorageType": "gp2",
            "Encrypted": false,
            "DBSnapshotArn": "arn:aws:rds:us-
east-1:123456789012:snapshot:mydbsnapshot",
            "IAMDatabaseAuthenticationEnabled": false,
```

```
            "ProcessorFeatures": [],
            "DbiResourceId": "db-AKIAIOSFODNN7EXAMPLE"
        }
    ]
}
```

자세한 내용은 Amazon RDS 사용 설명서의 DB 스냅샷 생성을 참조하세요.

예 2: 생성한 수동 스냅샷의 개수를 확인하려면

다음 `describe-db-snapshots` 예제에서는 `--query` 옵션의 `length` 연산자를 사용하여 특정 AWS 리전에서 촬영된 수동 스냅샷의 수를 반환합니다.

```
aws rds describe-db-snapshots \
    --snapshot-type manual \
    --query "length(*[].{DBSnapshots:SnapshotType})" \
    --region eu-central-1
```

출력:

```
35
```

자세한 내용은 Amazon RDS 사용 설명서의 DB 스냅샷 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DescribeDBSnapshots를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
type DbInstances struct {
 RdsClient *rds.Client
```

```
}



// GetSnapshot gets a DB instance snapshot.
func (instances *DbInstances) GetSnapshot(snapshotName string)
 (*types.DBSnapshot, error) {
 output, err := instances.RdsClient.DescribeDBSnapshots(context.TODO(),
  &rds.DescribeDBSnapshotsInput{
   DBSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBSnapshots[0], nil
 }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 DescribeDBSnapshots를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client
```

```
    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_snapshot(self, snapshot_id):
        """
        Gets a DB instance snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_snapshots(
                DBSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 DescribeDBSnapshots를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require "aws-sdk-rds" # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) DB instance
# snapshots.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return instance_snapshots [Array, nil] All instance snapshots, or nil if
 error.
def list_instance_snapshots(rds_resource)
  instance_snapshots = []
  rds_resource.db_snapshots.each do |s|
    instance_snapshots.append({
                                "id": s.snapshot_id,
                                "status": s.status
                              })
  end
  instance_snapshots
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list instance snapshots:\n #{e.message}"
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DescribeDBSnapshots를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DescribeOrderableDBInstanceOptions** 사용

다음 코드 예제는 DescribeOrderableDBInstanceOptions의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- DB 인스턴스 시작하기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptions(string engine, string engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
```

```
        await foreach (var instanceOptions in
 paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DescribeOrderableDBInstanceOptions를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available 'micro' DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseMicroDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::chooseMicroDBInstanceClass(const Aws::String &engine,
```

```
                                            const Aws::String &engineVersion,
                                            Aws::String &dbInstanceClass,
                                            const Aws::RDS::RDSClient &client) {
   std::vector<Aws::String> instanceClasses;
   Aws::String marker;
   do {
       Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
       request.SetEngine(engine);
       request.SetEngineVersion(engineVersion);
       if (!marker.empty()) {
           request.SetMarker(marker);
       }

       Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
               client.DescribeOrderableDBInstanceOptions(request);

       if (outcome.IsSuccess()) {
           const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption>
&options =
                   outcome.GetResult().GetOrderableDBInstanceOptions();
           for (const Aws::RDS::Model::OrderableDBInstanceOption &option:
options) {
               const Aws::String &instanceClass = option.GetDBInstanceClass();
               if (instanceClass.find("micro") != std::string::npos) {
                   if (std::find(instanceClasses.begin(), instanceClasses.end(),
                                 instanceClass) ==
                       instanceClasses.end()) {
                       instanceClasses.push_back(instanceClass);
                   }
               }
           }
           marker = outcome.GetResult().GetMarker();
       }
       else {
           std::cerr << "Error with RDS::DescribeOrderableDBInstanceOptions. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           return false;
       }
   } while (!marker.empty());

   std::cout << "The available micro DB instance classes for your database
engine are:"
             << std::endl;
```

```
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which micro DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DescribeOrderableDBInstanceOptions를 참조하십시오.

CLI

AWS CLI

주문 가능한 DB 인스턴스 옵션을 설명하려면

다음 `describe-orderable-db-instance-options` 예제에서는 MySQL DB 엔진을 실행 중인 DB 인스턴스의 주문 가능한 옵션에 대한 세부 정보를 검색합니다.

```
aws rds describe-orderable-db-instance-options \
    --engine mysql
```

출력:

```
{
    "OrderableDBInstanceOptions": [
        {
            "MinStorageSize": 5,
            "ReadReplicaCapable": true,
            "MaxStorageSize": 6144,
            "AvailabilityZones": [
                {
                    "Name": "us-east-1a"
                },
                {
                    "Name": "us-east-1b"
```

```
                },
                {
                    "Name": "us-east-1c"
                },
                {
                    "Name": "us-east-1d"
                }
            ],
            "SupportsIops": false,
            "AvailableProcessorFeatures": [],
            "MultiAZCapable": true,
            "DBInstanceClass": "db.m1.large",
            "Vpc": true,
            "StorageType": "gp2",
            "LicenseModel": "general-public-license",
            "EngineVersion": "5.5.46",
            "SupportsStorageEncryption": false,
            "SupportsEnhancedMonitoring": true,
            "Engine": "mysql",
            "SupportsIAMDatabaseAuthentication": false,
            "SupportsPerformanceInsights": false
        }
    ]
    ...some output truncated...
}
```

- API 세부 정보는 AWS CLI 명령 참조의 DescribeOrderableDBInstanceOptions를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
type DbInstances struct {
 RdsClient *rds.Client
}
```

```
// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (instances *DbInstances) GetOrderableInstances(engine string, engineVersion
 string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instanceOptions []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(instances.RdsClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(context.TODO())
  if err != nil {
   log.Printf("Couldn't get orderable DB instance options: %v\n", err)
   break
  } else {
   instanceOptions = append(instanceOptions,
 output.OrderableDBInstanceOptions...)
  }
 }
 return instanceOptions, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 DescribeOrderableDBInstanceOptions를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   // Get a list of allowed engine versions.
   public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
       try {
           DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
                   .dbParameterGroupFamily(dbParameterGroupFamily)
                   .engine("mysql")
                   .build();

           DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
           List<DBEngineVersion> dbEngines = response.dbEngineVersions();
           for (DBEngineVersion dbEngine : dbEngines) {
               System.out.println("The engine version is " +
dbEngine.engineVersion());
               System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeOrderableDBInstanceOptions를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
 instance.
        :param db_engine_version: The engine version that must be supported by
 the DB instance.
        :return: The list of DB instance options that can be used to create a
 compatible DB instance.
        """
        try:
            inst_opts = []
```

```
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts
```

- API 세부 정보는 Python(Boto3)용 AWS SDK API 참조의 DescribeOrderableDBInstanceOptions를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GenerateRDSAuthToken** 사용

다음 코드 예시에서는 `GenerateRDSAuthToken`을 사용하는 방법을 보여 줍니다.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

RDSUtilities 클래스를 사용하여 인증 토큰을 생성합니다.

```
public class GenerateRDSAuthToken {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier> <masterUsername>

                Where:
                    dbInstanceIdentifier - The database instance identifier.\s
                    masterUsername - The master user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String masterUsername = args[1];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        String token = getAuthToken(rdsClient, dbInstanceIdentifier,
 masterUsername);
        System.out.println("The token response is " + token);
    }

    public static String getAuthToken(RdsClient rdsClient, String
 dbInstanceIdentifier, String masterUsername) {

        RdsUtilities utilities = rdsClient.utilities();
        try {
            GenerateAuthenticationTokenRequest tokenRequest =
 GenerateAuthenticationTokenRequest.builder()
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .username(masterUsername)
                    .port(3306)
                    .hostname(dbInstanceIdentifier)
                    .build();

            return utilities.generateAuthenticationToken(tokenRequest);
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GenerateRDSAuthToken을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ModifyDBInstance** 사용

다음 코드 예제는 ModifyDBInstance의 사용 방법을 보여 줍니다.

CLI

AWS CLI

예제 1: DB 인스턴스를 수정하려면 다음과 같이 하세요.

다음 modify-db-instance 예제에서는 옵션 그룹과 파라미터 그룹을 호환되는 Microsoft SQL Server DB 인스턴스와 연결합니다. --apply-immediately 파라미터를 사용하면 다음 유지 관리 기간이 될 때까지 기다리는 대신 옵션과 파라미터 그룹이 즉시 연결됩니다.

```
aws rds modify-db-instance \
    --db-instance-identifier database-2 \
    --option-group-name test-se-2017 \
    --db-parameter-group-name test-sqlserver-se-2017 \
    --apply-immediately
```

출력:

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "database-2",
        "DBInstanceClass": "db.r4.large",
        "Engine": "sqlserver-se",
```

```
        "DBInstanceStatus": "available",

        ...output omitted...

        "DBParameterGroups": [
            {
                "DBParameterGroupName": "test-sqlserver-se-2017",
                "ParameterApplyStatus": "applying"
            }
        ],
        "AvailabilityZone": "us-west-2d",

        ...output omitted...

        "MultiAZ": true,
        "EngineVersion": "14.00.3281.6.v1",
        "AutoMinorVersionUpgrade": false,
        "ReadReplicaDBInstanceIdentifiers": [],
        "LicenseModel": "license-included",
        "OptionGroupMemberships": [
            {
                "OptionGroupName": "test-se-2017",
                "Status": "pending-apply"
            }
        ],
        "CharacterSetName": "SQL_Latin1_General_CP1_CI_AS",
        "SecondaryAvailabilityZone": "us-west-2c",
        "PubliclyAccessible": true,
        "StorageType": "gp2",

        ...output omitted...

        "DeletionProtection": false,
        "AssociatedRoles": [],
        "MaxAllocatedStorage": 1000
    }
}
```

자세한 내용은 Amazon RDS 사용 설명서의 Amazon RDS DB 인스턴스 수정을 참조하세요.

예제 2: VPC 보안 그룹을 DB 인스턴스와 연결하려면 다음과 같이 하세요.

다음 `modify-db-instance` 예제는 특정 VPC 보안 그룹을 연결하고 DB 인스턴스에서 DB 보안 그룹을 제거합니다.

```
aws rds modify-db-instance \
    --db-instance-identifier dbName \
    --vpc-security-group-ids sg-ID
```

출력:

```
{
"DBInstance": {
    "DBInstanceIdentifier": "dbName",
    "DBInstanceClass": "db.t3.micro",
    "Engine": "mysql",
    "DBInstanceStatus": "available",
    "MasterUsername": "admin",
    "Endpoint": {
        "Address": "dbName.abcdefghijk.us-west-2.rds.amazonaws.com",
        "Port": 3306,
        "HostedZoneId": "ABCDEFGHIJK1234"
    },
    "AllocatedStorage": 20,
    "InstanceCreateTime": "2024-02-15T00:37:58.793000+00:00",
    "PreferredBackupWindow": "11:57-12:27",
    "BackupRetentionPeriod": 7,
    "DBSecurityGroups": [],
    "VpcSecurityGroups": [
        {
            "VpcSecurityGroupId": "sg-ID",
            "Status": "active"
        }
    ],
    ... output omitted ...
    "MultiAZ": false,
    "EngineVersion": "8.0.35",
    "AutoMinorVersionUpgrade": true,
    "ReadReplicaDBInstanceIdentifiers": [],
    "LicenseModel": "general-public-license",

    ... output ommited ...
    }
}
```

자세한 내용은 Amazon RDS 사용 설명서의 보안 그룹을 통한 액세스 제어를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ModifyDBInstance를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ModifyDBInstance {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier> <dbSnapshotIdentifier>\s
                Where:
                    dbInstanceIdentifier - The database instance identifier.\s
                    masterUserPassword - The updated password that corresponds to
 the master user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String dbInstanceIdentifier = args[0];
        String masterUserPassword = args[1];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        updateIntance(rdsClient, dbInstanceIdentifier, masterUserPassword);
        rdsClient.close();
    }

    public static void updateIntance(RdsClient rdsClient, String
 dbInstanceIdentifier, String masterUserPassword) {
        try {
            // For a demo - modify the DB instance by modifying the master
 password.
            ModifyDbInstanceRequest modifyDbInstanceRequest =
 ModifyDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .publiclyAccessible(true)
                    .masterUserPassword(masterUserPassword)
                    .build();

            ModifyDbInstanceResponse instanceResponse =
 rdsClient.modifyDBInstance(modifyDbInstanceRequest);
            System.out.print("The ARN of the modified database is: " +
 instanceResponse.dbInstance().dbInstanceArn());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ModifyDBInstance를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun updateIntance(
    dbInstanceIdentifierVal: String?,
    masterUserPasswordVal: String?,
) {
    val request =
        ModifyDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            publiclyAccessible = true
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val instanceResponse = rdsClient.modifyDbInstance(request)
        println("The ARN of the modified database is
 ${instanceResponse.dbInstance?.dbInstanceArn}")
    }
}
```

- API 세부 정보는 Kotlin용 AWS SDK API 참조의 ModifyDBInstance를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ModifyDBParameterGroup** 사용

다음 코드 예제는 ModifyDBParameterGroup의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- DB 인스턴스 시작하기

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Update a DB parameter group. Use the action
 DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="parameters">List of parameters. Maximum of 20 per request.</
param>
    /// <returns>The updated DB parameter group name.</returns>
    public async Task<string> ModifyDBParameterGroup(
        string name, List<Parameter> parameters)
    {
        var response = await _amazonRDS.ModifyDBParameterGroupAsync(
            new ModifyDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                Parameters = parameters,
            });
        return response.DBParameterGroupName;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ModifyDBParameterGroup을 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::ModifyDBParameterGroupRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBParameterGroupOutcome outcome =
                client.ModifyDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::ModifyDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 ModifyDBParameterGroup을 참조하십시오.

CLI

AWS CLI

DB 파라미터 그룹을 수정하려면

다음 modify-db-parameter-group 예제에서는 DB 파라미터 그룹의 clr enabled 파라미터 값을 변경합니다. --apply-immediately 파라미터를 사용하면 다음 유지 관리 기간이 될 때까지 기다리는 대신 DB 파라미터 그룹이 즉시 수정됩니다.

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name test-sqlserver-se-2017 \
    --parameters "ParameterName='clr
 enabled',ParameterValue=1,ApplyMethod=immediate"
```

출력:

```
{
    "DBParameterGroupName": "test-sqlserver-se-2017"
}
```

자세한 내용은 Amazon RDS 사용 설명서의 DB 파라미터 그룹의 파라미터 수정을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ModifyDBParameterGroup을 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
type DbInstances struct {
 RdsClient *rds.Client
}



// UpdateParameters updates parameters in a named DB parameter group.
func (instances *DbInstances) UpdateParameters(parameterGroupName string, params
 []types.Parameter) error {
 _, err := instances.RdsClient.ModifyDBParameterGroup(context.TODO(),
  &rds.ModifyDBParameterGroupInput{
```

```
    DBParameterGroupName: aws.String(parameterGroupName),
    Parameters:           params,
   })
  if err != nil {
   log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
   return err
  } else {
   return nil
  }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 ModifyDBParameterGroup을 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    // Modify auto_increment_offset and auto_increment_increment parameters.
    public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
        try {
            Parameter parameter1 = Parameter.builder()
                    .parameterName("auto_increment_offset")
                    .applyMethod("immediate")
                    .parameterValue("5")
                    .build();

            List<Parameter> paraList = new ArrayList<>();
            paraList.add(parameter1);
            ModifyDbParameterGroupRequest groupRequest =
 ModifyDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .parameters(paraList)
                    .build();
```

```
            ModifyDbParameterGroupResponse response =
 rdsClient.modifyDBParameterGroup(groupRequest);
            System.out.println("The parameter group " +
 response.dbParameterGroupName() + " was successfully modified");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ModifyDBParameterGroup을 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)
```

```
    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
Parameters=update_parameters
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- API 세부 정보는 Python(Boto3)용 AWS SDK API 참조의 ModifyDBParameterGroup을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **RebootDBInstance** 사용

다음 코드 예제는 RebootDBInstance의 사용 방법을 보여 줍니다.

CLI

AWS CLI

DB 인스턴스를 재부팅하려면

다음 reboot-db-instance 예제에서는 지정된 DB 인스턴스의 재부팅을 시작합니다.

```
aws rds reboot-db-instance \
    --db-instance-identifier test-mysql-instance
```

출력:

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "test-mysql-instance",
        "DBInstanceClass": "db.t3.micro",
        "Engine": "mysql",
        "DBInstanceStatus": "rebooting",
        "MasterUsername": "admin",
        "Endpoint": {
            "Address": "test-mysql-instance.############.us-
west-2.rds.amazonaws.com",
            "Port": 3306,
            "HostedZoneId": "Z1PVIF0EXAMPLE"
        },

    ... output omitted...

    }
}
```

자세한 내용은 Amazon RDS 사용 설명서의 DB 인스턴스 재부팅을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 RebootDBInstance를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class RebootDBInstance {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier>\s

                Where:
                    dbInstanceIdentifier - The database instance identifier\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
```

```
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        rebootInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    public static void rebootInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
        try {
            RebootDbInstanceRequest rebootDbInstanceRequest =
RebootDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            RebootDbInstanceResponse instanceResponse =
rdsClient.rebootDBInstance(rebootDbInstanceRequest);
            System.out.print("The database " +
instanceResponse.dbInstance().dbInstanceArn() + " was rebooted");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 RebootDBInstance를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용한 Amazon RDS에 대한 시나리오

다음 코드 예제는 AWS SDK로 Amazon RDS에서 일반적인 시나리오를 구현하는 방법을 보여줍니다. 이러한 시나리오에서는 Amazon RDS 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여줍니다. 각 시나리오에는 GitHub에 대한 링크가 포함되어 있습니다. 여기에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

예시

- AWS SDK를 사용하여 Amazon RDS DB 인스턴스 시작하기

AWS SDK를 사용하여 Amazon RDS DB 인스턴스 시작하기

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 사용자 지정 DB 파라미터 그룹을 생성하고 파라미터 값을 설정합니다.
- 파라미터 그룹을 사용하도록 구성된 DB 인스턴스를 생성합니다. DB 인스턴스에는 데이터베이스도 포함되어 있습니다.
- 인스턴스의 스냅샷을 만듭니다.
- 인스턴스 및 파라미터 그룹을 삭제합니다.

.NET

SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
/// <summary>
/// Scenario for RDS DB instance example.
/// </summary>
public class RDSInstanceScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
    1.  Returns a list of the available DB engine families using the
 DescribeDBEngineVersionsAsync method.
```

```
   2.  Selects an engine family and creates a custom DB parameter group using
the CreateDBParameterGroupAsync method.
   3.  Gets the parameter groups using the DescribeDBParameterGroupsAsync
method.
   4.  Gets parameters in the group using the DescribeDBParameters method.
   5.  Parses and displays parameters in the group.
   6.  Modifies both the auto_increment_offset and auto_increment_increment
parameters
       using the ModifyDBParameterGroupAsync method.
   7.  Gets and displays the updated parameters using the DescribeDBParameters
method with a source of "user".
   8.  Gets a list of allowed engine versions using the
DescribeDBEngineVersionsAsync method.
   9.  Displays and selects from a list of micro instance classes available for
the selected engine and version.
   10. Creates an RDS DB instance that contains a MySql database and uses the
parameter group
       using the CreateDBInstanceAsync method.
   11. Waits for DB instance to be ready using the DescribeDBInstancesAsync
method.
   12. Prints out the connection endpoint string for the new DB instance.
   13. Creates a snapshot of the DB instance using the CreateDBSnapshotAsync
method.
   14. Waits for DB snapshot to be ready using the DescribeDBSnapshots method.
   15. Deletes the DB instance using the DeleteDBInstanceAsync method.
   16. Waits for DB instance to be deleted using the DescribeDbInstances method.
   17. Deletes the parameter group using the DeleteDBParameterGroupAsync.
   */

   private static readonly string sepBar = new('-', 80);
   private static RDSWrapper rdsWrapper = null!;
   private static ILogger logger = null!;
   private static readonly string engine = "mysql";
   static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon RDS service.
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
           .ConfigureServices((_, services) =>
```

```
                services.AddAWSService<IAmazonRDS>()
                    .AddTransient<RDSWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        }).CreateLogger<RDSInstanceScenario>();

        rdsWrapper = host.Services.GetRequiredService<RDSWrapper>();

        Console.WriteLine(sepBar);
        Console.WriteLine(
            "Welcome to the Amazon Relational Database Service (Amazon RDS) DB
instance scenario example.");
        Console.WriteLine(sepBar);

        try
        {
            var parameterGroupFamily = await ChooseParameterGroupFamily();

            var parameterGroup = await
CreateDbParameterGroup(parameterGroupFamily);

            var parameters = await
DescribeParametersInGroup(parameterGroup.DBParameterGroupName,
                new List<string> { "auto_increment_offset",
"auto_increment_increment" });

            await ModifyParameters(parameterGroup.DBParameterGroupName,
parameters);

            await
DescribeUserSourceParameters(parameterGroup.DBParameterGroupName);

            var engineVersionChoice = await
ChooseDbEngineVersion(parameterGroupFamily);

            var instanceChoice = await ChooseDbInstanceClass(engine,
engineVersionChoice.EngineVersion);

            var newInstanceIdentifier = "Example-Instance-" + DateTime.Now.Ticks;
```

```
            var newInstance = await CreateRdsNewInstance(parameterGroup, engine,
engineVersionChoice.EngineVersion,
                instanceChoice.DBInstanceClass, newInstanceIdentifier);
            if (newInstance != null)
            {
                DisplayConnectionString(newInstance);

                await CreateSnapshot(newInstance);

                await DeleteRdsInstance(newInstance);
            }

            await DeleteParameterGroup(parameterGroup);

            Console.WriteLine("Scenario complete.");
            Console.WriteLine(sepBar);
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
        }
    }

    /// <summary>
    /// Choose the RDS DB parameter group family from a list of available
options.
    /// </summary>
    /// <returns>The selected parameter group family.</returns>
    public static async Task<string> ChooseParameterGroupFamily()
    {
        Console.WriteLine(sepBar);
        // 1. Get a list of available engines.
        var engines = await rdsWrapper.DescribeDBEngineVersions(engine);

        Console.WriteLine("1. The following is a list of available DB parameter
group families:");
        int i = 1;
        var parameterGroupFamilies = engines.GroupBy(e =>
e.DBParameterGroupFamily).ToList();
        foreach (var parameterGroupFamily in parameterGroupFamilies)
        {
            // List the available parameter group families.
            Console.WriteLine(
                $"\t{i}. Family: {parameterGroupFamily.Key}");
```

```
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > parameterGroupFamilies.Count)
        {
            Console.WriteLine("Select an available DB parameter group family by
entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
        var parameterGroupFamilyChoice = parameterGroupFamilies[choiceNumber -
1];
        Console.WriteLine(sepBar);
        return parameterGroupFamilyChoice.Key;
    }

    /// <summary>
    /// Create and get information on a DB parameter group.
    /// </summary>
    /// <param name="dbParameterGroupFamily">The DBParameterGroupFamily for the
new DB parameter group.</param>
    /// <returns>The new DBParameterGroup.</returns>
    public static async Task<DBParameterGroup> CreateDbParameterGroup(string
dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"2. Create new DB parameter group with family
{dbParameterGroupFamily}:");

        var parameterGroup = await rdsWrapper.CreateDBParameterGroup(
            "ExampleParameterGroup-" + DateTime.Now.Ticks,
            dbParameterGroupFamily, "New example parameter group");

        var groupInfo =
            await rdsWrapper.DescribeDBParameterGroups(parameterGroup
                .DBParameterGroupName);

        Console.WriteLine(
            $"3. New DB parameter group: \n\t{groupInfo[0].Description}, \n\tARN
{groupInfo[0].DBParameterGroupArn}");
        Console.WriteLine(sepBar);
        return parameterGroup;
    }
```

```
    /// <summary>
    /// Get and describe parameters from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <param name="parameterNames">Optional specific names of parameters to
describe.</param>
    /// <returns>The list of requested parameters.</returns>
    public static async Task<List<Parameter>> DescribeParametersInGroup(string
parameterGroupName, List<string>? parameterNames = null)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("4. Get some parameters from the group.");
        Console.WriteLine(sepBar);

        var parameters =
            await rdsWrapper.DescribeDBParameters(parameterGroupName);

        var matchingParameters =
            parameters.Where(p => parameterNames == null ||
parameterNames.Contains(p.ParameterName)).ToList();

        Console.WriteLine("5. Parameter information:");
        matchingParameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);

        return matchingParameters;
    }

    /// <summary>
    /// Modify a parameter from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <param name="parameters">The parameters to modify.</param>
    /// <returns>Async task.</returns>
    public static async Task ModifyParameters(string parameterGroupName,
List<Parameter> parameters)
    {
```

```
        Console.WriteLine(sepBar);
        Console.WriteLine("6. Modify some parameters in the group.");

        foreach (var p in parameters)
        {
            if (p.IsModifiable && p.DataType == "integer")
            {
                int newValue = 0;
                while (newValue == 0)
                {
                    Console.WriteLine(
                        $"Enter a new value for {p.ParameterName} from the
allowed values {p.AllowedValues} ");

                    var choice = Console.ReadLine();
                    Int32.TryParse(choice, out newValue);
                }

                p.ParameterValue = newValue.ToString();
            }
        }

        await rdsWrapper.ModifyDBParameterGroup(parameterGroupName, parameters);

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Describe the user source parameters in the group.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <returns>Async task.</returns>
    public static async Task DescribeUserSourceParameters(string
parameterGroupName)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("7. Describe user source parameters in the group.");

        var parameters =
            await rdsWrapper.DescribeDBParameters(parameterGroupName, "user");


        parameters.ForEach(p =>
            Console.WriteLine(
```

```
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);
    }


    /// <summary>
    /// Choose a DB engine version.
    /// </summary>
    /// <param name="dbParameterGroupFamily">DB parameter group family for engine
choice.</param>
    /// <returns>The selected engine version.</returns>
    public static async Task<DBEngineVersion> ChooseDbEngineVersion(string
dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed engines.
        var allowedEngines =
            await rdsWrapper.DescribeDBEngineVersions(engine,
dbParameterGroupFamily);

        Console.WriteLine($"Available DB engine versions for parameter group
family {dbParameterGroupFamily}:");
        int i = 1;
        foreach (var version in allowedEngines)
        {
            Console.WriteLine(
                $"\t{i}. Engine: {version.Engine} Version
{version.EngineVersion}.");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedEngines.Count)
        {
            Console.WriteLine("8. Select an available DB engine version by
entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
```

```
        var engineChoice = allowedEngines[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return engineChoice;
    }

    /// <summary>
    /// Choose a DB instance class for a particular engine and engine version.
    /// </summary>
    /// <param name="engine">DB engine for DB instance choice.</param>
    /// <param name="engineVersion">DB engine version for DB instance choice.</
param>
    /// <returns>The selected orderable DB instance option.</returns>
    public static async Task<OrderableDBInstanceOption>
 ChooseDbInstanceClass(string engine, string engineVersion)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed DB instance classes.
        var allowedInstances =
            await rdsWrapper.DescribeOrderableDBInstanceOptions(engine,
 engineVersion);

        Console.WriteLine($"8. Available micro DB instance classes for engine
 {engine} and version {engineVersion}:");
        int i = 1;

        // Filter to micro instances for this example.
        allowedInstances = allowedInstances
            .Where(i => i.DBInstanceClass.Contains("micro")).ToList();

        foreach (var instance in allowedInstances)
        {
            Console.WriteLine(
                $"\t{i}. Instance class: {instance.DBInstanceClass} (storage type
 {instance.StorageType})");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedInstances.Count)
        {
            Console.WriteLine("9. Select an available DB instance class by
 entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
```

```
        }

        var instanceChoice = allowedInstances[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return instanceChoice;
    }

    /// <summary>
    /// Create a new RDS DB instance.
    /// </summary>
    /// <param name="parameterGroup">Parameter group to use for the DB
 instance.</param>
    /// <param name="engineName">Engine to use for the DB instance.</param>
    /// <param name="engineVersion">Engine version to use for the DB instance.</
param>
    /// <param name="instanceClass">Instance class to use for the DB instance.</
param>
    /// <param name="instanceIdentifier">Instance identifier to use for the DB
 instance.</param>
    /// <returns>The new DB instance.</returns>
    public static async Task<DBInstance?> CreateRdsNewInstance(DBParameterGroup
 parameterGroup,
        string engineName, string engineVersion, string instanceClass, string
 instanceIdentifier)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"10. Create a new DB instance with identifier
 {instanceIdentifier}.");
        bool isInstanceReady = false;
        DBInstance newInstance;
        var instances = await rdsWrapper.DescribeDBInstances();
        isInstanceReady = instances.FirstOrDefault(i =>
            i.DBInstanceIdentifier == instanceIdentifier)?.DBInstanceStatus ==
 "available";

        if (isInstanceReady)
        {
            Console.WriteLine("Instance already created.");
            newInstance = instances.First(i => i.DBInstanceIdentifier ==
 instanceIdentifier);
        }
        else
        {
            Console.WriteLine("Please enter an admin user name:");
```

```
            var username = Console.ReadLine();

            Console.WriteLine("Please enter an admin password:");
            var password = Console.ReadLine();

            newInstance = await rdsWrapper.CreateDBInstance(
                "ExampleInstance",
                instanceIdentifier,
                parameterGroup.DBParameterGroupName,
                engineName,
                engineVersion,
                instanceClass,
                20,
                username,
                password
            );

            // 11. Wait for the DB instance to be ready.

            Console.WriteLine("11. Waiting for DB instance to be ready...");
            while (!isInstanceReady)
            {
                instances = await
rdsWrapper.DescribeDBInstances(instanceIdentifier);
                isInstanceReady = instances.FirstOrDefault()?.DBInstanceStatus ==
"available";
                newInstance = instances.First();
                Thread.Sleep(30000);
            }
        }

        Console.WriteLine(sepBar);
        return newInstance;
    }

    /// <summary>
    /// Display a connection string for an RDS DB instance.
    /// </summary>
    /// <param name="instance">The DB instance to use to get a connection
string.</param>
    public static void DisplayConnectionString(DBInstance instance)
    {
        Console.WriteLine(sepBar);
        // Display the connection string.
```

```
        Console.WriteLine("12. New DB instance connection string: ");
        Console.WriteLine(
            $"\n{engine} -h {instance.Endpoint.Address} -P
 {instance.Endpoint.Port} "
            + $"-u {instance.MasterUsername} -p [YOUR PASSWORD]\n");

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Create a snapshot from an RDS DB instance.
    /// </summary>
    /// <param name="instance">DB instance to use when creating a snapshot.</
param>
    /// <returns>The snapshot object.</returns>
    public static async Task<DBSnapshot> CreateSnapshot(DBInstance instance)
    {
        Console.WriteLine(sepBar);
        // Create a snapshot.
        Console.WriteLine($"13. Creating snapshot from DB instance
 {instance.DBInstanceIdentifier}.");
        var snapshot = await
 rdsWrapper.CreateDBSnapshot(instance.DBInstanceIdentifier, "ExampleSnapshot-" +
 DateTime.Now.Ticks);

        // Wait for the snapshot to be available
        bool isSnapshotReady = false;

        Console.WriteLine($"14. Waiting for snapshot to be ready...");
        while (!isSnapshotReady)
        {
            var snapshots = await
 rdsWrapper.DescribeDBSnapshots(instance.DBInstanceIdentifier);
            isSnapshotReady = snapshots.FirstOrDefault()?.Status == "available";
            snapshot = snapshots.First();
            Thread.Sleep(30000);
        }

        Console.WriteLine(
            $"Snapshot {snapshot.DBSnapshotIdentifier} status is
 {snapshot.Status}.");
        Console.WriteLine(sepBar);
        return snapshot;
    }
```

```
    /// <summary>
    /// Delete an RDS DB instance.
    /// </summary>
    /// <param name="instance">The DB instance to delete.</param>
    /// <returns>Async task.</returns>
    public static async Task DeleteRdsInstance(DBInstance newInstance)
    {
        Console.WriteLine(sepBar);
        // Delete the DB instance.
        Console.WriteLine($"15. Delete the DB instance
{newInstance.DBInstanceIdentifier}.");
        await rdsWrapper.DeleteDBInstance(newInstance.DBInstanceIdentifier);

        // Wait for the DB instance to delete.
        Console.WriteLine($"16. Waiting for the DB instance to delete...");
        bool isInstanceDeleted = false;

        while (!isInstanceDeleted)
        {
            var instance = await rdsWrapper.DescribeDBInstances();
            isInstanceDeleted = instance.All(i => i.DBInstanceIdentifier !=
newInstance.DBInstanceIdentifier);
            Thread.Sleep(30000);
        }

        Console.WriteLine("DB instance deleted.");
        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Delete a DB parameter group.
    /// </summary>
    /// <param name="parameterGroup">The parameter group to delete.</param>
    /// <returns>Async task.</returns>
    public static async Task DeleteParameterGroup(DBParameterGroup
parameterGroup)
    {
        Console.WriteLine(sepBar);
        // Delete the parameter group.
        Console.WriteLine($"17. Delete the DB parameter group
{parameterGroup.DBParameterGroupName}.");
        await
rdsWrapper.DeleteDBParameterGroup(parameterGroup.DBParameterGroupName);
```

```
        Console.WriteLine(sepBar);
    }
```

시나리오에서 DB 인스턴스 작업에 대해 사용하는 래퍼 메서드입니다.

```
/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with
 DB instance operations.
/// </summary>
public partial class RDSWrapper
{
    private readonly IAmazonRDS _amazonRDS;
    public RDSWrapper(IAmazonRDS amazonRDS)
    {
        _amazonRDS = amazonRDS;
    }


    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="dbParameterGroupFamily">Optional parameter group family
 name.</param>
    /// <returns>List of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>> DescribeDBEngineVersions(string
 engine,
        string dbParameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = dbParameterGroupFamily
            });
        return response.DBEngineVersions;
    }
```

```
    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptions(string engine, string engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }



    /// <summary>
    /// Returns a list of DB instances.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
    /// <returns>List of DB instances.</returns>
    public async Task<List<DBInstance>> DescribeDBInstances(string
dbInstanceIdentifier = null)
    {
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
                DBInstanceIdentifier = dbInstanceIdentifier
```

```
            });
        // Get the entire list using the paginator.
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }



    /// <summary>
    /// Create an RDS DB instance with a particular set of properties. Use the
action DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbName">Name for the DB instance.</param>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="parameterGroupName">DB parameter group to associate with the
instance.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <param name="allocatedStorage">The amount of storage in gibibytes (GiB)
to allocate to the DB instance.</param>
    /// <param name="adminName">Admin user name.</param>
    /// <param name="adminPassword">Admin user password.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstance(string dbName, string
dbInstanceIdentifier,
        string parameterGroupName, string dbEngine, string dbEngineVersion,
        string instanceClass, int allocatedStorage, string adminName, string
adminPassword)
    {
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBName = dbName,
                DBInstanceIdentifier = dbInstanceIdentifier,
                DBParameterGroupName = parameterGroupName,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass,
                AllocatedStorage = allocatedStorage,
```

```
                MasterUsername = adminName,
                MasterUserPassword = adminPassword
            });

        return response.DBInstance;
    }



    /// <summary>
    /// Delete a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> DeleteDBInstance(string dbInstanceIdentifier)
    {
        var response = await _amazonRDS.DeleteDBInstanceAsync(
            new DeleteDBInstanceRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });

        return response.DBInstance;
    }
```

DB 파라미터 그룹에 대해 시나리오에서 사용하는 래퍼 메서드.

```
/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with
 parameter groups.
/// </summary>
public partial class RDSWrapper
{

    /// <summary>
    /// Get descriptions of DB parameter groups.
    /// </summary>
```

```
    /// <param name="name">Optional name of the DB parameter group to describe.</
param>
    /// <returns>The list of DB parameter group descriptions.</returns>
    public async Task<List<DBParameterGroup>> DescribeDBParameterGroups(string
 name = null)
    {
        var response = await _amazonRDS.DescribeDBParameterGroupsAsync(
            new DescribeDBParameterGroupsRequest()
            {
                DBParameterGroupName = name
            });
        return response.DBParameterGroups;
    }



    /// <summary>
    /// Create a new DB parameter group. Use the action
 DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="family">Family of the DB parameter group.</param>
    /// <param name="description">Description of the DB parameter group.</param>
    /// <returns>The new DB parameter group.</returns>
    public async Task<DBParameterGroup> CreateDBParameterGroup(
        string name, string family, string description)
    {
        var response = await _amazonRDS.CreateDBParameterGroupAsync(
            new CreateDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                DBParameterGroupFamily = family,
                Description = description
            });
        return response.DBParameterGroup;
    }



    /// <summary>
    /// Update a DB parameter group. Use the action
 DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
```

```
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="parameters">List of parameters. Maximum of 20 per request.</
param>
    /// <returns>The updated DB parameter group name.</returns>
    public async Task<string> ModifyDBParameterGroup(
        string name, List<Parameter> parameters)
    {
        var response = await _amazonRDS.ModifyDBParameterGroupAsync(
            new ModifyDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                Parameters = parameters,
            });
        return response.DBParameterGroupName;
    }



    /// <summary>
    /// Delete a DB parameter group. The group cannot be a default DB parameter
 group
    /// or be associated with any DB instances.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDBParameterGroup(string name)
    {
        var response = await _amazonRDS.DeleteDBParameterGroupAsync(
            new DeleteDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }



    /// <summary>
    /// Get a list of DB parameters from a specific parameter group.
    /// </summary>
    /// <param name="dbParameterGroupName">Name of a specific DB parameter
 group.</param>
    /// <param name="source">Optional source for selecting parameters.</param>
```

```
    /// <returns>List of parameter values.</returns>
    public async Task<List<Parameter>> DescribeDBParameters(string
 dbParameterGroupName, string source = null)
    {
        var results = new List<Parameter>();
        var paginateParameters = _amazonRDS.Paginators.DescribeDBParameters(
            new DescribeDBParametersRequest()
            {
                DBParameterGroupName = dbParameterGroupName,
                Source = source
            });
        // Get the entire list using the paginator.
        await foreach (var parameters in paginateParameters.Parameters)
        {
            results.Add(parameters);
        }
        return results;
    }
```

시나리오에서 DB 스냅샷 작업에 대해 사용하는 래퍼 메서드입니다.

```
/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with
 snapshots.
/// </summary>
public partial class RDSWrapper
{

    /// <summary>
    /// Create a snapshot of a DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBSnapshot> CreateDBSnapshot(string dbInstanceIdentifier,
 string snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBSnapshotAsync(
            new CreateDBSnapshotRequest()
            {
```

```
                DBSnapshotIdentifier = snapshotIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        return response.DBSnapshot;
    }



    /// <summary>
    /// Return a list of DB snapshots for a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBSnapshot>> DescribeDBSnapshots(string
dbInstanceIdentifier)
    {
        var results = new List<DBSnapshot>();
        var snapshotsPaginator = _amazonRDS.Paginators.DescribeDBSnapshots(
            new DescribeDBSnapshotsRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        // Get the entire list using the paginator.
        await foreach (var snapshots in snapshotsPaginator.DBSnapshots)
        {
            results.Add(snapshots);
        }
        return results;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 다음 주제를 참조하십시오.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions

- DescribeDBInstances
- DescribeDBParameterGroups
- DescribeDBParameters
- DescribeDBSnapshots
- DescribeOrderableDBInstanceOptions
- ModifyDBParameterGroup

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Routine which creates an Amazon RDS instance and demonstrates several
 operations
//! on that instance.
/*!
 \sa gettingStartedWithDBInstances()
 \param clientConfiguration: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::gettingStartedWithDBInstances(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::RDS::RDSClient client(clientConfig);

    printAsterisksLine();
    std::cout << "Welcome to the Amazon Relational Database Service (Amazon RDS)"
              << std::endl;
    std::cout << "get started with DB instances demo." << std::endl;
    printAsterisksLine();
```

```
    std::cout << "Checking for an existing DB parameter group named '" <<
              PARAMETER_GROUP_NAME << "'." << std::endl;
    Aws::String dbParameterGroupFamily("Undefined");
    bool parameterGroupFound = true;
    {
        // 1. Check if the DB parameter group already exists.
        Aws::RDS::Model::DescribeDBParameterGroupsRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBParameterGroupsOutcome outcome =
                client.DescribeDBParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB parameter group named '" <<
                      PARAMETER_GROUP_NAME << "' already exists." << std::endl;
            dbParameterGroupFamily = outcome.GetResult().GetDBParameterGroups()
[0].GetDBParameterGroupFamily();
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::RDS::RDSErrors::D_B_PARAMETER_GROUP_NOT_FOUND_FAULT) {
            std::cout << "DB parameter group named '" <<
                      PARAMETER_GROUP_NAME << "' does not exist." << std::endl;
            parameterGroupFound = false;
        }
        else {
            std::cerr << "Error with RDS::DescribeDBParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    if (!parameterGroupFound) {
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 2. Get available engine versions for the specified engine.
        if (!getDBEngineVersions(DB_ENGINE, NO_PARAMETER_GROUP_FAMILY,
                                 engineVersions, client)) {
            return false;
        }

        std::cout << "Getting available database engine versions for " <<
 DB_ENGINE
                  << "."
```

```
                  << std::endl;
        std::vector<Aws::String> families;
        for (const Aws::RDS::Model::DBEngineVersion &version: engineVersions) {
            Aws::String family = version.GetDBParameterGroupFamily();
            if (std::find(families.begin(), families.end(), family) ==
                families.end()) {
                families.push_back(family);
                std::cout << "  " << families.size() << ": " << family <<
std::endl;
            }
        }

        int choice = askQuestionForIntRange("Which family do you want to use? ",
1,
                                            static_cast<int>(families.size()));
        dbParameterGroupFamily = families[choice - 1];
    }
    if (!parameterGroupFound) {
        // 3.  Create a DB parameter group.
        Aws::RDS::Model::CreateDBParameterGroupRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example parameter group.");

        Aws::RDS::Model::CreateDBParameterGroupOutcome outcome =
                client.CreateDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "Let's set some parameter values in your parameter group."
              << std::endl;

    Aws::String marker;
```

```
    Aws::Vector<Aws::RDS::Model::Parameter> autoIncrementParameters;
    // 4.  Get the parameters in the DB parameter group.
    if (!getDBParameters(PARAMETER_GROUP_NAME, AUTO_INCREMENT_PREFIX, NO_SOURCE,
                         autoIncrementParameters,
                         client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }

    Aws::Vector<Aws::RDS::Model::Parameter> updateParameters;

    for (Aws::RDS::Model::Parameter &autoIncParameter: autoIncrementParameters) {
        if (autoIncParameter.GetIsModifiable() &&
            (autoIncParameter.GetDataType() == "integer")) {
            std::cout << "The " << autoIncParameter.GetParameterName()
                      << " is described as: " <<
                      autoIncParameter.GetDescription() << "." << std::endl;
            if (autoIncParameter.ParameterValueHasBeenSet()) {
                std::cout << "The current value is "
                          << autoIncParameter.GetParameterValue()
                          << "." << std::endl;
            }
            std::vector<int> splitValues = splitToInts(
                    autoIncParameter.GetAllowedValues(), '-');
            if (splitValues.size() == 2) {
                int newValue = askQuestionForIntRange(
                        Aws::String("Enter a new value in the range ") +
                        autoIncParameter.GetAllowedValues() + ": ",
                        splitValues[0], splitValues[1]);
                autoIncParameter.SetParameterValue(std::to_string(newValue));
                updateParameters.push_back(autoIncParameter);

            }
            else {
                std::cerr << "Error parsing " <<
autoIncParameter.GetAllowedValues()
                          << std::endl;
            }
        }
    }

    {
        // 5.  Modify the auto increment parameters in the group.
        Aws::RDS::Model::ModifyDBParameterGroupRequest request;
```

```
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBParameterGroupOutcome outcome =
                client.ModifyDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::ModifyDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    std::cout
            << "You can get a list of parameters you've set by specifying a
source of 'user'."
            << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> userParameters;
    // 6.  Display the modified parameters in the group.
    if (!getDBParameters(PARAMETER_GROUP_NAME, NO_NAME_PREFIX, "user",
userParameters,
                         client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }

    for (const auto &userParameter: userParameters) {
        std::cout << "  " << userParameter.GetParameterName() << ", " <<
                  userParameter.GetDescription() << ", parameter value - "
                  << userParameter.GetParameterValue() << std::endl;
    }

    printAsterisksLine();
    std::cout << "Checking for an existing DB instance." << std::endl;

    Aws::RDS::Model::DBInstance dbInstance;
    // 7.  Check if the DB instance already exists.
    if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
```

```
        return false;
    }

    if (dbInstance.DbInstancePortHasBeenSet()) {
        std::cout << "The DB instance already exists." << std::endl;
    }
    else {
        std::cout << "Let's create a DB instance." << std::endl;
        const Aws::String administratorName = askQuestion(
                "Enter an administrator username for the database: ");
        const Aws::String administratorPassword = askQuestion(
                "Enter a password for the administrator (at least 8 characters):
");
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 8.  Get a list of available engine versions.
        if (!getDBEngineVersions(DB_ENGINE, dbParameterGroupFamily,
engineVersions,
                                 client)) {
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }

        std::cout << "The available engines for your parameter group are:" <<
std::endl;

        int index = 1;
        for (const Aws::RDS::Model::DBEngineVersion &engineVersion:
engineVersions) {
            std::cout << "   " << index << ": " <<
engineVersion.GetEngineVersion()
                      << std::endl;
            ++index;
        }
        int choice = askQuestionForIntRange("Which engine do you want to use? ",
1,

static_cast<int>(engineVersions.size()));
        const Aws::RDS::Model::DBEngineVersion engineVersion =
engineVersions[choice -
                                                                              1];

        Aws::String dbInstanceClass;
        // 9.  Get a list of micro instance classes.
```

```
        if (!chooseMicroDBInstanceClass(engineVersion.GetEngine(),
                                        engineVersion.GetEngineVersion(),
                                        dbInstanceClass,
                                        client)) {
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }

        std::cout << "Creating a DB instance named '" << DB_INSTANCE_IDENTIFIER
                  << "' and database '" << DB_NAME << "'.\n"
                  << "The DB instance is configured to use your custom parameter
group '"
                  << PARAMETER_GROUP_NAME << "',\n"
                  << "selected engine version " <<
engineVersion.GetEngineVersion()
                  << ",\n"
                  << "selected DB instance class '" << dbInstanceClass << "',"
                  << " and " << DB_ALLOCATED_STORAGE << " GiB of " <<
DB_STORAGE_TYPE
                  << " storage.\nThis typically takes several minutes." <<
std::endl;

        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBName(DB_NAME);
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetEngine(engineVersion.GetEngine());
        request.SetEngineVersion(engineVersion.GetEngineVersion());
        request.SetDBInstanceClass(dbInstanceClass);
        request.SetStorageType(DB_STORAGE_TYPE);
        request.SetAllocatedStorage(DB_ALLOCATED_STORAGE);
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
```

```
                    << std::endl;
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }
    }

    std::cout << "Waiting for the DB instance to become available." << std::endl;

    int counter = 0;
    // 11. Wait for the DB instance to become available.
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 900) {
            std::cerr << "Wait for instance to become available timed out ofter "
                      << counter
                      << " seconds." << std::endl;
            cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }

        dbInstance = Aws::RDS::Model::DBInstance();
        if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
            cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB instance status is '"
                      << dbInstance.GetDBInstanceStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbInstance.GetDBInstanceStatus() != "available");

    if (dbInstance.GetDBInstanceStatus() == "available") {
        std::cout << "The DB instance has been created." << std::endl;
    }

    printAsterisksLine();

    // 12. Display the connection string that can be used to connect a 'mysql'
shell to the database.
```

```
    displayConnection(dbInstance);

    printAsterisksLine();

    if (askYesNoQuestion(
            "Do you want to create a snapshot of your DB instance (y/n)? ")) {
        Aws::String snapshotID(DB_INSTANCE_IDENTIFIER + "-" +
                               Aws::String(Aws::Utils::UUID::RandomUUID()));
        {
            std::cout << "Creating a snapshot named " << snapshotID << "." <<
std::endl;
            std::cout << "This typically takes a few minutes." << std::endl;

            // 13. Create a snapshot of the DB instance.
            Aws::RDS::Model::CreateDBSnapshotRequest request;
            request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBSnapshotOutcome outcome =
                    client.CreateDBSnapshot(request);

            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with RDS::CreateDBSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
        }

        std::cout << "Waiting for snapshot to become available." << std::endl;

        Aws::RDS::Model::DBSnapshot snapshot;
        counter = 0;
        do {
            std::this_thread::sleep_for(std::chrono::seconds(1));
            ++counter;
            if (counter > 600) {
                std::cerr << "Wait for snapshot to be available timed out ofter "
```

```
                        << counter
                        << " seconds." << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }

            // 14. Wait for the snapshot to become available.
            Aws::RDS::Model::DescribeDBSnapshotsRequest request;
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBSnapshotsOutcome outcome =
                    client.DescribeDBSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBSnapshots()[0];
            }
            else {
                std::cerr << "Error with RDS::DescribeDBSnapshots. "
                        << outcome.GetError().GetMessage()
                        << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }

            if ((counter % 20) == 0) {
                std::cout << "Current snapshot status is '"
                        << snapshot.GetStatus()
                        << "' after " << counter << " seconds." << std::endl;
            }
        } while (snapshot.GetStatus() != "available");

        if (snapshot.GetStatus() != "available") {
            std::cout << "A snapshot has been created." << std::endl;
        }
    }

    printAsterisksLine();

    bool result = true;
    if (askYesNoQuestion(
            "Do you want to delete the DB instance and parameter group (y/n)? "))
{
```

```
        result = cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
 client);
    }

    return result;
}


//! Routine which gets DB parameters using the 'DescribeDBParameters' api.
/*!
 \sa getDBParameters()
 \param parameterGroupName: The name of the parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBParameters(const Aws::String &parameterGroupName,
                                  const Aws::String &namePrefix,
                                  const Aws::String &source,
                                  Aws::Vector<Aws::RDS::Model::Parameter>
 &parametersResult,
                                  const Aws::RDS::RDSClient &client) {
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeDBParametersRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBParametersOutcome outcome =
                client.DescribeDBParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
```

```
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBEngineVersions(const Aws::String &engineName,
                                      const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                      const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }
```

```
    engineVersionsResult.clear();
    Aws::String marker; // Used for pagination.

    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }


        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            auto &engineVersions = outcome.GetResult().GetDBEngineVersions();
            engineVersionsResult.insert(engineVersionsResult.end(),
 engineVersions.begin(),
                                        engineVersions.end());
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }

    } while (!marker.empty());


    return true;
}


//! Routine which gets a DB instance description.
/*!
 \sa describeDBInstance()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                     Aws::RDS::Model::DBInstance &instanceResult,
                                     const Aws::RDS::RDSClient &client) {
```

```
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with RDS::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}


//! Routine which gets available 'micro' DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseMicroDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::chooseMicroDBInstanceClass(const Aws::String &engine,
                                             const Aws::String &engineVersion,
                                             Aws::String &dbInstanceClass,
                                             const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker;
    do {
```

```
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption>
&options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option:
options) {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (instanceClass.find("micro") != std::string::npos) {
                    if (std::find(instanceClasses.begin(), instanceClasses.end(),
                                  instanceClass) ==
                        instanceClasses.end()) {
                        instanceClasses.push_back(instanceClass);
                    }
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeOrderableDBInstanceOptions. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available micro DB instance classes for your database
engine are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which micro DB instance class do you want to use? ",
```

```
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}

//! Routine which deletes resources created by the scenario.
/*!
\sa cleanUpResources()
\param parameterGroupName: A parameter group name, this may be empty.
\param dbInstanceIdentifier: A DB instance identifier, this may be empty.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::RDS::cleanUpResources(const Aws::String &parameterGroupName,
                                   const Aws::String &dbInstanceIdentifier,
                                   const Aws::RDS::RDSClient &client) {
    bool result = true;
    if (!dbInstanceIdentifier.empty()) {
        {
            // 15. Delete the DB instance.
            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with RDS::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }

        std::cout
                << "Waiting for DB instance to delete before deleting the
 parameter group."
                << std::endl;
```

```
        std::cout << "This may take a while." << std::endl;

        int counter = 0;
        Aws::RDS::Model::DBInstance dbInstance;
        do {
            std::this_thread::sleep_for(std::chrono::seconds(1));
            ++counter;
            if (counter > 800) {
                std::cerr << "Wait for instance to delete timed out ofter " <<
counter
                          << " seconds." << std::endl;
                return false;
            }

            dbInstance = Aws::RDS::Model::DBInstance();
            // 16. Wait for the DB instance to be deleted.
            if (!describeDBInstance(dbInstanceIdentifier, dbInstance, client)) {
                return false;
            }

            if (dbInstance.DBInstanceIdentifierHasBeenSet() && (counter % 20) ==
0) {
                std::cout << "Current DB instance status is '"
                          << dbInstance.GetDBInstanceStatus()
                          << "' after " << counter << " seconds." << std::endl;
            }
        } while (dbInstance.DBInstanceIdentifierHasBeenSet());
    }

    if (!parameterGroupName.empty()) {
        // 17. Delete the parameter group.
        Aws::RDS::Model::DeleteDBParameterGroupRequest request;
        request.SetDBParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBParameterGroupOutcome outcome =
                client.DeleteDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::DeleteDBParameterGroup. "
                      << outcome.GetError().GetMessage()
```

```
                        << std::endl;
            result = false;
        }
    }

    return result;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 다음 주제를 참조하십시오.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
// GetStartedInstances is an interactive example that shows you how to use the
 AWS SDK for Go
// with Amazon Relation Database Service (Amazon RDS) to do the following:
//
//  1. Create a custom DB parameter group and set parameter values.
//  2. Create a DB instance that is configured to use the parameter group. The DB
 instance
//     also contains a database.
//  3. Take a snapshot of the DB instance.
//  4. Delete the DB instance and parameter group.
type GetStartedInstances struct {
 sdkConfig  aws.Config
 instances  actions.DbInstances
 questioner demotools.IQuestioner
 helper     IScenarioHelper
 isTestRun  bool
}

// NewGetStartedInstances constructs a GetStartedInstances instance from a
 configuration.
// It uses the specified config to get an Amazon RDS
// client and create wrappers for the actions used in the scenario.
func NewGetStartedInstances(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
 helper IScenarioHelper) GetStartedInstances {
 rdsClient := rds.NewFromConfig(sdkConfig)
 return GetStartedInstances{
  sdkConfig:  sdkConfig,
  instances:  actions.DbInstances{RdsClient: rdsClient},
  questioner: questioner,
  helper:     helper,
 }
}

// Run runs the interactive scenario.
func (scenario GetStartedInstances) Run(dbEngine string, parameterGroupName
 string,
 instanceName string, dbName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
  }
 }()
```

```
 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon Relational Database Service (Amazon RDS) DB
 Instance demo.")
 log.Println(strings.Repeat("-", 88))

 parameterGroup := scenario.CreateParameterGroup(dbEngine, parameterGroupName)
 scenario.SetUserParameters(parameterGroupName)
 instance := scenario.CreateInstance(instanceName, dbEngine, dbName,
 parameterGroup)
 scenario.DisplayConnection(instance)
 scenario.CreateSnapshot(instance)
 scenario.Cleanup(instance, parameterGroup)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}

// CreateParameterGroup shows how to get available engine versions for a
 specified
// database engine and create a DB parameter group that is compatible with a
// selected engine family.
func (scenario GetStartedInstances) CreateParameterGroup(dbEngine string,
 parameterGroupName string) *types.DBParameterGroup {

 log.Printf("Checking for an existing DB parameter group named %v.\n",
  parameterGroupName)
 parameterGroup, err := scenario.instances.GetParameterGroup(parameterGroupName)
 if err != nil {
  panic(err)
 }
 if parameterGroup == nil {
  log.Printf("Getting available database engine versions for %v.\n", dbEngine)
  engineVersions, err := scenario.instances.GetEngineVersions(dbEngine, "")
  if err != nil {
   panic(err)
  }

  familySet := map[string]struct{}{}
  for _, family := range engineVersions {
   familySet[*family.DBParameterGroupFamily] = struct{}{}
  }
  var families []string
  for family := range familySet {
```

```
    families = append(families, family)
   }
   sort.Strings(families)
   familyIndex := scenario.questioner.AskChoice("Which family do you want to use?
\n", families)
   log.Println("Creating a DB parameter group.")
   _, err = scenario.instances.CreateParameterGroup(
    parameterGroupName, families[familyIndex], "Example parameter group.")
   if err != nil {
    panic(err)
   }
   parameterGroup, err = scenario.instances.GetParameterGroup(parameterGroupName)
   if err != nil {
    panic(err)
   }
  }
  log.Printf("Parameter group %v:\n", *parameterGroup.DBParameterGroupFamily)
  log.Printf("\tName: %v\n", *parameterGroup.DBParameterGroupName)
  log.Printf("\tARN: %v\n", *parameterGroup.DBParameterGroupArn)
  log.Printf("\tFamily: %v\n", *parameterGroup.DBParameterGroupFamily)
  log.Printf("\tDescription: %v\n", *parameterGroup.Description)
  log.Println(strings.Repeat("-", 88))
  return parameterGroup
}

// SetUserParameters shows how to get the parameters contained in a custom
 parameter
// group and update some of the parameter values in the group.
func (scenario GetStartedInstances) SetUserParameters(parameterGroupName string)
 {
  log.Println("Let's set some parameter values in your parameter group.")
  dbParameters, err := scenario.instances.GetParameters(parameterGroupName, "")
  if err != nil {
   panic(err)
  }
  var updateParams []types.Parameter
  for _, dbParam := range dbParameters {
   if strings.HasPrefix(*dbParam.ParameterName, "auto_increment") &&
    dbParam.IsModifiable && *dbParam.DataType == "integer" {
    log.Printf("The %v parameter is described as:\n\t%v",
     *dbParam.ParameterName, *dbParam.Description)
    rangeSplit := strings.Split(*dbParam.AllowedValues, "-")
    lower, _ := strconv.Atoi(rangeSplit[0])
    upper, _ := strconv.Atoi(rangeSplit[1])
```

```
    newValue := scenario.questioner.AskInt(
     fmt.Sprintf("Enter a value between %v and %v:", lower, upper),
     demotools.InIntRange{Lower: lower, Upper: upper})
    dbParam.ParameterValue = aws.String(strconv.Itoa(newValue))
    updateParams = append(updateParams, dbParam)
   }
  }
  err = scenario.instances.UpdateParameters(parameterGroupName, updateParams)
  if err != nil {
   panic(err)
  }
  log.Println("To get a list of parameters that you set previously, specify a
  source of 'user'.")
  userParameters, err := scenario.instances.GetParameters(parameterGroupName,
  "user")
  if err != nil {
   panic(err)
  }
  log.Println("Here are the parameters you set:")
  for _, param := range userParameters {
   log.Printf("\t%v: %v\n", *param.ParameterName, *param.ParameterValue)
  }
  log.Println(strings.Repeat("-", 88))
}

// CreateInstance shows how to create a DB instance that contains a database of a
// specified type. The database is also configured to use a custom DB parameter
 group.
func (scenario GetStartedInstances) CreateInstance(instanceName string, dbEngine
 string,
 dbName string, parameterGroup *types.DBParameterGroup) *types.DBInstance {

 log.Println("Checking for an existing DB instance.")
 instance, err := scenario.instances.GetInstance(instanceName)
 if err != nil {
  panic(err)
 }
 if instance == nil {
  adminUsername := scenario.questioner.Ask(
   "Enter an administrator username for the database: ", demotools.NotEmpty{})
  adminPassword := scenario.questioner.AskPassword(
   "Enter a password for the administrator (at least 8 characters): ", 7)
  engineVersions, err := scenario.instances.GetEngineVersions(dbEngine,
   *parameterGroup.DBParameterGroupFamily)
```

```
 if err != nil {
  panic(err)
 }
 var engineChoices []string
 for _, engine := range engineVersions {
  engineChoices = append(engineChoices, *engine.EngineVersion)
 }
 engineIndex := scenario.questioner.AskChoice(
  "The available engines for your parameter group are:\n", engineChoices)
 engineSelection := engineVersions[engineIndex]
 instOpts, err :=
scenario.instances.GetOrderableInstances(*engineSelection.Engine,
  *engineSelection.EngineVersion)
 if err != nil {
  panic(err)
 }
 optSet := map[string]struct{}{}
 for _, opt := range instOpts {
  if strings.Contains(*opt.DBInstanceClass, "micro") {
   optSet[*opt.DBInstanceClass] = struct{}{}
  }
 }
 var optChoices []string
 for opt := range optSet {
  optChoices = append(optChoices, opt)
 }
 sort.Strings(optChoices)
 optIndex := scenario.questioner.AskChoice(
  "The available micro DB instance classes for your database engine are:\n",
optChoices)
 storageType := "standard"
 allocatedStorage := int32(5)
 log.Printf("Creating a DB instance named %v and database %v.\n"+
  "The DB instance is configured to use your custom parameter group %v,\n"+
  "selected engine %v,\n"+
  "selected DB instance class %v,"+
  "and %v GiB of %v storage.\n"+
  "This typically takes several minutes.",
  instanceName, dbName, *parameterGroup.DBParameterGroupName,
*engineSelection.EngineVersion,
  optChoices[optIndex], allocatedStorage, storageType)
 instance, err = scenario.instances.CreateInstance(
  instanceName, dbName, *engineSelection.Engine, *engineSelection.EngineVersion,
  *parameterGroup.DBParameterGroupName, optChoices[optIndex], storageType,
```

```
    allocatedStorage, adminUsername, adminPassword)
   if err != nil {
    panic(err)
   }
   for *instance.DBInstanceStatus != "available" {
    scenario.helper.Pause(30)
    instance, err = scenario.instances.GetInstance(instanceName)
    if err != nil {
     panic(err)
    }
   }
   log.Println("Instance created and available.")
  }
  log.Println("Instance data:")
  log.Printf("\tDBInstanceIdentifier: %v\n", *instance.DBInstanceIdentifier)
  log.Printf("\tARN: %v\n", *instance.DBInstanceArn)
  log.Printf("\tStatus: %v\n", *instance.DBInstanceStatus)
  log.Printf("\tEngine: %v\n", *instance.Engine)
  log.Printf("\tEngine version: %v\n", *instance.EngineVersion)
  log.Println(strings.Repeat("-", 88))
  return instance
}

// DisplayConnection displays connection information about a DB instance and tips
// on how to connect to it.
func (scenario GetStartedInstances) DisplayConnection(instance *types.DBInstance)
 {
  log.Println(
   "You can now connect to your database by using your favorite MySQL client.\n" +
    "One way to connect is by using the 'mysql' shell on an Amazon EC2 instance\n"
  +
    "that is running in the same VPC as your DB instance. Pass the endpoint,\n" +
    "port, and administrator username to 'mysql'. Then, enter your password\n" +
    "when prompted:")
  log.Printf("\n\tmysql -h %v -P %v -u %v -p\n",
   *instance.Endpoint.Address, instance.Endpoint.Port, *instance.MasterUsername)
  log.Println("For more information, see the User Guide for RDS:\n" +
   "\thttps://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
CHAP_GettingStarted.CreatingConnecting.MySQL.html#CHAP_GettingStarted.Connecting.MySQL")
  log.Println(strings.Repeat("-", 88))
}

// CreateSnapshot shows how to create a DB instance snapshot and wait until it's
 available.
```

```
func (scenario GetStartedInstances) CreateSnapshot(instance *types.DBInstance) {
 if scenario.questioner.AskBool(
  "Do you want to create a snapshot of your DB instance (y/n)? ", "y") {
  snapshotId := fmt.Sprintf("%v-%v", *instance.DBInstanceIdentifier,
 scenario.helper.UniqueId())
  log.Printf("Creating a snapshot named %v. This typically takes a few minutes.
\n", snapshotId)
  snapshot, err :=
 scenario.instances.CreateSnapshot(*instance.DBInstanceIdentifier, snapshotId)
  if err != nil {
   panic(err)
  }
  for *snapshot.Status != "available" {
   scenario.helper.Pause(30)
   snapshot, err = scenario.instances.GetSnapshot(snapshotId)
   if err != nil {
    panic(err)
   }
  }
  log.Println("Snapshot data:")
  log.Printf("\tDBSnapshotIdentifier: %v\n", *snapshot.DBSnapshotIdentifier)
  log.Printf("\tARN: %v\n", *snapshot.DBSnapshotArn)
  log.Printf("\tStatus: %v\n", *snapshot.Status)
  log.Printf("\tEngine: %v\n", *snapshot.Engine)
  log.Printf("\tEngine version: %v\n", *snapshot.EngineVersion)
  log.Printf("\tDBInstanceIdentifier: %v\n", *snapshot.DBInstanceIdentifier)
  log.Printf("\tSnapshotCreateTime: %v\n", *snapshot.SnapshotCreateTime)
  log.Println(strings.Repeat("-", 88))
 }
}

// Cleanup shows how to clean up a DB instance and DB parameter group.
// Before the DB parameter group can be deleted, all associated DB instances must
 first be deleted.
func (scenario GetStartedInstances) Cleanup(
 instance *types.DBInstance, parameterGroup *types.DBParameterGroup) {

 if scenario.questioner.AskBool(
  "\nDo you want to delete the database instance and parameter group (y/n)? ",
 "y") {
  log.Printf("Deleting database instance %v.\n", *instance.DBInstanceIdentifier)
  err := scenario.instances.DeleteInstance(*instance.DBInstanceIdentifier)
  if err != nil {
   panic(err)
```

```
  }
  log.Println(
   "Waiting for the DB instance to delete. This typically takes several
 minutes.")
  for instance != nil {
   scenario.helper.Pause(30)
   instance, err = scenario.instances.GetInstance(*instance.DBInstanceIdentifier)
   if err != nil {
    panic(err)
   }
  }
  log.Printf("Deleting parameter group %v.",
 *parameterGroup.DBParameterGroupName)
  err =
 scenario.instances.DeleteParameterGroup(*parameterGroup.DBParameterGroupName)
  if err != nil {
   panic(err)
  }
 }
}
```

Amazon RDS 작업을 관리하기 위해 시나리오에서 호출하는 함수를 정의합니다.

```
type DbInstances struct {
 RdsClient *rds.Client
}


// GetParameterGroup gets a DB parameter group by name.
func (instances *DbInstances) GetParameterGroup(parameterGroupName string) (
 *types.DBParameterGroup, error) {
 output, err := instances.RdsClient.DescribeDBParameterGroups(
  context.TODO(), &rds.DescribeDBParameterGroupsInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
```

```
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBParameterGroups[0], err
 }
}



// CreateParameterGroup creates a DB parameter group that is based on the
 specified
// parameter group family.
func (instances *DbInstances) CreateParameterGroup(
 parameterGroupName string, parameterGroupFamily string, description string) (
 *types.DBParameterGroup, error) {

 output, err := instances.RdsClient.CreateDBParameterGroup(context.TODO(),
  &rds.CreateDBParameterGroupInput{
   DBParameterGroupName:   aws.String(parameterGroupName),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
   Description:            aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBParameterGroup, err
 }
}



// DeleteParameterGroup deletes the named DB parameter group.
func (instances *DbInstances) DeleteParameterGroup(parameterGroupName string)
 error {
 _, err := instances.RdsClient.DeleteDBParameterGroup(context.TODO(),
  &rds.DeleteDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
```

```
 } else {
  return nil
 }
}



// GetParameters gets the parameters that are contained in a DB parameter group.
func (instances *DbInstances) GetParameters(parameterGroupName string, source
 string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator := rds.NewDescribeDBParametersPaginator(instances.RdsClient,
  &rds.DescribeDBParametersInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Source:               aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(context.TODO())
  if err != nil {
   log.Printf("Couldn't get parameters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}



// UpdateParameters updates parameters in a named DB parameter group.
func (instances *DbInstances) UpdateParameters(parameterGroupName string, params
 []types.Parameter) error {
 _, err := instances.RdsClient.ModifyDBParameterGroup(context.TODO(),
  &rds.ModifyDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Parameters:           params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
```

```
  return err
 } else {
  return nil
 }
}



// CreateSnapshot creates a snapshot of a DB instance.
func (instances *DbInstances) CreateSnapshot(instanceName string, snapshotName
 string) (
 *types.DBSnapshot, error) {
 output, err := instances.RdsClient.CreateDBSnapshot(context.TODO(),
 &rds.CreateDBSnapshotInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBSnapshot, nil
 }
}



// GetSnapshot gets a DB instance snapshot.
func (instances *DbInstances) GetSnapshot(snapshotName string)
 (*types.DBSnapshot, error) {
 output, err := instances.RdsClient.DescribeDBSnapshots(context.TODO(),
  &rds.DescribeDBSnapshotsInput{
   DBSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBSnapshots[0], nil
 }
}
```

```
// CreateInstance creates a DB instance.
func (instances *DbInstances) CreateInstance(instanceName string, dbName string,
 dbEngine string, dbEngineVersion string, parameterGroupName string,
 dbInstanceClass string,
 storageType string, allocatedStorage int32, adminName string, adminPassword
 string) (
 *types.DBInstance, error) {
 output, err := instances.RdsClient.CreateDBInstance(context.TODO(),
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBName:               aws.String(dbName),
  DBParameterGroupName: aws.String(parameterGroupName),
  Engine:               aws.String(dbEngine),
  EngineVersion:        aws.String(dbEngineVersion),
  DBInstanceClass:      aws.String(dbInstanceClass),
  StorageType:          aws.String(storageType),
  AllocatedStorage:     aws.Int32(allocatedStorage),
  MasterUsername:       aws.String(adminName),
  MasterUserPassword:   aws.String(adminPassword),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}



// GetInstance gets data about a DB instance.
func (instances *DbInstances) GetInstance(instanceName string) (
 *types.DBInstance, error) {
 output, err := instances.RdsClient.DescribeDBInstances(context.TODO(),
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
```

```
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}



// DeleteInstance deletes a DB instance.
func (instances *DbInstances) DeleteInstance(instanceName string) error {
 _, err := instances.RdsClient.DeleteDBInstance(context.TODO(),
 &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      true,
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (instances *DbInstances) GetEngineVersions(engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := instances.RdsClient.DescribeDBEngineVersions(context.TODO(),
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
```

```
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (instances *DbInstances) GetOrderableInstances(engine string, engineVersion
 string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instanceOptions []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(instances.RdsClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(context.TODO())
  if err != nil {
   log.Printf("Couldn't get orderable DB instance options: %v\n", err)
   break
  } else {
   instanceOptions = append(instanceOptions,
 output.OrderableDBInstanceOptions...)
  }
 }
 return instanceOptions, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하십시오.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup

- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeDBParameterGroups
- DescribeDBParameters
- DescribeDBSnapshots
- DescribeOrderableDBInstanceOptions
- ModifyDBParameterGroup

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

여러 작업을 실행합니다.

```
import com.google.gson.Gson;
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotRequest;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotResponse;
import software.amazon.awssdk.services.rds.model.DBEngineVersion;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.DBParameterGroup;
import software.amazon.awssdk.services.rds.model.DBSnapshot;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsRequest;
import
 software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsResponse;
```

```
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import
 software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsResponse;
import
 software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsResponse;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.OrderableDBInstanceOption;
import software.amazon.awssdk.services.rds.model.Parameter;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupRequest;
import
 software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupRequest;
import
 software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbParameterGroupRequest;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
```

```
 *
 * This Java example performs these tasks:
 *
 * 1. Returns a list of the available DB engines.
 * 2. Selects an engine family and create a custom DB parameter group.
 * 3. Gets the parameter groups.
 * 4. Gets parameters in the group.
 * 5. Modifies the auto_increment_offset parameter.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions.
 * 8. Gets a list of micro instance classes available for the selected engine.
 * 9. Creates an RDS database instance that contains a MySql database and uses
 * the parameter group.
 * 10. Waits for the DB instance to be ready and prints out the connection
 * endpoint value.
 * 11. Creates a snapshot of the DB instance.
 * 12. Waits for an RDS DB snapshot to be ready.
 * 13. Deletes the RDS DB instance.
 * 14. Deletes the parameter group.
 */
public class RDSScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = """

                Usage:
                    <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier>
 <dbName> <dbSnapshotIdentifier> <secretName>

                Where:
                    dbGroupName - The database group name.\s
                    dbParameterGroupFamily - The database parameter group name
 (for example, mysql8.0).
                    dbInstanceIdentifier - The database instance identifier\s
                    dbName - The database name.\s
                    dbSnapshotIdentifier - The snapshot identifier.\s
                    secretName - The name of the AWS Secrets Manager secret that
 contains the database credentials"
                """;

        if (args.length != 6) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String dbGroupName = args[0];
        String dbParameterGroupFamily = args[1];
        String dbInstanceIdentifier = args[2];
        String dbName = args[3];
        String dbSnapshotIdentifier = args[4];
        String secretName = args[5];

        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
        String masterUsername = user.getUsername();
        String masterUserPassword = user.getPassword();

        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();
        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon RDS example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Return a list of the available DB engines");
        describeDBEngines(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a custom parameter group");
        createDBParameterGroup(rdsClient, dbGroupName, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get the parameter group");
        describeDbParameterGroups(rdsClient, dbGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the parameters in the group");
        describeDbParameters(rdsClient, dbGroupName, 0);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("5. Modify the auto_increment_offset parameter");
        modifyDBParas(rdsClient, dbGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Display the updated value");
        describeDbParameters(rdsClient, dbGroupName, -1);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Get a list of allowed engine versions");
        getAllowedEngines(rdsClient, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Get a list of micro instance classes available for
the selected engine");
        getMicroInstances(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "9. Create an RDS database instance that contains a MySql
database and uses the parameter group");
        String dbARN = createDatabaseInstance(rdsClient, dbGroupName,
dbInstanceIdentifier, dbName, masterUsername,
                masterUserPassword);
        System.out.println("The ARN of the new database is " + dbARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Wait for DB instance to be ready");
        waitForInstanceReady(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Create a snapshot of the DB instance");
        createSnapshot(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Wait for DB snapshot to be ready");
```

```
        waitForSnapshotReady(rdsClient, dbInstanceIdentifier,
dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the DB instance");
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the parameter group");
        deleteParaGroup(rdsClient, dbGroupName, dbARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);

        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
                .region(region)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                .build();
    }

    public static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                .secretId(secretName)
                .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    // Delete the parameter group after database has been deleted.
    // An exception is thrown if you attempt to delete the para group while
database
```

```
   // exists.
   public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
           throws InterruptedException {
       try {
           boolean isDataDel = false;
           boolean didFind;
           String instanceARN;

           // Make sure that the database has been deleted.
           while (!isDataDel) {
               DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
               List<DBInstance> instanceList = response.dbInstances();
               int listSize = instanceList.size();
               didFind = false;
               int index = 1;
               for (DBInstance instance : instanceList) {
                   instanceARN = instance.dbInstanceArn();
                   if (instanceARN.compareTo(dbARN) == 0) {
                       System.out.println(dbARN + " still exists");
                       didFind = true;
                   }
                   if ((index == listSize) && (!didFind)) {
                       // Went through the entire list and did not find the
database ARN.
                       isDataDel = true;
                   }
                   Thread.sleep(sleepTime * 1000);
                   index++;
               }
           }

           // Delete the para group.
           DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
                   .dbParameterGroupName(dbGroupName)
                   .build();

           rdsClient.deleteDBParameterGroup(parameterGroupRequest);
           System.out.println(dbGroupName + " was deleted.");

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
```

```
            System.exit(1);
        }
    }

    // Delete the DB instance.
    public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .deleteAutomatedBackups(true)
                    .skipFinalSnapshot(true)
                    .build();

            DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.print("The status of the database is " +
response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the snapshot instance is available.
    public static void waitForSnapshotReady(RdsClient rdsClient, String
dbInstanceIdentifier,
            String dbSnapshotIdentifier) {
        try {
            boolean snapshotReady = false;
            String snapshotReadyStr;
            System.out.println("Waiting for the snapshot to become available.");

            DescribeDbSnapshotsRequest snapshotsRequest =
DescribeDbSnapshotsRequest.builder()
                    .dbSnapshotIdentifier(dbSnapshotIdentifier)
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            while (!snapshotReady) {
                DescribeDbSnapshotsResponse response =
rdsClient.describeDBSnapshots(snapshotsRequest);
```

```
                List<DBSnapshot> snapshotList = response.dbSnapshots();
                for (DBSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");
        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Create an Amazon RDS snapshot.
    public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
        try {
            CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();

            CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
            System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
```

```
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            String endpoint = "";
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint().address();
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Create a database instance and return the ARN of the database.
    public static String createDatabaseInstance(RdsClient rdsClient,
            String dbGroupName,
            String dbInstanceIdentifier,
            String dbName,
            String masterUsername,
            String masterUserPassword) {

        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
```

```
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .allocatedStorage(100)
                    .dbName(dbName)
                    .dbParameterGroupName(dbGroupName)
                    .engine("mysql")
                    .dbInstanceClass("db.m4.large")
                    .engineVersion("8.0")
                    .storageType("standard")
                    .masterUsername(masterUsername)
                    .masterUserPassword(masterUserPassword)
                    .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
            return response.dbInstance().dbInstanceArn();

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }

        return "";
    }

    // Get a list of micro instances.
    public static void getMicroInstances(RdsClient rdsClient) {
        try {
            DescribeOrderableDbInstanceOptionsRequest dbInstanceOptionsRequest =
DescribeOrderableDbInstanceOptionsRequest
                    .builder()
                    .engine("mysql")
                    .build();

            DescribeOrderableDbInstanceOptionsResponse response = rdsClient

.describeOrderableDBInstanceOptions(dbInstanceOptionsRequest);
            List<OrderableDBInstanceOption> orderableDBInstances =
response.orderableDBInstanceOptions();
            for (OrderableDBInstanceOption dbInstanceOption :
orderableDBInstances) {
                System.out.println("The engine version is " +
dbInstanceOption.engineVersion());
```

```
                System.out.println("The engine description is " +
dbInstanceOption.engine());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Get a list of allowed engine versions.
    public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
        try {
            DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .engine("mysql")
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
            List<DBEngineVersion> dbEngines = response.dbEngineVersions();
            for (DBEngineVersion dbEngine : dbEngines) {
                System.out.println("The engine version is " +
dbEngine.engineVersion());
                System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Modify auto_increment_offset and auto_increment_increment parameters.
    public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
        try {
            Parameter parameter1 = Parameter.builder()
                    .parameterName("auto_increment_offset")
                    .applyMethod("immediate")
                    .parameterValue("5")
                    .build();
```

```
            List<Parameter> paraList = new ArrayList<>();
            paraList.add(parameter1);
            ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .parameters(paraList)
                    .build();

            ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
            System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Retrieve parameters in the group.
    public static void describeDbParameters(RdsClient rdsClient, String
dbGroupName, int flag) {
        try {
            DescribeDbParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .source("user")
                        .build();
            }

            DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
```

```
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
        try {
            DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .maxRecords(20)
                    .build();

            DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
            List<DBParameterGroup> groups = response.dbParameterGroups();
            for (DBParameterGroup group : groups) {
                System.out.println("The group name is " +
group.dbParameterGroupName());
                System.out.println("The group description is " +
group.description());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
```

```
        }
    }

    public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
        try {
            CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .description("Created by using the AWS SDK for Java")
                    .build();

            CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
            System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .defaultOnly(true)
                    .engine("mysql")
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
engineOb.engine());
```

```
                System.out.println("The version number of the database engine " +
 engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 항목을 참조하세요.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
Before running this code example, set up your development environment, including
 your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This example requires an AWS Secrets Manager secret that contains the database
 credentials. If you do not create a
secret, this example will not work. For more details, see:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html

This example performs the following tasks:

1. Returns a list of the available DB engines by invoking the
 DescribeDbEngineVersions method.
2. Selects an engine family and create a custom DB parameter group by invoking
 the createDBParameterGroup method.
3. Gets the parameter groups by invoking the DescribeDbParameterGroups method.
4. Gets parameters in the group by invoking the DescribeDbParameters method.
5. Modifies both the auto_increment_offset and auto_increment_increment
 parameters by invoking the modifyDbParameterGroup method.
6. Gets and displays the updated parameters.
7. Gets a list of allowed engine versions by invoking the
 describeDbEngineVersions method.
8. Gets a list of micro instance classes available for the selected engine.
9. Creates an Amazon Relational Database Service (Amazon RDS) database instance
 that contains a MySQL database and uses the parameter group.
10. Waits for DB instance to be ready and prints out the connection endpoint
 value.
11. Creates a snapshot of the DB instance.
12. Waits for the DB snapshot to be ready.
13. Deletes the DB instance.
14. Deletes the parameter group.
 */

var sleepTime: Long = 20

suspend fun main(args: Array<String>) {
```

```
   val usage = """
       Usage:
           <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier>
<dbName> <dbSnapshotIdentifier><secretName>

       Where:
           dbGroupName - The database group name.
           dbParameterGroupFamily - The database parameter group name.
           dbInstanceIdentifier - The database instance identifier.
           dbName -  The database name.
           dbSnapshotIdentifier - The snapshot identifier.
           secretName - The name of the AWS Secrets Manager secret that contains
the database credentials.
   """

   if (args.size != 6) {
       println(usage)
       exitProcess(1)
   }

   val dbGroupName = args[0]
   val dbParameterGroupFamily = args[1]
   val dbInstanceIdentifier = args[2]
   val dbName = args[3]
   val dbSnapshotIdentifier = args[4]
   val secretName = args[5]

   val gson = Gson()
   val user = gson.fromJson(getSecretValues(secretName).toString(),
User::class.java)
   val username = user.username
   val userPassword = user.password

   println("1. Return a list of the available DB engines")
   describeDBEngines()

   println("2. Create a custom parameter group")
   createDBParameterGroup(dbGroupName, dbParameterGroupFamily)

   println("3. Get the parameter groups")
   describeDbParameterGroups(dbGroupName)

   println("4. Get the parameters in the group")
   describeDbParameters(dbGroupName, 0)
```

```
    println("5. Modify the auto_increment_offset parameter")
    modifyDBParas(dbGroupName)

    println("6. Display the updated value")
    describeDbParameters(dbGroupName, -1)

    println("7. Get a list of allowed engine versions")
    getAllowedEngines(dbParameterGroupFamily)

    println("8. Get a list of micro instance classes available for the selected
 engine")
    getMicroInstances()

    println("9. Create an RDS database instance that contains a MySql database
 and uses the parameter group")
    val dbARN = createDatabaseInstance(dbGroupName, dbInstanceIdentifier, dbName,
 username, userPassword)
    println("The ARN of the new database is $dbARN")

    println("10. Wait for DB instance to be ready")
    waitForDbInstanceReady(dbInstanceIdentifier)

    println("11. Create a snapshot of the DB instance")
    createDbSnapshot(dbInstanceIdentifier, dbSnapshotIdentifier)

    println("12. Wait for DB snapshot to be ready")
    waitForSnapshotReady(dbInstanceIdentifier, dbSnapshotIdentifier)

    println("13. Delete the DB instance")
    deleteDbInstance(dbInstanceIdentifier)

    println("14. Delete the parameter group")
    if (dbARN != null) {
        deleteParaGroup(dbGroupName, dbARN)
    }

    println("The Scenario has successfully completed.")
}

suspend fun deleteParaGroup(
    dbGroupName: String,
    dbARN: String,
) {
```

```
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false // Reset this value.
            didFind = false // Reset this value.
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(dbARN) == 0) {
                        println("$dbARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database name.
                        isDataDel = true
                    }
                    index++
                }
            }
        }

        // Delete the para group.
        val parameterGroupRequest =
            DeleteDbParameterGroupRequest {
                dbParameterGroupName = dbGroupName
            }
        rdsClient.deleteDbParameterGroup(parameterGroupRequest)
        println("$dbGroupName was deleted.")
    }
}

suspend fun deleteDbInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
```

```
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}

// Waits until the snapshot instance is available.
suspend fun waitForSnapshotReady(
    dbInstanceIdentifierVal: String?,
    dbSnapshotIdentifierVal: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbSnapshotsRequest {
            dbSnapshotIdentifier = dbSnapshotIdentifierVal
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    while (!snapshotReady) {
        RdsClient { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbSnapshots(snapshotsRequest)
            val snapshotList: List<DbSnapshot>? = response.dbSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        print(".")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
```

```
}

// Create an Amazon RDS snapshot.
suspend fun createDbSnapshot(
    dbInstanceIdentifierVal: String?,
    dbSnapshotIdentifierVal: String?,
) {
    val snapshotRequest =
        CreateDbSnapshotRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbSnapshotIdentifier = dbSnapshotIdentifierVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbSnapshot(snapshotRequest)
        print("The Snapshot id is ${response.dbSnapshot?.dbiResourceId}")
    }
}

// Waits until the database instance is available.
suspend fun waitForDbInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }
    var endpoint = ""
    while (!instanceReady) {
        RdsClient { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus.toString()
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint?.address.toString()
                        instanceReady = true
                    } else {
                        print(".")
                        delay(sleepTime * 1000)
                    }
```

```
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is
 $endpoint")
}

// Create a database instance and return the ARN of the database.
suspend fun createDatabaseInstance(
    dbGroupNameVal: String?,
    dbInstanceIdentifierVal: String?,
    dbNameVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNameVal
            dbParameterGroupName = dbGroupNameVal
            engine = "mysql"
            dbInstanceClass = "db.m4.large"
            engineVersion = "8.0"
            storageType = "standard"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}

// Get a list of micro instances.
suspend fun getMicroInstances() {
    val dbInstanceOptionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "mysql"
        }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
```

```
        val response =
 rdsClient.describeOrderableDbInstanceOptions(dbInstanceOptionsRequest)
        val orderableDBInstances = response.orderableDbInstanceOptions
        if (orderableDBInstances != null) {
            for (dbInstanceOption in orderableDBInstances) {
                println("The engine version is
 ${dbInstanceOption.engineVersion}")
                println("The engine description is ${dbInstanceOption.engine}")
            }
        }
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "mysql"
        }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        val dbEngines: List<DbEngineVersion>? = response.dbEngineVersions
        if (dbEngines != null) {
            for (dbEngine in dbEngines) {
                println("The engine version is ${dbEngine.engineVersion}")
                println("The engine description is
 ${dbEngine.dbEngineDescription}")
            }
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBParas(dbGroupName: String) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.Immediate
            parameterValue = "5"
        }

    val paraList: ArrayList<Parameter> = ArrayList()
    paraList.add(parameter1)
```

```
    val groupRequest =
        ModifyDbParameterGroupRequest {
            dbParameterGroupName = dbGroupName
            parameters = paraList
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbParameterGroup(groupRequest)
        println("The parameter group ${response.dbParameterGroupName} was
 successfully modified")
    }
}

// Retrieve parameters in the group.
suspend fun describeDbParameters(
    dbGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbParametersRequest {
                dbParameterGroupName = dbGroupName
            }
        } else {
            DescribeDbParametersRequest {
                dbParameterGroupName = dbGroupName
                source = "user"
            }
        }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbParameters(dbParameterGroupsRequest)
        val dbParameters: List<Parameter>? = response.parameters
        var paraName: String
        if (dbParameters != null) {
            for (para in dbParameters) {
                // Only print out information about either auto_increment_offset
 or auto_increment_increment.
                paraName = para.parameterName.toString()
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    System.out.println("*** The parameter value is
 ${para.parameterValue}")
```

```
                    System.out.println("*** The parameter data type is
 ${para.dataType}")
                    System.out.println("*** The parameter description is
 ${para.description}")
                    System.out.println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}

suspend fun describeDbParameterGroups(dbGroupName: String?) {
    val groupsRequest =
        DescribeDbParameterGroupsRequest {
            dbParameterGroupName = dbGroupName
            maxRecords = 20
        }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbParameterGroups(groupsRequest)
        val groups = response.dbParameterGroups
        if (groups != null) {
            for (group in groups) {
                println("The group name is ${group.dbParameterGroupName}")
                println("The group description is ${group.description}")
            }
        }
    }
}

// Create a parameter group.
suspend fun createDBParameterGroup(
    dbGroupName: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbParameterGroupRequest {
            dbParameterGroupName = dbGroupName
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbParameterGroup(groupRequest)
```

```
        println("The group name is
 ${response.dbParameterGroup?.dbParameterGroupName}")
    }
}

// Returns a list of the available DB engines.
suspend fun describeDBEngines() {
    val engineVersionsRequest =
        DescribeDbEngineVersionsRequest {
            defaultOnly = true
            engine = "mysql"
            maxRecords = 20
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        val engines: List<DbEngineVersion>? = response.dbEngineVersions

        // Get all DbEngineVersion objects.
        if (engines != null) {
            for (engineOb in engines) {
                println("The name of the DB parameter group family for the
 database engine is ${engineOb.dbParameterGroupFamily}.")
                println("The name of the database engine ${engineOb.engine}.")
                println("The version number of the database engine
 ${engineOb.engineVersion}")
            }
        }
    }
}

suspend fun getSecretValues(secretName: String?): String? {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient { region = "us-west-2" }.use { secretsClient ->
        val valueResponse = secretsClient.getSecretValue(valueRequest)
        return valueResponse.secretString
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하십시오.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
class RdsInstanceScenario:
    """Runs a scenario that shows how to get started using Amazon RDS DB
 instances."""

    def __init__(self, instance_wrapper):
        """
        :param instance_wrapper: An object that wraps Amazon RDS DB instance
 actions.
        """
        self.instance_wrapper = instance_wrapper
```

```
    def create_parameter_group(self, parameter_group_name, db_engine):
        """
        Shows how to get available engine versions for a specified database
engine and
        create a DB parameter group that is compatible with a selected engine
family.

        :param parameter_group_name: The name given to the newly created
parameter group.
        :param db_engine: The database engine to use as a basis.
        :return: The newly created parameter group.
        """
        print(
            f"Checking for an existing DB instance parameter group named
{parameter_group_name}."
        )
        parameter_group = self.instance_wrapper.get_parameter_group(
            parameter_group_name
        )
        if parameter_group is None:
            print(f"Getting available database engine versions for {db_engine}.")
            engine_versions =
self.instance_wrapper.get_engine_versions(db_engine)
            families = list({ver["DBParameterGroupFamily"] for ver in
engine_versions})
            family_index = q.choose("Which family do you want to use? ",
families)
            print(f"Creating a parameter group.")
            self.instance_wrapper.create_parameter_group(
                parameter_group_name, families[family_index], "Example parameter
group."
            )
            parameter_group = self.instance_wrapper.get_parameter_group(
                parameter_group_name
            )
        print(f"Parameter group {parameter_group['DBParameterGroupName']}:")
        pp(parameter_group)
        print("-" * 88)
        return parameter_group

    def update_parameters(self, parameter_group_name):
        """
        Shows how to get the parameters contained in a custom parameter group and
        update some of the parameter values in the group.
```

```
        :param parameter_group_name: The name of the parameter group to query and
modify.
        """
        print("Let's set some parameter values in your parameter group.")
        auto_inc_parameters = self.instance_wrapper.get_parameters(
            parameter_group_name, name_prefix="auto_increment"
        )
        update_params = []
        for auto_inc in auto_inc_parameters:
            if auto_inc["IsModifiable"] and auto_inc["DataType"] == "integer":
                print(f"The {auto_inc['ParameterName']} parameter is described
as:")
                print(f"\t{auto_inc['Description']}")
                param_range = auto_inc["AllowedValues"].split("-")
                auto_inc["ParameterValue"] = str(
                    q.ask(
                        f"Enter a value between {param_range[0]} and
{param_range[1]}: ",
                        q.is_int,
                        q.in_range(int(param_range[0]), int(param_range[1])),
                    )
                )
                update_params.append(auto_inc)
        self.instance_wrapper.update_parameters(parameter_group_name,
update_params)
        print(
            "You can get a list of parameters you've set by specifying a source
of 'user'."
        )
        user_parameters = self.instance_wrapper.get_parameters(
            parameter_group_name, source="user"
        )
        pp(user_parameters)
        print("-" * 88)

    def create_instance(self, instance_name, db_name, db_engine,
parameter_group):
        """
        Shows how to create a DB instance that contains a database of a specified
        type and is configured to use a custom DB parameter group.

        :param instance_name: The name given to the newly created DB instance.
        :param db_name: The name given to the created database.
```

```
        :param db_engine: The engine of the created database.
        :param parameter_group: The parameter group that is associated with the
DB instance.
        :return: The newly created DB instance.
        """
        print("Checking for an existing DB instance.")
        db_inst = self.instance_wrapper.get_db_instance(instance_name)
        if db_inst is None:
            print("Let's create a DB instance.")
            admin_username = q.ask(
                "Enter an administrator user name for the database: ",
q.non_empty
            )
            admin_password = q.ask(
                "Enter a password for the administrator (at least 8 characters):
",
                q.non_empty,
            )
            engine_versions = self.instance_wrapper.get_engine_versions(
                db_engine, parameter_group["DBParameterGroupFamily"]
            )
            engine_choices = [ver["EngineVersion"] for ver in engine_versions]
            print("The available engines for your parameter group are:")
            engine_index = q.choose("Which engine do you want to use? ",
engine_choices)
            engine_selection = engine_versions[engine_index]
            print(
                "The available micro DB instance classes for your database engine
are:"
            )
            inst_opts = self.instance_wrapper.get_orderable_instances(
                engine_selection["Engine"], engine_selection["EngineVersion"]
            )
            inst_choices = list(
                {
                    opt["DBInstanceClass"]
                    for opt in inst_opts
                    if "micro" in opt["DBInstanceClass"]
                }
            )
            inst_index = q.choose(
                "Which micro DB instance class do you want to use? ",
inst_choices
            )
```

```
            group_name = parameter_group["DBParameterGroupName"]
            storage_type = "standard"
            allocated_storage = 5
            print(
                f"Creating a DB instance named {instance_name} and database
{db_name}.\n"
                f"The DB instance is configured to use your custom parameter
group {group_name},\n"
                f"selected engine {engine_selection['EngineVersion']},\n"
                f"selected DB instance class {inst_choices[inst_index]},"
                f"and {allocated_storage} GiB of {storage_type} storage.\n"
                f"This typically takes several minutes."
            )
            db_inst = self.instance_wrapper.create_db_instance(
                db_name,
                instance_name,
                group_name,
                engine_selection["Engine"],
                engine_selection["EngineVersion"],
                inst_choices[inst_index],
                storage_type,
                allocated_storage,
                admin_username,
                admin_password,
            )
            while db_inst.get("DBInstanceStatus") != "available":
                wait(10)
                db_inst = self.instance_wrapper.get_db_instance(instance_name)
        print("Instance data:")
        pp(db_inst)
        print("-" * 88)
        return db_inst

    @staticmethod
    def display_connection(db_inst):
        """
        Displays connection information about a DB instance and tips on how to
        connect to it.

        :param db_inst: The DB instance to display.
        """
        print(
            "You can now connect to your database using your favorite MySql
client.\n"
```

```
            "One way to connect is by using the 'mysql' shell on an Amazon EC2
 instance\n"
            "that is running in the same VPC as your DB instance. Pass the
 endpoint,\n"
            "port, and administrator user name to 'mysql' and enter your password
\n"
            "when prompted:\n"
        )
        print(
            f"\n\tmysql -h {db_inst['Endpoint']['Address']} -P
 {db_inst['Endpoint']['Port']} "
            f"-u {db_inst['MasterUsername']} -p\n"
        )
        print(
            "For more information, see the User Guide for Amazon RDS:\n"
            "\thttps://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
CHAP_GettingStarted.CreatingConnecting.MySQL.html#CHAP_GettingStarted.Connecting.MySQL"
        )
        print("-" * 88)

    def create_snapshot(self, instance_name):
        """
        Shows how to create a DB instance snapshot and wait until it's available.

        :param instance_name: The name of a DB instance to snapshot.
        """
        if q.ask(
            "Do you want to create a snapshot of your DB instance (y/n)? ",
 q.is_yesno
        ):
            snapshot_id = f"{instance_name}-{uuid.uuid4()}"
            print(
                f"Creating a snapshot named {snapshot_id}. This typically takes a
 few minutes."
            )
            snapshot = self.instance_wrapper.create_snapshot(snapshot_id,
 instance_name)
            while snapshot.get("Status") != "available":
                wait(10)
                snapshot = self.instance_wrapper.get_snapshot(snapshot_id)
            pp(snapshot)
            print("-" * 88)

    def cleanup(self, db_inst, parameter_group_name):
```

```
        """
        Shows how to clean up a DB instance and parameter group.
        Before the parameter group can be deleted, all associated DB instances
must first
        be deleted.

        :param db_inst: The DB instance to delete.
        :param parameter_group_name: The DB parameter group to delete.
        """
        if q.ask(
            "\nDo you want to delete the DB instance and parameter group (y/n)?
",
            q.is_yesno,
        ):
            print(f"Deleting DB instance {db_inst['DBInstanceIdentifier']}.")

self.instance_wrapper.delete_db_instance(db_inst["DBInstanceIdentifier"])
            print(
                "Waiting for the DB instance to delete. This typically takes
several minutes."
            )
            while db_inst is not None:
                wait(10)
                db_inst = self.instance_wrapper.get_db_instance(
                    db_inst["DBInstanceIdentifier"]
                )
            print(f"Deleting parameter group {parameter_group_name}.")
            self.instance_wrapper.delete_parameter_group(parameter_group_name)

    def run_scenario(self, db_engine, parameter_group_name, instance_name,
db_name):
        logging.basicConfig(level=logging.INFO, format="%(levelname)s:
%(message)s")

        print("-" * 88)
        print(
            "Welcome to the Amazon Relational Database Service (Amazon RDS)\n"
            "get started with DB instances demo."
        )
        print("-" * 88)

        parameter_group = self.create_parameter_group(parameter_group_name,
db_engine)
        self.update_parameters(parameter_group_name)
```

```
        db_inst = self.create_instance(
            instance_name, db_name, db_engine, parameter_group
        )
        self.display_connection(db_inst)
        self.create_snapshot(instance_name)
        self.cleanup(db_inst, parameter_group_name)

        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    try:
        scenario = RdsInstanceScenario(InstanceWrapper.from_client())
        scenario.run_scenario(
            "mysql",
            "doc-example-parameter-group",
            "doc-example-instance",
            "docexampledb",
        )
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Amazon RDS 작업을 관리하기 위해 시나리오에서 호출하는 함수를 정의합니다.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)
```

```
   def get_parameter_group(self, parameter_group_name):
       """
       Gets a DB parameter group.

       :param parameter_group_name: The name of the parameter group to retrieve.
       :return: The parameter group.
       """
       try:
           response = self.rds_client.describe_db_parameter_groups(
               DBParameterGroupName=parameter_group_name
           )
           parameter_group = response["DBParameterGroups"][0]
       except ClientError as err:
           if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
               logger.info("Parameter group %s does not exist.",
parameter_group_name)
           else:
               logger.error(
                   "Couldn't get parameter group %s. Here's why: %s: %s",
                   parameter_group_name,
                   err.response["Error"]["Code"],
                   err.response["Error"]["Message"],
               )
               raise
       else:
           return parameter_group


   def create_parameter_group(
       self, parameter_group_name, parameter_group_family, description
   ):
       """
       Creates a DB parameter group that is based on the specified parameter
group
       family.

       :param parameter_group_name: The name of the newly created parameter
group.
       :param parameter_group_family: The family that is used as the basis of
the new
                                      parameter group.
       :param description: A description given to the parameter group.
       :return: Data about the newly created parameter group.
```

```
        """
        try:
            response = self.rds_client.create_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            self.rds_client.delete_db_parameter_group(
                DBParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB parameter group.
```

```
        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
filtered
                            to contain only parameters that start with this
prefix.
        :param source: When specified, only parameters from this source are
retrieved.
                       For example, a source of 'user' retrieves only parameters
that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator = self.rds_client.get_paginator("describe_db_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters


    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
```

```
        try:
            response = self.rds_client.modify_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
Parameters=update_parameters
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def create_snapshot(self, snapshot_id, instance_id):
        """
        Creates a snapshot of a DB instance.

        :param snapshot_id: The ID to give the created snapshot.
        :param instance_id: The ID of the DB instance to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_snapshot(
                DBSnapshotIdentifier=snapshot_id,
DBInstanceIdentifier=instance_id
            )
            snapshot = response["DBSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def get_snapshot(self, snapshot_id):
```

```
        """
        Gets a DB instance snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_snapshots(
                DBSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned
list of
                                       engine versions to those that are
compatible with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
```

```
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by
the DB instance.
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts


    def get_db_instance(self, instance_id):
        """
```

```
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst


    def create_db_instance(
        self,
        db_name,
        instance_id,
        parameter_group_name,
        db_engine,
        db_engine_version,
        instance_class,
        storage_type,
        allocated_storage,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB instance.

        :param db_name: The name of the database that is created in the DB
instance.
        :param instance_id: The ID to give the newly created DB instance.
```

```
        :param parameter_group_name: A parameter group to associate with the DB
instance.
        :param db_engine: The database engine of a database to create in the DB
instance.
        :param db_engine_version: The engine version for the created database.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :param storage_type: The storage type of the DB instance.
        :param allocated_storage: The amount of storage allocated on the DB
instance, in GiBs.
        :param admin_name: The name of the admin user for the created database.
        :param admin_password: The admin password for the created database.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBName=db_name,
                DBInstanceIdentifier=instance_id,
                DBParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
                DBInstanceClass=instance_class,
                StorageType=storage_type,
                AllocatedStorage=allocated_storage,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.
```

```
        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하십시오.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하는 Amazon RDS의 서버리스 예제

다음 코드 예제에서는 Amazon RDS를 AWS SDK와 함께 사용하는 방법을 보여줍니다.

예시

- Lambda 함수를 사용하여 Amazon RDS 데이터베이스에 연결

Lambda 함수를 사용하여 Amazon RDS 데이터베이스에 연결

다음 코드 예시는 RDS 데이터베이스에 연결하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 간단한 데이터베이스 요청을 하고 결과를 반환합니다.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda 함수에서 Amazon RDS 데이터베이스에 연결

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
/*
Golang v2 code here.
*/

package main

import (
 "context"
 "database/sql"
 "encoding/json"
 "fmt"
```

```
 "os"

 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
 _ "github.com/go-sql-driver/mysql"
)

type MyEvent struct {
 Name string `json:"name"`
}

func HandleRequest(event *MyEvent) (map[string]interface{}, error) {

 var dbName string = os.Getenv("DatabaseName")
 var dbUser string = os.Getenv("DatabaseUser")
 var dbHost string = os.Getenv("DBHost") // Add hostname without https
 var dbPort int = os.Getenv("Port")      // Add port number
 var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
 var region string = os.Getenv("AWS_REGION")

 cfg, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  panic("configuration error: " + err.Error())
 }

 authenticationToken, err := auth.BuildAuthToken(
  context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
 if err != nil {
  panic("failed to create authentication token: " + err.Error())
 }

 dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
  dbUser, authenticationToken, dbEndpoint, dbName,
 )

 db, err := sql.Open("mysql", dsn)
 if err != nil {
  panic(err)
 }

 defer db.Close()

 var sum int
```

```
 err = db.QueryRow("SELECT ?+? AS sum", 3, 2).Scan(&sum)
 if err != nil {
  panic(err)
 }
 s := fmt.Sprint(sum)
 message := fmt.Sprintf("The selected sum is: %s", s)

 messageBytes, err := json.Marshal(message)
 if err != nil {
  return nil, err
 }

 messageString := string(messageBytes)
 return map[string]interface{}{
  "statusCode": 200,
  "headers":    map[string]string{"Content-Type": "application/json"},
  "body":       messageString,
 }, nil
}

func main() {
 lambda.Start(HandleRequest)
}
```

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda 함수에서 Amazon RDS 데이터베이스에 연결

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyRequestEvent;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyResponseEvent;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rdsdata.RdsDataClient;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.rdsdata.model.Field;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class RdsLambdaHandler implements
 RequestHandler<APIGatewayProxyRequestEvent, APIGatewayProxyResponseEvent> {

    @Override
    public APIGatewayProxyResponseEvent handleRequest(APIGatewayProxyRequestEvent
 event, Context context) {
        APIGatewayProxyResponseEvent response = new
 APIGatewayProxyResponseEvent();

        try {
            // Obtain auth token
            String token = createAuthToken();

            // Define connection configuration
            String connectionString = String.format("jdbc:mysql://%s:%s/%s?
useSSL=true&requireSSL=true",
                    System.getenv("ProxyHostName"),
                    System.getenv("Port"),
                    System.getenv("DBName"));

            // Establish a connection to the database
            try (Connection connection =
 DriverManager.getConnection(connectionString, System.getenv("DBUserName"),
 token);
                 PreparedStatement statement =
 connection.prepareStatement("SELECT ? + ? AS sum")) {

                statement.setInt(1, 3);
                statement.setInt(2, 2);

                try (ResultSet resultSet = statement.executeQuery()) {
                    if (resultSet.next()) {
                        int sum = resultSet.getInt("sum");
```

```
                        response.setStatusCode(200);
                        response.setBody("The selected sum is: " + sum);
                    }
                }
            }

        } catch (Exception e) {
            response.setStatusCode(500);
            response.setBody("Error: " + e.getMessage());
        }

        return response;
    }

    private String createAuthToken() {
        // Create RDS Data Service client
        RdsDataClient rdsDataClient = RdsDataClient.builder()
                .region(Region.of(System.getenv("AWS_REGION")))
                .credentialsProvider(DefaultCredentialsProvider.create())
                .build();

        // Define authentication request
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
                .resourceArn(System.getenv("ProxyHostName"))
                .secretArn(System.getenv("DBUserName"))
                .database(System.getenv("DBName"))
                .sql("SELECT 'RDS IAM Authentication'")
                .build();

        // Execute request and obtain authentication token
        ExecuteStatementResponse response =
 rdsDataClient.executeStatement(request);
        Field tokenField = response.records().get(0).get(0);

        return tokenField.stringValue();
    }
}
```

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

JavaScript를 사용하여 Lambda 함수에서 Amazon RDS 데이터베이스에 연결

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
/*
Node.js code here.
*/
// ES6+ example
import { Signer } from "@aws-sdk/rds-signer";
import mysql from 'mysql2/promise';

async function createAuthToken() {
  // Define connection authentication parameters
  const dbinfo = {

    hostname: process.env.ProxyHostName,
    port: process.env.Port,
    username: process.env.DBUserName,
    region: process.env.AWS_REGION,

  }

  // Create RDS Signer object
  const signer = new Signer(dbinfo);

  // Request authorization token from RDS, specifying the username
  const token = await signer.getAuthToken();
  return token;
}

async function dbOps() {

  // Obtain auth token
```

```
  const token = await createAuthToken();
  // Define connection configuration
  let connectionConfig = {
    host: process.env.ProxyHostName,
    user: process.env.DBUserName,
    password: token,
    database: process.env.DBName,
    ssl: 'Amazon RDS'
  }
  // Create the connection to the DB
  const conn = await mysql.createConnection(connectionConfig);
  // Obtain the result of the query
  const [res,] = await conn.execute('select ?+? as sum', [3, 2]);
  return res;

}

export const handler = async (event) => {
  // Execute database flow
  const result = await dbOps();
  // Return result
  return {
    statusCode: 200,
    body: JSON.stringify("The selected sum is: " + result[0].sum)
  }
};
```

TypeScript를 사용하여 Lambda 함수에서 Amazon RDS 데이터베이스에 연결

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import { Signer } from "@aws-sdk/rds-signer";
import mysql from 'mysql2/promise';

// RDS settings
// Using '!' (non-null assertion operator) to tell the TypeScript compiler that
 the DB settings are not null or undefined,
const proxy_host_name = process.env.PROXY_HOST_NAME!
const port = parseInt(process.env.PORT!)
const db_name = process.env.DB_NAME!
const db_user_name = process.env.DB_USER_NAME!
```

```
const aws_region = process.env.AWS_REGION!


async function createAuthToken(): Promise<string> {

    // Create RDS Signer object
    const signer = new Signer({
        hostname: proxy_host_name,
        port: port,
        region: aws_region,
        username: db_user_name
    });

    // Request authorization token from RDS, specifying the username
    const token = await signer.getAuthToken();
    return token;
}

async function dbOps(): Promise<mysql.QueryResult | undefined> {
    try {
        // Obtain auth token
        const token = await createAuthToken();
        const conn = await mysql.createConnection({
            host: proxy_host_name,
            user: db_user_name,
            password: token,
            database: db_name,
            ssl: 'Amazon RDS' // Ensure you have the CA bundle for SSL connection
        });
        const [rows, fields] = await conn.execute('SELECT ? + ? AS sum', [3, 2]);
        console.log('result:', rows);
        return rows;
    }
    catch (err) {
        console.log(err);
    }
}

export const lambdaHandler = async (event: any): Promise<{ statusCode: number;
 body: string }> => {
    // Execute database flow
    const result = await dbOps();

    // Return error is result is undefined
```

```
    if (result == undefined)
        return {
            statusCode: 500,
            body: JSON.stringify(`Error with connection to DB host`)
        }

    // Return result
    return {
        statusCode: 200,
        body: JSON.stringify(`The selected sum is: ${result[0].sum}`)
    };
};
```

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda 함수에서 Amazon RDS 데이터베이스에 연결

```
<?php
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
use Aws\Rds\AuthTokenGenerator;
use Aws\Credentials\CredentialProvider;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
```

```
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }


    private function getAuthToken(): string {
        // Define connection authentication parameters
        $dbConnection = [
            'hostname' => getenv('DB_HOSTNAME'),
            'port' => getenv('DB_PORT'),
            'username' => getenv('DB_USERNAME'),
            'region' => getenv('AWS_REGION'),
        ];

        // Create RDS AuthTokenGenerator object
        $generator = new
AuthTokenGenerator(CredentialProvider::defaultProvider());

        // Request authorization token from RDS, specifying the username
        return $generator->createToken(
            $dbConnection['hostname'] . ':' . $dbConnection['port'],
            $dbConnection['region'],
            $dbConnection['username']
        );
    }

    private function getQueryResults() {
        // Obtain auth token
        $token = $this->getAuthToken();

        // Define connection configuration
        $connectionConfig = [
            'host' => getenv('DB_HOSTNAME'),
            'user' => getenv('DB_USERNAME'),
            'password' => $token,
            'database' => getenv('DB_NAME'),
        ];

        // Create the connection to the DB
        $conn = new PDO(

"mysql:host={$connectionConfig['host']};dbname={$connectionConfig['database']}",
            $connectionConfig['user'],
```

```
            $connectionConfig['password'],
            [
                PDO::MYSQL_ATTR_SSL_CA => '/path/to/rds-ca-2019-root.pem',
                PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => true,
            ]
        );

        // Obtain the result of the query
        $stmt = $conn->prepare('SELECT ?+? AS sum');
        $stmt->execute([3, 2]);

        return $stmt->fetch(PDO::FETCH_ASSOC);
    }

    /**
     * @param mixed $event
     * @param Context $context
     * @return array
     */
    public function handle(mixed $event, Context $context): array
    {
        $this->logger->info("Processing query");

        // Execute database flow
        $result = $this->getQueryResults();

        return [
            'sum' => $result['sum']
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Python을 사용하여 Lambda 함수에서 Amazon RDS 데이터베이스에 연결

```
import json
import os
import boto3
import pymysql

# RDS settings
proxy_host_name = os.environ['PROXY_HOST_NAME']
port = int(os.environ['PORT'])
db_name = os.environ['DB_NAME']
db_user_name = os.environ['DB_USER_NAME']
aws_region = os.environ['AWS_REGION']


# Fetch RDS Auth Token
def get_auth_token():
    client = boto3.client('rds')
    token = client.generate_db_auth_token(
        DBHostname=proxy_host_name,
        Port=port
        DBUsername=db_user_name
        Region=aws_region
    )
    return token

def lambda_handler(event, context):
    token = get_auth_token()
    try:
        connection = pymysql.connect(
            host=proxy_host_name,
            user=db_user_name,
            password=token,
```

```
            db=db_name,
            port=port,
            ssl={'ca': 'Amazon RDS'}  # Ensure you have the CA bundle for SSL
 connection
        )

        with connection.cursor() as cursor:
            cursor.execute('SELECT %s + %s AS sum', (3, 2))
            result = cursor.fetchone()

        return result

    except Exception as e:
        return (f"Error: {str(e)}")  # Return an error message if an exception
 occurs
```

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda 함수에서 Amazon RDS 데이터베이스에 연결

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
# Ruby code here.

require 'aws-sdk-rds'
require 'json'
require 'mysql2'

def lambda_handler(event:, context:)
  endpoint = ENV['DBEndpoint'] # Add the endpoint without https"
  port = ENV['Port']           # 3306
  user = ENV['DBUser']
  region = ENV['DBRegion']     # 'us-east-1'
```

```
  db_name = ENV['DBName']

  credentials = Aws::Credentials.new(
    ENV['AWS_ACCESS_KEY_ID'],
    ENV['AWS_SECRET_ACCESS_KEY'],
    ENV['AWS_SESSION_TOKEN']
  )
  rds_client = Aws::RDS::AuthTokenGenerator.new(
    region: region,
    credentials: credentials
  )

  token = rds_client.auth_token(
    endpoint: endpoint+ ':' + port,
    user_name: user,
    region: region
  )

  begin
    conn = Mysql2::Client.new(
      host: endpoint,
      username: user,
      password: token,
      port: port,
      database: db_name,
      sslca: '/var/task/global-bundle.pem',
      sslverify: true,
      enable_cleartext_plugin: true
    )
    a = 3
    b = 2
    result = conn.query("SELECT #{a} + #{b} AS sum").first['sum']
    puts result
    conn.close
    {
      statusCode: 200,
      body: result.to_json
    }
  rescue => e
    puts "Database connection failed due to #{e}"
  end
end
```

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Rust를 사용하여 Lambda 함수에서 Amazon RDS 데이터베이스에 연결

```
use aws_config::BehaviorVersion;
use aws_credential_types::provider::ProvideCredentials;
use aws_sigv4::{
    http_request::{sign, SignableBody, SignableRequest, SigningSettings},
    sign::v4,
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use serde_json::{json, Value};
use sqlx::postgres::PgConnectOptions;
use std::env;
use std::time::{Duration, SystemTime};

const RDS_CERTS: &[u8] = include_bytes!("global-bundle.pem");

async fn generate_rds_iam_token(
    db_hostname: &str,
    port: u16,
    db_username: &str,
) -> Result<String, Error> {
    let config = aws_config::load_defaults(BehaviorVersion::v2024_03_28()).await;

    let credentials = config
        .credentials_provider()
        .expect("no credentials provider found")
        .provide_credentials()
        .await
        .expect("unable to load credentials");
    let identity = credentials.into();
    let region = config.region().unwrap().to_string();

    let mut signing_settings = SigningSettings::default();
```

```
    signing_settings.expires_in = Some(Duration::from_secs(900));
    signing_settings.signature_location =
 aws_sigv4::http_request::SignatureLocation::QueryParams;

    let signing_params = v4::SigningParams::builder()
        .identity(&identity)
        .region(&region)
        .name("rds-db")
        .time(SystemTime::now())
        .settings(signing_settings)
        .build()?;

    let url = format!(
        "https://{db_hostname}:{port}/?Action=connect&DBUser={db_user}",
        db_hostname = db_hostname,
        port = port,
        db_user = db_username
    );

    let signable_request =
        SignableRequest::new("GET", &url, std::iter::empty(),
 SignableBody::Bytes(&[]))
            .expect("signable request");

    let (signing_instructions, _signature) =
        sign(signable_request, &signing_params.into())?.into_parts();

    let mut url = url::Url::parse(&url).unwrap();
    for (name, value) in signing_instructions.params() {
        url.query_pairs_mut().append_pair(name, &value);
    }

    let response = url.to_string().split_off("https://".len());

    Ok(response)
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    run(service_fn(handler)).await
}

async fn handler(_event: LambdaEvent<Value>) -> Result<Value, Error> {
    let db_host = env::var("DB_HOSTNAME").expect("DB_HOSTNAME must be set");
```

```
    let db_port = env::var("DB_PORT")
        .expect("DB_PORT must be set")
        .parse::<u16>()
        .expect("PORT must be a valid number");
    let db_name = env::var("DB_NAME").expect("DB_NAME must be set");
    let db_user_name = env::var("DB_USERNAME").expect("DB_USERNAME must be set");

    let token = generate_rds_iam_token(&db_host, db_port, &db_user_name).await?;

    let opts = PgConnectOptions::new()
        .host(&db_host)
        .port(db_port)
        .username(&db_user_name)
        .password(&token)
        .database(&db_name)
        .ssl_root_cert_from_pem(RDS_CERTS.to_vec())
        .ssl_mode(sqlx::postgres::PgSslMode::Require);

    let pool = sqlx::postgres::PgPoolOptions::new()
        .connect_with(opts)
        .await?;

    let result: i32 = sqlx::query_scalar("SELECT $1 + $2")
        .bind(3)
        .bind(2)
        .fetch_one(&pool)
        .await?;

    println!("Result: {:?}", result);

    Ok(json!({
        "statusCode": 200,
        "content-type": "text/plain",
        "body": format!("The selected sum is: {result}")
    }))
}
```

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용한 Amazon RDS용 교차 서비스 예제

다음 샘플 애플리케이션에서는 AWS SDK를 사용하여 Amazon RDS를 다른 AWS 서비스와 결합합니다. 각 예시에는 애플리케이션을 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 GitHub 링크가 포함되어 있습니다.

예시

- Aurora 서버리스 작업 항목 트래커 만들기

Aurora 서버리스 작업 항목 트래커 만들기

다음 코드 예제는 Amazon Aurora Serverless 데이터베이스에서 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 보내는 웹 애플리케이션 생성 방법을 보여줍니다.

.NET

SDK for .NET

AWS SDK for .NET을 사용하여 Amazon Aurora 데이터베이스에서 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 통해 보고서를 이메일로 보내는 웹 애플리케이션 생성 방법을 보여줍니다. 이 예제에서는 RESTful .NET 백엔드와의 상호 작용을 위해 React.js로 빌드된 프런트엔드를 사용합니다.

- React 웹 애플리케이션을 AWS 서비스와 통합합니다.
- Aurora 테이블의 항목을 나열, 추가, 업데이트 및 삭제합니다.
- Amazon SES를 사용하여 필터링된 작업 항목에 대한 이메일 보고서를 보냅니다.
- 포함된 AWS CloudFormation 스크립트를 통해 예제 리소스를 배포하고 관리합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하십시오.

이 예시에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스
- Amazon SES

C++

SDK for C++

Amazon Aurora Serverless 데이터베이스에 저장된 작업 항목을 추적하고 보고하는 웹 애플리케이션 생성 방법을 보여줍니다.

Amazon Aurora Serverless 데이터를 쿼리하고 React 애플리케이션에서 사용하도록 C++ REST API를 설정하는 방법에 대한 지침과 전체 소스 코드는 GitHub에서 전체 예제를 참조하십시오.

이 예시에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스
- Amazon SES

Java

SDK for Java 2.x

Amazon RDS 데이터베이스에 저장된 작업 항목을 추적하고 보고하는 웹 애플리케이션 생성 방법을 보여줍니다.

Amazon Aurora Serverless 데이터를 쿼리하고 React 애플리케이션에서 사용하도록 Spring REST API를 설정하는 방법에 대한 지침과 전체 소스 코드는 GitHub에서 전체 예제를 참조하십시오.

JDBC API를 사용한 예제를 설정하고 실행하는 방법에 대한 전체 소스 코드와 지침은 GitHub에서 전체 예제를 참조하십시오.

이 예시에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스
- Amazon SES

JavaScript

SDK for JavaScript (v3)

AWS SDK for JavaScript(v3)을 사용하여 Amazon Aurora 데이터베이스에서 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 통해 보고서를 이메일로 보내는 웹 애플리케이션 생성 방법을 보여줍니다. 이 예제에서는 Express Node.js 백엔드와의 상호 작용을 위해 React.js로 빌드된 프런트엔드를 사용합니다.

- React.js 웹 애플리케이션을 AWS 서비스와 통합합니다.
- Aurora 테이블의 항목을 나열, 추가 및 업데이트합니다.
- Amazon SES를 사용하여 필터링된 작업 항목에 대한 이메일 보고서를 보냅니다.
- 포함된 AWS CloudFormation 스크립트를 통해 예제 리소스를 배포하고 관리합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하십시오.

이 예시에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스
- Amazon SES

Kotlin

SDK for Kotlin

Amazon RDS 데이터베이스에 저장된 작업 항목을 추적하고 보고하는 웹 애플리케이션 생성 방법을 보여줍니다.

Amazon Aurora Serverless 데이터를 쿼리하고 React 애플리케이션에서 사용하도록 Spring REST API를 설정하는 방법에 대한 지침과 전체 소스 코드는 GitHub에서 전체 예제를 참조하십시오.

이 예시에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스

- Amazon SES

PHP

SDK for PHP

AWS SDK for PHP을 사용하여 Amazon RDS 데이터베이스에서 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 통해 보고서를 이메일로 보내는 웹 애플리케이션 생성 방법을 보여줍니다. 이 예제에서는 RESTful PHP 백엔드와의 상호 작용을 위해 React.js로 빌드된 프런트엔드를 사용합니다.

- React.js 웹 애플리케이션을 AWS 서비스와 통합합니다.
- Amazon RDS 테이블의 항목을 나열, 추가, 업데이트 및 삭제합니다.
- Amazon SES를 사용하여 필터링된 작업 항목에 대한 이메일 보고서를 보냅니다.
- 포함된 AWS CloudFormation 스크립트를 통해 예제 리소스를 배포하고 관리합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하십시오.

이 예시에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스
- Amazon SES

Python

SDK for Python (Boto3)

AWS SDK for Python (Boto3)을 사용하여 Amazon Aurora Serverless 데이터베이스에서 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 통해 보고서를 이메일로 보내는 REST 서비스 생성 방법을 보여 줍니다. 이 예제는 Flask 웹 프레임워크를 사용하여 HTTP 라우팅을 처리하고 React 웹 페이지와 통합하여 완전한 기능을 갖춘 웹 애플리케이션을 제공합니다.

- AWS 서비스와 통합되는 Flask REST 서비스를 구축합니다.
- Aurora Serverless 데이터베이스에 저장된 작업 항목을 읽고, 쓰고, 업데이트합니다.
- 데이터베이스 보안 인증 정보가 포함된 AWS Secrets Manager 암호를 생성하고 이를 사용하여 데이터베이스에 대한 호출을 인증합니다.

- Amazon SES를 사용하여 작업 항목에 대한 이메일 보고서를 보냅니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하십시오.

이 예시에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스
- Amazon SES

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon RDS의 보안

AWS에서는 클라우드 보안을 가장 중요하게 생각합니다. AWS 고객은 보안에 가장 보안에 민감한 조직의 요구 사항에 부합하도록 빌드된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 AWS과 사용자의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드 내 보안 및 클라우드의 보안으로 설명합니다.

- 클라우드의 보안 - AWS는 AWS클라우드에서 AWS서비스를 실행하는 인프라를 보호합니다. AWS는 또한 안전하게 사용할 수 있는 서비스를 제공합니다. 서드 파티 감사원은 정기적으로 AWS규정 준수 프로그램의 일환으로 보안 효과를 테스트하고 검증합니다. Amazon RDS 에 적용되는 규정 준수 프로그램에 대해 알아보려면 AWS services in scope by compliance program을 참조하세요.
- 클라우드 내 보안 – 귀하의 책임은 귀하가 사용하는 AWS서비스에 의해 결정됩니다. 또한 데이터의 민감도, 조직의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 Amazon RDS 사용 시 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 항목에서는 보안 및 규정 준수 목표를 충족하도록 Amazon RDS 를 구성하는 방법을 보여 줍니다. 또한 Amazon RDS 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 알아봅니다.

DB 인스턴스 에서 Amazon RDS 리소스 및 데이터베이스에 대한 액세스를 관리할 수 있습니다. 액세스 관리에 사용하는 방법은 사용자가 Amazon RDS 를 사용하여 수행해야 하는 작업 유형에 따라 다릅니다.

- 네트워크 액세스 제어를 최대한 강화할 목적으로 Amazon VPC 서비스에 따라 DB 인스턴스 를가상 프라이빗 클라우드(VPC)에서 실행합니다. DB 인스턴스 를 VPC에서 만드는 방법에 대한 자세한 내용은 Amazon VPC 및 Amazon RDS 섹션을 참조하세요.
- Amazon RDS 리소스를 관리할 수 있는 사용자를 결정하는 권한을 할당하려면 AWS Identity and Access Management(IAM) 정책을 사용합니다. 예를 들면, IAM을 사용하여 DB 인스턴스 , 태그 리소스를 생성, 설명, 수정, 삭제하거나 보안 그룹을 수정할 수 있는 사용자를 결정할 수 있습니다.
- 보안 그룹을 사용하여 어떤 IP 주소 또는 Amazon EC2 인스턴스가 DB 인스턴스 에 있는 데이터베이스에 연결할 수 있는지 제어합니다. DB 인스턴스 를 처음 생성하면, 방화벽이 연결된 보안 그룹에서 지정한 규칙을 통한 액세스 이외의 데이터베이스 액세스를 차단합니다.
- Db2, MySQL, MariaDB, PostgreSQL, Oracle 또는 Microsoft SQL Server 데이터베이스 엔진을 실행하는 DB 인스턴스에 보안 소켓 계층(SSL) 또는 전송 계층 보안(TLS) 연결을 사용합니다. DB 인스턴

스 에서 SSL/TLS를 사용하는 것에 대한 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 섹션을 참조하세요.

- Amazon RDS 암호화를 사용하여 DB 인스턴스 및 저장 중인 스냅샷을 보호합니다. Amazon RDS 암호화는 DB 인스턴스 를 호스팅하는 서버의 데이터를 업계 표준 AES-256 암호화 알고리즘을 사용하여 암호화합니다. 자세한 정보는 Amazon RDS 리소스 암호화을 참조하세요.
- Oracle DB 인스턴스와 함께 네트워크 암호화 및 Transparent Data Encryption을 사용합니다. 자세한 내용은 Oracle 기본 네트워크 암호화 및 Oracle Transparent Data Encryption 단원을 참조하십시오.
- DB 엔진의 보안 기능을 사용하여 DB 인스턴스 에 있는 데이터베이스에 누가 로그인할 수 있는지 제어합니다. 이러한 보안 기능은 데이터베이스가 마치 로컬 네트워크에 있는 것처럼 실행됩니다.

> **Note**
>
> 사용 사례에 따라 보안을 구성해야 합니다. Amazon RDS가 관리하는 프로세스에 대한 보안 액세스를 구성할 필요가 없습니다. 이러한 프로세스로는 백업 생성, 기본 DB 인스턴스와 읽기 전용 복제본 간 데이터 복제 등이 있습니다.

Amazon RDS 리소스에 대한 액세스 관리와 DB 인스턴스 의 데이터베이스에 대한 액세스 관리에 대해서는 아래 주제를 참조하세요.

주제

- Amazon RDS을 사용한 데이터베이스 인증
- Amazon RDS 및 AWS Secrets Manager를 통한 암호 관리
- Amazon RDS의 데이터 보호
- Amazon RDS의 자격 증명 및 액세스 관리
- Amazon RDS의 로깅 및 모니터링
- Amazon RDS의 규정 준수 확인
- Amazon RDS의 복원성
- Amazon RDS의 인프라 보안
- Amazon RDS API 및 인터페이스 VPC 엔드포인트(AWS PrivateLink)
- Amazon RDS의 보안 모범 사례
- 보안 그룹을 통한 액세스 제어
- 마스터 사용자 계정 권한

- Amazon RDS에 서비스 연결 역할 사용
- Amazon VPC 및 Amazon RDS

Amazon RDS을 사용한 데이터베이스 인증

Amazon RDS 는 데이터베이스 사용자를 인증하는 여러 가지 방법을 지원합니다.

암호, Kerberos 및 IAM 데이터베이스 인증은 데이터베이스에 대해 서로 다른 인증 방법을 사용합니다. 따라서 특정 사용자는 하나의 인증 방법만 사용하여 데이터베이스에 로그인할 수 있습니다.

PostgreSQL의 경우, 특정 데이터베이스 사용자에 대해 다음 역할 설정 중 하나만 사용하세요.

- IAM 데이터베이스 인증을 사용하려면 `rds_iam` 역할을 사용자에게 할당합니다.
- Kerberos 인증을 사용하려면 `rds_ad` 역할을 사용자에게 할당합니다.
- 암호 인증을 사용하려면 `rds_iam` 또는 `rds_ad` 역할을 사용자에게 할당하지 않습니다.

`rds_iam` 및 `rds_ad` 역할을 직접적으로 또는 중첩된 액세스 권한 부여를 통해 간접적으로 PostgreSQL 데이터베이스 사용자에게 둘 다 할당하지 않도록 합니다. `rds_iam` 역할이 마스터 사용자에게 추가되면, IAM 인증이 암호 인증보다 우선하므로 마스터 사용자가 IAM 사용자로 로그인해야 합니다.

> **Important**
>
> 애플리케이션에서 직접 마스터 사용자를 사용하지 않는 것이 좋습니다. 대신에 애플리케이션에 필요한 최소 권한으로 생성한 데이터베이스 사용자를 사용하는 모범 사례를 준수하십시오.

주제

- 암호 인증
- IAM 데이터베이스 인증
- Kerberos 인증

암호 인증

암호 인증을 통해 데이터베이스는 모든 사용자 계정 관리를 수행합니다. `CREATE USER`와 같은 DB 엔진에서 암호를 지정하는 데 필요한 적절한 절이 있는 SQL 문을 사용하여 사용자를 생성합니다. 예를

들어 MySQL에서 명령문은 `CREATE USER ## IDENTIFIED BY ##`이지만 PostgreSQL에서 명령문은 `CREATE USER ## WITH PASSWORD ##`입니다.

암호 인증을 통해 데이터베이스는 사용자 계정을 제어하고 인증합니다. DB 엔진에 강력한 암호 관리 기능이 있는 경우 보안을 강화할 수 있습니다. 소규모 사용자 커뮤니티가 있는 경우 암호 인증을 사용하여 데이터베이스 인증을 쉽게 관리할 수 있습니다. 이 경우 일반 텍스트 암호가 생성되므로 AWS Secrets Manager과 통합하면 보안이 강화될 수 있습니다.

Secrets Manager를 Amazon RDS 와 함께 사용하는 것에 대한 자세한 내용은 AWS Secrets Manager 사용 설명서에서 Creating a basic secret 및 Rotating secrets for supported Amazon RDS databases를 참조하세요. 사용자 지정 애플리케이션에서 프로그래밍 방식으로 비밀을 검색하는 방법에 대한 자세한 내용은 AWS Secrets Manager 사용 설명서의 비밀 값 검색을 참조하세요.

IAM 데이터베이스 인증

AWS Identity and Access Management(IAM) 데이터베이스 인증을 사용하여 DB 인스턴스 에 인증할 수 있습니다. 이러한 인증 방식은 DB 인스턴스 에 연결할 때 암호를 사용할 필요가 없습니다. 대신에 인증 토큰을 사용합니다.

특정 DB 엔진의 가용성에 대한 정보를 포함하여 IAM 데이터베이스 인증에 대한 자세한 내용은 MariaDB, MySQL 및 PostgreSQL IAM 데이터베이스 인증 섹션을 참조하세요.

Kerberos 인증

Amazon RDS에서는 Kerberos 및 Microsoft Active Directory를 사용한 데이터베이스 사용자의 외부 인증을 지원합니다. Kerberos는 티켓과 대칭 키 암호화를 사용하여 네트워크를 통해 암호를 전송할 필요가 없는 네트워크 인증 프로토콜입니다. Kerberos는 Active Directory에 내장되어 있으며 데이터베이스와 같은 네트워크 리소스에 대해 사용자를 인증하도록 설계되었습니다.

Kerberos 및 Active Directory에 대한 Amazon RDS의 지원은 데이터베이스 사용자에게 Single Sign-On 및 중앙 집중식 인증의 이점을 제공합니다. 사용자 자격 증명을 Active Directory에 보관할 수 있습니다. Active Directory는 여러 DB 인스턴스 에 대한 자격 증명을 보관하고 관리할 수 있는 중앙 집중식 공간을 제공합니다.

자체 관리형 Active Directory의 자격 증명을 사용하려면 DB 인스턴스 가 조인된 Microsoft Active Directory의 AWS Directory Service에 대한 신뢰 관계를 설정해야 합니다.

RDS for PostgreSQL 및 RDS for MySQL 은 포리스트 전체 인증 또는 선택적 인증을 통해 단방향 및 양방향 포리스트 신뢰 관계를 지원합니다.

일부 시나리오에서는 외부 신뢰 관계를 통해 Kerberos 인증을 구성할 수 있습니다. 이를 위해서는 자체 관리형 Active Directory에 추가 설정이 필요합니다. 여기에는 Kerberos 포리스트 검색 순서가 포함되지만, 이에 국한되지는 않습니다.

Microsoft SQL Server 및 PostgreSQL DB 인스턴스는 단방향 및 양방향 포리스트 신뢰 관계를 지원합니다. Oracle DB 인스턴스는 단방향 및 양방향 외부 및 포리스트 신뢰 관계를 지원합니다. 자세한 내용은 AWS Directory Service 관리 안내서의 신뢰 관계를 생성해야 하는 경우를 참조하십시오.

특정 DB 엔진의 Kerberos 인증에 대한 자세한 내용은 다음을 참조하십시오.

- RDS for SQL Server를 사용하여 AWS 관리형 Active Directory 작업
- Amazon RDS for MySQL에 Kerberos 인증 사용
- Amazon RDS for Oracle에 대한 Kerberos 인증 구성
- Amazon RDS for PostgreSQL과 함께 Kerberos 인증 사용
- Amazon RDS for Db2에 대한 Kerberos 인증 사용 .

> **Note**
>
> 현재 MariaDB DB 인스턴스에 대해서는 Kerberos 인증이 지원되지 않습니다.

Amazon RDS 및 AWS Secrets Manager를 통한 암호 관리

Amazon RDS는 Secrets Manager와 통합되어 DB 인스턴스와 다중 AZ DB 클러스터의 마스터 사용자 암호를 관리합니다.

주제

- Secrets Manager와 Amazon RDS 통합에 대한 제한 사항
- AWS Secrets Manager를 통한 마스터 사용자 암호 관리 개요
- Secrets Manager를 통한 마스터 사용자 암호 관리의 이점
- Secrets Manager 통합에 필요한 권한
- AWS Secrets Manager 마스터 사용자 암호의 RDS 관리 적용
- Secrets Manager를 통해 DB 인스턴스의 마스터 사용자 암호 관리
- Secrets Manager를 사용하여 RDS for Oracle 테넌트 데이터베이스의 마스터 사용자 암호 관리
- Secrets Manager를 사용하여 다중 AZ DB 클러스터의 마스터 사용자 암호 관리
- DB 인스턴스의 마스터 사용자 암호 비밀 교체
- 다중 AZ DB 클러스터의 마스터 사용자 암호 비밀 교체
- DB 인스턴스의 비밀에 대한 세부 정보 보기
- 다중 AZ DB 클러스터의 비밀에 대한 세부 정보 보기
- 테넌트 데이터베이스의 보안 암호에 대한 세부 정보 보기
- 리전 및 버전 사용 가능 여부

Secrets Manager와 Amazon RDS 통합에 대한 제한 사항

다음 기능에서는 Secrets Manager를 사용한 마스터 사용자 암호 관리가 지원되지 않습니다.

- 소스 DB 또는 DB 클러스터가 Secrets Manager로 자격 증명을 관리할 때 읽기 전용 복제본 생성. 이는 RDS for SQL Server를 제외한 모든 DB 엔진에 적용됩니다.
- Amazon RDS 블루/그린 배포
- Amazon RDS Custom
- Oracle Data Guard 전환

AWS Secrets Manager를 통한 마스터 사용자 암호 관리 개요

AWS Secrets Manager를 사용하면 데이터베이스 암호를 포함한 하드 코딩된 보안 인증 정보를 Secrets Manager에서 프로그래밍 방식으로 비밀을 검색하게 하는 API 호출로 바꿀 수 있습니다. Secrets Manager 사용에 대한 자세한 내용은 AWS Secrets Manager 사용 설명서를 참조하세요.

Secrets Manager에 데이터베이스 암호를 저장하면 AWS 계정에서 요금을 부과합니다. 요금에 대한 자세한 내용은 AWS Secrets Manager 요금을 참조하십시오.

다음 작업 중 하나를 수행할 때 RDS가 Secrets Manager에서 Amazon RDS DB 인스턴스 또는 다중 AZ DB 클러스터의 마스터 사용자 암호를 관리하도록 지정할 수 있습니다.

- DB 인스턴스 생성
- 다중 AZ DB 클러스터 만들기
- RDS for Oracle CDB에서 테넌트 데이터베이스 만들기
- DB 인스턴스 수정
- 다중 AZ DB 클러스터를 수정하는 방법
- 테넌트 데이터베이스 수정(RDS for Oracle만 해당)
- Amazon S3에서 DB 인스턴스 복원
- 스냅샷에서 또는 특정 시점으로 DB 인스턴스 복원(RDS for Oracle만 해당)

RDS가 Secrets Manager에서 마스터 사용자 암호를 관리하도록 지정하면 RDS 가 암호를 생성하여 Secrets Manager에 저장합니다. 암호와 직접 상호 작용하여 마스터 사용자의 보안 인증 정보를 검색할 수 있습니다. 고객 관리 키를 지정하여 암호를 암호화하거나 Secrets Manager에서 제공하는 KMS 키를 사용할 수도 있습니다.

RDS는 비밀 설정을 관리하고 기본적으로 7일마다 비밀을 교체합니다. 교체 일정 같은 일부 설정을 수정할 수 있습니다. Secrets Manager에서 암호를 관리하는 DB 인스턴스를 삭제하면 암호 및 관련 메타데이터도 삭제됩니다.

비밀의 보안 인증 정보를 사용하여 DB 인스턴스 또는 다중 AZ DB 클러스터에 연결하려면 Secrets Manager에서 비밀을 검색하면 됩니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 AWS Secrets Manager에서 비밀 검색 및 AWS Secrets Manager 비밀에 있는 보안 인증 정보를 사용하여 SQL 데이터베이스에 연결을 참조하세요.

Secrets Manager를 통한 마스터 사용자 암호 관리의 이점

Secrets Manager를 사용하여 RDS 마스터 사용자 암호를 관리하면 다음과 같은 이점이 있습니다.

- RDS가 데이터베이스 보안 인증 정보를 자동으로 생성합니다.
- RDS가 데이터베이스 보안 인증 정보를 AWS Secrets Manager에 자동으로 저장하고 관리합니다.
- 애플리케이션을 변경할 필요 없이 RDS가 정기적으로 데이터베이스 보안 인증 정보를 교체합니다.
- Secrets Manager가 사용자 액세스 및 일반 텍스트 보기로부터 데이터베이스 보안 인증 정보를 보호합니다.
- Secrets Manager를 사용하면 데이터베이스 연결을 위한 비밀의 데이터베이스 보안 인증 정보를 검색할 수 있습니다.
- Secrets Manager에서 IAM을 사용하여 비밀의 데이터베이스 보안 인증 정보에 대한 액세스를 세밀하게 제어할 수 있습니다.
- 필요에 따라 다른 KMS 키를 사용하여 데이터베이스 암호화와 보안 인증 정보 암호화를 분리할 수 있습니다.
- 데이터베이스 보안 인증 정보를 수동으로 관리하고 교체하지 않아도 됩니다.
- AWS CloudTrail 및 Amazon CloudWatch를 사용하여 데이터베이스 보안 인증 정보를 쉽게 모니터링할 수 있습니다.

Secrets Manager의 이점에 대한 자세한 내용은 AWS Secrets Manager 사용 설명서를 참조하세요.

Secrets Manager 통합에 필요한 권한

사용자는 Secrets Manager 통합과 관련된 작업을 수행하는 데 필요한 권한이 있어야 합니다. 사용자에게 필요한 지정된 리소스에서 특정 API 작업을 수행할 수 있는 권한을 부여하는 IAM 정책을 생성할 수 있습니다. 그런 다음 해당 권한이 필요한 IAM 권한 세트 또는 역할에 이러한 정책을 연결할 수 있습니다. 자세한 내용은 Amazon RDS의 자격 증명 및 액세스 관리 섹션을 참조하세요.

생성, 수정 또는 복원 작업의 경우 Amazon RDS가 Secrets Manager에서 마스터 사용자 암호를 관리하도록 지정하는 사용자는 다음 작업을 수행할 권한이 있어야 합니다.

- `kms:DescribeKey`
- `secretsmanager:CreateSecret`
- `secretsmanager:TagResource`

`MasterUserSecretKmsKeyId`에 대한 고객 관리형 키에 액세스하고 `aws/secretsmanager`를 설명하려면 `kms:DescribeKey` 권한이 필요합니다.

생성, 수정 또는 복원 작업의 경우 Secrets Manager에서 비밀을 암호화하는 고객 관리형 키를 지정하는 사용자는 다음 작업을 수행할 권한이 있어야 합니다.

- kms:Decrypt
- kms:GenerateDataKey
- kms:CreateGrant

수정 작업의 경우 Secrets Manager에서 마스터 사용자 암호를 교체하는 사용자는 다음 작업을 수행할 권한이 있어야 합니다.

- secretsmanager:RotateSecret

AWS Secrets Manager 마스터 사용자 암호의 RDS 관리 적용

IAM 조건 키를 사용하여 AWS Secrets Manager에서 마스터 사용자 암호의 RDS 관리를 적용할 수 있습니다. 다음 정책은 Secrets Manager에서 RDS 가 마스터 사용자 암호를 관리하지 않는 한 사용자가 DB 인스턴스 또는 DB 클러스터를 생성 또는 복원하거나 테넌트 데이터베이스를 생성 또는 수정하는 것을 허용하지 않습니다.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": ["rds:CreateDBInstance", "rds:CreateDBCluster",
 "rds:RestoreDBInstanceFromS3", "rds:RestoreDBClusterFromS3",
                       "rds:RestoreDBInstanceFromDBSnapshot",
 "rds:RestoreDBInstanceToPointInTime", "rds:CreateTenantDatabase",
                       "rds:ModifyTenantDatabase"],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "rds:ManageMasterUserPassword": false
                }
            }
        }
    ]
}
```

> **Note**
>
> 이 정책은 생성 시 AWS Secrets Manager에서 암호 관리를 적용합니다. 그러나 인스턴스를 수정하여 여전히 Secrets Manager 통합을 비활성화하고 마스터 암호를 수동으로 설정할 수 있습니다.
> 이를 방지하려면 정책의 Action 블록에 `rds:ModifyDBInstance`, `rds:ModifyDBCluster`를 포함하세요. 이렇게 하면 사용자가 Secrets Manager 통합이 활성화되지 않은 기존 인스턴스에 추가 수정 사항을 적용할 수 없습니다.

IAM 정책에서 조건 키 사용에 관한 자세한 내용은 Amazon RDS의 정책 조건 키 및 정책 예: 조건 키 사용을 참조하세요.

Secrets Manager를 통해 DB 인스턴스의 마스터 사용자 암호 관리

다음 작업을 수행할 때 Secrets Manager에서 마스터 사용자 암호의 RDS 관리를 구성할 수 있습니다.

- Amazon RDS DB 인스턴스 생성
- Amazon RDS DB 인스턴스 수정
- MySQL DB 인스턴스로 백업 복원
- DB 인스턴스 복원(RDS for Oracle만 해당)
- Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원(RDS for Oracle만 해당)

RDS 콘솔, AWS CLI 또는 RDS API를 사용하여 이전 작업을 수행할 수 있습니다.

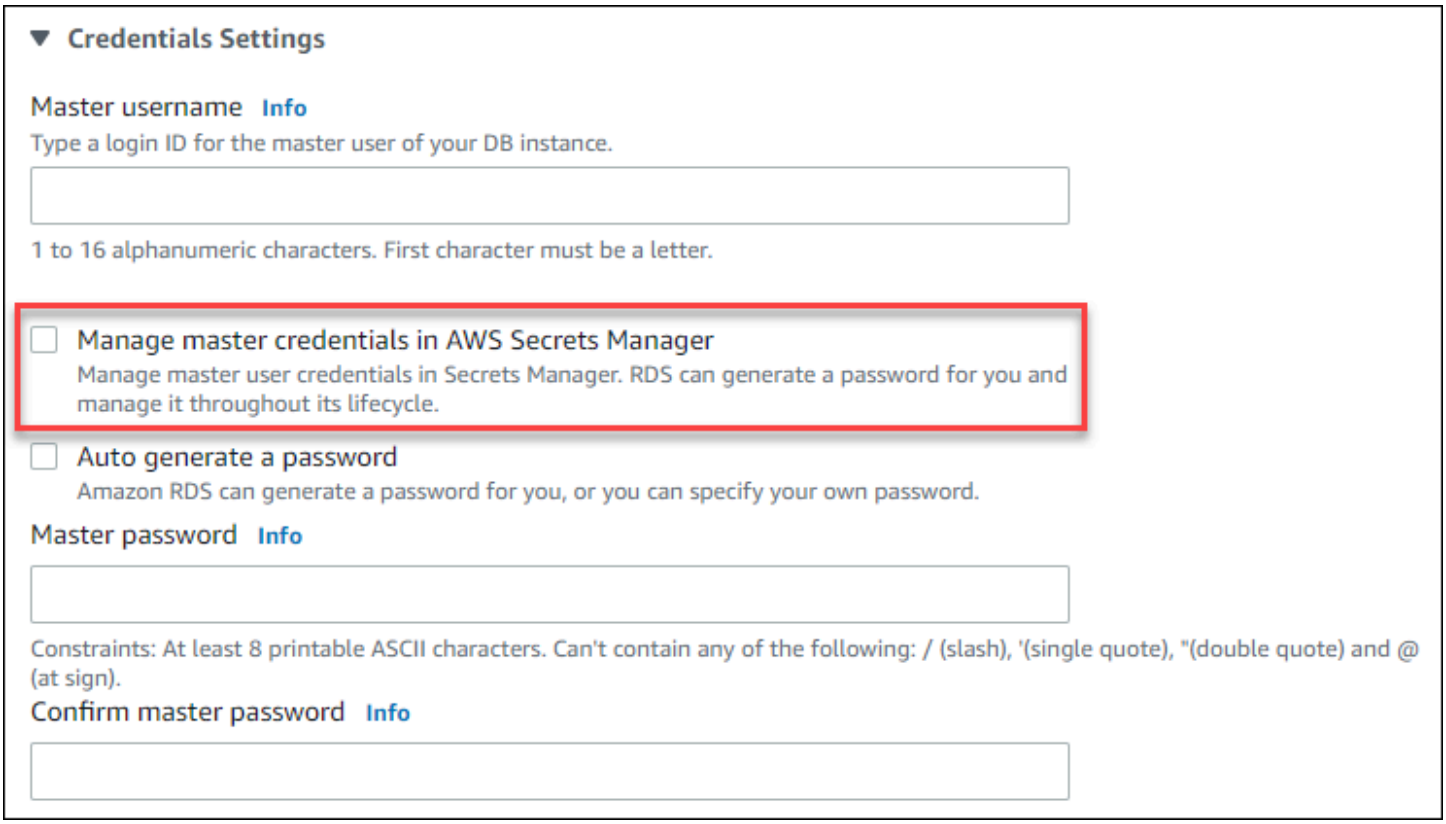
콘솔

지침에 따라 RDS 콘솔을 사용하여 DB 인스턴스를 생성하거나 수정합니다.

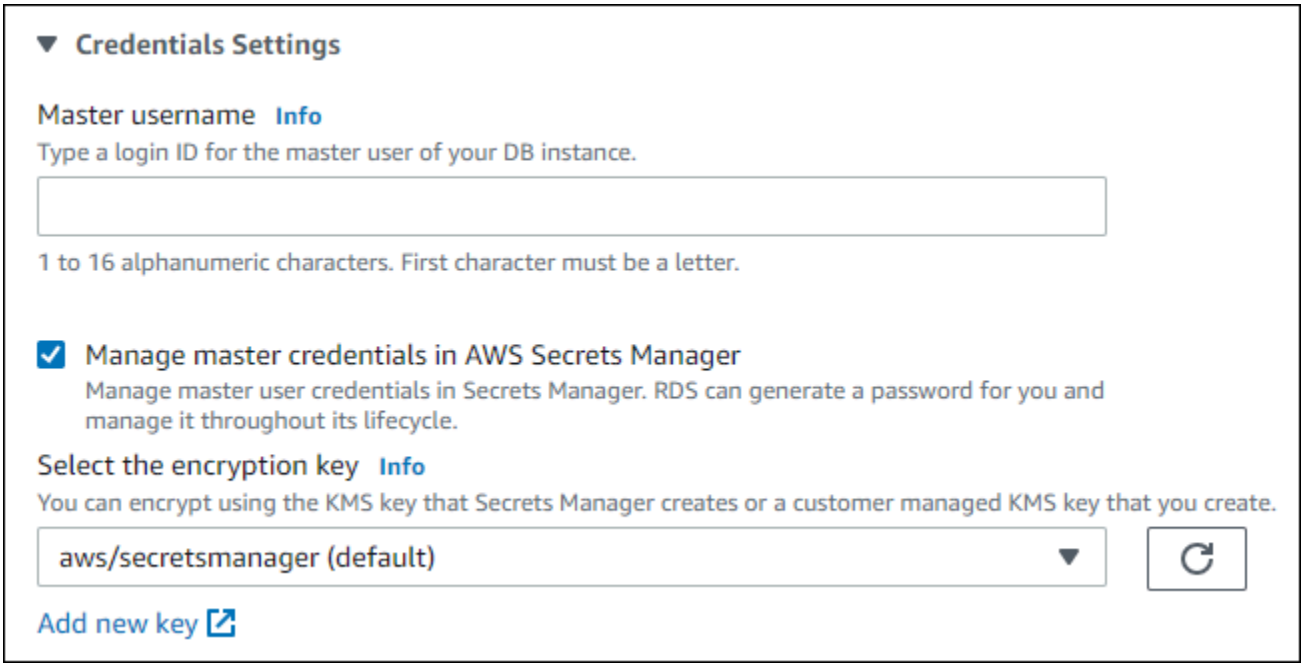
- DB 인스턴스 생성
- Amazon RDS DB 인스턴스 수정
- Amazon S3에서 새 MySQL DB 인스턴스로 데이터 가져오기

RDS 콘솔을 사용하여 이러한 작업 중 하나를 수행하는 경우 Secrets Manager에서 RDS가 마스터 사용자 암호를 관리하도록 지정할 수 있습니다. DB 인스턴스를 만들거나 복원할 때 자격 증명 설정에서 AWS Secrets Manager에서 마스터 자격 증명 관리를 선택합니다. DB 인스턴스를 수정할 때 설정에서 AWS Secrets Manager에서 마스터 자격 증명 관리를 선택합니다.

다음 이미지는 DB 인스턴스를 생성 또는 복원할 때의 Manage master credentials in AWS Secrets Manager 설정의 예입니다.



이 옵션을 선택하면 RDS가 Secrets Manager에서 마스터 사용자 암호를 생성하고 해당 수명 주기 동안 이를 관리합니다.



Secrets Manager가 제공하는 KMS 키 또는 사용자가 생성한 고객 관리 키를 사용하여 비밀을 암호화하도록 선택할 수 있습니다. RDS가 DB 인스턴스의 데이터베이스 자격 증명을 관리한 후에는 보안 암호를 암호화하는 데 사용된 KMS 키를 변경할 수 없습니다.

요구 사항에 맞는 다른 설정을 선택할 수 있습니다. DB 인스턴스를 만들 때 사용 가능한 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 섹션을 참조하세요. DB 인스턴스를 수정할 때 사용 가능한 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 섹션을 참조하세요.

AWS CLI

Secrets Manager에서 RDS를 사용하여 마스터 사용자 암호를 관리하려면 다음 AWS CLI 명령 중 하나에서 `--manage-master-user-password` 옵션을 지정하세요.

- create-db-instance
- modify-db-instance
- restore-db-instance-from-s3
- restore-db-instance-from-db-snapshot(RDS for Oracle만 해당)
- restore-db-instance-to-point-in-time(RDS for Oracle만 해당)

이러한 명령에서 `--manage-master-user-password` 옵션을 지정하면 RDS가 Secrets Manager에서 마스터 사용자 암호를 생성하고 해당 수명 주기 동안 이를 관리합니다.

비밀을 암호화하려면 고객 관리형 키를 지정하거나 Secrets Manager에서 제공하는 기본 KMS 키를 사용하면 됩니다. 고객 관리형 키를 지정하려면 `--master-user-secret-kms-key-id` 옵션을 사용하세요. AWS KMS 키 식별자는 KMS 키의 키 ARN, 키 ID, 별칭 ARN 또는 별칭 이름입니다. 다른 AWS 계정에서 KMS 키를 사용하려면 키 ARN 또는 별칭 ARN을 지정하세요. RDS가 DB 인스턴스의 데이터베이스 보안 인증 정보를 관리한 후에는 비밀을 암호화하는 데 사용된 KMS 키를 변경할 수 없습니다.

요구 사항에 맞는 다른 설정을 선택할 수 있습니다. DB 인스턴스를 생성할 때 사용 가능한 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정을 참조하세요. DB 인스턴스를 수정할 때 사용 가능한 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정을 참조하세요.

다음 예에서는 DB 인스턴스를 만들고 RDS 가 Secrets Manager에서 마스터 사용자 암호를 관리하도록 지정합니다. 비밀은 Secrets Manager에서 제공하는 KMS 키를 사용하여 암호화됩니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --engine mysql \
    --engine-version 8.0.39 \
    --db-instance-class db.r5b.large \
    --allocated-storage 200 \
    --master-username testUser \
    --manage-master-user-password
```

Windows의 경우:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --engine mysql ^
    --engine-version 8.0.39 ^
    --db-instance-class db.r5b.large ^
    --allocated-storage 200 ^
    --master-username testUser ^
    --manage-master-user-password
```

RDS API

RDS가 Secrets Manager에서 마스터 사용자 암호를 관리하도록 지정하려면 다음 RDS API 작업 중 하나에서 `ManageMasterUserPassword` 파라미터를 `true`로 설정합니다.

- CreateDBInstance
- ModifyDBInstance
- RestoreDBInstanceFromS3
- RestoreDBInstanceFromSnapshot(RDS for Oracle만 해당)
- RestoreDBInstanceToPointInTime(RDS for Oracle만 해당)

이러한 명령에서 `ManageMasterUserPassword` 파라미터를 `true`로 설정하면 RDS가 Secrets Manager에서 마스터 사용자 암호를 생성하고 해당 수명 주기 동안 이를 관리합니다.

비밀을 암호화하려면 고객 관리형 키를 지정하거나 Secrets Manager에서 제공하는 기본 KMS 키를 사용하면 됩니다. `MasterUserSecretKmsKeyId` 파라미터를 사용하여 고객 관리형 키를 지정합니다. AWS KMS 키 식별자는 KMS 키의 키 ARN, 키 ID, 별칭 ARN 또는 별칭 이름입니다. 다른 AWS 계정에서 KMS 키를 사용하려면 키 ARN 또는 별칭 ARN을 지정하세요. RDS가 DB 인스턴스의 데이터베이스 보안 인증 정보를 관리한 후에는 비밀을 암호화하는 데 사용된 KMS 키를 변경할 수 없습니다.

Secrets Manager를 사용하여 RDS for Oracle 테넌트 데이터베이스의 마스터 사용자 암호 관리

다음 작업을 수행할 때 Secrets Manager에서 마스터 사용자 암호의 RDS 관리를 구성할 수 있습니다.

- CDB 인스턴스에 RDS for Oracle 테넌트 데이터베이스 추가
- RDS for Oracle 테넌트 데이터베이스 수정

RDS 콘솔, AWS CLI 또는 RDS API를 사용하여 앞서 설명한 작업을 수행할 수 있습니다.

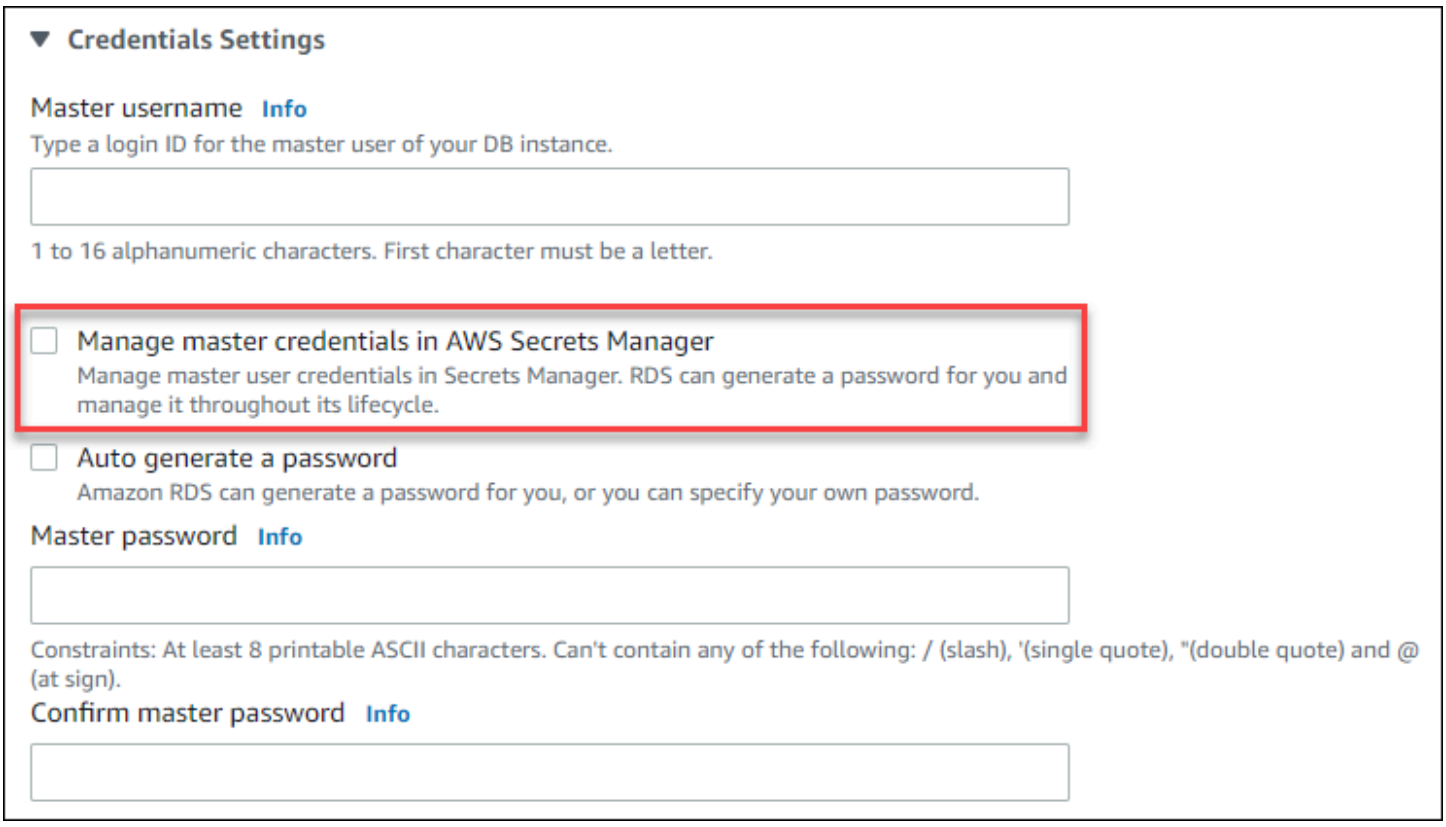
콘솔

지침에 따라 RDS 콘솔을 사용하여 RDS for Oracle 테넌트 데이터베이스를 만들거나 수정합니다.

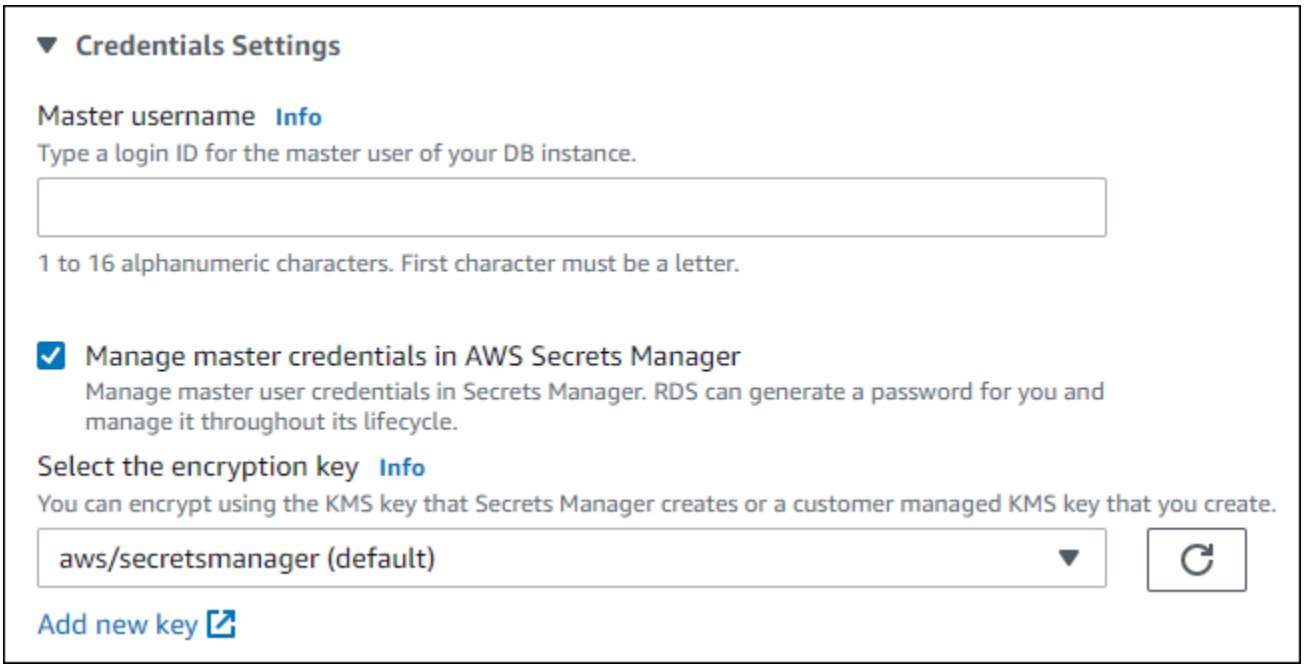
- CDB 인스턴스에 RDS for Oracle 테넌트 데이터베이스 추가
- RDS for Oracle 테넌트 데이터베이스 수정

RDS 콘솔을 사용하여 앞서 설명한 작업 중 하나를 수행하는 경우 Secrets Manager에서 RDS가 마스터 사용자 암호를 관리하도록 지정할 수 있습니다. 테넌트 데이터베이스를 만들 때 자격 증명 설정에서 AWS Secrets Manager에서 마스터 자격 증명 관리를 선택합니다. 테넌트 데이터베이스를 수정할 때 설정에서 AWS Secrets Manager에서 마스터 자격 증명 관리를 선택합니다.

다음 이미지는 테넌트 데이터베이스를 만들 때의 AWS Secrets Manager에서 마스터 자격 증명 관리 설정을 예로 든 것입니다.



이 옵션을 선택하면 RDS가 Secrets Manager에서 마스터 사용자 암호를 생성하고 해당 수명 주기 동안 이를 관리합니다.



Secrets Manager가 제공하는 KMS 키 또는 사용자가 생성한 고객 관리 키를 사용하여 비밀을 암호화하도록 선택할 수 있습니다. RDS가 테넌트 데이터베이스의 데이터베이스 자격 증명을 관리한 후에는 보안 암호를 암호화하는 데 사용된 KMS 키를 변경할 수 없습니다.

요구 사항에 맞는 다른 설정을 선택할 수 있습니다. 테넌트 데이터베이스를 만들 때 사용 가능한 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 섹션을 참조하세요. 테넌트 데이터베이스를 수정할 때 사용 가능한 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 섹션을 참조하세요.

AWS CLI

Secrets Manager에서 RDS를 사용하여 마스터 사용자 암호를 관리하려면 다음 AWS CLI 명령 중 하나에서 `--manage-master-user-password` 옵션을 지정하세요.

- create-tenant-database
- modify-tenant-database

앞서 설명한 명령에서 `--manage-master-user-password` 옵션을 지정하면 RDS가 Secrets Manager에서 마스터 사용자 암호를 생성하고 해당 수명 주기 동안 이를 관리합니다.

비밀을 암호화하려면 고객 관리형 키를 지정하거나 Secrets Manager에서 제공하는 기본 KMS 키를 사용하면 됩니다. 고객 관리형 키를 지정하려면 `--master-user-secret-kms-key-id` 옵션을 사용하세요. AWS KMS 키 식별자는 KMS 키의 키 ARN, 키 ID, 별칭 ARN 또는 별칭 이름입니다. 다른 AWS 계정에서 KMS 키를 사용하려면 키 ARN 또는 별칭 ARN을 지정하세요. RDS가 테넌트 데이터베이스의 데이터베이스 자격 증명을 관리한 후에는 보안 암호를 암호화하는 데 사용된 KMS 키를 변경할 수 없습니다.

요구 사항에 맞는 다른 설정을 선택할 수 있습니다. 테넌트 데이터베이스를 만들 때 사용 가능한 설정에 대한 자세한 내용은 create-tenant-database를 참조하세요. 테넌트 데이터베이스를 수정할 때 사용 가능한 설정에 대한 자세한 내용은 modify-tenant-database를 참조하세요.

다음 예에서는 RDS for Oracle 테넌트 데이터베이스를 만들고 RDS가 Secrets Manager에서 마스터 사용자 암호를 관리하도록 지정합니다. 비밀은 Secrets Manager에서 제공하는 KMS 키를 사용하여 암호화됩니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-tenant-database --region us-east-1 \
    --db-instance-identifier my-cdb-inst \
```

```
    --tenant-db-name mypdb2 \
    --master-username mypdb2-admin \
    --character-set-name UTF-16 \
    --manage-master-user-password
```

Windows의 경우:

```
aws rds create-tenant-database --region us-east-1 ^
    --db-instance-identifier my-cdb-inst ^
    --tenant-db-name mypdb2 ^
    --master-username mypdb2-admin ^
    --character-set-name UTF-16 ^
    --manage-master-user-password
```

RDS API

RDS가 Secrets Manager에서 마스터 사용자 암호를 관리하도록 지정하려면 다음 RDS API 작업 중 하나에서 `ManageMasterUserPassword` 파라미터를 `true`로 설정합니다.

- CreateTenantDatabase
- ModifyTenantDatabase

이러한 명령에서 `ManageMasterUserPassword` 파라미터를 `true`로 설정하면 RDS가 Secrets Manager에서 마스터 사용자 암호를 생성하고 해당 수명 주기 동안 이를 관리합니다.

비밀을 암호화하려면 고객 관리형 키를 지정하거나 Secrets Manager에서 제공하는 기본 KMS 키를 사용하면 됩니다. `MasterUserSecretKmsKeyId` 파라미터를 사용하여 고객 관리형 키를 지정합니다. AWS KMS 키 식별자는 KMS 키의 키 ARN, 키 ID, 별칭 ARN 또는 별칭 이름입니다. 다른 AWS 계정에서 KMS 키를 사용하려면 키 ARN 또는 별칭 ARN을 지정하세요. RDS가 테넌트 데이터베이스의 데이터베이스 자격 증명을 관리한 후에는 보안 암호를 암호화하는 데 사용된 KMS 키를 변경할 수 없습니다.

Secrets Manager를 사용하여 다중 AZ DB 클러스터의 마스터 사용자 암호 관리

다음 작업을 수행할 때 Secrets Manager에서 마스터 사용자 암호의 RDS 관리를 구성할 수 있습니다.

- Amazon RDS용 다중 AZ DB 클러스터 생성
- Amazon RDS용 다중 AZ DB 클러스터 수정

콘솔, AWS CLI 또는 RDS API를 사용하여 이러한 작업을 수행할 수 있습니다.

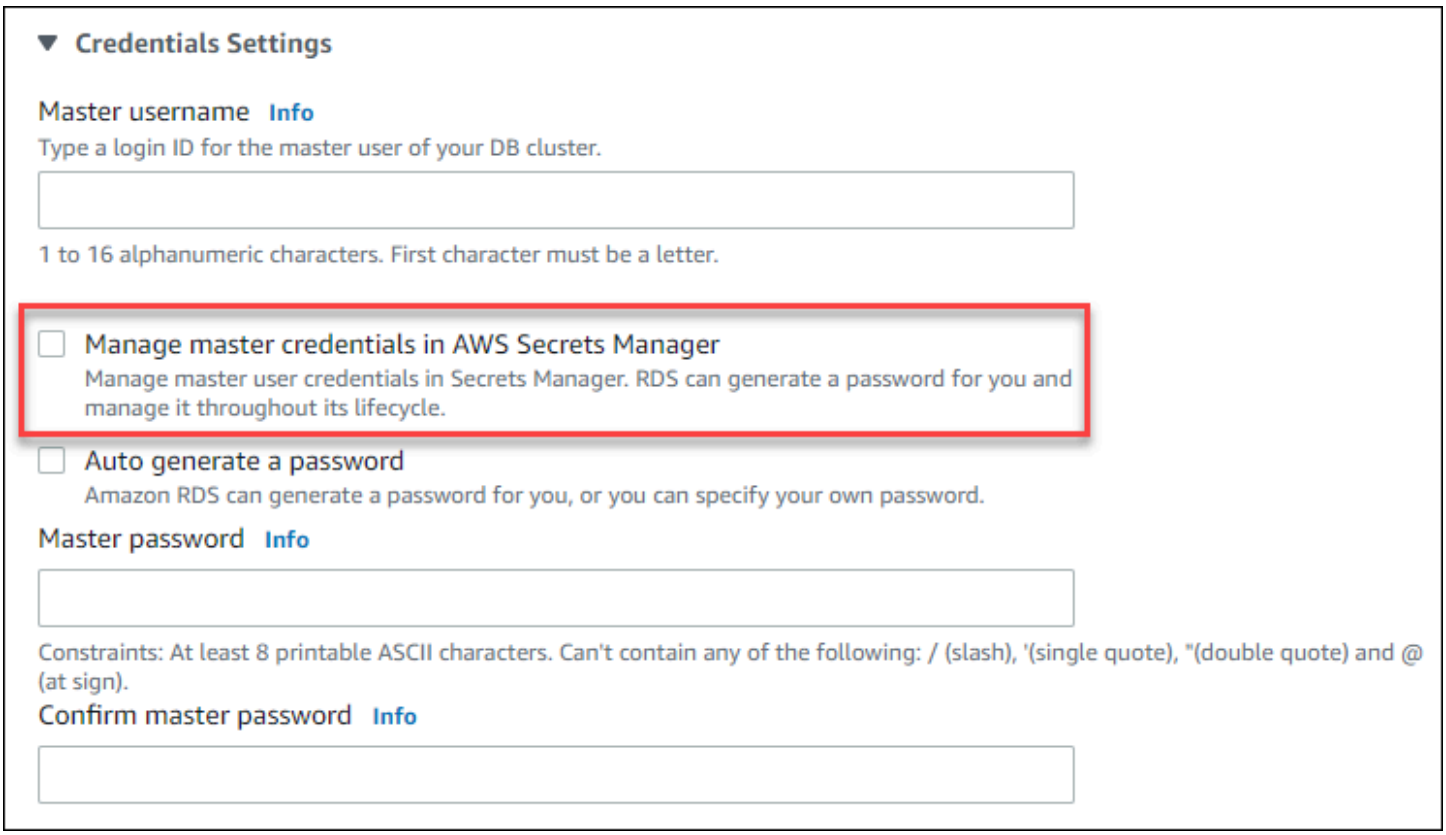
콘솔

다음 지침에 따라 RDS 콘솔을 사용하여 다중 AZ DB 클러스터를 생성하거나 수정합니다.

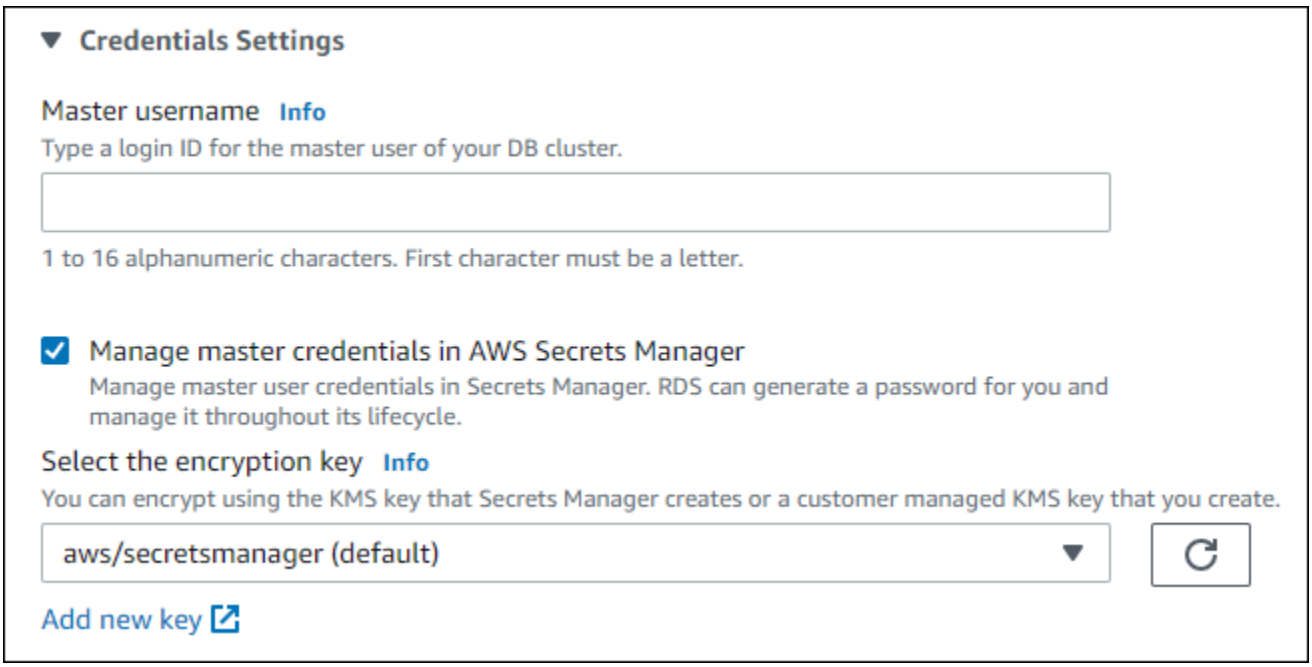
- DB 클러스터 생성
- Amazon RDS용 다중 AZ DB 클러스터 수정

RDS 콘솔을 사용하여 이러한 작업 중 하나를 수행하는 경우 Secrets Manager에서 RDS가 마스터 사용자 암호를 관리하도록 지정할 수 있습니다. DB 클러스터를 생성 할 때 이를 수행하려면 Credential settings(보안 인증 정보 설정)에서 Manage master credentials in AWS Secrets Manager를 선택합니다. DB 클러스터를 수정할 경우 Settings(설정)에서 Manage master credentials in AWS Secrets Manager를 선택합니다.

다음 이미지는 DB 클러스터를 생성 할 때의 Manage master credentials in AWS Secrets Manager 설정의 예입니다.



이 옵션을 선택하면 RDS가 Secrets Manager에서 마스터 사용자 암호를 생성하고 수명 주기 동안 이를 관리합니다.



Secrets Manager가 제공하는 KMS 키 또는 사용자가 생성한 고객 관리 키를 사용하여 비밀을 암호화하도록 선택할 수 있습니다. RDS가 DB 클러스터의 데이터베이스 보안 인증 정보를 관리한 후에는 비밀을 암호화하는 데 사용되는 KMS 키를 변경할 수 없습니다.

요구 사항에 맞는 다른 설정을 선택할 수 있습니다.

다중 AZ DB 클러스터를 생성할 때 사용 가능한 설정에 대한 자세한 내용은 다중 AZ DB 클러스터를 생성하기 위한 설정을 참조하세요. 다중 AZ DB 클러스터를 수정할 때 사용 가능한 설정에 대한 자세한 내용은 다중 AZ DB 클러스터를 수정하기 위한 설정을 참조하세요.

AWS CLI

RDS가 Secrets Manager에서 마스터 사용자 암호를 관리하도록 지정하려면 다음 명령 중 하나에서 `--manage-master-user-password` 옵션을 지정하세요.

- create-db-cluster
- modify-db-cluster

이러한 명령에서 `--manage-master-user-password` 옵션을 지정하면 RDS가 Secrets Manager에서 마스터 사용자 암호를 생성하고 해당 수명 주기 동안 이를 관리합니다.

비밀을 암호화하려면 고객 관리형 키를 지정하거나 Secrets Manager에서 제공하는 기본 KMS 키를 사용하면 됩니다. 고객 관리형 키를 지정하려면 `--master-user-secret-kms-key-id` 옵션을 사용

하세요. AWS KMS 키 식별자는 KMS 키의 키 ARN, 키 ID, 별칭 ARN 또는 별칭 이름입니다. 다른 AWS 계정에서 KMS 키를 사용하려면 키 ARN 또는 별칭 ARN을 지정하세요. RDS DB 클러스터의 데이터베이스 보안 인증 정보를 관리한 후에는 비밀을 암호화하는 데 사용되는 KMS 키를 변경할 수 없습니다.

요구 사항에 맞는 다른 설정을 선택할 수 있습니다.

다중 AZ DB 클러스터를 생성할 때 사용 가능한 설정에 대한 자세한 내용은 다중 AZ DB 클러스터를 생성하기 위한 설정을 참조하세요. 다중 AZ DB 클러스터를 수정할 때 사용 가능한 설정에 대한 자세한 내용은 다중 AZ DB 클러스터를 수정하기 위한 설정을 참조하세요.

이 예에서는 다중 AZ DB 클러스터를 생성하고 RDS가 Secrets Manager에서 암호를 관리하도록 지정합니다. 비밀은 Secrets Manager에서 제공하는 KMS 키를 사용하여 암호화됩니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-cluster \
   --db-cluster-identifier mysql-multi-az-db-cluster \
   --engine mysql \
   --engine-version 8.0.39  \
   --backup-retention-period 1  \
   --allocated-storage 4000 \
   --storage-type io1 \
   --iops 10000 \
   --db-cluster-instance-class db.r6gd.xlarge \
   --master-username testUser \
   --manage-master-user-password
```

Windows의 경우:

```
aws rds create-db-cluster ^
   --db-cluster-identifier mysql-multi-az-db-cluster ^
   --engine mysql ^
   --engine-version 8.0.39 ^
   --backup-retention-period 1 ^
   --allocated-storage 4000 ^
   --storage-type io1 ^
   --iops 10000 ^
   --db-cluster-instance-class db.r6gd.xlarge ^
   --master-username testUser ^
```

```
    --manage-master-user-password
```

RDS API

RDS가 Secrets Manager에서 마스터 사용자 암호를 관리하도록 지정하려면 다음 작업 중 하나에서 `ManageMasterUserPassword` 파라미터를 `true`로 설정하세요.

- CreateDBCluster
- ModifyDBCluster

이러한 명령에서 `ManageMasterUserPassword` 파라미터를 `true`로 설정하면 RDS가 Secrets Manager에서 마스터 사용자 암호를 생성하고 해당 수명 주기 동안 이를 관리합니다.

비밀을 암호화하려면 고객 관리형 키를 지정하거나 Secrets Manager에서 제공하는 기본 KMS 키를 사용하면 됩니다. `MasterUserSecretKmsKeyId` 파라미터를 사용하여 고객 관리형 키를 지정합니다. AWS KMS 키 식별자는 KMS 키의 키 ARN, 키 ID, 별칭 ARN 또는 별칭 이름입니다. 다른 AWS 계정에서 KMS 키를 사용하려면 키 ARN 또는 별칭 ARN을 지정하세요. RDS DB 클러스터의 데이터베이스 보안 인증 정보를 관리한 후에는 비밀을 암호화하는 데 사용되는 KMS 키를 변경할 수 없습니다.

DB 인스턴스의 마스터 사용자 암호 비밀 교체

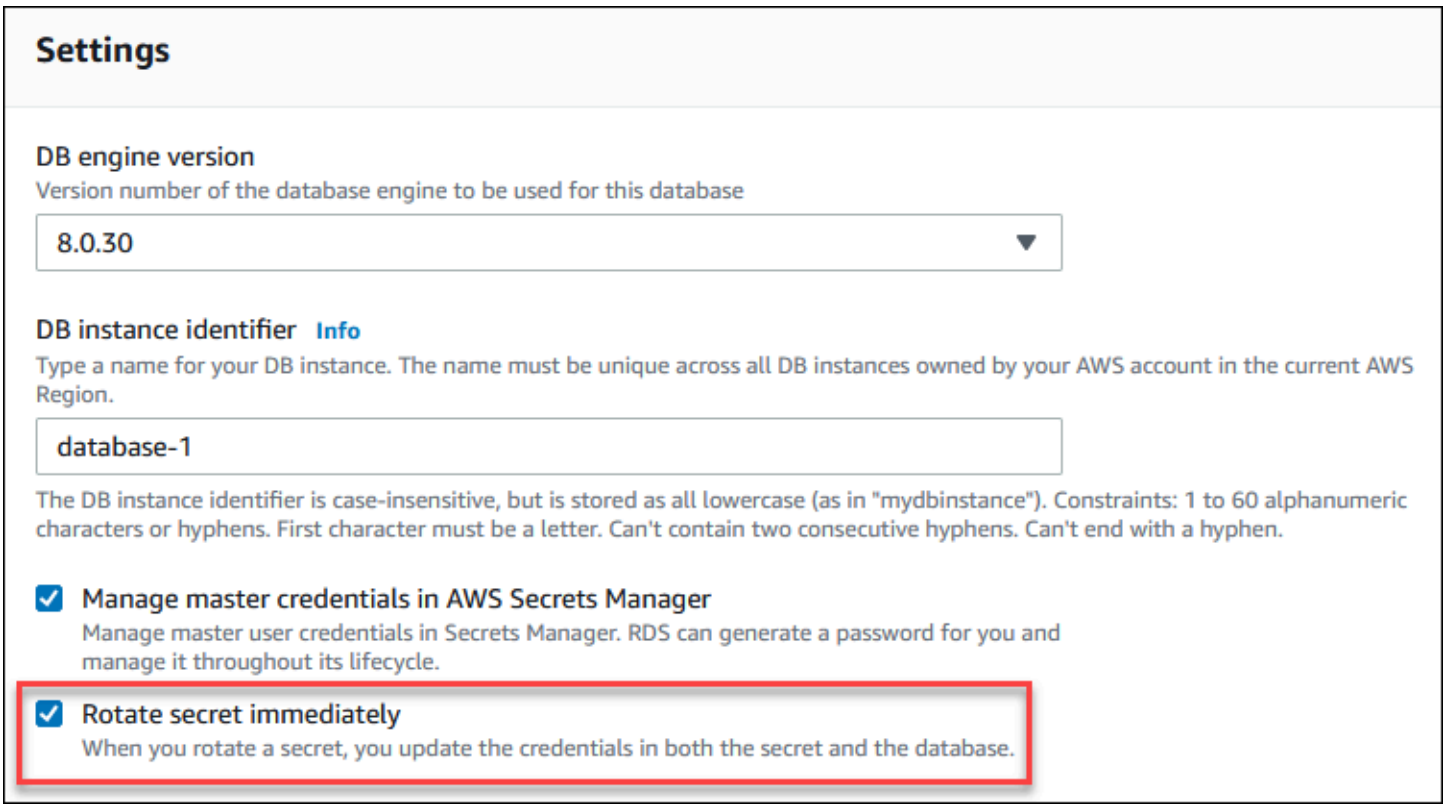
RDS가 마스터 사용자 암호 비밀을 교체하면 Secrets Manager는 기존 비밀의 새 비밀 버전을 생성합니다. 새 버전의 비밀에는 새 마스터 사용자 암호가 포함됩니다. Amazon RDS는 새 비밀 버전의 암호와 일치하도록 DB 인스턴스의 마스터 사용자 암호를 변경합니다.

예약된 교체를 기다리지 않고 비밀을 즉시 교체할 수 있습니다. Secrets Manager에서 마스터 사용자 암호 비밀을 교체하려면 DB 인스턴스를 수정합니다. DB 인스턴스 수정에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

RDS 콘솔, AWS CLI 또는 RDS API를 사용하여 마스터 사용자 암호 비밀을 즉시 교체할 수 있습니다. 새 암호는 항상 28자이며 하나 이상의 대문자와 소문자, 하나의 숫자, 하나의 구두점을 포함합니다.

콘솔

RDS 콘솔을 사용하여 마스터 사용자 암호 비밀을 교체하려면 DB 인스턴스를 수정하고 Settings(설정)에서 Rotate secret immediately(비밀 즉시 교체)를 선택합니다.



RDS 콘솔을 사용하여 DB 인스턴스를 수정하려면 Amazon RDS DB 인스턴스 수정의 지침에 따르세요. 확인 페이지에서 Apply immediately(즉시 적용)를 선택해야 합니다.

AWS CLI

AWS CLI를 사용하여 마스터 사용자 암호 비밀을 교체하려면 modify-db-instance 명령을 사용하고 `--rotate-master-user-password` 옵션을 지정합니다. 마스터 암호를 교체할 때 `--apply-immediately` 옵션을 지정해야 합니다.

이 예에서는 마스터 사용자 암호 비밀을 교체합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --rotate-master-user-password \
    --apply-immediately
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --rotate-master-user-password ^
    --apply-immediately
```

RDS API

ModifyDBInstance 작업을 사용하고 RotateMasterUserPassword 파라미터를 true로 설정하여 마스터 사용자 암호 비밀을 교체할 수 있습니다. 마스터 암호를 교체할 때 ApplyImmediately 파라미터를 true로 설정해야 합니다.

다중 AZ DB 클러스터의 마스터 사용자 암호 비밀 교체

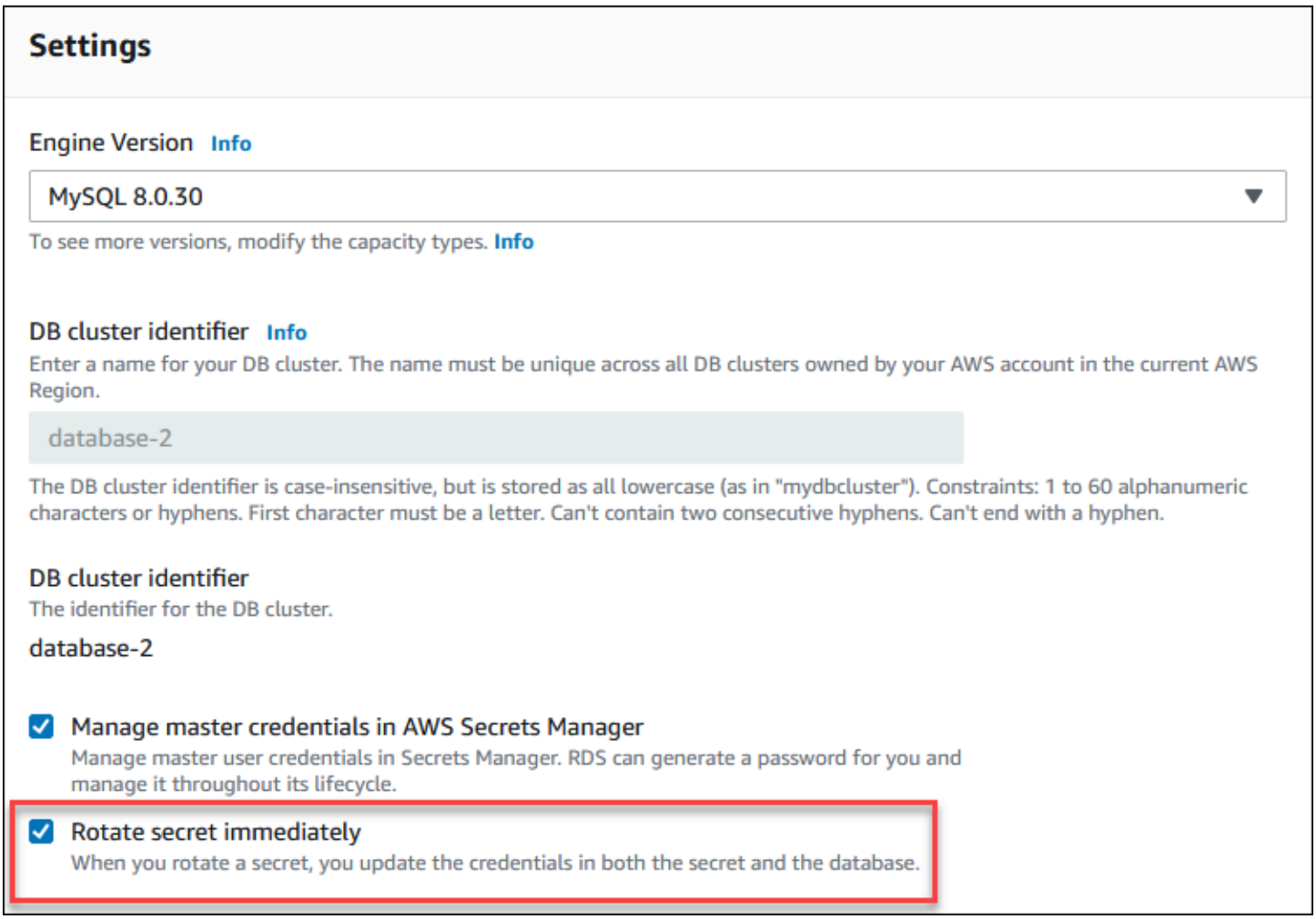
RDS가 마스터 사용자 암호 비밀을 교체하면 Secrets Manager는 기존 비밀의 새 비밀 버전을 생성합니다. 새 버전의 비밀에는 새 마스터 사용자 암호가 포함됩니다. Amazon RDS는 새 비밀 버전의 암호와 일치하도록 다중 AZ DB 클러스터의 마스터 사용자 암호를 변경합니다.

예약된 교체를 기다리지 않고 비밀을 즉시 교체할 수 있습니다. Secrets Manager에서 마스터 사용자 암호 비밀을 교체하려면 다중 AZ DB 클러스터를 수정하세요. 다중 AZ DB 클러스터 수정에 대한 자세한 내용은 Amazon RDS용 다중 AZ DB 클러스터 수정을 참조하세요.

RDS 콘솔, AWS CLI 또는 RDS API를 사용하여 마스터 사용자 암호 비밀을 즉시 교체할 수 있습니다. 새 암호는 항상 28자이며 하나 이상의 대문자와 소문자, 하나의 숫자, 하나의 구두점을 포함합니다.

콘솔

RDS 콘솔을 사용하여 마스터 사용자 암호 비밀을 교체하려면 다중 AZ DB 클러스터를 수정하고 Settings(설정)에서 Rotate secret immediately(암호 즉시 교체)를 선택합니다.



RDS 콘솔을 사용하여 다중 AZ DB 클러스터를 수정하려면 Amazon RDS용 다중 AZ DB 클러스터 수정 의 지침에 따르세요. 확인 페이지에서 Apply immediately(즉시 적용)를 선택해야 합니다.

AWS CLI

AWS CLI를 사용하여 마스터 사용자 암호를 교체하려면 modify-db-cluster 명령을 사용하고 `--rotate-master-user-password` 옵션을 지정합니다. 마스터 암호를 교체할 때 `--apply-immediately` 옵션을 지정해야 합니다.

이 예에서는 마스터 사용자 암호 비밀을 교체합니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-cluster \
```

```
    --db-cluster-identifier mydbcluster \
    --rotate-master-user-password \
    --apply-immediately
```

Windows의 경우:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --rotate-master-user-password ^
    --apply-immediately
```

RDS API

ModifyDBCluster 작업을 사용하고 RotateMasterUserPassword 파라미터를 true로 설정하여 마스터 사용자 암호 비밀을 교체할 수 있습니다. 마스터 암호를 교체할 때 ApplyImmediately 파라미터를 true로 설정해야 합니다.

DB 인스턴스의 비밀에 대한 세부 정보 보기

콘솔(https://console.aws.amazon.com/secretsmanager/) 또는 AWS CLI(get-secret-value Secrets Manager 명령)을 사용하여 비밀을 검색할 수 있습니다.

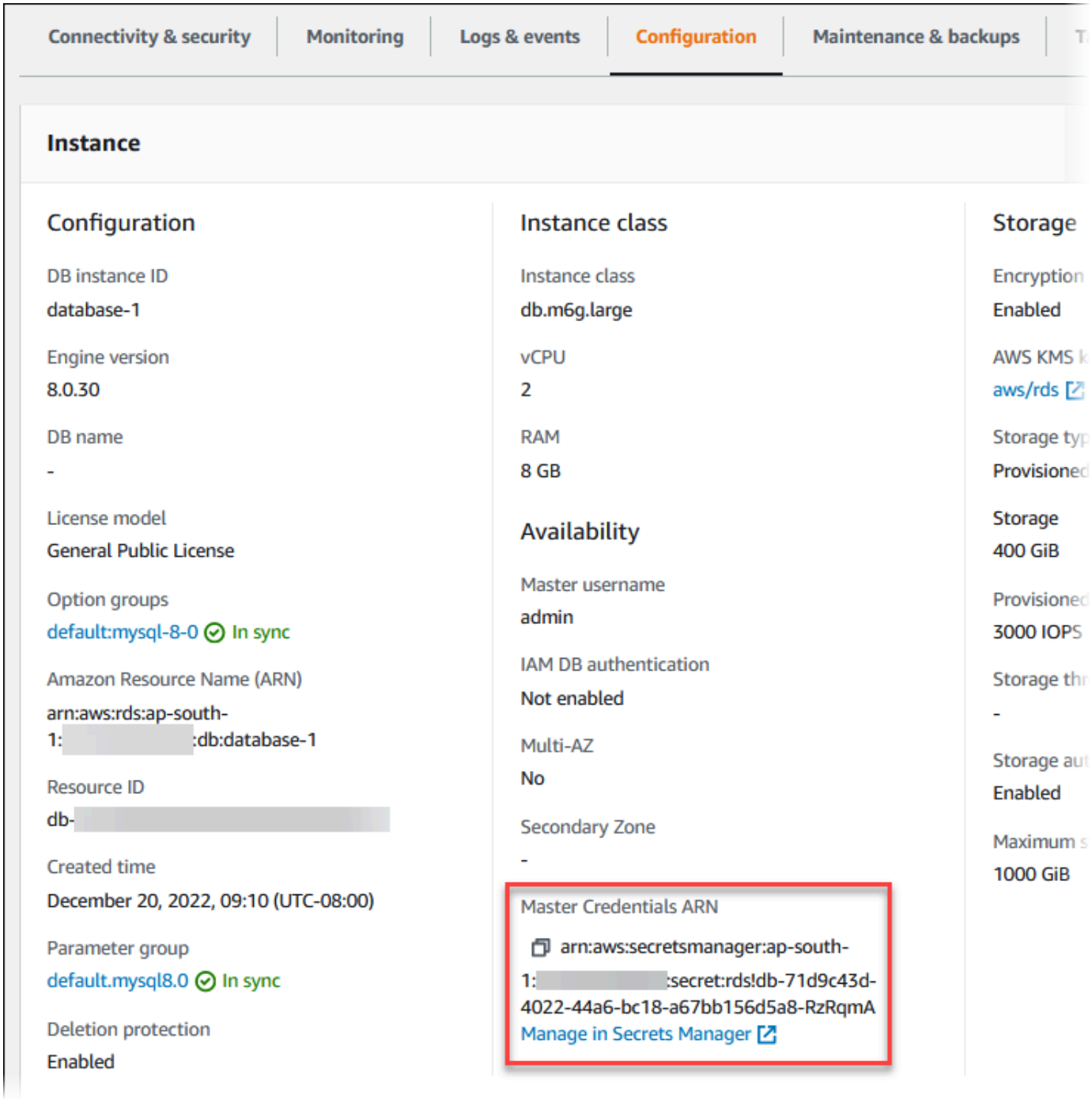
RDS 콘솔, AWS CLI 또는 RDS API를 사용하여 Secrets Manager에서 RDS가 관리하는 비밀의 Amazon 리소스 이름(ARN)을 찾을 수 있습니다.

콘솔

Secrets Manager에서 RDS가 관리하는 비밀에 대한 세부 정보 보기

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
3. 세부 정보를 표시할 DB 인스턴스의 이름을 선택합니다.
4. Configuration(구성) 탭을 선택합니다.

   마스터 보안 인증 정보 ARN에서 비밀 ARN을 볼 수 있습니다.



Manage in Secrets Manager(Secrets Manager에서 관리) 링크를 클릭하면 Secrets Manager 콘솔에서 비밀을 보고 관리할 수 있습니다.

AWS CLI

describe-db-instance RDS CLI 명령을 사용하면 Secrets Manager에서 RDS가 관리하는 비밀에 대한 다음 정보를 찾을 수 있습니다.

- `SecretArn` - 비밀의 ARN
- `SecretStatus` - 비밀의 상태

  가능한 상태 값에는 다음이 포함됩니다.

  - `creating` - 비밀이 생성 중입니다.
  - `active` - 비밀의 일반적 사용 및 교체가 가능합니다.
  - `rotating` - 비밀이 교체 중입니다.
  - `impaired` - 비밀을 데이터베이스 보안 인증 정보에 액세스하는 데 사용할 수 있지만 교체할 수는 없습니다. 예를 들어 권한이 변경되어 RDS가 더 이상 비밀 또는 비밀의 KMS 키에 액세스할 수 없는 경우 비밀이 이 상태가 될 수 있습니다.

    암호가 이 상태인 경우 상태를 초래한 조건을 수정할 수 있습니다. 상태를 초래한 조건을 수정하면 다음 교체까지 상태가 `impaired`로 유지됩니다. 또는 DB 인스턴스를 수정하여 데이터베이스 보안 인증 정보 자동 관리를 해제한 다음 DB 인스턴스를 다시 수정하여 데이터베이스 보안 인증 정보 자동 관리를 켤 수 있습니다. DB 인스턴스를 수정하려면 modify-db-instance 명령에서 `--manage-master-user-password` 옵션을 사용하세요.
- `KmsKeyId` - 비밀을 암호화하는 데 사용된 KMS 키의 ARN입니다.

특정 DB 인스턴스의 출력을 표시하려면 `--db-instance-identifier` 옵션을 지정합니다. 이 예제는 DB 인스턴스가 사용하는 암호의 출력을 보여 줍니다.

Example

```
aws rds describe-db-instances --db-instance-identifier mydbinstance
```

다음은 비밀의 샘플 출력입니다.

```
"MasterUserSecret": {
                "SecretArn": "arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
db-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx",
                "SecretStatus": "active",
                "KmsKeyId": "arn:aws:kms:eu-
west-1:123456789012:key/0987dcba-09fe-87dc-65ba-ab0987654321"
            }
```

보안 ARN이 있으면 get-secret-value Secrets Manager CLI 명령을 사용하여 비밀에 대한 세부 정보를 볼 수 있습니다.

이 예제는 이전 샘플 출력의 비밀에 대한 세부 정보를 보여 줍니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws secretsmanager get-secret-value \
    --secret-id 'arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
db-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx'
```

Windows의 경우:

```
aws secretsmanager get-secret-value ^
    --secret-id 'arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
db-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx'
```

RDS API

DescribedBInstances 작업을 사용하고 `DBInstanceIdentifier` 파라미터를 DB 인스턴스 식별자로 설정하면 Secrets Manager에서 RDS가 관리하는 비밀의 ARN, 상태, KMS 키를 볼 수 있습니다. 비밀에 대한 세부 정보가 출력에 포함됩니다.

비밀 ARN이 있으면 GetSecretValue Secrets Manager 작업을 사용하여 비밀에 대한 세부 정보를 볼 수 있습니다.

다중 AZ DB 클러스터의 비밀에 대한 세부 정보 보기

콘솔(https://console.aws.amazon.com/secretsmanager/) 또는 AWS CLI(get-secret-value Secrets Manager 명령)을 사용하여 비밀을 검색할 수 있습니다.

RDS 콘솔, AWS CLI 또는 RDS API를 사용하여 Secrets Manager에서 RDS 가 관리하는 비밀의 Amazon 리소스 이름(ARN)을 찾을 수 있습니다.

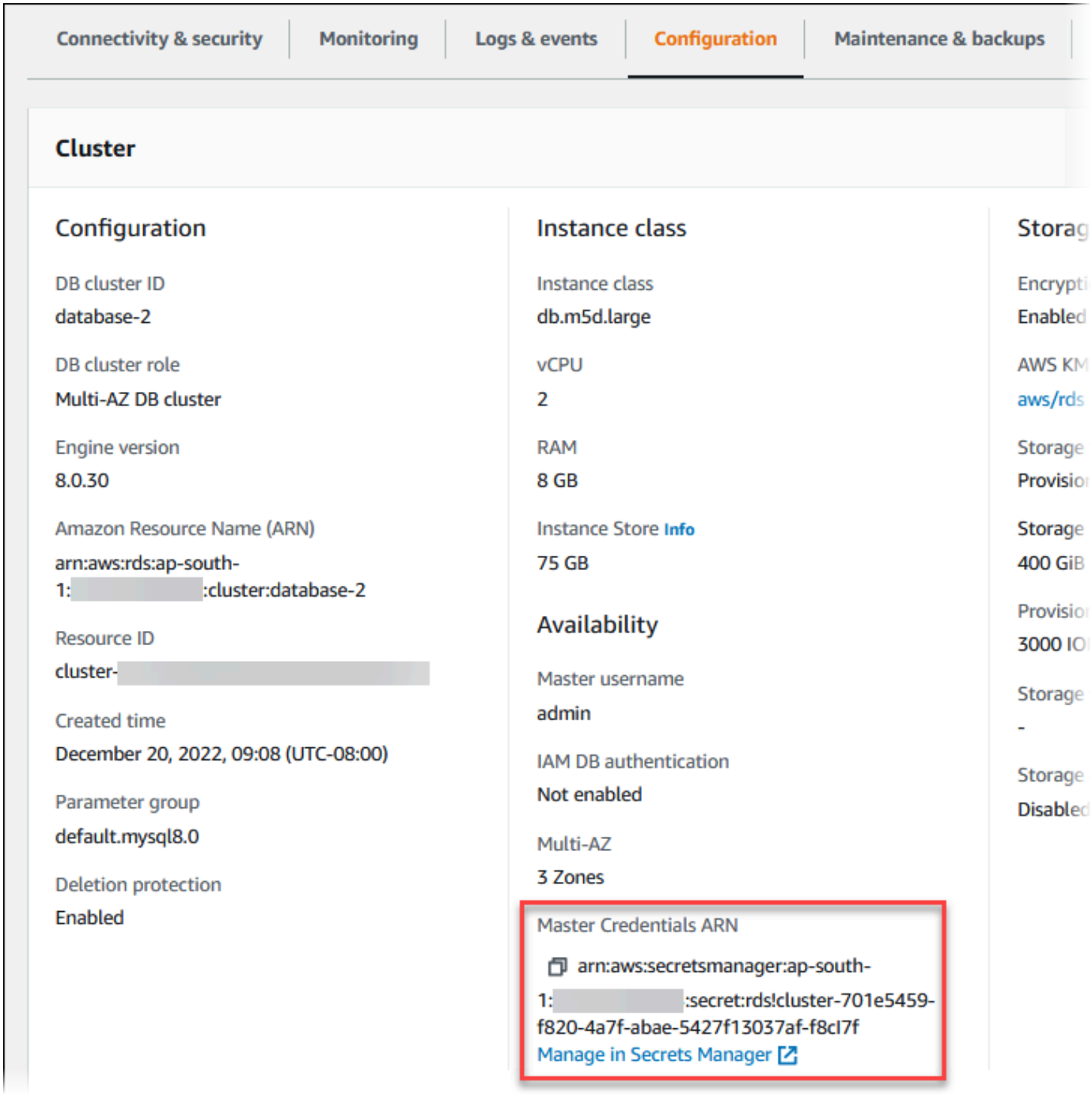
콘솔

Secrets Manager에서 RDS가 관리하는 비밀에 대한 세부 정보 보기

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택합니다.

3. 그런 다음 세부 정보를 표시할 다중 AZ DB 클러스터 이름을 선택합니다.
4. Configuration(구성) 탭을 선택합니다.

   마스터 보안 인증 정보 ARN에서 비밀 ARN을 볼 수 있습니다.



Manage in Secrets Manager(Secrets Manager에서 관리) 링크를 클릭하면 Secrets Manager 콘솔에서 비밀을 보고 관리할 수 있습니다.

AWS CLI

RDS AWS CLI describe-db-clusters 명령을 사용하면 Secrets Manager에서 RDS가 관리하는 비밀에 대한 다음 정보를 찾을 수 있습니다.

- `SecretArn` - 비밀의 ARN
- `SecretStatus` - 비밀의 상태

  가능한 상태 값에는 다음이 포함됩니다.

  - `creating` - 비밀이 생성 중입니다.
  - `active` - 비밀의 일반적 사용 및 교체가 가능합니다.
  - `rotating` - 비밀이 교체 중입니다.
  - `impaired` - 비밀을 데이터베이스 보안 인증 정보에 액세스하는 데 사용할 수 있지만 교체할 수는 없습니다. 예를 들어 권한이 변경되어 RDS가 더 이상 비밀 또는 비밀의 KMS 키에 액세스할 수 없는 경우 비밀이 이 상태가 될 수 있습니다.

    암호가 이 상태인 경우 상태를 초래한 조건을 수정할 수 있습니다. 상태를 초래한 조건을 수정하면 다음 교체까지 상태가 `impaired`로 유지됩니다. 또는 DB 클러스터를 수정하여 데이터베이스 보안 인증 정보 자동 관리를 해제한 다음 DB 클러스터를 다시 수정하여 데이터베이스 보안 인증 정보 자동 관리를 켤 수 있습니다. DB 클러스터를 수정하려면 modify-db-cluster 명령에서 `--manage-master-user-password` 옵션을 사용하세요.
- `KmsKeyId` - 비밀을 암호화하는 데 사용된 KMS 키의 ARN입니다.

특정 DB 클러스터의 출력을 표시하려면 `--db-cluster-identifier` 옵션을 지정합니다. 이 예제는 DB 클러스터가 사용하는 비밀의 출력을 보여 줍니다.

Example

```
aws rds describe-db-clusters --db-cluster-identifier mydbcluster
```

다음 샘플은 비밀 출력을 보여 줍니다.

```
"MasterUserSecret": {
                "SecretArn": "arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
cluster-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx",
                "SecretStatus": "active",
                "KmsKeyId": "arn:aws:kms:eu-
west-1:123456789012:key/0987dcba-09fe-87dc-65ba-ab0987654321"
```

```
        }
```

보안 ARN이 있으면 get-secret-value Secrets Manager CLI 명령을 사용하여 비밀에 대한 세부 정보를 볼 수 있습니다.

이 예제는 이전 샘플 출력의 비밀에 대한 세부 정보를 보여 줍니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws secretsmanager get-secret-value \
    --secret-id 'arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
cluster-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx'
```

Windows의 경우:

```
aws secretsmanager get-secret-value ^
    --secret-id 'arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
cluster-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx'
```

RDS API

DescribedBClusters RDS 작업을 사용하고 `DBClusterIdentifier` 파라미터를 DB 클러스터 식별자로 설정하여 Secrets Manager에서 RDS가 관리하는 비밀의 ARN, 상태, KMS 키를 볼 수 있습니다. 비밀에 대한 세부 정보가 출력에 포함됩니다.

비밀 ARN이 있으면 GetSecretValue Secrets Manager 작업을 사용하여 비밀에 대한 세부 정보를 볼 수 있습니다.

테넌트 데이터베이스의 보안 암호에 대한 세부 정보 보기

콘솔(https://console.aws.amazon.com/secretsmanager/) 또는 AWS CLI(get-secret-value Secrets Manager 명령)을 사용하여 비밀을 검색할 수 있습니다.

Amazon RDS 콘솔, AWS CLI 또는 Amazon RDS API를 사용하여 AWS Secrets Manager에서 Amazon RDS가 관리하는 보안 암호의 Amazon 리소스 이름(ARN)을 찾을 수 있습니다.

콘솔

테넌트 데이터베이스의 AWS Secrets Manager에서 Amazon RDS가 관리하는 보안 암호에 대한 세부 정보를 보려면 다음을 수행하세요.

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
3. 세부 정보를 표시할 테넌트 데이터베이스를 포함하는 DB 인스턴스의 이름을 선택합니다.
4. 구성 탭을 선택합니다.

   테넌트 데이터베이스 섹션에서 테넌트 데이터베이스를 찾고 마스터 자격 증명 ARN을 확인합니다.

   Manage in Secrets Manager(Secrets Manager에서 관리) 링크를 클릭하면 Secrets Manager 콘솔에서 비밀을 보고 관리할 수 있습니다.

AWS CLI

describe-tenant-databases Amazon RDS AWS CLI 명령을 사용하여 테넌트 데이터베이스의에서 AWS Secrets Manager의 Amazon RDS가 관리하는 보안 암호에 대한 다음 정보를 찾을 수 있습니다.

- `SecretArn` - 비밀의 ARN
- `SecretStatus` - 비밀의 상태

  가능한 상태 값에는 다음이 포함됩니다.

  - `creating` - 비밀이 생성 중입니다.
  - `active` - 비밀의 일반적 사용 및 교체가 가능합니다.
  - `rotating` - 비밀이 교체 중입니다.
  - `impaired` - 비밀을 데이터베이스 보안 인증 정보에 액세스하는 데 사용할 수 있지만 교체할 수는 없습니다. 예를 들어 권한이 변경되어 Amazon RDS가 더 이상 보안 암호 또는 보안 암호의 KMS 키에 액세스할 수 없는 경우 보안 비밀이 이 상태가 될 수 있습니다.

    암호가 이 상태인 경우 상태를 초래한 조건을 수정할 수 있습니다. 상태를 초래한 조건을 수정하면 다음 교체까지 상태가 `impaired`로 유지됩니다. 또는 테넌트 데이터베이스를 수정하여 데이터베이스 자격 증명 자동 관리를 끈 다음 테넌트 데이터베이스를 다시 수정하여 데이터베이스 보안 인

증 정보 자동 관리를 켤 수 있습니다. 테넌트 데이터베이스를 수정하려면 modify-tenant-database 명령의 `--manage-master-user-password` 옵션을 사용합니다.

- `KmsKeyId` - 비밀을 암호화하는 데 사용된 KMS 키의 ARN입니다.

특정 DB 인스턴스에 있는 테넌트 데이터베이스의 출력을 표시하려면 `--db-instance-identifier` 옵션을 지정합니다. 특정 테넌트 데이터베이스에 대한 출력을 표시하는 `--tenant-db-name` 옵션을 지정할 수도 있습니다. 이 예제는 테넌트 데이터베이스가 사용하는 보안 비밀의 출력을 보여 줍니다.

Example

```
aws rds describe-tenant-databases \
    --db-instance-identifier database-3 \
    --query "TenantDatabases[0].MasterUserSecret"
```

다음은 비밀의 샘플 출력입니다.

```
{
    "SecretArn": "arn:aws:secretsmanager:us-east-2:123456789012:secret:rds!db-ABC123",
    "SecretStatus": "active",
    "KmsKeyId": "arn:aws:kms:us-east-2:123456789012:key/aa11bb22-####-####-####-
fedcba123456"
}
```

보안 비밀 ARN이 있으면 get-secret-value Secrets Manager AWS CLI 명령을 사용하여 보안 비밀에 대한 세부 정보를 볼 수 있습니다.

이 예제는 이전 샘플 출력의 비밀에 대한 세부 정보를 보여 줍니다.

Example

대상 LinuxmacOS, 또는Unix:

```
aws secretsmanager get-secret-value \
    --secret-id 'arn:aws:secretsmanager:us-east-2:123456789012:secret:rds!db-ABC123'
```

Windows의 경우:

```
aws secretsmanager get-secret-value ^
```

```
    --secret-id 'arn:aws:secretsmanager:us-east-2:123456789012:secret:rds!db-ABC123'
```

Amazon RDS API

DescribeTenantDatabases 작업을 사용하고 DBInstanceIdentifier 파라미터를 DB 인스턴스 식별자로 설정하면 AWS Secrets Manager에서 Amazon RDS가 관리하는 보안 비밀의 ARN, 상태와 KMS 키를 볼 수 있습니다. TenantDBName 파라미터를 특정 테넌트 데이터베이스 이름으로 설정할 수도 있습니다. 비밀에 대한 세부 정보가 출력에 포함됩니다.

비밀 ARN이 있으면 GetSecretValue Secrets Manager 작업을 사용하여 비밀에 대한 세부 정보를 볼 수 있습니다.

리전 및 버전 사용 가능 여부

기능 가용성 및 해당 지원은 각 데이터베이스 엔진의 특정 버전 및 AWS 리전에 따라 다릅니다. 버전 및 Secrets Manager와 Amazon RDS 통합의 리전 가용성에 대한 자세한 내용은 Secrets Manager와 Amazon RDS 통합을 지원하는 리전 및 DB 엔진을 참조하세요.

Amazon RDS의 데이터 보호

AWS 공동 책임 모델은 Amazon Relational Database Service에서 데이터 보호에 적용됩니다. 이 모델에서 설명하는 것처럼 AWS은 모든 AWS 클라우드를 실행하는 글로벌 인프라를 보호할 책임이 있습니다. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터를 보호하려면 AWS 계정 자격 증명을 보호하고 AWS IAM Identity Center 또는 AWS Identity and Access Management(IAM)를 통해 개별 사용자 계정을 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용하세요.
- SSL/TLS를 사용하여 AWS 리소스와 통신하세요. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- AWS CloudTrail로 API 및 사용자 활동 로깅을 설정하세요. AWS 활동 캡처에 CloudTrail 추적을 사용하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- AWS 암호화 솔루션을 AWS 서비스 내의 모든 기본 보안 컨트롤과 함께 사용하세요.

- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API를 통해 AWS에 액세스할 때 FIPS 140-3 검증된 암호화 모듈이 필요한 경우, FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 Amazon RDS 또는 기타 AWS 서비스에서 콘솔, API, AWS CLI 또는 AWS SDK를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버로 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명 정보를 URL에 포함해서는 안 됩니다.

주제

- 암호화를 사용하여 데이터 보호
- 인터네트워크 트래픽 개인 정보

암호화를 사용하여 데이터 보호

데이터베이스 리소스에 대한 암호화를 활성화할 수 있습니다. 또한 DB 인스턴스 에 대한 연결도 암호화가 가능합니다.

주제

- Amazon RDS 리소스 암호화
- AWS KMS key 관리
- SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화
- SSL/TLS 인증서 교체

Amazon RDS 리소스 암호화

Amazon RDS는 Amazon RDS DB 인스턴스를 암호화할 수 있습니다. 유휴 시 암호화되는 데이터로는 DB 인스턴스에 대한 기본 스토리지, 자동 백업 파일, 읽기 전용 복제본 및 스냅샷이 포함됩니다.

Amazon RDS 암호화된 DB 인스턴스는 Amazon RDS DB 인스턴스를 호스팅하는 서버의 데이터를 업계 표준 AES-256 암호화 알고리즘을 사용하여 암호화합니다. 데이터가 암호화된 이후 Amazon

RDS가 성능에 미치는 영향을 최소화한 상태에서 데이터 액세스 및 암호 해독의 인증을 투명하게 처리합니다. 암호화를 사용하도록 데이터베이스 클라이언트 애플리케이션을 수정하지 않아도 됩니다.

> **Note**
>
> 암호화/비암호화 DB 인스턴스의 경우에는 AWS 리전 간 복제에서도 원본과 읽기 전용 복제본 사이에 전송되는 데이터가 암호화됩니다.

주제

- Amazon RDS 리소스 암호화 개요
- DB 인스턴스 암호화
- DB 인스턴스에 대해 암호화가 켜져 있는지 확인
- Amazon RDS 암호화 가용성
- 전송 중 암호화
- Amazon RDS 암호화된 DB 인스턴스의 제한

Amazon RDS 리소스 암호화 개요

Amazon RDS 암호화된 DB 인스턴스는 기본 스토리지에 대한 무단 액세스로부터 데이터의 보안을 유지해 추가 계층의 데이터 보호를 제공합니다. 클라우드에 배포된 애플리케이션의 데이터 보호를 강화하고 저장된 데이터 암호화를 위한 규정 준수 요구 사항을 만족하기 위해 Amazon RDS 암호화를 사용할 수 있습니다. Amazon RDS 암호화된 DB 인스턴스의 경우 모든 로그, 백업 및 스냅샷이 암호화됩니다. 암호화의 가용성 및 제한에 대한 자세한 내용은 Amazon RDS 암호화 가용성 및 Amazon RDS 암호화된 DB 인스턴스의 제한 섹션을 참조하세요.

Amazon RDS는 AWS Key Management Service 키를 사용하여 이러한 리소스를 암호화합니다. AWS KMS는 안전하고 가용성이 높은 하드웨어와 소프트웨어를 결합하여 클라우드에 맞게 규모가 조정된 키 관리 시스템을 제공합니다. AWS 관리형 키을(를) 사용하거나 고객 관리형 키를 생성할 수 있습니다.

암호화된 DB 인스턴스를 생성할 때 Amazon RDS에 사용할 고객 관리형 키 또는 AWS 관리형 키을(를) 선택하여 DB 인스턴스를 암호화할 수 있습니다. 고객 관리형 키의 키 식별자를 지정하지 않으면 Amazon RDS는 새 DB 인스턴스에 AWS 관리형 키을(를) 사용합니다. Amazon RDS는 AWS 계정에 대해 Amazon RDS용 AWS 관리형 키을(를) 생성합니다. AWS 계정에 각 AWS 리전의 Amazon RDS에 대한 다른 AWS 관리형 키이(가) 있습니다.

Amazon RDS 리소스 암호화 및 복호화에 사용되는 고객 관리형 키를 관리하려면 AWS Key Management Service(AWS KMS)를 사용합니다.

AWS KMS를 사용하면 고객 관리형 키를 생성하고 이 키의 사용을 제어하는 정책을 정의할 수 있습니다. AWS KMS는 CloudTrail을 지원하므로 KMS 키 사용을 감사하여 고객 관리형 키가 적절하게 사용되고 있는지 확인할 수 있습니다. 고객 관리형 키는 Amazon Aurora를 비롯해 Amazon S3, Amazon EBS, Amazon Redshift 등 지원되는 AWS 서비스에서 사용할 수 있습니다. AWS KMS와 통합되는 서비스 목록은 AWS 서비스 통합을 참조하세요. KMS 키 사용에 대한 몇 가지 고려 사항:

- 암호화된 DB 인스턴스를 생성한 후에는 해당 DB 인스턴스에서 사용된 KMS 키를 변경할 수 없습니다. 따라서 암호화된 DB 인스턴스를 생성하기 전에 KMS 키 요구 사항을 결정해야 합니다.

  DB 인스턴스의 암호화 키를 변경해야 하는 경우 인스턴스의 수동 스냅샷을 생성하고 스냅샷을 복사하는 동안 암호화를 활성화합니다. 자세한 내용은 re:Post 지식 센터 문서를 참조하세요.
- 암호화된 스냅샷을 복사하는 경우 소스 스냅샷을 암호화하는 데 사용된 것과 다른 KMS 키를 사용하여 대상 스냅샷을 암호화할 수 있습니다.
- 동일한 AWS 리전에 있는 경우, Amazon RDS 암호화된 인스턴스의 읽기 전용 복제본은 기본 DB 인스턴스와 동일한 KMS 키를 사용하여 암호화되어야 합니다.
- 기본 DB 인스턴스와 읽기 전용 복제본이 서로 다른 AWS 리전에 있을 경우 해당 AWS 리전의 KMS 키를 사용하여 읽기 전용 복제본을 암호화합니다.
- 스냅샷을 공유한 AWS 계정의 AWS 관리형 키을(를) 사용하여 암호화된 스냅샷은 공유할 수 없습니다.
- 또한 Amazon RDS에서는 TDE(Transparent Data Encryption)를 사용하여 Oracle 또는 SQL Server DB 인스턴스 암호화를 지원합니다. TDE를 RDS 저장 데이터 암호화와 함께 사용하면 데이터베이스의 성능에 약간의 영향을 미칠 수 있지만, TDE를 RDS 저장 데이터 암호화와 함께 사용할 수 있습니다. 암호화 방법별로 다른 키를 관리해야 합니다. TDE에 대한 자세한 내용은 Oracle Transparent Data Encryption 또는 SQL Server에서 TDE(투명한 데이터 암호화) 지원 단원을 참조하십시오.

> **Important**
>
> Amazon RDS는 KMS 키를 비활성화하면 DB 인스턴스의 KMS 키에 대한 액세스 권한을 잃습니다. KMS 키에 대한 액세스 권한을 잃으면 백업이 활성화된 인스턴스에서 감지 후 2시간이 지났을 때 암호화된 DB 인스턴스가 `inaccessible-encryption-credentials-recoverable` 상태가 됩니다. DB 인스턴스는 7일 동안 이 상태로 유지되며, 이 기간 동안 인스턴스가 중지됩니다. 이 기간 동안 DB 인스턴스에 대한 API 호출은 성공하지 못할 수 있습니다. DB 인스턴스를 복구하려면 KMS 키를 활성화하고 이 DB 인스턴스를 다시 시작하세요.

AWS Management Console, AWS CLI 또는 RDS API에서 KMS 키를 활성화합니다. AWS CLI 명령 start-db-instance 또는 AWS Management Console을 사용하여 DB 인스턴스를 다시 시작합니다.
`inaccessible-encryption-credentials-recoverable` 상태는 중지할 수 있는 DB 인스턴스에만 적용됩니다. 읽기 전용 복제본 및 읽기 전용 복제본이 있는 인스턴스와 같이 중지할 수 없는 인스턴스에는 이 복구 가능 상태를 적용할 수 없습니다. 자세한 내용은 DB 인스턴스 중지 제한 사항 섹션을 참조하세요.
7일 이내에 DB 인스턴스가 복구되지 않으면 터미널 `inaccessible-encryption-credentials` 상태가 됩니다. 이 상태에서는 DB 인스턴스를 더 이상 사용할 수 없으며 백업에서 DB 인스턴스를 복원할 수만 있습니다. 데이터베이스에서 암호화된 데이터가 손실되지 않도록 보호하려면 암호화된 DB 인스턴스에 대해 항상 백업을 활성화하는 것이 좋습니다.
DB 인스턴스를 생성하는 동안 Amazon RDS는 호출 주체가 KMS 키에 대한 액세스 권한이 있는지 확인하고 DB 인스턴스의 전체 수명 기간 동안 사용하는 KMS 키에서 부여를 생성합니다. MS 키에 대한 호출 주체의 액세스 권한을 취소해도 실행 중인 데이터베이스에는 영향을 미치지 않습니다. 스냅샷을 다른 계정으로 복사하는 등 계정 간 시나리오에서 KMS 키를 사용하는 경우, KMS 키를 다른 계정과 공유해야 합니다. 다른 KMS 키를 지정하지 않고 스냅샷에서 DB 인스턴스를 만들면 새 인스턴스는 소스 계정의 KMS 키를 사용합니다. DB 인스턴스를 생성한 후 키에 대한 액세스 권한을 취소해도 인스턴스에는 영향을 미치지 않습니다. 그러나 키를 비활성화하면 해당 키로 암호화된 모든 DB 인스턴스에 영향을 미칩니다. 이를 방지하려면 스냅샷 복사 작업 중에 다른 키를 지정하세요.
백업이 비활성화된 DB 인스턴스는 인스턴스 수정 또는 복구 중에 볼륨이 호스트에서 분리될 때까지 계속 사용할 수 있습니다. RDS는 해당하는 경우 인스턴스를 `inaccessible-encryption-credentials-recoverable` 상태 또는 `inaccessible-encryption-credentials` 상태로 이동합니다.

KMS 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 AWS KMS keys 섹션 및 AWS KMS key 관리 섹션을 참조하세요.

DB 인스턴스 암호화

새로운 DB 인스턴스를 암호화하려면 Amazon RDS 콘솔에서 암호화 활성화(Enable encryption)를 선택합니다. DB 인스턴스 생성에 대한 자세한 내용은 Amazon RDS DB 인스턴스 생성 단원을 참조하십시오.

create-db-instance AWS CLI 명령을 사용하여 암호화된 DB 인스턴스를 생성할 경우 `--storage-encrypted` 파라미터를 설정합니다. CreateDBInstance API 작업을 사용할 경우 `StorageEncrypted` 파라미터를 true로 설정하십시오.

AWS CLI `create-db-instance` 명령을 사용하여 고객 관리형 키로 암호화된 DB 인스턴스를 생성하는 경우 `--kms-key-id` 파라미터를 KMS 키의 키 식별자로 설정합니다. Amazon RDS API `CreateDBInstance` 작업을 사용하는 경우 `KmsKeyId` 파라미터를 KMS 키의 키 식별자로 설정합니다. 다른 AWS 계정에서 고객 관리형 키를 사용하려면 키 ARN 또는 별칭 ARN을 지정합니다.

DB 인스턴스에 대해 암호화가 켜져 있는지 확인

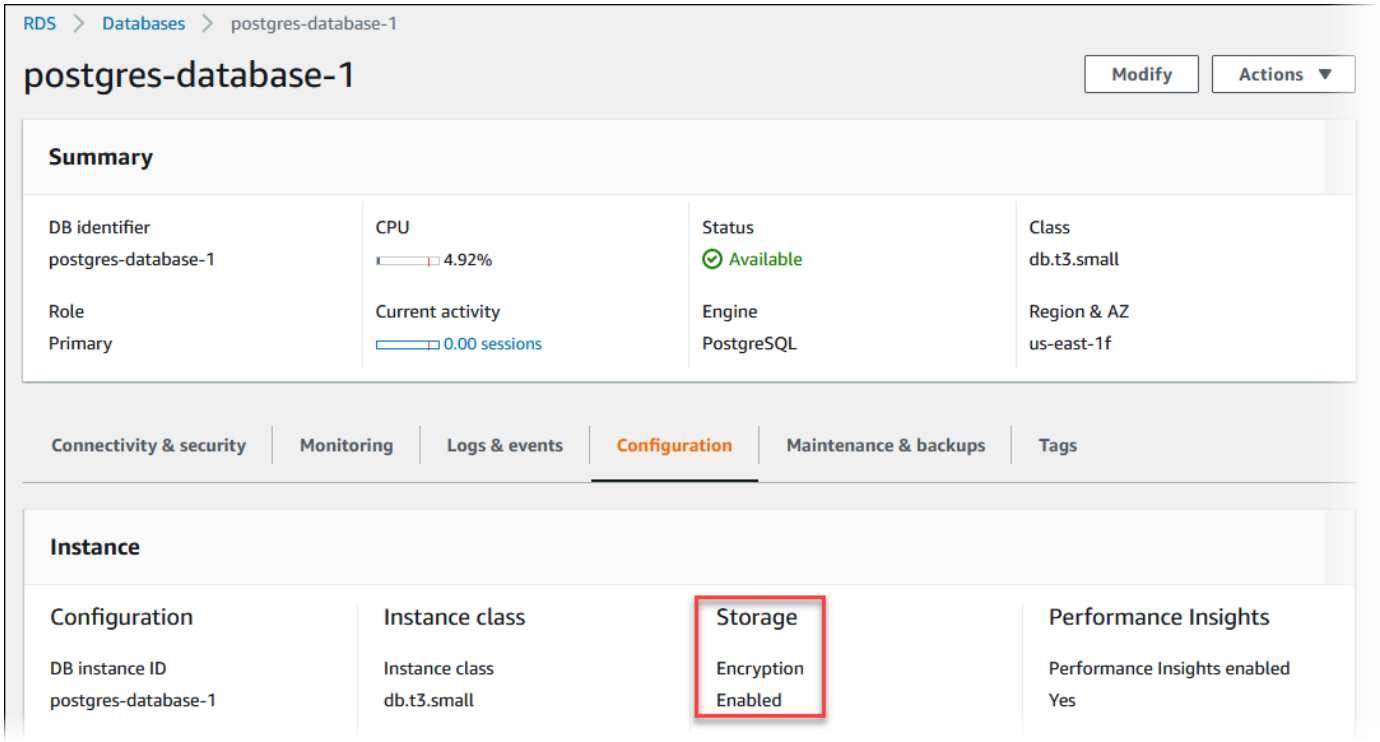
AWS Management Console, AWS CLI 또는 RDS API를 사용하여 DB 인스턴스에 대해 저장 데이터 암호화가 켜져 있는지 확인합니다.

콘솔

DB 인스턴스에 대해 저장 데이터 암호화가 켜져 있는지 확인하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
3. 세부 정보를 보기 위해 확인하려는 DB 인스턴스의 이름을 선택합니다.
4. 구성(Configuration) 탭을 선택하고 스토리지(Storage)에서 암호화(Encryption) 값을 확인합니다.

   그러면 활성화(Enabled) 또는 비활성화(Not enabled) 중 하나가 표시됩니다.



AWS CLI

AWS CLI를 사용하여 DB 인스턴스에 대해 저장 데이터 암호화가 켜져 있는지 확인하려면 다음 옵션을 사용하여 describe-db-instances 명령을 호출합니다.

- `--db-instance-identifier` – DB 인스턴스의 이름입니다.

다음 예에서는 쿼리를 사용하여 `mydb` DB 인스턴스의 저장 데이터 암호화에 대해 `TRUE` 또는 `FALSE` 중 하나를 반환합니다.

Example

```
aws rds describe-db-instances --db-instance-identifier mydb --query "*[].
{StorageEncrypted:StorageEncrypted}" --output text
```

RDS API

Amazon RDS API를 사용하여 DB 인스턴스에 대해 저장 데이터 암호화가 켜져 있는지 확인하려면 다음 파라미터를 사용하여 DescribeDBInstances를 호출합니다.

- `DBInstanceIdentifier` – DB 인스턴스의 이름입니다.

Amazon RDS 암호화 가용성

Amazon RDS 암호화는 현재 모든 데이터베이스 엔진과 스토리지 유형에 사용할 수 있습니다.

Amazon RDS 암호화는 대부분의 DB 인스턴스 클래스에서 사용 가능합니다. 다음 테이블에는 Amazon RDS 암호화를 지원하지 않는 DB 인스턴스 클래스가 열거되어 있습니다.

인스턴스 유형	인스턴스 클래스
범용(M1)	db.m1.small db.m1.medium db.m1.large db.m1.xlarge
메모리 최적화(M2)	db.m2.xlarge

인스턴스 유형	인스턴스 클래스
	db.m2.2xlarge db.m2.4xlarge
버스트 가능(T2)	db.t2.micro

전송 중 암호화

물리적 계층에서의 암호화

AWS 글로벌 네트워크를 통해 AWS 리전으로 이동하는 모든 데이터는 AWS 보안 시설에서 나가기 전에 물리적 계층에서 자동으로 암호화됩니다. AZ 간에 전송되는 모든 트래픽은 암호화됩니다. 이 섹션에 나열된 암호화 계층을 비롯한 추가 암호화 계층은 추가적인 보호 기능을 제공할 수 있습니다.

Amazon VPC 피어링 및 전송 게이트웨이 교차 리전 피어링에서 제공하는 암호화

Amazon VPC 및 Transit Gateway 피어링을 사용하는 모든 리전 간 트래픽은 리전을 종료하면 자동으로 대량 암호화됩니다. AWS 보안 시설에서 나가기 전에 모든 트래픽에 대한 추가 암호화 계층이 물리적 계층에서 자동으로 제공됩니다.

인스턴스 간 암호화

AWS는 모든 유형의 DB 인스턴스 간 보안 프라이빗 연결을 제공합니다. 또한 일부 인스턴스 유형은 기본 Nitro 시스템 하드웨어의 오프로드 기능을 사용하여 인스턴스 간 전송 중 트래픽을 자동으로 암호화합니다. 이 암호화는 256비트 암호화와 함께 관련 데이터로 인증된 암호화(AEAD) 알고리즘을 사용합니다. 네트워크 성능에는 영향을 미치지 않습니다. 인스턴스 간에 이러한 전송 중 트래픽 암호화를 추가로 지원하려면 다음 요구 사항을 충족해야 합니다.

- 이러한 인스턴스는 다음 인스턴스 유형을 사용합니다.
  - 범용: M6i, M6id, M6in, M6idn, M7g
  - 메모리 최적화: R6i, R6id, R6in, R6idn, R7g, X2idn, X2iedn, X2iezn
- 인스턴스가 동일한 AWS 리전에 있습니다.
- 인스턴스가 동일한 VPC 또는 피어링된 VPC에 있으며, 트래픽이 로드 밸런서나 전송 게이트웨이 같은 가상 네트워크 디바이스 또는 서비스를 통과하지 않습니다.

Amazon RDS 암호화된 DB 인스턴스의 제한

Amazon RDS 암호화된 DB 인스턴스에는 다음과 같은 제한이 있습니다.

- Amazon RDS DB 인스턴스의 암호화는 인스턴스를 생성할 때에만 가능하며 DB 인스턴스가 생성된 후에는 불가능합니다.

  다만 암호화되지 않은 스냅샷의 사본을 암호화할 수 있기 때문에 암호화되지 않은 DB 인스턴스에 실질적으로 암호화를 추가할 수 있습니다. 즉, DB 인스턴스의 스냅샷을 만든 다음 해당 스냅샷의 암호화된 사본을 만들 수 있습니다. 그런 다음 암호화된 스냅샷에서 DB 인스턴스를 복구할 수 있고, 원본 DB 인스턴스의 암호화된 사본이 생깁니다. 자세한 내용은 Amazon RDS용 DB 스냅샷 복사 단원을 참조하십시오.
- 암호화된 DB 인스턴스의 암호화를 비활성화할 수 없습니다.
- 암호화되지 않은 DB 인스턴스의 암호화된 스냅샷은 생성할 수 없습니다.
- 암호화된 DB 인스턴스의 스냅샷은 DB 인스턴스와 동일한 KMS 키를 사용하여 암호화해야 합니다.
- 암호화되지 않은 DB 인스턴스의 암호화된 읽기 전용 복제본이나 암호화된 DB 인스턴스의 암호화되지 않은 읽기 전용 복제본은 보유할 수 없습니다.
- 암호화된 읽기 전용 복제본이 소스 DB 인스턴스와 동일한 AWS 리전에 있는 경우 해당 인스턴스와 동일한 KMS 키를 사용하여 암호화해야 합니다.
- 암호화되지 않은 백업 또는 스냅샷을 암호화된 DB 인스턴스로 복원할 수 없습니다.
- 암호화된 스냅샷을 한 AWS 리전에서 다른 리전으로 복사하려면 대상 AWS 리전에 KMS 키를 지정해야 합니다. 이는 KMS 키가 생성된 AWS 리전에만 해당하기 때문입니다.

  소스 스냅샷은 복사 프로세스 전체에서 암호화를 유지합니다. Amazon RDS는 봉투 암호화를 사용하여 복사 프로세스 중에 데이터를 보호합니다. 봉투 암호화에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서에서 봉투 암호화를 참조하세요.
- 암호화된 DB 인스턴스의 암호화를 해제할 수 없습니다. 하지만 암호화된 DB 인스턴스에서 데이터를 내보내고 암호화되지 않은 DB 인스턴스로 해당 데이터를 가져올 수 있습니다.

AWS KMS key 관리

Amazon RDS 는 AWS Key Management Service(AWS KMS)와 자동으로 통합하여 키 관리를 수행합니다. Amazon RDS 는 봉투 암호화를 사용합니다. 봉투 암호화에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서에서 봉투 암호화를 참조하세요.

두 가지 유형의 AWS KMS 키를 사용하여 DB 인스턴스 를 암호화할 수 있습니다.

- KMS 키를 완전히 제어하기 위해서는 고객 관리형 키를 생성해야 합니다. 고객 관리형 키에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 고객 관리형 키를 참조하세요.
- AWS 관리형 키 은(는) AWS KMS와 통합된 AWS 서비스가 고객의 계정에서 고객 대신 생성, 관리 및 사용하는 KMS 키입니다. 기본적으로 RDS AWS 관리형 키(`aws/rds`)는 암호화에 사용됩니다. RDS AWS 관리형 키는 관리, 교체 또는 삭제할 수 없습니다. AWS 관리형 키에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 AWS 관리형 키를 참조하십시오.

Amazon RDS 암호화된 DB 인스턴스 에 사용되는 KMS 키를 관리하려면 AWS KMS 콘솔, AWS CLI 또는 AWS KMS API에서 AWS Key Management Service(AWS KMS)를 사용합니다. AWS 관리형 또는 고객 관리형 키로 수행한 모든 작업의 감사 로그를 보려면 AWS CloudTrail을 사용합니다. 키 교체에 대한 자세한 내용은 AWS KMS 키 교체를 참조하세요.

고객 관리형 키의 사용 권한 부여

RDS 가 암호화 작업에 고객 관리형 키를 사용하는 경우 RDS 리소스를 만들거나 변경하는 사용자를 대신해 작업합니다.

고객 관리형 키를 사용하여 RDS 리소스를 만들려면 고객 관리형 키에서 다음 작업을 직접적으로 호출할 수 있는 권한이 사용자에게 있어야 합니다.

- `kms:CreateGrant`
- `kms:DescribeKey`

키 정책에서 허용하는 경우 키 정책 또는 IAM 정책에서 이러한 필수 권한을 지정할 수 있습니다.

> **Important**
>
> Amazon RDS와 같은 관리형 서비스에서 AWS KMS 키 정책의 모든 리소스(*)에 대해 명시적 거부 문을 사용하는 경우 리소스 소유 계정을 허용하는 조건을 지정해야 합니다. 거부 규칙에 IAM 사용자에 대한 예외가 포함되어 있더라도 이 조건이 없다면 작업이 실패할 수 있습니다.

> **Tip**
>
> 최소 권한의 원칙을 따르려면 `kms:CreateGrant`에 대한 전체 액세스 권한을 허용하지 마세요. 대신에 kms:ViaService 조건 키를 사용하여 AWS 서비스에서 사용자를 대신하여 권한이 부여되는 경우에만 사용자가 KMS 키에 대한 권한을 부여할 수 있도록 허용합니다.

다양한 방법으로 IAM 정책을 더 엄격하게 설정할 수 있습니다. 예를 들어 RDS 에서 생성된 요청에 대해서만 고객 관리형 키를 사용할 수 있도록 허용하고 싶다면 `rds.<region>.amazonaws.com` 값과 함께 kms:ViaService 조건 키를 사용합니다. 또한 암호화 작업에 대한 고객 관리형 키 사용 조건으로 Amazon RDS 암호화 컨텍스트의 키 또는 값을 사용할 수도 있습니다.

자세한 내용을 알아보려면 AWS Key Management Service 개발자 안내서의 다른 계정의 사용자가 KMS 키를 사용하도록 허용 및 AWS KMS의 키 정책을 참조하세요.

Amazon RDS 암호화 컨텍스트

RDS 가 사용자의 KMS 키를 사용하거나 Amazon EBS가 RDS 를 대신하여 KMS 키를 사용하는 경우, 서비스가 암호화 컨텍스트를 지정합니다. 암호화 컨텍스트는 AWS KMS가 데이터 무결성을 보장하기 위해 사용하는 추가 인증 데이터(AAD)입니다. 암호화 작업에 대해 암호화 컨텍스트가 지정되면 서비스가 암호화 해제 작업에 대해 동일한 암호화 컨텍스트를 지정해야 합니다. 그렇지 않으면 암호화 해제가 실패합니다. 암호화 컨텍스트는 AWS CloudTrail 로그에도 기록되어, 해당 KMS 키가 사용된 이유를 이해하는 데 도움을 줍니다. CloudTrail 로그에 CMK 사용을 설명하는 여러 항목이 포함될 수 있지만, 각 로그 항목의 암호화 컨텍스트는 특히 해당 KMS 키를 사용한 이유를 파악하는 데 도움이 될 수 있습니다.

최소한, 다음 JSON 형식 예에서 보듯이 Amazon RDS 는 항상 암호화 컨텍스트에 DB 인스턴스 ID를 사용합니다.

```
{ "aws:rds:db-id": "db-CQYSMDPBRZ7BPMH7Y3RTDG5QY" }
```

이 암호화 컨텍스트는 KMS 키가 사용된 DB 인스턴스 를 식별하는 데 도움이 될 수 있습니다.

KMS 키가 특정 DB 인스턴스 와 특정 Amazon EBS 볼륨에 사용되는 경우, 다음 JSON 형식 예에서 보듯이 암호화 컨텍스트에 DB 인스턴스 ID와 Amazon EBS 볼륨 ID가 모두 사용됩니다.

```
{
  "aws:rds:db-id": "db-BRG7VYS3SVIFQW7234EJQOM5RQ",
  "aws:ebs:id": "vol-ad8c6542"
}
```

SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화

애플리케이션에서 보안 소켓 계층(SSL) 또는 전송 계층 보안(TLS)을 사용하여 Db2, MariaDB, Microsoft SQL Server, MySQL, Oracle 또는 PostgreSQL을 실행하는 데이터베이스에 대한 연결을 암호화할 수 있습니다.

SSL/TLS 연결은 클라이언트와 DB 인스턴스 또는 클러스터 사이에 전송되는 데이터를 암호화하여 하나의 보안 계층을 제공합니다. 필요에 따라 SSL/TLS 연결에서는 데이터베이스에 설치된 서버 인증서를 검증하여 서버 ID 확인을 수행할 수 있습니다. 서버 ID 확인을 요구하려면 다음의 일반적인 절차를 따르세요.

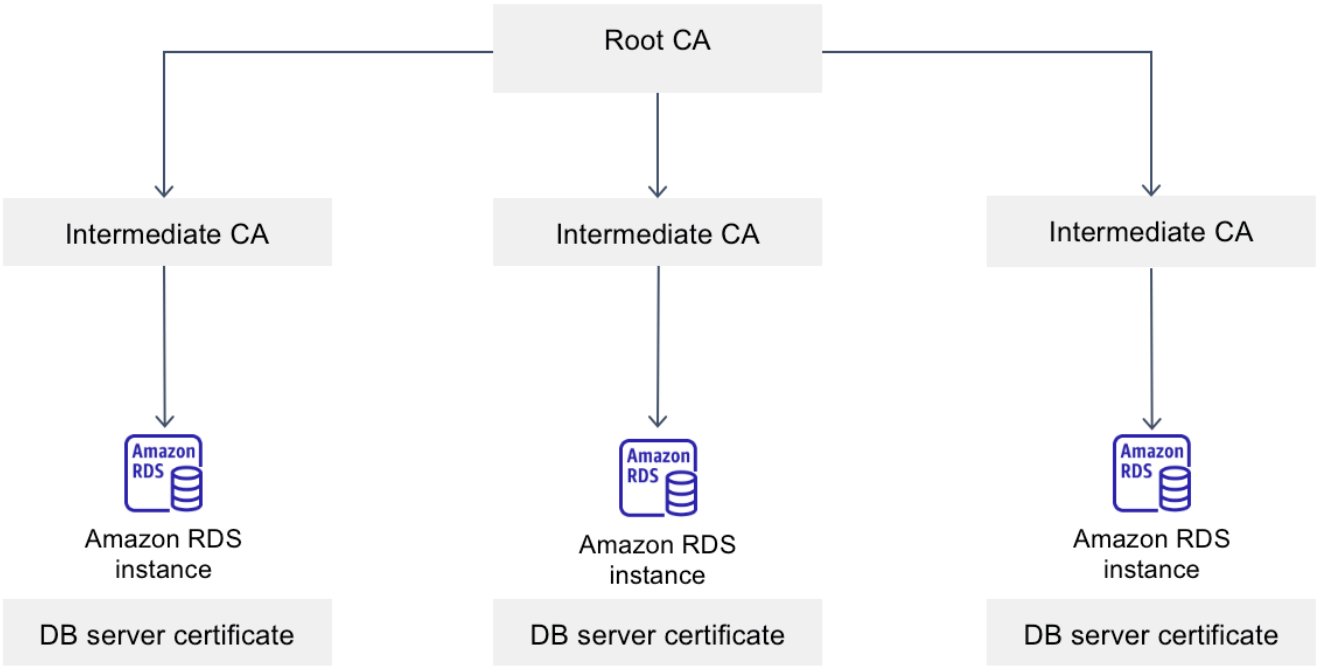
1. 데이터베이스의 DB 서버 인증서에 서명하는 인증 기관(CA)을 선택합니다. 인증 기관에 관한 자세한 내용은 인증 기관 단원을 참조하세요.
2. 데이터베이스에 연결할 때 사용할 인증서 번들을 다운로드합니다. 인증서 번들을 다운로드하려면 AWS 리전별 인증서 번들 단원을 참조하세요.

   > **Note**
   >
   > 모든 인증서는 SSL/TLS 연결을 통한 다운로드에만 사용 가능합니다.

3. SSL/TLS 연결을 구현하는 DB 엔진 프로세스를 사용하여 데이터베이스에 연결합니다. 각 DB 엔진에는 SSL/TLS를 구현하기 위한 고유한 프로세스가 있습니다. 데이터베이스에서 SSL/TLS를 구현하는 방법을 알아보려면 DB 엔진에 해당하는 아래 링크로 이동하세요.
   - Amazon RDS for Db2 DB 인스턴스에 SSL/TLS 사용
   - Amazon RDS의 MariaDB DB 인스턴스에 대한 SSL/TLS 지원
   - ???
   - Amazon RDS의 MySQL DB 인스턴스에 대한 SSL/TLS 지원
   - RDS for Oracle DB 인스턴스에 SSL 사용
   - PostgreSQL DB 인스턴스와 함께 SSL 사용

인증 기관

인증 기관(CA)은 인증서 체인의 맨 위에 있는 루트 CA를 식별하는 인증서입니다. CA는 각 DB 인스턴스에 설치된 DB 서버 인증서에 서명합니다. DB 서버 인증서는 DB 인스턴스를 신뢰할 수 있는 서버로 식별합니다.



Amazon RDS는 데이터베이스의 DB 서버 인증서에 서명할 수 있는 다음 CA를 제공합니다.

인증 기관(CA)	설명	일반 이름(CN)
rds-ca-2019	RSA 2048 프라이빗 키 알고리즘과 SHA256 서명 알고리즘을 갖춘 인증 기관을 사용합니다. 이 CA는 2024년에 만료되며 자동 서버 인증서 교체를 지원하지 않습니다. 이 CA를 사용 중이고 동일한 표준을 유지하려면 rds-ca-rsa2048-g1 CA로 전환하는 것이 좋습니다.	Amazon RDS *region-id entifier* CA 2019
rds-ca-rsa2048-g1	대부분의 AWS 리전에서 RSA 2048 프라이빗 키 알고리즘과 SHA256 서명 알고리즘을 갖춘 인증 기관을 사용합니다. AWS GovCloud (US) Regions에서 이 CA는 RSA 2048 프라이빗 키 알고리즘과 SHA384 서명 알고리즘을 갖춘 인증 기관을 사용합니다.	Amazon RDS *region-id entifier* RSA2048 G1

인증 기관(CA)	설명	일반 이름(CN)
	이 CA의 유효 기간은 rds-ca-2019 CA보다 깁니다. 이 CA는 자동 서버 인증서 교체를 지원합니다.	
rds-ca-rsa4096-g1	RSA 4096 프라이빗 키 알고리즘과 SHA384 서명 알고리즘을 갖춘 인증 기관을 사용합니다. 이 CA는 자동 서버 인증서 교체를 지원합니다.	Amazon RDS *region-id entifier* RSA4096 G1
rds-ca-ecc384-g1	ECC 384 프라이빗 키 알고리즘과 SHA384 서명 알고리즘을 갖춘 인증 기관을 사용합니다. 이 CA는 자동 서버 인증서 교체를 지원합니다.	Amazon RDS *region-id entifier* ECC384 G1

> **Note**
>
> AWS CLI를 사용하는 경우 describe-certificates를 사용하여 위에 나열된 인증 기관의 유효성을 확인할 수 있습니다.

이러한 CA 인증서는 지역 및 글로벌 인증서 번들에 포함되어 있습니다. rds-ca-rsa2048-g1, rds-ca-rsa4096-g1, rds-ca-ecc384-g1 CA를 데이터베이스에 사용하면 RDS가 데이터베이스에서 DB 서버 인증서를 관리합니다. RDS는 DB 서버 인증서가 만료되기 전에 자동으로 교체합니다.

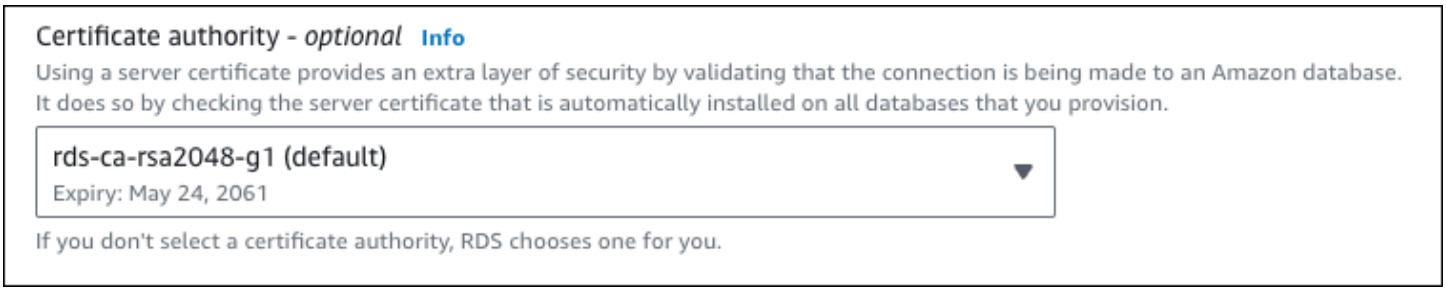
데이터베이스의 CA 설정

다음 작업을 수행할 때 데이터베이스의 CA를 설정할 수 있습니다.

- DB 인스턴스 또는 다중 AZ DB 인스턴스 생성 - DB 인스턴스 또는 클러스터를 생성할 때 CA를 설정할 수 있습니다. 지침은 the section called “DB 인스턴스 생성” 또는 the section called “다중 AZ DB 클러스터 생성” 섹션을 참조하십시오.
- DB 인스턴스 또는 다중 AZ DB 클러스터 수정 - DB 인스턴스 또는 클러스터를 수정하여 해당 CA를 설정할 수 있습니다. 지침은 the section called “DB 인스턴스 수정” 또는 the section called “다중 AZ DB 클러스터 수정” 섹션을 참조하십시오.

> **Note**
>
> 기본 CA는 rds-ca-rsa2048-g1로 설정되어 있습니다. modify-certificates 명령을 사용하여 AWS 계정에 기본 CA를 재정의할 수 있습니다.

사용 가능한 CA는 DB 엔진 및 DB 엔진 버전에 따라 다릅니다. AWS Management Console을 사용하는 경우 다음 이미지에 표시된 것처럼 인증 기관 설정을 사용하여 CA를 선택할 수 있습니다.



콘솔에는 DB 엔진 및 DB 엔진 버전에 사용할 수 있는 CA만 표시됩니다. AWS CLI를 사용하는 경우 create-db-instance 또는 modify-db-instance 명령을 사용하여 DB 인스턴스의 CA를 설정할 수 있습니다. create-db-cluster 또는 modify-db-cluster 명령을 사용하여 다중 AZ DB 클러스터의 CA를 설정할 수 있습니다.

AWS CLI를 사용하는 경우 describe-certificate 명령을 사용하여 계정에 사용할 수 있는 CA를 확인할 수 있습니다. 이 명령은 출력의 `ValidTill`에 각 CA의 만료 날짜도 표시합니다. describe-db-engine-version 명령을 사용하여 특정 DB 엔진 및 DB 엔진 버전에 사용할 수 있는 CA를 찾을 수 있습니다.

다음 예제는 기본 RDS for PostgreSQL DB 엔진 버전에 사용할 수 있는 CA를 보여 줍니다.

```
aws rds describe-db-engine-versions --default-only --engine postgres
```

다음과 같은 출력이 표시됩니다. 사용 가능한 CA는 `SupportedCACertificateIdentifiers`에 나열되어 있습니다. 출력은 `SupportsCertificateRotationWithoutRestart`에서 DB 엔진 버전이 다시 시작하지 않고 인증서를 교체하는 기능을 지원하는지 여부도 표시합니다.

```
{
    "DBEngineVersions": [
        {
            "Engine": "postgres",
            "MajorEngineVersion": "13",
            "EngineVersion": "13.4",
```

```
            "DBParameterGroupFamily": "postgres13",
            "DBEngineDescription": "PostgreSQL",
            "DBEngineVersionDescription": "PostgreSQL 13.4-R1",
            "ValidUpgradeTarget": [],
            "SupportsLogExportsToCloudwatchLogs": false,
            "SupportsReadReplica": true,
            "SupportedFeatureNames": [
                "Lambda"
            ],
            "Status": "available",
            "SupportsParallelQuery": false,
            "SupportsGlobalDatabases": false,
            "SupportsBabelfish": false,
            "SupportsCertificateRotationWithoutRestart": true,
            "SupportedCACertificateIdentifiers": [
                "rds-ca-2019",
                "rds-ca-rsa2048-g1",
                "rds-ca-ecc384-g1",
                "rds-ca-rsa4096-g1"
            ]
        }
    ]
}
```

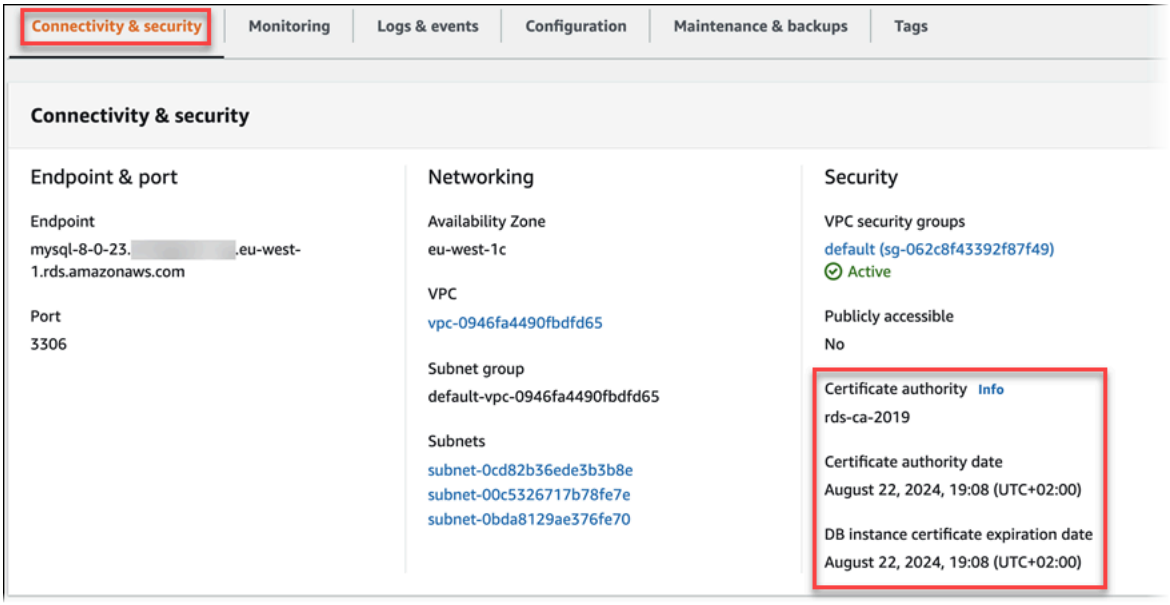
DB 서버 인증서 유효 기간

DB 서버 인증서의 유효 기간은 DB 엔진 및 DB 엔진 버전에 따라 다릅니다. DB 엔진 버전에서 재시작 없이 인증서를 교체하는 기능을 지원하는 경우 DB 서버 인증서의 유효 기간은 1년입니다. 그렇지 않으면 유효 기간은 3년입니다.

DB 서버 인증서 교체에 대한 자세한 내용은 자동 서버 인증서 교체 단원을 참조하세요.

DB 인스턴스의 CA 보기

콘솔의 연결 및 보안 탭에서 다음 이미지와 같이 데이터베이스의 CA에 대한 세부 정보를 볼 수 있습니다.



AWS CLI를 사용 중인 경우 describe-db-instances 명령을 사용하여 DB 인스턴스의 CA에 대한 세부 정보를 확인할 수 있습니다. describe-db-cluster 명령을 사용하여 다중 AZ DB 클러스터의 CA에 대한 세부 정보를 볼 수 있습니다.

Amazon RDS용 인증서 번들 다운로드

SSL 또는 TLS로 데이터베이스에 연결하는 경우 데이터베이스 인스턴스에 Amazon RDS의 신뢰 인증서가 필요합니다. 다음 표에서 적절한 링크를 선택하여 데이터베이스를 호스팅하는 AWS 리전에 해당하는 번들을 다운로드하세요.

AWS 리전별 인증서 번들

모든 AWS 리전 리전과 GovCloud(US) 리전의 인증서 번들에는 다음과 같은 인증서가 포함되어 있습니다.

- `rds-ca-2019` 중간 및 루트 인증서.
- `rds-ca-rsa2048-g1`, `rds-ca-rsa4096-g1` 및 `rds-ca-ecc384-g1` 루트 CA 인증서. 애플리케이션 트러스트 스토어에는 루트 CA 인증서만 등록하면 됩니다.

> **Note**
>
> Amazon RDS Proxy 에서는 AWS Certificate Manager(ACM)의 인증서를 사용합니다. RDS 프록시를 사용하는 경우 Amazon RDS 인증서를 다운로드하거나 RDS 프록시 연결을 사용하는

애플리케이션을 업데이트할 필요가 없습니다. 자세한 내용은 RDS Proxy에서 TLS/SSL 사용 단원을 참조하십시오.

AWS 리전에 대한 인증서 번들을 다운로드하려면 다음 표에서 해당 데이터베이스를 호스팅하는 AWS 리전에 대한 링크를 선택하세요.

AWS 리전	인증서 번들(PEM)	인증서 번들(PKCS7)
모든 상업용 AWS 리전	global-bundle.pem	global-bundle.p7b
미국 동부(버지니아 북부)	us-east-1-bundle.pem	us-east-1-bundle.p7b
미국 동부(오하이오)	us-east-2-bundle.pem	us-east-2-bundle.p7b
미국 서부(캘리포니아 북부)	us-west-1-bundle.pem	us-west-1-bundle.p7b
미국 서부(오리건)	us-west-2-bundle.pem	us-west-2-bundle.p7b
Africa (Cape Town)	af-south-1-bundle.pem	af-south-1-bundle.p7b
Asia Pacific (Hong Kong)	ap-east-1-bundle.pem	ap-east-1-bundle.p7b
아시아 태평양(하이데라바드)	ap-south-2-bundle.pem	ap-south-2-bundle.p7b
아시아 태평양(자카르타)	ap-southeast-3-bundle.pem	ap-southeast-3-bundle.p7b
아시아 태평양(멜버른)	ap-southeast-4-bundle.pem	ap-southeast-4-bundle.p7b
아시아 태평양(뭄바이)	ap-south-1-bundle.pem	ap-south-1-bundle.p7b
Asia Pacific (Osaka)	ap-northeast-3-bundle.pem	ap-northeast-3-bundle.p7b
아시아 태평양(도쿄)	ap-northeast-1-bundle.pem	ap-northeast-1-bundle.p7b
Asia Pacific (Seoul)	ap-northeast-2-bundle.pem	ap-northeast-2-bundle.p7b
아시아 태평양(싱가포르)	ap-southeast-1-bundle.pem	ap-southeast-1-bundle.p7b
아시아 태평양(시드니)	ap-southeast-2-bundle.pem	ap-southeast-2-bundle.p7b

AWS 리전	인증서 번들(PEM)	인증서 번들(PKCS7)
Canada (Central)	ca-central-1-bundle.pem	ca-central-1-bundle.p7b
캐나다 서부(캘거리)	ca-west-1-bundle.pem	ca-west-1-bundle.p7b
유럽(프랑크푸르트)	eu-central-1-bundle.pem	eu-central-1-bundle.p7b
유럽(아일랜드)	eu-west-1-bundle.pem	eu-west-1-bundle.p7b
Europe (London)	eu-west-2-bundle.pem	eu-west-2-bundle.p7b
Europe (Milan)	eu-south-1-bundle.pem	eu-south-1-bundle.p7b
Europe (Paris)	eu-west-3-bundle.pem	eu-west-3-bundle.p7b
유럽(스페인)	eu-south-2-bundle.pem	eu-south-2-bundle.p7b
유럽(스톡홀름)	eu-north-1-bundle.pem	eu-north-1-bundle.p7b
유럽(취리히)	eu-central-2-bundle.pem	eu-central-2-bundle.p7b
이스라엘(텔아비브)	il-central-1-bundle.pem	il-central-1-bundle.p7b
Middle East (Bahrain)	me-south-1-bundle.pem	me-south-1-bundle.p7b
중동(UAE)	me-central-1-bundle.pem	me-central-1-bundle.p7b
남아메리카(상파울루)	sa-east-1-bundle.pem	sa-east-1-bundle.p7b
모든 AWS GovCloud (US) Region	global-bundle.pem	global-bundle.p7b
AWS GovCloud(미국 동부)	us-gov-east-1-bundle.pem	us-gov-east-1-bundle.p7b
AWS GovCloud(미국 서부)	us-gov-west-1-bundle.pem	us-gov-west-1-bundle.p7b

CA 인증서의 내용 보기

CA 인증서 번들의 내용을 확인하려면 다음 명령을 사용합니다.

```
keytool -printcert -v -file global-bundle.pem
```

SSL/TLS 인증서 교체

Amazon RDS 인증 기관 인증서 rds-ca-2019는 2024년 8월에 만료됩니다. RDS DB 인스턴스 또는 다중 AZ DB 클러스터에 연결하기 위해 인증서 확인과 함께 보안 소켓 계층(SSL) 또는 Transport Layer Security(TLS)를 사용하거나 사용할 계획이라면 새 CA 인증서인 rds-ca-rsa2048-g1, rds-ca-rsa4096-g1 또는 rds-ca-ecc384-g1 중 하나를 사용하는 것을 고려하세요. 현재 인증서 확인과 함께 SSL/TLS를 사용하지 않는 경우에도 CA 인증서가 만료되었을 수 있으며, 인증서 확인과 함께 SSL/TLS를 사용하여 RDS 데이터베이스에 연결하려는 경우 새 CA 인증서로 업데이트해야 합니다.

Amazon RDS는 AWS 보안 모범 사례로 새 CA 인증서를 제공합니다. 새 인증서 및 지원되는 AWS 리전에 대한 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 섹션을 참조하세요.

데이터베이스의 CA 인증서를 업데이트하려면 다음 방법을 사용하세요.

- DB 인스턴스 또는 클러스터를 수정하여 CA 인증서 업데이트
- 유지 관리를 적용하여 CA 인증서 업데이트

DB 인스턴스 또는 다중 AZ DB 클러스터에서 새로운 CA 인증서를 사용하도록 업데이트하기 전에 RDS 데이터베이스에 연결하는 클라이언트 또는 애플리케이션을 업데이트해야 합니다.

인증서 교체 고려 사항

인증서를 교체하기 전에 다음 상황을 고려하세요.

- Amazon RDS Proxy 에서는 AWS Certificate Manager(ACM)의 인증서를 사용합니다. RDS 프록시를 사용하는 경우 SSL/TLS 인증서를 교체할 때 RDS 프록시 연결을 사용하는 애플리케이션을 업데이트할 필요가 없습니다. 자세한 정보는 RDS Proxy에서 TLS/SSL 사용을 참조하세요.
- 2020년 7월 28일 이전에 생성되었거나 rds-ca-2019 인증서로 업데이트된 DB 인스턴스 또는 다중 AZ DB 클러스터가 있는 Go 버전 1.15 애플리케이션을 사용하는 경우 인증서를 다시 업데이트해야 합니다. 엔진에 따라 rds-ca-rsa2048-g1, rds-ca-rsa4096-g1, rds-ca-ecc384-g1 로 인증서를 업데이트하세요.

  새 CA 인증서 식별자를 사용하여 DB 인스턴스의 경우 `modify-db-instance` 명령 또는 다중 AZ DB 클러스터의 경우 `modify-db-cluster` 명령을 사용합니다. `describe-db-engine-versions` 명령을 사용하여 특정 DB 엔진 및 DB 엔진 버전에 사용할 수 있는 CA를 찾을 수 있습니다.

2020년 7월 28일 이후에 데이터베이스를 생성하거나 인증서를 업데이트한 경우에는 아무 조치도 필요하지 않습니다. 자세한 내용은 Go GitHub 문제 #39568를 참조하세요.

DB 인스턴스 또는 클러스터를 수정하여 CA 인증서 업데이트

다음 예제에서는 rds-ca-2019에서 rds-ca-rsa2048-g1로 CA 인증서를 업데이트합니다. 다른 인증서를 선택할 수 있습니다. 자세한 내용은 인증 기관 섹션을 참조하세요.

애플리케이션 트러스트 스토어를 업데이트하여 CA 인증서 업데이트와 관련된 가동 중단 시간을 줄이세요. CA 인증서 교체와 관련된 다시 시작에 대한 자세한 내용은 자동 서버 인증서 교체 섹션을 참조하세요.

DB 인스턴스 또는 클러스터를 수정하여 CA 인증서를 업데이트하는 방법

1. SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화에 설명된 대로 새 SSL/TLS 인증서를 다운로드합니다.
2. 새 SSL/TLS 인증서를 사용하도록 애플리케이션을 업데이트합니다.

   새 SSL/TLS 인증서를 위해 애플리케이션을 업데이트하는 방법은 애플리케이션에 따라 다릅니다. 애플리케이션 개발자와 함께 애플리케이션의 SSL/TLS 인증서를 업데이트하십시오.

   SSL/TLS 연결을 확인하고 각 DB 엔진의 애플리케이션을 업데이트하는 방법에 대한 자세한 내용은 다음 주제를 참조하십시오.

   - 새 SSL/TLS 인증서를 사용해 MariaDB 인스턴스에 연결할 애플리케이션 업데이트
   - 새 SSL/TLS 인증서를 사용해 Microsoft SQL Server DB 인스턴스에 연결할 애플리케이션을 업데이트
   - 새 SSL/TLS 인증서를 사용해 MySQL DB 인스턴스에 연결할 애플리케이션 업데이트
   - 새 SSL/TLS 인증서를 사용해 Oracle DB에 연결할 애플리케이션을 업데이트
   - 새 SSL/TLS 인증서를 사용해 PostgreSQL DB 인스턴스에 연결할 애플리케이션 업데이트

   Linux 운영 체제의 트러스트 스토어를 업데이트하는 샘플 스크립트는 트러스트 스토어로 인증서를 가져오기 위한 샘플 스크립트 섹션을 참조하세요.

> **Note**
>
> 인증서 번들에는 이전 및 신규 CA의 인증서가 들어 있으므로 애플리케이션을 안전하게 업그레이드하고 전환 기간에 연결성을 유지할 수 있습니다. AWS Database Migration Service를 사용하여 데이터베이스를 DB 인스턴스 또는 클러스터 로 마이그레이션하는 경우, 연결이 끊기지 않고 마이그레이션이 진행되도록 인증서 번들을 사용하는 것이 좋습니다.

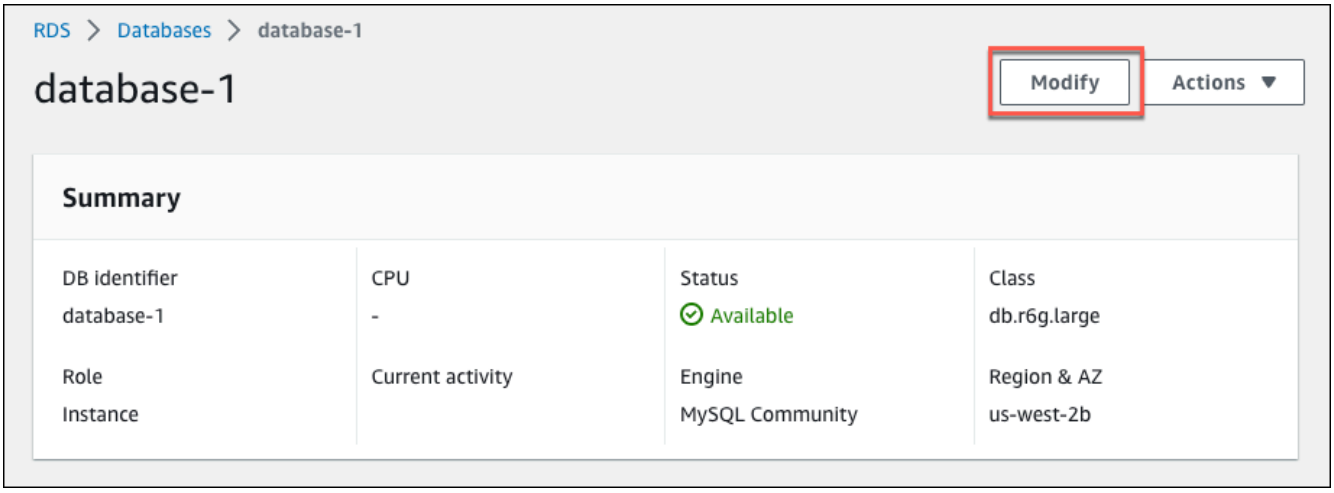
3. DB 인스턴스 또는 다중 AZ DB 클러스터를 수정하여 CA를 rds-ca-2019에서 rds-ca-rsa2048-g1로 변경합니다. CA 인증서를 업데이트하기 위해 데이터베이스를 다시 시작해야 하는지 확인하려면 describe-db-engine-versions 명령을 사용하여 `SupportsCertificateRotationWithoutRestart` 플래그를 확인합니다.

> **Important**
>
> 인증서 만료 후 연결 문제가 발생하는 경우 콘솔에서 즉시 적용을 지정하거나 `--apply-immediately`를 통해 AWS CLI 옵션을 지정하여 즉시 적용 옵션을 사용합니다. 기본적으로 이 작업은 다음 유지 관리 기간 중에 실행되도록 예약되어 있습니다.
> RDS for Oracle DB 인스턴스의 경우 연결 오류를 방지하려면 Oracle DB를 다시 시작하는 것이 좋습니다.
> AlwaysOn 또는 미러링 옵션이 활성화된 RDS for SQL Server 다중 AZ 인스턴스의 경우 인증서 교체 후 인스턴스를 재부팅하면 장애 조치가 수행될 수 있습니다.
> 기본 RDS CA와 다른 인스턴스 CA에 재정의를 설정하려면 modify-certificates CLI 명령을 사용합니다.

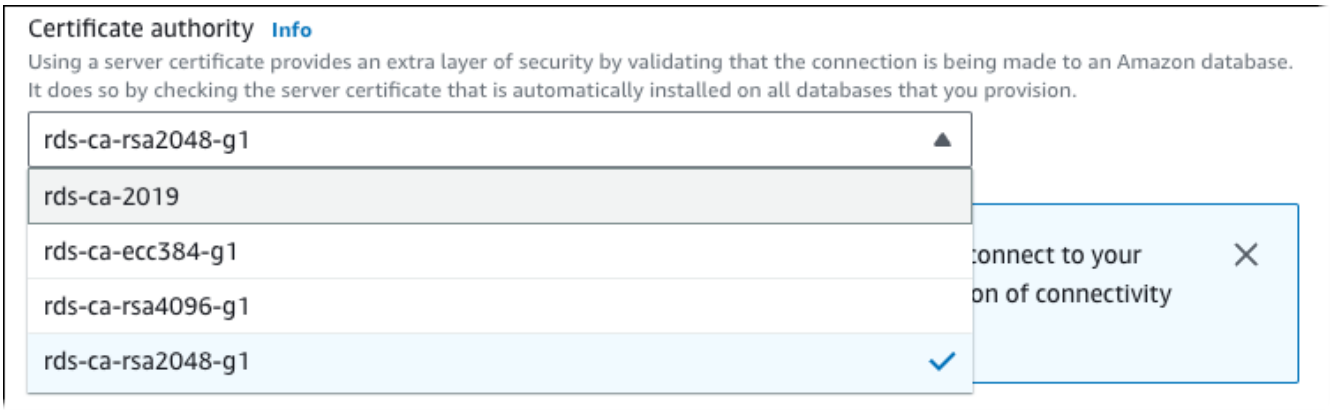
AWS Management Console 또는 AWS CLI를 사용하여 DB 인스턴스 또는 다중 AZ DB 클러스터의 CA 인증서를 rds-ca-2019에서 rds-ca-rsa2048-g1로 변경할 수 있습니다.

Console

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택한 다음 수정하려는 DB 인스턴스 또는 다중 AZ DB 클러스터를 선택합니다.
3. 수정을 선택합니다.



4. 연결 섹션에서 rds-ca-rsa2048-g1을 선택합니다.



5. [Continue]를 수정 사항을 요약한 내용을 확인합니다.
6. 변경 사항을 즉시 적용하려면 즉시 적용을 선택합니다.
7. 확인 페이지에서 변경 내용을 검토합니다. 내용이 정확할 경우 DB 인스턴스 수정 또는 클러스터 수정을 선택하여 변경 사항을 저장합니다.

> **Important**
>
> 이 작업을 예약하는 경우 클라이언트 측 트러스트 스토어를 미리 업데이트했는지 확인하십시오.

또는 뒤로를 선택하여 변경 내용을 편집하거나 취소를 선택하여 변경 내용을 취소합니다.

AWS CLI

AWS CLI를 사용하여 DB 인스턴스 또는 다중 AZ DB 클러스터의 CA를 rds-ca-2019에서 rds-ca-rsa2048-g1로 변경하려면 modify-db-instance 또는 modify-db-cluster 명령을 직접 호출합니다. DB 인스턴스 또는 클러스터 식별자 및 `--ca-certificate-identifier` 옵션을 지정합니다.

`--apply-immediately` 파라미터를 사용하여 업데이트를 즉시 적용합니다. 기본적으로 이 작업은 다음 유지 관리 기간 중에 실행되도록 예약되어 있습니다.

> **Important**
>
> 이 작업을 예약하는 경우 클라이언트 측 트러스트 스토어를 미리 업데이트했는지 확인하십시오.

Example

DB 인스턴스

다음 예시에서는 CA 인증서를 `rds-ca-rsa2048-g1`로 설정하여 `mydbinstance`를 수정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --ca-certificate-identifier rds-ca-rsa2048-g1
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --ca-certificate-identifier rds-ca-rsa2048-g1
```

> **Note**
>
> 인스턴스 재부팅이 필요한 경우 modify-db-instance CLI 명령을 사용하여 `--no-certificate-rotation-restart` 옵션을 지정할 수 있습니다.

Example

다중 AZ DB 클러스터

다음 예시에서는 CA 인증서를 `rds-ca-rsa2048-g1`로 설정하여 `mydbcluster`를 수정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --ca-certificate-identifier rds-ca-rsa2048-g1
```

Windows의 경우:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --ca-certificate-identifier rds-ca-rsa2048-g1
```

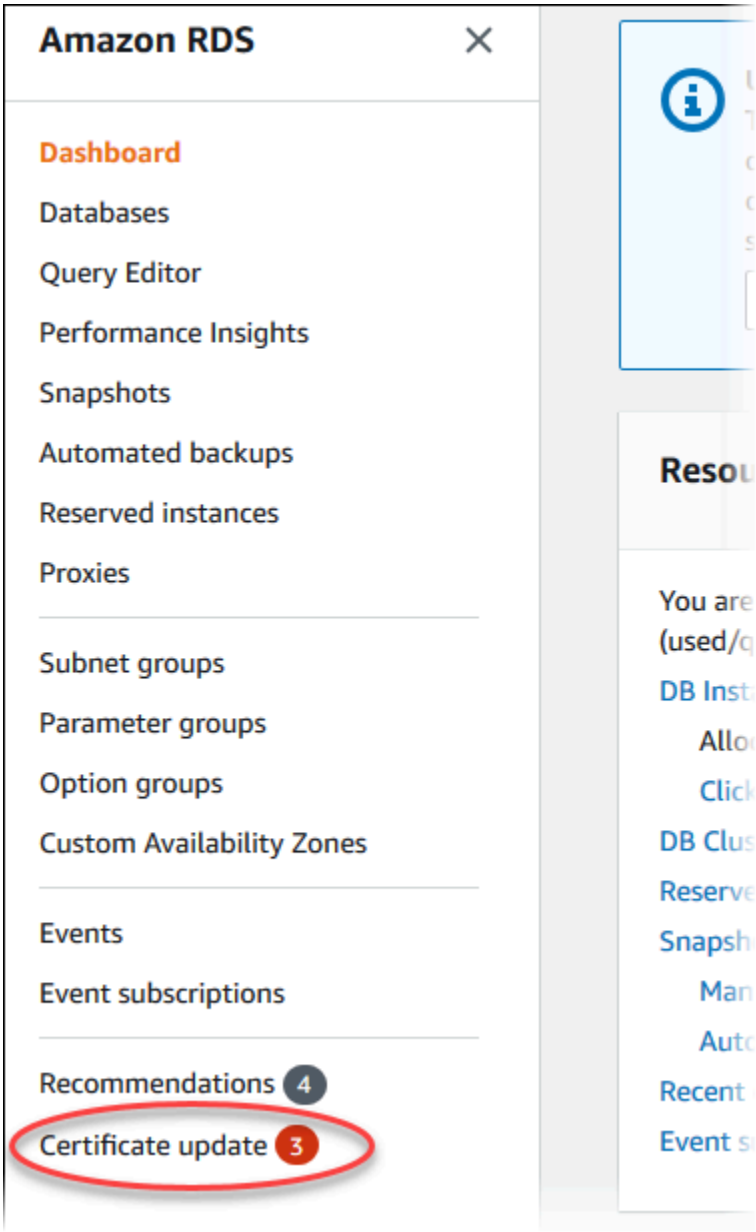
유지 관리를 적용하여 CA 인증서 업데이트

유지 관리를 적용하여 CA 인증서를 업데이트하려면 다음 단계를 수행합니다.

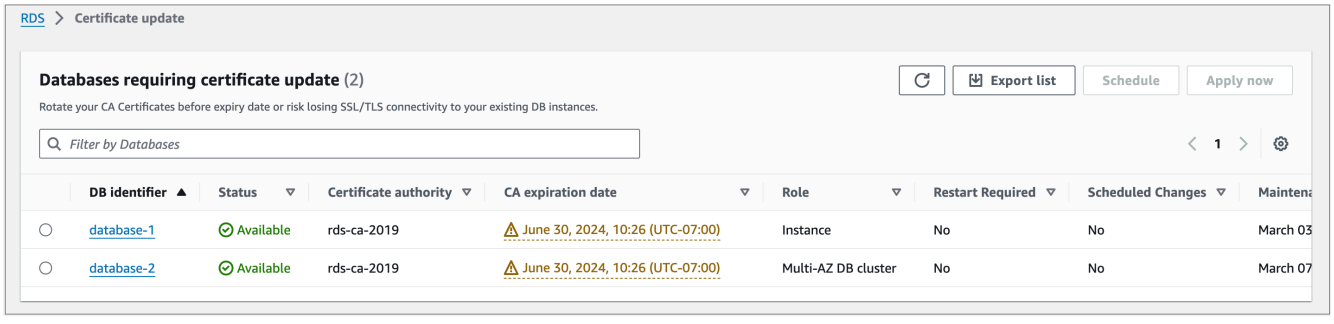
Console

유지 관리를 적용하여 CA 인증서를 업데이트하는 방법

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 인증서 업데이트를 선택합니다.



인증서 업데이트가 필요한 데이터베이스 페이지가 표시됩니다.



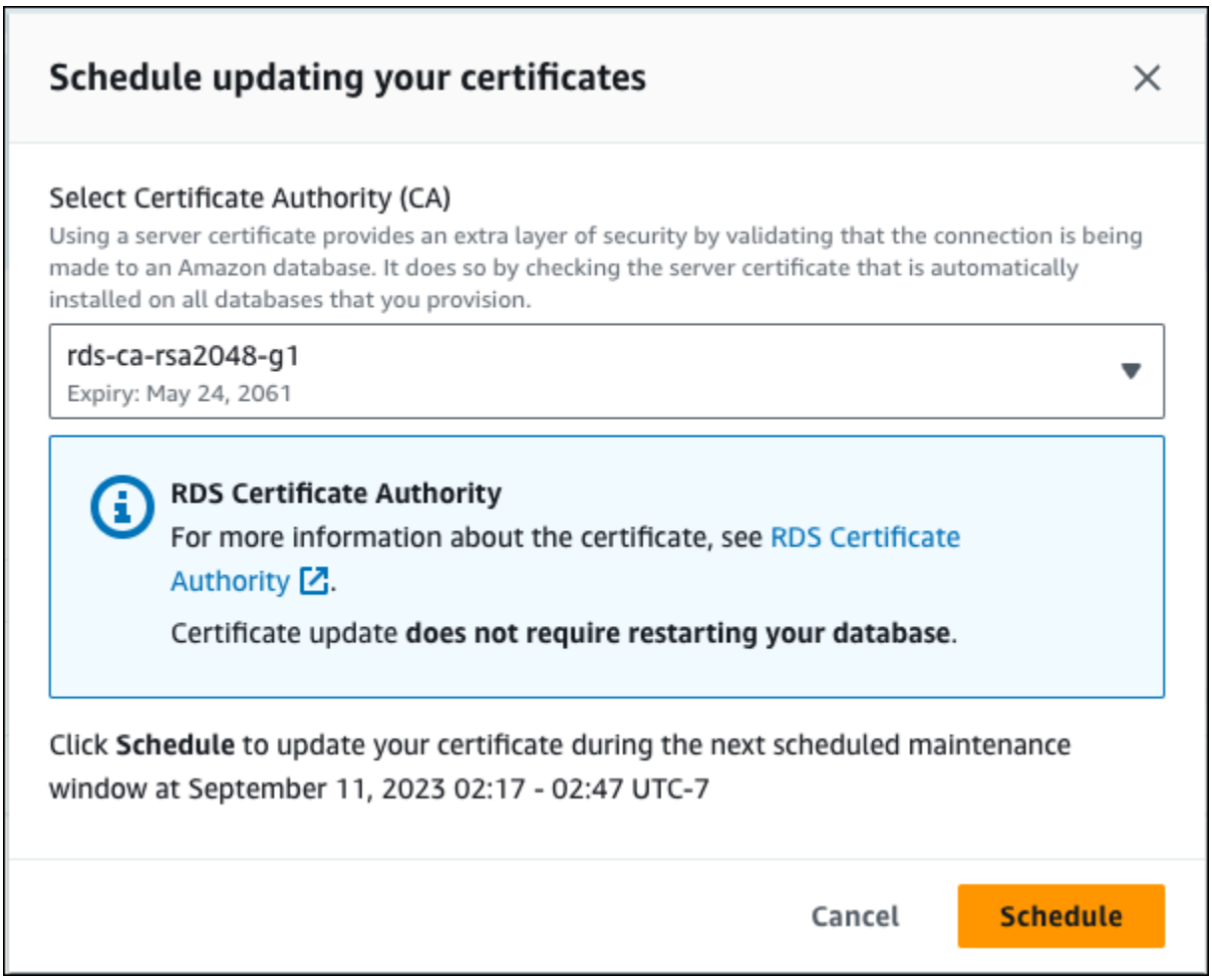
> **Note**
>
> 이 페이지에는 현재 AWS 리전의 DB 인스턴스 및 클러스터만 표시됩니다. 둘 이상의 AWS 리전에 데이터베이스가 있는 경우 각 AWS 리전에서 이 페이지를 확인하여 이전 SSL/TLS 인증서가 있는 모든 DB 인스턴스를 확인합니다.

3. 업데이트할 DB 인스턴스 또는 다중 AZ DB 클러스터를 선택합니다.

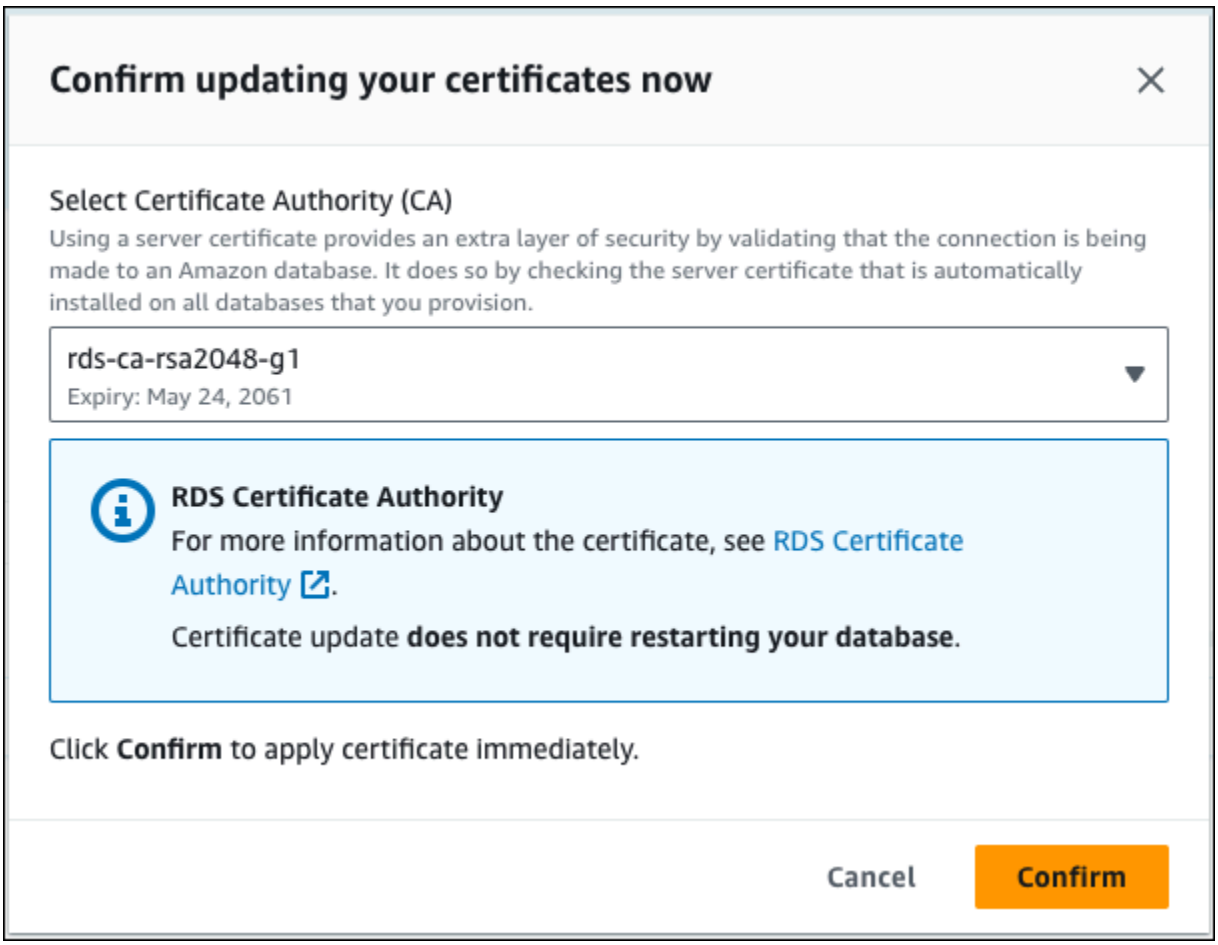
   일정을 선택하여 다음 유지 관리 기간에 따른 인증서 교체를 예약할 수 있습니다. 지금 적용을 선택하여 즉시 교체를 적용합니다.

> **Important**
>
> 인증서 만료 후 연결 문제가 발생하면 지금 적용 옵션을 사용합니다.

4. a. 일정을 선택하면 CA 인증서 교체를 확인하라는 메시지가 표시됩니다. 이 메시지에는 예약된 업데이트 기간도 표시됩니다.



b. 지금 적용을 선택하면 CA 인증서 교체를 확인하라는 메시지가 표시됩니다.



> **Important**
>
> 데이터베이스에서 CA 인증서 교체를 예약하기 전에 SSL/TLS 및 서버 인증서를 사용하여 연결하는 모든 클라이언트 애플리케이션을 업데이트합니다. 이러한 업데이트는 DB 엔진에만 적용됩니다. 이러한 클라이언트 애플리케이션을 업데이트한 후 CA 인증서 교체를 확인할 수 있습니다.

계속하려면 확인란을 선택한 다음 확인선택합니다.

5. 업데이트할 각 DB 인스턴스 및 클러스터에 대해 3단계와 4단계를 반복합니다.

자동 서버 인증서 교체

루트 CA에서 자동 서버 인증서 교체를 지원하는 경우 RDS는 DB 서버 인증서의 교체를 자동으로 처리합니다. RDS는 자동 교체에 동일한 루트 CA를 사용하므로 사용자가 새 CA 번들을 다운로드할 필요가 없습니다. 인증 기관 섹션을 참조하세요.

DB 서버 인증서의 교체 및 유효 기간은 다음의 DB 엔진에 따라 다릅니다.

- DB 엔진이 재시작 없는 교체를 지원하는 경우 RDS는 사용자가 별도의 조치를 취하지 않아도 DB 서버 인증서를 자동으로 교체합니다. RDS는 사용자가 선호하는 유지 관리 기간 내에 DB 서버 인증서의 유효 기간이 반쯤 남은 시점에서 DB 서버 인증서 교체를 시도합니다. 새 DB 서버 인증서는 12개월 동안 유효합니다.
- DB 엔진에서 재시작 없이는 교체할 수 없는 경우, RDS는 DB 서버 인증서가 만료되기 최소 6개월 전에 유지 관리 이벤트에 대해 사용자에게 알려줍니다. 새 DB 서버 인증서는 36개월 동안 유효합니다.

describe-db-engine-versions 명령을 사용하고 `SupportsCertificateRotationWithoutRestart` 플래그를 검사하여 DB 엔진 버전이 재시작 없이 인증서를 교체하는 기능을 지원하는지 파악합니다. 자세한 정보는 데이터베이스의 CA 설정을 참조하세요.

트러스트 스토어로 인증서를 가져오기 위한 샘플 스크립트

다음은 인증서 번들을 트러스트 스토어로 가져오는 샘플 셸 스크립트입니다.

각 샘플 셸 스크립트는 Java Devopment Kit(JDK)의 일부인 keytool을 사용합니다. Docker 설치에 대한 자세한 내용은 Docker 설치 안내서를 참조하세요.

Linux

다음은 Linux 운영 체제에서 트러스트 스토어로 인증서 번들을 가져오는 샘플 셸 스크립트입니다.

```
mydir=tmp/certs
if [ ! -e "${mydir}" ]
then
mkdir -p "${mydir}"
fi truststore=${mydir}/rds-truststore.jks storepassword=changeit

curl -sS "https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem">
 ${mydir}/global-bundle.pem
```

```
awk 'split_after == 1 {n++;split_after=0} /-----END CERTIFICATE-----/
 {split_after=1}{print > "rds-ca-" n+1 ".pem"}' < ${mydir}/global-bundle.pem

for CERT in rds-ca-*; do alias=$(openssl x509 -noout -text -in $CERT | perl -ne
 'next unless /Subject:/; s/.*(CN=|CN = )//; print')
  echo "Importing $alias"
  keytool -import -file ${CERT} -alias "${alias}" -storepass ${storepassword} -
keystore ${truststore} -noprompt
  rm $CERT
done

rm ${mydir}/global-bundle.pem

echo "Trust store content is: "

keytool -list -v -keystore "$truststore" -storepass ${storepassword} | grep Alias |
 cut -d " " -f3- | while read alias
do expiry=`keytool -list -v -keystore "$truststore" -storepass ${storepassword} -
alias "${alias}" | grep Valid | perl -ne 'if(/until: (.*?)\n/) { print "$1\n"; }'`
   echo " Certificate ${alias} expires in '$expiry'"
done
```

macOS

다음은 macOS에서 트러스트 스토어로 인증서 번들을 가져오는 샘플 셸 스크립트입니다.

```
mydir=tmp/certs
if [ ! -e "${mydir}" ]
then
mkdir -p "${mydir}"
fi truststore=${mydir}/rds-truststore.jks storepassword=changeit

curl -sS "https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem">
 ${mydir}/global-bundle.pem
split -p "-----BEGIN CERTIFICATE-----" ${mydir}/global-bundle.pem rds-ca-

for CERT in rds-ca-*; do alias=$(openssl x509 -noout -text -in $CERT | perl -ne
 'next unless /Subject:/; s/.*(CN=|CN = )//; print')
  echo "Importing $alias"
  keytool -import -file ${CERT} -alias "${alias}" -storepass ${storepassword} -
keystore ${truststore} -noprompt
  rm $CERT
done
```

```
rm ${mydir}/global-bundle.pem

echo "Trust store content is: "

keytool -list -v -keystore "$truststore" -storepass ${storepassword} | grep Alias |
 cut -d " " -f3- | while read alias
do expiry=`keytool -list -v -keystore "$truststore" -storepass ${storepassword} -
alias "${alias}" | grep Valid | perl -ne 'if(/until: (.*?)\n/) { print "$1\n"; }'`
   echo " Certificate ${alias} expires in '$expiry'"
done
```

인터네트워크 트래픽 개인 정보

Amazon RDS 와 온프레미스 애플리케이션 간의 연결을 비롯해 Amazon RDS 및 동일한 AWS 리전의 다른 AWS 리소스 간의 연결이 보호됩니다.

서비스와 온프레미스 클라이언트 및 애플리케이션 간의 트래픽

프라이빗 네트워크와 AWS 사이에 두 연결 옵션이 있습니다.

- AWS Site-to-Site VPN 연결. 자세한 내용은 AWS Site-to-Site VPN란 무엇입니까?를 참조하십시오.
- AWS Direct Connect 연결. 자세한 내용은 AWS Direct Connect란 무엇인가요?를 참조하세요.

AWS에서 게시한 API 작업을 사용하면 네트워크를 통해 Amazon RDS 에 액세스할 수 있습니다. 고객은 다음을 지원해야 합니다.

- Transport Layer Security(TLS) TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군 Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 위탁자와 관련된 보안 암호 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 자격 증명을 생성하여 요청에 서명할 수 있습니다.

Amazon RDS의 자격 증명 및 액세스 관리

AWS Identity and Access Management(IAM)은 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 있도록 지원하는 AWS 서비스입니다. IAM 관리자는 어떤 사용자가 Amazon RDS 리소스를 사용할 수 있도록 인증(로그인)되고 권한이 부여(권한 있음)될 수 있는지 제어합니다. IAM은 추가 비용 없이 사용할 수 있는 AWS 서비스입니다.

주제

- 고객
- 자격 증명을 통한 인증
- 정책을 사용하여 액세스 관리
- Amazon RDS에서 IAM을 사용하는 방법
- Amazon RDS 자격 증명 기반 정책 예
- Amazon RDS에 대한 AWS 관리형 정책
- AWS 관리형 정책에 대한 Amazon RDS 업데이트
- 교차 서비스 혼동된 대리자 문제 방지
- MariaDB, MySQL 및 PostgreSQL IAM 데이터베이스 인증
- Amazon RDS 자격 증명 및 액세스 문제 해결

고객

AWS Identity and Access Management(IAM)를 사용하는 방법은 Amazon RDS에서 수행하는 작업에 따라 달라집니다.

서비스 사용자 – Amazon RDS 서비스를 사용하여 작업을 수행하는 경우 필요한 자격 증명과 권한을 관리자가 제공합니다. 더 많은 Amazon RDS 기능을 사용하여 작업을 수행하게 되면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수 있습니다. Amazon RDS의 기능에 액세스할 수 없는 경우 Amazon RDS 자격 증명 및 액세스 문제 해결 단원을 참조하십시오.

서비스 관리자 – 회사에서 Amazon RDS 리소스를 책임지고 있는 경우 Amazon RDS에 대한 전체 액세스를 가지고 있을 것입니다. 서비스 관리자는 직원이 액세스해야 하는 Amazon RDS 기능과 리소스를

결정합니다. 그런 다음 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하세요. 회사가 Amazon RDS에서 IAM을 사용하는 방법에 대해 자세히 알아보려면 Amazon RDS에서 IAM을 사용하는 방법 단원을 참조하십시오.

관리자 – 관리자는 Amazon RDS에 대한 액세스 권한을 관리하는 정책을 작성하는 방법에 대해 자세히 알아보는 것이 좋습니다. IAM에서 사용할 수 있는 예제 Amazon RDS 자격 증명 기반 정책 예제를 보려면 Amazon RDS 자격 증명 기반 정책 예 단원을 참조하십시오.

자격 증명을 통한 인증

인증은 ID 자격 증명을 사용하여 AWS에 로그인하는 방식입니다. AWS 계정 루트 사용자이나 IAM 사용자로, 또는 IAM 역할을 수임하여 인증(AWS에 로그인)받아야 합니다.

ID 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 ID로 AWS에 로그인할 수 있습니다. AWS IAM Identity Center (IAM Identity Center) 사용자, 회사의 Single Sign-On 인증, Google 또는 Facebook 보안 인증이 페더레이션 ID의 예제입니다. 페더레이션형 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 연동을 사용하여 AWS에 액세스하면 간접적으로 역할을 수임합니다.

사용자 유형에 따라AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. AWS에 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS 계정에 로그인하는 방법을 참조하세요.

AWS에 프로그래밍 방식으로 액세스하는 경우, AWS에서는 자격 증명을 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트(SDK) 및 명령줄 인터페이스(CLI)를 제공합니다. AWS 도구를 사용하지 않는 경우, 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 요청에 직접 서명하는 자세한 방법은 IAM 사용 설명서에서 API 요청용 AWS Signature Version 4를 참조하세요.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, AWS은(는) 다중 인증(MFA)을 사용하여 계정의 보안을 강화하는 것을 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 다중 인증 및 IAM 사용 설명서에서 IAM의 AWS 다중 인증을 참조하세요.

AWS 계정 루트 사용자

AWS 계정을(를) 생성할 때는 해당 계정의 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 단일 로그인 ID로 시작합니다. 이 ID는 AWS 계정루트 사용자라고 하며, 계정을 생성할 때 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명을 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는

데 사용합니다. 루트 사용자로 로그인해야 하는 전체 작업 목록은 IAM 사용 설명서의 루트 사용자 보안 인증이 필요한 작업을 참조하세요.

페더레이션 자격 증명

가장 좋은 방법은 관리자 액세스가 필요한 사용자를 포함한 사용자가 ID 공급자와의 페더레이션을 사용하여 임시 자격 증명을 사용하여 AWS 서비스에 액세스하도록 요구합니다.

페더레이션 ID는 엔터프라이즈 사용자 디렉터리, 웹 ID 공급자, AWS Directory Service, Identity Center 디렉터리의 사용자 또는 자격 증명 소스를 통해 제공된 자격 증명을 사용하여 AWS 서비스에 액세스하는 모든 사용자입니다. 페더레이션 ID는 AWS 계정에 액세스할 때 역할을 수임하고 역할은 임시 자격 증명을 제공합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center을(를) 사용하는 것이 좋습니다. IAM Identity Center에서 사용자 및 그룹을 생성하거나 모든 AWS 계정 및 애플리케이션에서 사용하기 위해 고유한 ID 소스의 사용자 및 그룹 집합에 연결하고 동기화할 수 있습니다. IAM Identity Center에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 IAM Identity Center란 무엇인가요?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가지고 있는 AWS 계정 내 ID입니다. 가능하면 암호 및 액세스 키와 같은 장기 자격 증명이 있는 IAM 사용자를 생성하는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명이 필요한 특정 사용 사례가 있는 경우, 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 장기 보안 인증이 필요한 사용 사례의 경우, 정기적으로 액세스 키 교체를 참조하세요.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 자격 증명을 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세한 내용은 IAM 사용 설명서에서 IAM 사용자 사용 사례를 참조하세요.

IAM 데이터베이스 인증을 사용하여 DB 인스턴스에 인증할 수 있습니다.

IAM 데이터베이스 인증은 다음 DB 엔진에서 유효합니다.

- RDS for MariaDB

- RDS for MySQL
- RDS for PostgreSQL

IAM을 사용한 DB 인스턴스 인증에 대한 자세한 내용은 MariaDB, MySQL 및 PostgreSQL IAM 데이터베이스 인증 섹션을 참조하세요.

IAM 역할

IAM 역할은 특정 권한을 가지고 있는 AWS 계정 계정 내 ID입니다. 이 역할은 사용자와 비슷하지만, 특정 개인과 연결되지 않습니다. 역할 전환하여 AWS Management Console에서 IAM 역할을 임시로 수임할 수 있습니다. AWS CLI 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 정보는 IAM 사용 설명서의 IAM 역할 사용을 참조하세요.

임시 자격 증명이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 임시 사용자 권한 - 사용자는 IAM 역할을 수임하여 특정 작업에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 페더레이션 사용자 액세스 - 페더레이션 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 ID가 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 관련 역할에 대한 자세한 내용은 IAM 사용 설명서의 Create a role for a third-party identity provider (federation)를 참조하세요. IAM Identity Center를 사용하는 경우, 권한 집합을 구성합니다. 인증 후 ID가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 집합을 IAM의 역할과 연관짓습니다. 권한 집합에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 집합을 참조하세요.
- 교차 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 AWS 서비스를 사용하면 (역할을 프록시로 사용하는 대신) 리소스에 정책을 직접 연결할 수 있습니다. 크로스 계정 액세스를 위한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하십시오.
- 교차 서비스 액세스 - 일부 AWS 서비스는 다른 AWS 서비스의 기능을 사용합니다. 예를 들어, 서비스에서 호출하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 직접적으로 호출하는 위탁자의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
  - 전달 액세스 세션 - IAM 사용자 또는 역할을 사용하여 AWS에서 작업을 수행하는 사람은 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우, 다른 서비스에서 다른 작업을 시작하는 작업을 수

행할 수 있습니다. FAS는 AWS 서비스를 직접 호출하는 위탁자의 권한과 요청하는 AWS 서비스를 함께 사용하여 다운스트림 서비스에 대한 요청을 수행합니다. FAS 요청은 서비스에서 완료를 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 받은 경우에만 이루어집니다. 이 경우, 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

- 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 Create a role to delegate permissions to an AWS 서비스를 참조하세요.
- 서비스 연결 역할 – 서비스 연결 역할은 AWS 서비스에 연결된 서비스 역할의 한 유형입니다. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 링크 역할은 AWS 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집할 수는 없습니다.

- Amazon EC2에서 실행 중인 애플리케이션 – IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS역할을 할당하고 해당 역할을 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로필을 생성합니다. 인스턴스 프로필에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

IAM 역할을 사용할지 여부를 알아보려면 IAM 사용 설명서의 사용자 대신 IAM 역할을 생성해야 하는 경우를 참조하십시오.

정책을 사용하여 액세스 관리

정책을 생성하고 IAM 자격 증명 또는 AWS 리소스에 연결하여 AWS에서 액세스를 제어합니다. 정책은 자격 증명이나 리소스와 연결될 때 해당 권한을 정의하는 AWS의 객체입니다. AWS는 엔터티(루트 사용자, 사용자 또는 IAM 역할)가 요청을 보낼 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은 AWS에 JSON 문서로 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용 설명서의 JSON 정책 개요를 참조하세요.

관리자는 AWS 리소스에 액세스할 수 있는 사람과 해당 리소스에 대해 수행할 수 있는 작업을 지정할 수 있습니다. 모든 IAM 엔터티(권한 세트 또는 역할)는 처음에는 권한이 없습니다. 다시 말해, 기본적으로 사용자는 아무 작업도 수행할 수 없으며, 자신의 암호를 변경할 수도 없습니다. 사용자에게 태스크를 수행할 권한을 부여하기 위해 관리자는 사용자에게 권한 정책을 연결해야 합니다. 또한 관리자는 의도한 권한을 가지고 있는 그룹에 사용자를 추가할 수 있습니다. 관리자가 그룹에 권한을 부여하면 그룹의 모든 사용자가 해당 권한을 받습니다.

IAM 정책은 태스크를 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole` 작업을 허용하는 정책이 있다고 가정합니다. 해당 정책이 있는 사용자는 AWS Management Console, AWS CLI 또는 AWS API에서 역할 정보를 가져올 수 있습니다.

ID 기반 정책

자격 증명 기반 정책은 권한 세트나 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 자격 증명이 수행할 수 있는 작업, 대상 리소스 및 이에 관한 조건을 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

보안 인증 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 권한 세트 또는 역할에 직접 포함됩니다. 관리형 정책은 AWS 계정에 속한 다수의 권한 세트 및 역할에 추가할 수 있는 독립형 정책입니다. 관리형 정책에는 AWS관리형 정책과 고객 관리형 정책이 포함되어 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책과 인라인 정책의 선택을 참조하세요.

Amazon RDS와 관련된 AWS 관리형 정책에 대한 자세한 내용은 Amazon RDS에 대한 AWS 관리형 정책 섹션을 참조하세요.

기타 정책 타입

AWS는 비교적 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 자격 증명 기반 정책에 따라 IAM 엔터티(권한 세트 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 엔터티에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 개체의 자격 증명 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 권한 세트 또는 역할을 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 내용은 IAM 사용 설명서의 IAM 개체에 대한 권한 경계를 참조하세요.
- 서비스 제어 정책(SCP) – SCP는 AWS Organizations에서 조직 또는 조직 단위(OU)에 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations는 기업이 소유하는 여러 개의 AWS 계정을 그룹화하고 중앙에서 관리하기 위한 서비스입니다. 조직에서 모든 특성을 활성화할 경우, 서비스 제어 정책(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 각 AWS 계정 루트 사용자을 비롯하여 멤버 계정의 엔터티에 대한 권한을 제한합니다. 조직 및 SCP에 대한 자세한 정보는 AWS Organizations 사용 설명서의 SCP 작동 방식을 참조하십시오.
- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 권한 세트 또는 역할

의 자격 증명 기반 정책과 세션 정책이 만나는 지점입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 여러 정책 유형이 관련될 때 AWS가 요청을 허용할지를 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

Amazon RDS에서 IAM을 사용하는 방법

IAM을 사용하여 Amazon RDS에 대한 액세스를 관리하려면 먼저 어떤 IAM 기능을 Amazon RDS에서 사용할 수 있는지를 이해해야 합니다.

다음 표에는 Amazon RDS에서 사용할 수 있는 IAM 기능이 나열되어 있습니다.

IAM 기능	Amazon RDS 지원
ID 기반 정책	예
리소스 기반 정책	아니요
정책 작업	예
정책 리소스	예
정책 조건 키(서비스별)	예
ACLs	No
속성 기반 액세스 제어(ABAC)(정책 태그)	예
임시 보안 인증	예
전달 액세스 세션	예
서비스 역할	예
서비스 연결 역할	예

Amazon RDS 및 기타 AWS 서비스에서 IAM을 사용하는 방법을 개괄적으로 알아보려면 IAM 사용 설명서에서 IAM으로 작업하는 AWS 서비스를 참조하세요.

주제

- Amazon RDS 자격 증명 기반 정책
- Amazon RDS 내의 리소스 기반 정책
- Amazon RDS의 정책 작업
- Amazon RDS의 정책 리소스
- Amazon RDS의 정책 조건 키
- Amazon RDS의 액세스 제어 목록(ACL)
- Amazon RDS 태그가 있는 정책의 속성 기반 액세스 제어(ABAC)
- Amazon RDS에서 임시 자격 증명 사용
- Amazon RDS를 위한 전달 액세스 세션
- Amazon RDS에 대한 서비스 역할
- Amazon RDS에 대한 서비스 연결 역할

Amazon RDS 자격 증명 기반 정책

ID 기반 정책 지원: 예.

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

IAM ID 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. 보안 인증 기반 정책에서는 위탁자가 연결된 사용자 또는 역할에 적용되므로 보안 주체를 지정할 수 없습니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

Amazon RDS 자격 증명 기반 정책 예

Amazon RDS 자격 증명 기반 정책 예제를 보려면 Amazon RDS 자격 증명 기반 정책 예 단원을 참조하십시오.

Amazon RDS 내의 리소스 기반 정책

리소스 기반 정책 지원: 아니요.

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 위탁자가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 위탁자를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는 AWS 서비스가 포함될 수 있습니다.

교차 계정 액세스를 활성화하려는 경우, 전체 계정이나 다른 계정의 IAM 개체를 리소스 기반 정책의 위탁자로 지정할 수 있습니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 트러스트 관계 설정의 절반밖에 되지 않는다는 것을 유념하세요. 보안 주체와 리소스가 서로 다른 AWS 계정에 있는 경우, 신뢰할 수 있는 계정의 IAM 관리자는 보안 주체 엔터티(사용자 또는 역할)에도 리소스 액세스 권한을 부여해야 합니다. 엔터티에 ID 기반 정책을 연결하여 권한을 부여합니다. 하지만 리소스 기반 정책이 동일 계정의 위탁자에 액세스를 부여하는 경우, 추가 자격 증명 기반 정책이 필요하지 않습니다. 자세한 내용은 IAM 사용 설명서의 크로스 계정 리소스 액세스를 참조하세요.

Amazon RDS의 정책 작업

정책 작업 지원: 예.

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 위탁자가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action` 요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 일반적으로 정책 작업의 이름은 연결된 AWSAPI 작업의 이름과 동일합니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하세요.

Amazon RDS의 정책 작업은 작업 앞에 다음 접두사를 사용합니다. `rds:`. 예를 들어 Amazon RDS `DescribeDBInstances` API 작업으로 DB 인스턴스를 설명할 수 있는 권한을 부여하려면 해당 정책에 `rds:DescribeDBInstances` 작업을 포함합니다. 정책 문에는 `Action` 또는 `NotAction` 요소가 반드시 추가되어야 합니다. Amazon RDS는 이 서비스로 수행할 수 있는 태스크를 설명하는 고유한 작업 세트를 정의합니다.

명령문 하나에 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "rds:action1",
      "rds:action2"
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, Describe라는 단어로 시작하는 모든 작업을 지정하려면 다음 작업을 포함합니다.

```
"Action": "rds:Describe*"
```

Amazon RDS 작업 목록을 보려면 서비스 권한 부여 참조에서 Amazon RDS에서 정의한 작업을 참조하세요.

Amazon RDS의 정책 리소스

정책 리소스 지원: 예.

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

Resource JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 문에는 Resource또는 NotResource요소가 반드시 추가되어야 합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 지원하는 작업에 대해 이를 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

DB 인스턴스 리소스는 다음과 같은 Amazon 리소스 이름(ARN)을 갖습니다.

```
arn:${Partition}:rds:${Region}:${Account}:{ResourceType}/${Resource}
```

ARN 형식에 대한 자세한 내용은 Amazon 리소스 이름(ARN) 및 AWS 서비스 네임스페이스 단원을 참조하세요.

예를 들어, 문에서 dbtest DB 인스턴스를 지정하려면 다음 ARN을 사용합니다.

```
"Resource": "arn:aws:rds:us-west-2:123456789012:db:dbtest"
```

특정 계정에 속하는 모든 DB 인스턴스를 지정하려면 와일드카드(*)를 사용합니다.

```
"Resource": "arn:aws:rds:us-east-1:123456789012:db:*"
```

리소스를 생성하기 위한 작업과 같은 일부 RDS API 작업은 특정 리소스에서 수행할 수 없습니다. 이러한 경우 와일드카드(*)를 사용하면 됩니다.

```
"Resource": "*"
```

다양한 Amazon RDS API 작업에는 여러 리소스가 관여합니다. 예를 들어 `CreateDBInstance`는 DB 인스턴스를 생성합니다. DB 인스턴스를 생성할 때는 사용자가 특정 보안 그룹과 파라미터 그룹을 사용해야 한다고 지정할 수 있습니다. 단일 문에서 여러 리소스를 지정하려면 ARN을 쉼표로 구분합니다.

```
"Resource": [
      "resource1",
      "resource2"
```

Amazon RDS 리소스 유형 및 해당 ARN의 목록을 보려면 서비스 권한 부여 참조에서 Amazon RDS에서 정의한 리소스를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 Amazon RDS에서 정의한 작업을 참조하세요.

Amazon RDS의 정책 조건 키

서비스별 정책 조건 키 지원: 예.

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Condition` 요소(또는 `Condition` 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. `Condition` 요소는 옵션입니다. 같거나 작음과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 `Condition` 요소를 지정하거나 단일 `Condition` 요소에서 여러 키를 지정하는 경우, AWS는 논리적 `AND` 작업을 사용하여 평가합니다. 단일 조건 키의 여러 값을 지정하는 경우, AWS는 논리적 `OR` 작업을 사용하여 조건을 평가합니다. 문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하세요.

AWS는 전역 조건 키와 서비스별 조건 키를 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

Amazon RDS에서는 자체 조건 키 집합을 정의하고 일부 전역 조건 키 사용도 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

모든 RDS API 작업은 `aws:RequestedRegion` 조건 키를 지원합니다.

Amazon RDS 조건 키 목록을 보려면 서비스 권한 부여 참조에서 Amazon RDS에서 정의한 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 Amazon RDS에서 정의한 작업을 참조하세요.

Amazon RDS의 액세스 제어 목록(ACL)

액세스 제어 목록(ACL) 지원; 아니요.

액세스 제어 목록(ACL)은 어떤 위탁자(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon RDS 태그가 있는 정책의 속성 기반 액세스 제어(ABAC)

정책의 ABAC(속성 기반 액세스 제어) 태그 지원: 예.

속성 기반 액세스 제어(ABAC)는 속성에 근거하여 권한을 정의하는 권한 부여 전략입니다. AWS에서는 이러한 속성을 태그라고 합니다. IAM 엔티티(사용자 또는 역할) 및 많은 AWS리소스에 태그를 연결할 수 있습니다. ABAC의 첫 번째 단계로 개체 및 리소스에 태그를 지정합니다. 그런 다음 위탁자의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하도록 ABAC 정책을 설계합니다.

ABAC는 빠르게 성장하는 환경에서 유용하며 정책 관리가 번거로운 상황에 도움이 됩니다.

태그에 근거하여 액세스를 제어하려면 `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 유형에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

ABAC에 대한 자세한 내용은 IAM 사용 설명서의 ABAC 권한 부여를 통한 권한 정의를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하세요.

Amazon RDS 리소스 태그 지정에 대한 자세한 내용은 조건 지정: 사용자 지정 태그 사용 단원을 참조하십시오. 리소스의 태그를 기반으로 리소스에 대한 액세스를 제한하는 자격 증명 기반 정책의 예제는 두 가지 값이 있는 특정 태그를 사용하는 리소스 작업에 대한 권한 부여에서 확인할 수 있습니다.

Amazon RDS에서 임시 자격 증명 사용

임시 자격 증명 지원: 예.

일부 AWS 서비스는 임시 자격 증명을 사용하여 로그인할 때 작동하지 않습니다. 임시 보안 인증으로 작동하는 AWS 서비스를 비롯한 추가 정보는 IAM 사용 설명서의 IAM으로 작업하는 AWS 서비스를 참조하세요.

사용자 이름과 암호를 제외한 다른 방법을 사용하여 AWS Management Console에 로그인하면 임시 자격 증명을 사용하는 것입니다. 예를 들어, 회사의 Single Sign-On(SSO) 링크를 사용하여 AWS에 액세스하면 해당 프로세스에서 자동으로 임시 자격 증명을 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 자격 증명을 자동으로 생성합니다. 역할 전환에 대한 자세한 내용은 IAM 사용 설명서의 사용자에서 IAM 역할로 전환(콘솔)을 참조하세요.

AWS CLI 또는 AWS API를 사용하여 임시 자격 증명을 수동으로 만들 수 있습니다 그런 다음 이러한 임시 자격 증명을 사용하여 AWS에 액세스할 수 있습니다. AWS에서는 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성할 것을 권장합니다. 자세한 정보는 IAM의 임시 보안 자격 증명 섹션을 참조하세요.

Amazon RDS를 위한 전달 액세스 세션

전달 액세스 세션 지원: 예.

IAM 사용자 또는 역할을 사용하여 AWS에서 작업을 수행하는 사람은 위탁자로 간주됩니다. 일부 서비스를 사용하는 경우, 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 AWS 서비스를 직접 호출하는 위탁자의 권한과 요청하는 AWS 서비스를 함께 사용하여 다운스트림 서비스에 대한 요청을 수행합니다. FAS 요청은 서비스에서 완료를 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 받은 경우에만 이루어집니다. 이 경우, 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

Amazon RDS에 대한 서비스 역할

서비스 역할 지원: 예.

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 Create a role to delegate permissions to an AWS 서비스를 참조하세요.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 Amazon RDS 기능이 중단될 수 있습니다. Amazon RDS에서 관련 지침을 제공하는 경우에만 서비스 역할을 편집합니다.

Amazon RDS에 대한 서비스 연결 역할

서비스 연결 역할 지원: 예.

서비스 연결 역할은 AWS 서비스에 연결된 서비스 역할의 한 유형입니다. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 링크 역할은 AWS 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

Amazon RDS 서비스 연결 역할을 사용하는 방법에 대한 자세한 내용은 Amazon RDS에 서비스 연결 역할 사용 섹션을 참조하세요.

Amazon RDS 자격 증명 기반 정책 예

기본적으로 권한 세트 및 역할은 Amazon RDS 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console, AWS CLI 또는 AWS API를 사용해 태스크를 수행할 수 없습니다. 관리자는 필요한 지정된 리소스에서 특정 API 작업을 수행할 수 있는 권한을 권한 세트와 역할에 부여하는 IAM 정책을 생성해야 합니다. 그런 다음 관리자는 해당 권한이 필요한 권한 세트 또는 역할에 이러한 정책을 연결해야 합니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 JSON 탭에서 정책 생성을 참조하세요.

주제

- 정책 모범 사례
- Amazon RDS 콘솔 사용
- 콘솔 사용에 필요한 권한
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용
- Amazon RDS에서 리소스를 생성, 수정 및 삭제하기 위한 권한 정책
- 정책 예: 조건 키 사용
- 조건 지정: 사용자 지정 태그 사용
- 생성 시 Amazon RDS 리소스를 태그하는 권한 부여

정책 모범 사례

ID 기반 정책에 따라 계정에서 사용자가 Amazon RDS 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따릅니다.

- AWS 관리형 정책으로 시작하고 최소 권한을 향해 나아가기 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. AWS 계정에서 사용할 수 있습니다. 사용 사례에 고유한 AWS 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS직무에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. AWS CloudFormation와 같이, 특정 AWS 서비스를 통해 사용되는 경우에만 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을 확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer에서 정책 검증을 참조하세요.
- 다중 인증(MFA) 필요 – AWS 계정에 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우, 추가 보안을 위해 MFA를 설정합니다. API 작업을 직접 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 MFA를 통한 보안 API 액세스를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

Amazon RDS 콘솔 사용

Amazon RDS 콘솔에 액세스하려면 최소 권한 집합이 있어야 합니다. 이러한 권한을 통해 AWS 계정에서 Amazon RDS 리소스에 대한 세부 정보를 나열하고 볼 수 있어야 합니다. 최소 필수 권한보다 더 제한적인 ID 기반 정책을 생성하는 경우, 콘솔이 해당 정책에 연결된 엔티티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 AWS API만 직접 호출하는 사용자에게 최소 콘솔 권한을 허용할 필요가 없습니다. 그 대신 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

해당 엔터티가 Amazon RDS 콘솔을 여전히 사용할 수 있도록 하려면 AWS 관리형 정책도 엔터티에 연결합니다.

```
AmazonRDSReadOnlyAccess
```

자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하십시오.

콘솔 사용에 필요한 권한

콘솔에서 작업하려면 최소한의 권한이 사용자에게 필요합니다. 이러한 권한이 있어야만 사용자가 자신의 AWS 계정에서 사용할 Amazon RDS 리소스를 설명하고, Amazon EC2 보안 및 네트워크 정보 등 다른 관련 정보를 입력할 수 있습니다.

최소 필수 권한보다 더 제한적인 IAM 정책을 만들면 콘솔은 해당 IAM 정책에 연결된 사용자에 대해 의도대로 작동하지 않습니다. 이 사용자가 콘솔을 사용할 수 있도록 하려면 `AmazonRDSReadOnlyAccess` 관리형 정책을 사용자에게 연결합니다(정책을 사용하여 액세스 관리 참조).

AWS CLI 또는 Amazon RDS API만 호출하는 사용자에게 최소 콘솔 권한을 허용할 필요가 없습니다.

다음 정책은 루트 AWS 계정의 모든 Amazon RDS 리소스에 대해 전체 액세스 권한을 부여합니다.

```
AmazonRDSFullAccess
```

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로 이 태스크를 완료할 수 있는 권한이 포함됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
```

```
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon RDS에서 리소스를 생성, 수정 및 삭제하기 위한 권한 정책

다음 섹션에는 리소스에 대한 액세스를 허용하고 제한하는 권한 정책의 예제가 나와 있습니다.

사용자에게 AWS 계정에서 DB 인스턴스를 생성하도록 허용

다음은 ID가 123456789012인 계정에 AWS 계정에 대한 DB 인스턴스를 생성하도록 허용하는 정책 예제입니다. 이 정책에 따라 새 DB 인스턴스의 이름은 `test`로 시작해야 합니다. 또한 새 DB 인스턴스는 MySQL 데이터베이스 엔진 및 `db.t2.micro` DB 인스턴스 클래스를 사용해야 합니다. 또한 새로운 DB 인스턴스는 `default`로 시작하는 옵션 그룹과 DB 파라미터 그룹을, 그리고 `default` 서브넷 그룹을 사용해야 합니다.

JSON

```
{
   "Version": "2012-10-17",
```

```
   "Statement": [
      {
         "Sid": "AllowCreateDBInstanceOnly",
         "Effect": "Allow",
         "Action": [
            "rds:CreateDBInstance"
         ],
         "Resource": [
            "arn:aws:rds:*:123456789012:db:test*",
            "arn:aws:rds:*:123456789012:og:default*",
            "arn:aws:rds:*:123456789012:pg:default*",
            "arn:aws:rds:*:123456789012:subgrp:default"
         ],
         "Condition": {
            "StringEquals": {
               "rds:DatabaseEngine": "mysql",
               "rds:DatabaseClass": "db.t2.micro"
            }
         }
      }
   ]
}
```

다음 사용자 권한을 지정하는 단일 명령문이 정책에 포함됩니다.

- 정책은 계정이 CreateDBInstance API 작업을 사용하여 DB 인스턴스를 생성할 수 있도록 허용합니다. 이는 create-db-instance AWS CLI 명령과 AWS Management Console에도 적용됩니다.
- `Resource` 요소는 사용자가 리소스 위치에서 또는 리소스를 사용하여 작업을 수행할 수 있도록 지정합니다. 리소스는 Amazon 리소스 이름(ARN)을 사용하여 지정합니다. 이 ARN에는 리소스가 속하는 서비스 이름(`rds`), AWS 리전(`*`는 위의 예제에서 사용하는 모든 리전을 의미함), AWS 계정 번호(이 예에서는 `123456789012`가 계정 번호임) 및 리소스 유형이 포함됩니다. ARN 생성에 대한 자세한 내용은 Amazon RDS의 Amazon 리소스 이름(ARN) 단원을 참조하십시오.

  위의 예제에서 `Resource` 요소는 사용자 리소스에 대해 다음과 같은 정책 제약 조건을 지정합니다.

  - 새 DB 인스턴스에 대한 DB 인스턴스 식별자는 `test`로 시작해야 합니다(예: `testCustomerData1`, `test-region2-data`).
  - 새로운 DB 인스턴스의 옵션 그룹은 `default`로 시작해야 합니다.
  - 새로운 DB 인스턴스의 DB 파라미터 그룹은 `default`로 시작해야 합니다.
  - 새로운 DB 인스턴스의 서브넷 그룹은 `default` 서브넷 그룹이 되어야 합니다.

- Condition 요소는 DB 엔진은 MySQL이 되고, DB 인스턴스 클래스는 db.t2.micro가 되도록 지정합니다. Condition 요소는 정책 적용 시 조건을 지정합니다. 그 밖에도 Condition 요소를 사용하여 다른 권한이나 제한을 추가할 수 있습니다. 조건 지정에 대한 자세한 내용은 Amazon RDS의 정책 조건 키 단원을 참조하십시오. 이 예제에서는 rds:DatabaseEngine 및 rds:DatabaseClass 조건을 지정합니다. rds:DatabaseEngine의 유효 조건 값에 대한 정보는 CreateDBInstance의 Engine 파라미터 아래 목록을 참조하십시오. rds:DatabaseClass의 유효 조건 값에 대한 정보는 DB 인스턴스 클래스에 지원되는 DB 엔진 단원을 참조하십시오.

자격 증명 기반 정책에서는 권한을 가질 보안 주체를 지정하지 않으므로 이 정책은 Principal 요소를 지정하지 않습니다. 정책을 사용자에게 연결할 경우 사용자는 암시적인 보안 주체입니다. IAM 역할에 권한 정책을 연결할 경우 역할의 신뢰 정책에 식별된 보안 주체는 권한을 가집니다.

Amazon RDS 작업 목록을 보려면 서비스 권한 부여 참조에서 Amazon RDS에서 정의한 작업을 참조하세요.

사용자가 모든 RDS 리소스에서 Describe 작업을 수행할 수 있도록 허용

다음 권한 정책은 사용자에게 Describe로 시작하는 모든 작업을 실행할 수 있는 권한을 부여합니다. 이러한 작업은 DB 인스턴스와 같은 RDS 리소스에 대한 정보를 보여 줍니다. Resource 요소에 와일드카드 문자(*)가 있으면 계정이 소유한 모든 Amazon RDS 리소스에 대해 작업이 허용됩니다.

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "AllowRDSDescribe",
         "Effect": "Allow",
         "Action": "rds:Describe*",
         "Resource": "*"
      }
   ]
}
```

사용자가 지정된 DB 파라미터 그룹 및 서브넷 그룹을 사용하는 DB 인스턴스를 생성할 수 있도록 허용

다음 권한 정책은 사용자가 mydbpg DB 파라미터 그룹 및 mydbsubnetgroup DB 서브넷 그룹을 사용해야 하는 DB 인스턴스만 생성할 수 있도록 허용하는 권한을 부여합니다.

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": "rds:CreateDBInstance",
         "Resource": [
            "arn:aws:rds:*:*:pg:mydbpg",
            "arn:aws:rds:*:*:subgrp:mydbsubnetgroup"
         ]
      }
   ]
}
```

두 가지 값이 있는 특정 태그를 사용하는 리소스 작업에 대한 권한 부여

자격 증명 기반 정책의 조건을 사용하여 태그를 기반으로 Amazon RDS 리소스에 대한 액세스를 제어할 수 있습니다. 다음 정책은 `development` 또는 `test`로 설정된 `stage` 태그를 사용하여 DB 인스턴스에서 `CreateDBSnapshot` API 작업을 수행할 수 있는 권한을 허용합니다.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowAnySnapshotName",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:snapshot:*"
      },
      {
         "Sid":"AllowDevTestToCreateSnapshot",
         "Effect":"Allow",
         "Action":[
```

```
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
}
```

다음 정책은 development 또는 test로 설정된 stage 태그를 사용하여 DB 인스턴스에서 ModifyDBInstance API 작업을 수행할 수 있는 권한을 허용합니다.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowChangingParameterOptionSecurityGroups",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
         ],
         "Resource": [
            "arn:aws:rds:*:123456789012:pg:*",
            "arn:aws:rds:*:123456789012:secgrp:*",
            "arn:aws:rds:*:123456789012:og:*"
         ]
      },
      {
         "Sid":"AllowDevTestToModifyInstance",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
         ],
```

```
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
}
```

사용자의 DB 인스턴스 삭제 방지

다음 권한 정책은 사용자의 특정 DB 인스턴스 삭제를 방지하는 권한을 부여합니다. 예를 들어, 관리자가 아닌 모든 사용자에 대해 프로덕션 DB 인스턴스를 삭제할 수 있는 권한을 거부해야 할 수 있습니다.

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "DenyDelete1",
         "Effect": "Deny",
         "Action": "rds:DeleteDBInstance",
         "Resource": "arn:aws:rds:us-west-2:123456789012:db:my-mysql-instance"
      }
   ]
}
```

리소스에 대한 모든 액세스 거부

리소스에 대한 액세스를 명시적으로 거부할 수 있습니다. 거부 정책은 허용 정책보다 우선합니다. 다음 정책은 사용자가 리소스를 관리할 수 있는 권한을 명시적으로 거부합니다.

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Deny",
         "Action": "rds:*",
         "Resource": "arn:aws:rds:us-east-1:123456789012:db:mydb"
      }
   ]
}
```

정책 예: 조건 키 사용

다음은 Amazon RDS IAM 권한 정책에서 조건 키를 사용할 수 있는 방법의 예입니다.

예제 1: 특정 DB 엔진을 사용하고 MultiAZ가 아닌 DB 인스턴스를 생성할 수 있는 권한 부여

다음 정책은 RDS 조건 키를 사용하며, 사용자가 MySQL 데이터베이스 엔진을 사용하는 DB 인스턴스만 생성할 수 있도록 허용하며, MultiAZ를 사용하지 않습니다. `Condition` 요소는 데이터베이스 엔진이 MySQL이라는 요구 사항을 나타냅니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "AllowMySQLCreate",
         "Effect": "Allow",
         "Action": "rds:CreateDBInstance",
         "Resource": "*",
         "Condition": {
            "StringEquals": {
               "rds:DatabaseEngine": "mysql"
            },
            "Bool": {
               "rds:MultiAz": false
            }
         }
      }
   ]
```

```
}
```

예제 2: 특정 DB 인스턴스 클래스에 대한 DB 인스턴스를 만들고 프로비저닝된 IOPS를 사용하는 DB 인스턴스를 만들 수 있는 권한을 명시적으로 거부

다음 정책은 가장 크고 가장 비싼 DB 인스턴스 클래스인 DB 인스턴스 클래스 `r3.8xlarge` 및 `m4.10xlarge`를 사용하는 DB 인스턴스를 만들 수 있는 권한을 명시적으로 거부합니다. 또한 이 정책은 추가 비용이 발생하는 프로비저닝된 IOPS를 사용하는 DB 인스턴스를 사용자가 생성하지 못하도록 합니다.

명시적으로 거부하는 권한은 이미 부여된 다른 모든 권한에 우선합니다. 따라서 부여하지 않으려는 권한을 자격 증명이 우연히 획득하지 않도록 할 수 있습니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "DenyLargeCreate",
         "Effect": "Deny",
         "Action": "rds:CreateDBInstance",
         "Resource": "*",
         "Condition": {
            "StringEquals": {
               "rds:DatabaseClass": [
                  "db.r3.8xlarge",
                  "db.m4.10xlarge"
               ]
            }
         }
      },
      {
         "Sid": "DenyPIOPSCreate",
         "Effect": "Deny",
         "Action": "rds:CreateDBInstance",
         "Resource": "*",
         "Condition": {
            "NumericNotEquals": {
               "rds:Piops": "0"
            }
         }
      }
   ]
```

```
}
```

예제 3: 리소스에 태그 지정하는 데 사용할 수 있는 태그 키와 값 집합 제한

다음 정책은 RDS 조건 키를 사용하고 키가 `stage`인 태그를 값이 `test`, `qa`, `production`인 리소스에 추가할 수 있도록 허용합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "rds:AddTagsToResource",
            "rds:RemoveTagsFromResource"
         ],
         "Resource": "*",
         "Condition": {
            "streq": {
               "rds:req-tag/stage": [
                  "test",
                  "qa",
                  "production"
               ]
            }
         }
      }
   ]
}
```

조건 지정: 사용자 지정 태그 사용

Amazon RDS에서는 사용자 지정 태그를 사용하여 IAM 정책에서 조건을 지정할 수 있습니다.

예를 들어 이름이 `environment`인 태그를 `beta`, `staging`, `production` 등의 값으로 DB 인스턴스에 추가한다고 가정하겠습니다. 그러면 `environment` 태그 값에 따라 특정 사용자를 DB 인스턴스로 제한하는 정책을 생성할 수 있습니다.

> **Note**
>
> 사용자 지정 태그 식별자는 대/소문자를 구분합니다.

다음 표에는 Condition 요소에서 사용할 수 있는 RDS 태그 식별자가 나와 있습니다.

RDS 태그 식별자	적용 대상
db-tag	읽기 전용 복제본을 포함하는 DB 인스턴스입니다.
snapshot-tag	DB 스냅샷
ri-tag	예약 DB 인스턴스
og-tag	DB 옵션 그룹
pg-tag	DB 파라미터 그룹
subgrp-tag	DB 서브넷 그룹
es-tag	이벤트 구독
cluster-tag	DB 클러스터
cluster-pg-tag	DB 클러스터 파라미터 그룹
cluster-snapshot-tag	DB 클러스터 스냅샷

사용자 지정 태그 조건의 구문은 다음과 같습니다.

"Condition":{"StringEquals":{"rds:*rds-tag-identifier*/*tag-name*": ["*value*"]} }

예를 들어, 다음 Condition 요소는 태그 이름이 environment이고 태그 값이 production인 DB 인스턴스에 적용됩니다.

"Condition":{"StringEquals":{"rds:db-tag/**environment**": ["**production**"]} }

태그 생성에 대한 자세한 내용은 Amazon RDS 리소스에 태그 지정 단원을 참조하십시오.

> **Important**
>
> 태깅을 사용하여 RDS 리소스에 대한 액세스를 관리하는 경우 RDS 리소스의 태그에 대한 액세스의 보안을 유지하는 것이 좋습니다. AddTagsToResource 및

RemoveTagsFromResource 작업에 대한 정책을 생성하여 태그에 대한 액세스를 관리할 수 있습니다. 예를 들어, 다음 정책은 모든 리소스에 대해 태그를 추가하거나 제거할 수 있는 사용자의 권한을 거부합니다. 그런 다음 특정 사용자가 태그를 추가하거나 제거할 수 있도록 허용하기 위한 정책을 생성할 수 있습니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"DenyTagUpdates",
         "Effect":"Deny",
         "Action":[
            "rds:AddTagsToResource",
            "rds:RemoveTagsFromResource"
         ],
         "Resource":"*"
      }
   ]
}
```

Amazon RDS 작업 목록을 보려면 서비스 권한 부여 참조에서 Amazon RDS에서 정의한 작업을 참조하세요.

정책 예: 사용자 지정 태그 사용

다음은 Amazon RDS IAM 권한 정책에서 사용자 지정 태그를 사용할 수 있는 방법의 예입니다. Amazon RDS 리소스에 태그를 추가하는 방법에 대한 자세한 내용은 Amazon RDS의 Amazon 리소스 이름(ARN) 단원을 참조하십시오.

> **Note**
>
> 모든 예는 us-west-2 리전을 사용하며 가상의 계정 ID를 포함합니다.

예제 1: 두 개의 값을 갖는 특정 태그를 사용하는 리소스 작업에 대한 권한 부여

다음 정책은 development 또는 test로 설정된 stage 태그를 사용하여 DB 인스턴스에서 CreateDBSnapshot API 작업을 수행할 수 있는 권한을 허용합니다.

```
{
```

```
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowAnySnapshotName",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:snapshot:*"
      },
      {
         "Sid":"AllowDevTestToCreateSnapshot",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
}
```

다음 정책은 development 또는 test로 설정된 stage 태그를 사용하여 DB 인스턴스에서 ModifyDBInstance API 작업을 수행할 수 있는 권한을 허용합니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowChangingParameterOptionSecurityGroups",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
         ],
         "Resource":" [
            "arn:aws:rds:*:123456789012:pg:*",
```

```
            "arn:aws:rds:*:123456789012:secgrp:*",
            "arn:aws:rds:*:123456789012:og:*"
         ]
      },
      {
         "Sid":"AllowDevTestToModifyInstance",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
}
```

예제 2: 지정된 DB 파라미터 그룹을 사용하는 DB 인스턴스를 만들 수 있는 권한을 명시적으로 거부

다음 정책은 특정 태그 값이 있는 DB 파라미터 그룹을 사용하는 DB 인스턴스를 만들 수 있는 권한을 명시적으로 거부합니다. DB 인스턴스를 생성할 때 특정 고객 생성 DB 파라미터 그룹을 사용해야 할 경우 이 정책을 적용할 수 있습니다. Deny를 사용하는 정책은 더 광범위한 정책에서 부여한 액세스 권한을 제한하기 위해 가장 자주 사용됩니다.

명시적으로 거부하는 권한은 이미 부여된 다른 모든 권한에 우선합니다. 따라서 부여하지 않으려는 권한을 자격 증명이 우연히 획득하지 않도록 할 수 있습니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"DenyProductionCreate",
         "Effect":"Deny",
         "Action":"rds:CreateDBInstance",
         "Resource":"arn:aws:rds:*:123456789012:pg:*",
```

```
         "Condition":{
            "StringEquals":{
               "rds:pg-tag/usage":"prod"
            }
         }
      }
   ]
}
```

예제 3: 인스턴스 이름에 사용자 이름이 접두사로 붙은 DB 인스턴스 작업에 대한 권한 부여

다음 정책은 DB 인스턴스 이름에 사용자 이름이 접두사로 붙어 있고 AddTagsToResource와 동일한 RemoveTagsFromResource라는 태그가 있거나 stage라는 태그가 없는 DB 인스턴스에서 API(devo 또는 stage 제외)를 호출할 수 있는 권한을 허용합니다.

정책의 Resource 줄은 Amazon 리소스 이름(ARN)을 기준으로 리소스를 식별합니다. Amazon RDS 리소스에서 ARN을 사용하는 방법에 대한 자세한 내용은 Amazon RDS의 Amazon 리소스 이름(ARN) 단원을 참조하십시오.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowFullDevAccessNoTags",
         "Effect":"Allow",
         "NotAction":[
            "rds:AddTagsToResource",
            "rds:RemoveTagsFromResource"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:${aws:username}*",
         "Condition":{
            "StringEqualsIfExists":{
               "rds:db-tag/stage":"devo"
            }
         }
      }
   ]
}
```

생성 시 Amazon RDS 리소스를 태그하는 권한 부여

일부 RDS API 작업을 사용하면 리소스를 생성할 때 태그를 지정할 수 있습니다. 리소스 태그를 사용하여 속성 기반 제어(ABAC)를 구현할 수 있습니다. 자세한 내용은 What is ABAC for AWS? 및 태그를 사용한 AWS 리소스 액세스 제어를 참조하세요.

사용자가 생성 시 리소스에 태그를 지정할 수 있으려면 `rds:CreateDBInstance`와 같은 리소스를 생성하는 작업을 사용할 권한이 있어야 합니다. 생성 작업에서 태그가 지정되면 RDS는 `rds:AddTagsToResource` 작업에서 추가 권한 부여를 수행해 사용자에게 태그를 생성할 권한이 있는지 확인합니다. 따라서 사용자는 `rds:AddTagsToResource` 작업을 사용할 명시적 권한도 가지고 있어야 합니다.

`rds:AddTagsToResource` 작업에 대한 IAM 정책 정의에서 `aws:RequestTag` 조건 키를 사용하여 리소스에 태그를 지정하는 요청에 태그를 요구할 수 있습니다.

예를 들어 다음 정책은 사용자가 DB 인스턴스를 생성하고 DB 인스턴스 생성 중에 태그를 적용할 수 있도록 허용하지만 특정 태그 키(`environment` 또는 `project`)를 사용해야 합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": [
               "rds:CreateDBInstance"
           ],
           "Resource": "*"
       },
       {
           "Effect": "Allow",
           "Action": [
               "rds:AddTagsToResource"
           ],
           "Resource": "*",
           "Condition": {
               "StringEquals": {
                   "aws:RequestTag/environment": ["production", "development"],
                   "aws:RequestTag/project": ["dataanalytics", "webapp"]
               },
               "ForAllValues:StringEquals": {
                   "aws:TagKeys": ["environment", "project"]
               }
```

```
            }
        }
    ]
}
```

이 정책은 environment 또는 project 태그 이외의 태그를 포함하거나 이러한 태그를 지정하지 않는 DB 인스턴스 생성 요청을 거부합니다. 또한 사용자는 정책의 허용된 값과 일치하는 태그 값을 지정해야 합니다.

다음 정책은 사용자가 DB 클러스터를 생성하고 생성 중에 environment=prod 태그를 제외하고는 어떤 태그든 적용하도록 허용합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": [
               "rds:CreateDBCluster"
           ],
           "Resource": "*"
       },
       {
           "Effect": "Allow",
           "Action": [
               "rds:AddTagsToResource"
           ],
           "Resource": "*",
           "Condition": {
               "StringNotEquals": {
                   "aws:RequestTag/environment": "prod"
               }
           }
       }
   ]
}
```

생성 시 태그 지정이 지원되는 RDS API 작업

리소스를 생성할 때 태그를 지원하는 RDS API 작업은 다음과 같습니다. 이러한 작업의 경우 리소스를 생성할 때 태그를 지정할 수 있습니다.

- CreateBlueGreenDeployment
- CreateCustomDBEngineVersion
- CreateDBCluster
- CreateDBClusterEndpoint
- CreateDBClusterParameterGroup
- CreateDBClusterSnapshot
- CreateDBInstance
- CreateDBInstanceReadReplica
- CreateDBParameterGroup
- CreateDBProxy
- CreateDBProxyEndpoint
- CreateDBSecurityGroup
- CreateDBShardGroup
- CreateDBSnapshot
- CreateDBSubnetGroup
- CreateEventSubscription
- CreateGlobalCluster
- CreateIntegration
- CreateOptionGroup
- CreateTenantDatabase
- CopyDBClusterParameterGroup
- CopyDBClusterSnapshot
- CopyDBParameterGroup
- CopyDBSnapshot
- CopyOptionGroup
- RestoreDBClusterFromS3
- RestoreDBClusterFromSnapshot
- RestoreDBClusterToPointInTime
- RestoreDBInstanceFromDBSnapshot
- RestoreDBInstanceFromS3

- RestoreDBInstanceToPointInTime
- PurchaseReservedDBInstancesOffering

AWS CLI 또는 API를 사용하여 태그가 있는 리소스를 생성하는 경우 생성 중에 리소스에 태그를 적용하는 데 Tags 파라미터가 사용됩니다.

이러한 API 작업의 경우 태그 지정에 실패하면 리소스가 생성되지 않고 오류와 함께 요청이 실패합니다. 리소스가 태그와 함께 생성되거나 전혀 생성되지 않으므로 의도한 태그 없이는 리소스가 생성되지 않습니다.

Amazon RDS에 대한 AWS 관리형 정책

권한 세트 및 역할에 권한을 추가하려면 정책을 직접 작성하는 것보다 AWS 관리형 정책을 사용하는 것이 편리합니다. 팀에 필요한 권한만 제공하는 IAM 고객 관리형 정책을 생성하려면 시간과 전문 지식이 필요합니다. 빠르게 시작하려면 AWS 관리형 정책을 사용하면 됩니다. 이 정책은 일반적인 사용 사례를 다루며 사용자의 AWS 계정에서 사용할 수 있습니다. AWS 관리형 정책에 대한 자세한 정보는 IAM 사용 설명서에서 AWS 관리형 정책을 참조하십시오.

AWS 서비스는 AWS 관리형 정책을 유지하고 업데이트합니다. AWS 관리형 정책에서는 권한을 변경할 수 없습니다. 서비스는 때때로 추가 권한을 AWS관리형 정책에 추가하여 새로운 기능을 지원합니다. 이 유형의 업데이트는 정책이 연결된 모든 자격 증명(권한 세트 및 역할)에 적용됩니다. 서비스는 새로운 기능이 시작되거나 새 작업을 사용할 수 있을 때 AWS 관리형 정책에 업데이트됩니다. 서비스는 AWS 관리형 정책에서 권한을 제거하지 않기 때문에 정책 업데이트로 인해 기존 권한이 손상되지 않습니다.

또한 AWS(은)는 여러 서비스의 직무에 대한 관리형 정책을 지원합니다. 예를 들어 `ReadOnlyAccess` AWS 관리형 정책은 모든 AWS 서비스 및 리소스에 대한 읽기 전용 액세스 권한을 제공합니다. 서비스에서 새 기능을 시작하면 AWS(이)가 새 작업 및 리소스에 대한 읽기 전용 권한을 추가합니다. 직무 정책의 목록과 설명은 IAM 사용 설명서의 직무에 관한 AWS 관리형 정책을 참조하십시오.

주제

- AWS 관리형 정책: AmazonRDSReadOnlyAccess
- AWS 관리형 정책: AmazonRDSFullAccess
- AWS 관리형 정책: AmazonRDSDataFullAccess
- AWS 관리형 정책: AmazonRDSEnhancedMonitoringRole
- AWS 관리형 정책: AmazonRDSPerformanceInsightsReadOnly
- AWS 관리형 정책: AmazonRDSPerformanceInsightsFullAccess
- AWS 관리형 정책: AmazonRDSDirectoryServiceAccess
- AWS 관리형 정책: AmazonRDSServiceRolePolicy
- AWS 관리형 정책: AmazonRDSCustomServiceRolePolicy
- AWS 관리형 정책: AmazonRDSCustomInstanceProfileRolePolicy
- AWS 관리형 정책: AmazonRDSPreviewServiceRolePolicy
- AWS 관리형 정책: AmazonRDSBetaServiceRolePolicy

AWS 관리형 정책: AmazonRDSReadOnlyAccess

이 정책은 AWS Management Console을 통해 Amazon RDS에 대한 읽기 전용 액세스를 허용합니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `rds` - 보안 주체가 Amazon RDS 리소스를 설명하고 Amazon RDS 리소스의 태그를 나열하도록 허용합니다.
- `cloudwatch` - 보안 주체가 Amazon CloudWatch 지표 통계를 가져오도록 허용합니다.
- `ec2` - 보안 주체가 가용 영역 및 네트워킹 리소스를 설명하도록 허용합니다.
- `logs` - 보안 주체가 로그 그룹의 CloudWatch Logs 로그 스트림을 설명하고 CloudWatch Logs 로그 이벤트를 가져오도록 허용합니다.
- `devops-guru` - 보안 주체가 CloudFormation 스택 이름 또는 리소스 태그로 지정되는 Amazon DevOps Guru 적용 범위가 있는 리소스를 설명할 수 있습니다.

JSON 정책 문서를 포함하여 이 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 안내서의 AmazonRDSReadOnlyAccess를 참조하세요.

AWS 관리형 정책: AmazonRDSFullAccess

이 정책은 AWS Management Console을 통해 Amazon RDS에 대한 전체 액세스 권한을 제공합니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `rds` – 보안 주체에게 Amazon RDS에 대한 전체 액세스 권한을 허용합니다.
- `application-autoscaling` - 보안 주체가 Application Auto Scaling 크기 조정 대상 및 정책을 설명하고 관리하도록 허용합니다.
- `cloudwatch` - 보안 주체가 CloudWatch 지표 통계를 가져오고 CloudWatch 경보를 관리하도록 허용합니다.
- `ec2` - 보안 주체가 가용 영역 및 네트워킹 리소스를 설명하도록 허용합니다.
- `logs` - 보안 주체가 로그 그룹의 CloudWatch Logs 로그 스트림을 설명하고 CloudWatch Logs 로그 이벤트를 가져오도록 허용합니다.
- `outposts` - 보안 주체가 AWS Outposts 인스턴스 유형을 가져오도록 허용합니다.

- `pi` - 보안 주체가 성능 개선 도우미 지표를 가져오도록 허용합니다.
- `sns` - 보안 주체에게 Amazon Simple Notification Service(Amazon SNS) 구독 및 주제를 허용하고 Amazon SNS 메시지를 게시하도록 허용합니다.
- `devops-guru` - 보안 주체가 CloudFormation 스택 이름 또는 리소스 태그로 지정되는 Amazon DevOps Guru 적용 범위가 있는 리소스를 설명할 수 있습니다.

JSON 정책 문서를 포함하여 이 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 안내서의 AmazonRDSFullAccess를 참조하세요.

AWS 관리형 정책: AmazonRDSDataFullAccess

이 정책은 특정 AWS 계정의 Aurora Serverless 클러스터에서 데이터 API 및 쿼리 편집기를 사용할 수 있는 전체 액세스 권한을 허용합니다. 이 정책은 AWS 계정가 AWS Secrets Manager에서 암호 값을 가져오도록 허용합니다.

`AmazonRDSDataFullAccess` 정책을 IAM 보안 인증에 연결할 수 있습니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `dbqms` - 보안 주체에게 쿼리를 액세스, 생성, 삭제, 설명, 업데이트를 할 수 있도록 허용합니다. Database Query Metadata Service(dbqms)는 내부 전용 서비스입니다. Amazon RDS를 포함한 여러 AWS 서비스에 사용되는 AWS Management Console의 쿼리 편집기에 대해 최근 쿼리와 저장된 쿼리를 제공합니다.
- `rds-data` - 보안 주체가 Aurora Serverless 데이터베이스에 SQL 문을 실행하도록 허용합니다.
- `secretsmanager` - 보안 주체가 AWS Secrets Manager에서 암호 값을 가져오도록 허용합니다.

JSON 정책 문서를 포함하여 이 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 안내서의 AmazonRDSDataFullAccess를 참조하세요.

AWS 관리형 정책: AmazonRDSEnhancedMonitoringRole

이 정책은 Amazon RDS Enhanced Monitoring을 위해 Amazon CloudWatch Logs 로그에 대한 액세스를 제공합니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `logs` - 보안 주체가 CloudWatch Logs 로그 그룹 및 보존 정책을 생성하고 로그 그룹의 CloudWatch Logs 로그 스트림을 생성 및 설명하도록 허용합니다. 또한 보안 주체가 CloudWatch Logs 로그 이벤트를 배치하고 가져오도록 허용합니다.

JSON 정책 문서를 포함하여 이 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 안내서의 AmazonRDSEnhancedMonitoringRole을 참조하세요.

AWS 관리형 정책: AmazonRDSPerformanceInsightsReadOnly

이 정책은 Amazon RDS DB 인스턴스 및 Amazon Aurora DB 클러스터에 대한 Amazon RDS 성능 개선 도우미에 읽기 전용 액세스를 제공합니다.

이제 이 정책에 `Sid`(문 ID)가 정책 문의 식별자로 포함됩니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `rds` - 보안 주체가 Amazon RDS DB 인스턴스와 Amazon Aurora DB 클러스터를 설명하도록 허용합니다.
- `pi` - 보안 주체가 Amazon RDS 성능 개선 도우미 API를 호출하고 성능 개선 도우미 지표에 액세스하도록 허용합니다.

JSON 정책 문서를 포함하여 이 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 안내서의 AmazonRDSPerformanceInsightsReadOnly를 참조하세요.

AWS 관리형 정책: AmazonRDSPerformanceInsightsFullAccess

이 정책은 Amazon RDS DB 인스턴스 및 Amazon Aurora DB 클러스터에 대해 Amazon RDS 성능 개선 도우미에 전체 액세스를 제공합니다.

이제 이 정책에 `Sid`(문 ID)가 정책 문의 식별자로 포함됩니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `rds` - 보안 주체가 Amazon RDS DB 인스턴스와 Amazon Aurora DB 클러스터를 설명하도록 허용합니다.

- `pi` - 보안 주체가 Amazon RDS 성능 개선 도우미 API를 호출하고 성능 분석 보고서를 생성, 확인 및 삭제하도록 허용합니다.
- `cloudwatch` - 보안 주체가 모든 Amazon CloudWatch 지표를 나열하고 지표 데이터와 통계를 가져오도록 허용합니다.

JSON 정책 문서를 포함하여 이 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 가이드의 AmazonRDSPerformanceInsightsFullAccess를 참조하세요.

AWS 관리형 정책: AmazonRDSDirectoryServiceAccess

이 정책은 Amazon RDS에서 AWS Directory Service를 호출하도록 허용합니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `ds` - 보안 주체가 AWS Directory Service 디렉터리를 설명하고 AWS Directory Service 디렉터리에 대한 권한 부여를 제어하도록 허용합니다.

JSON 정책 문서를 포함하여 이 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 안내서의 AmazonRDSDirectoryServiceAccess를 참조하세요.

AWS 관리형 정책: AmazonRDSServiceRolePolicy

`AmazonRDSServiceRolePolicy` 정책을 IAM 엔터티에 연결할 수 없습니다. 이 정책은 Amazon RDS에 사용자를 대신하여 작업을 수행할 수 있도록 하는 서비스 연결 역할에 연결됩니다. 자세한 내용은 Amazon RDS에 대한 서비스 연결 역할 권한 단원을 참조하십시오.

AWS 관리형 정책: AmazonRDSCustomServiceRolePolicy

`AmazonRDSCustomServiceRolePolicy` 정책을 IAM 엔터티에 연결할 수 없습니다. 이 정책은 Amazon RDS가 RDS DB 리소스를 대신하여 AWS 서비스를 직접 호출할 수 있는 서비스 연결 역할에 연결되어 있습니다.

이 정책에는 다음 권한이 포함되어 있습니다.

- `ec2` - RDS Custom이 특정 시점 복원 기능을 제공하는 DB 인스턴스에서 백업 작업을 수행할 수 있도록 허용합니다.
- `secretsmanager` - RDS Custom이 RDS Custom으로 만든 DB 인스턴스별 암호를 관리할 수 있도록 허용합니다.

- `cloudwatch` - RDS Custom이 CloudWatch 에이전트를 통해 DB 인스턴스 지표와 로그를 CloudWatch에 업로드할 수 있도록 허용합니다.
- `events`, `sqs` - RDS Custom이 DB 인스턴스에 대한 상태 정보를 보내고 받을 수 있도록 허용합니다.

JSON 정책 문서를 포함하여 이 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 안내서의 AmazonRDSCustomServiceRolePolicy를 참조하세요.

AWS 관리형 정책: AmazonRDSCustomInstanceProfileRolePolicy

`AmazonRDSCustomInstanceProfileRolePolicy`를 IAM 엔터티에 연결하면 안 됩니다. Amazon RDS Custom DB 인스턴스에 다양한 자동화 작업 및 데이터베이스 관리 작업을 수행할 수 있는 권한을 부여하는 데 사용되는 인스턴스 프로파일 역할에만 연결해야 합니다. RDS Custom 인스턴스 생성 중에 인스턴스 프로파일을 `custom-iam-instance-profile` 파라미터로 전달하면 RDS Custom에서 해당 인스턴스 프로파일을 DB 인스턴스에 연결합니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `ssm`, `ssmmessages`, `ec2messages` - RDS Custom이 Systems Manager를 통해 DB 인스턴스에서 통신하고, 자동화를 실행하고, 에이전트를 유지 관리할 수 있도록 허용합니다.
- `ec2`, `s3` - RDS Custom이 특정 시점 복원 기능을 제공하는 DB 인스턴스에서 백업 작업을 수행할 수 있도록 허용합니다.
- `secretsmanager` - RDS Custom이 RDS Custom으로 만든 DB 인스턴스별 암호를 관리할 수 있도록 허용합니다.
- `cloudwatch`, `logs` - RDS Custom이 CloudWatch 에이전트를 통해 DB 인스턴스 지표와 로그를 CloudWatch에 업로드할 수 있도록 허용합니다.
- `events`, `sqs` - RDS Custom이 DB 인스턴스에 대한 상태 정보를 보내고 받을 수 있도록 허용합니다.
- `kms` - RDS Custom이 인스턴스별 KMS 키를 사용하여 RDS Custom이 관리하는 암호 및 S3 개체의 암호화를 수행할 수 있도록 허용합니다.

JSON 정책 문서를 포함하여 이 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 가이드의 AmazonRDSCustomInstanceProfileRolePolicy를 참조하세요.

AWS 관리형 정책: AmazonRDSPreviewServiceRolePolicy

`AmazonRDSPreviewServiceRolePolicy`를 IAM 엔터티에 연결하면 안 됩니다. 이 정책은 Amazon RDS가 RDS DB 리소스를 대신하여 AWS 서비스를 직접 호출할 수 있는 서비스 연결 역할에 연결되어 있습니다. 자세한 내용은 Amazon RDS 미리 보기에 대한 서비스 연결 역할 단원을 참조하십시오.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `ec2` - 보안 주체가 가용 영역 및 네트워킹 리소스를 설명하도록 허용합니다.
- `secretsmanager` - 보안 주체가 AWS Secrets Manager에서 암호 값을 가져오도록 허용합니다.
- `cloudwatch`, `logs` - Amazon RDS가 CloudWatch 에이전트를 통해 DB 인스턴스 지표와 로그를 CloudWatch에 업로드할 수 있도록 허용합니다.

JSON 정책 문서를 포함하여 이 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 안내서의 AmazonRDSDataFullAccess를 참조하세요.

AWS 관리형 정책: AmazonRDSBetaServiceRolePolicy

`AmazonRDSBetaServiceRolePolicy`를 IAM 엔터티에 연결하면 안 됩니다. 이 정책은 Amazon RDS가 RDS DB 리소스를 대신하여 AWS 서비스를 직접 호출할 수 있는 서비스 연결 역할에 연결되어 있습니다. 자세한 내용은 Amazon RDS 베타에 대한 서비스 연결 역할 권한 단원을 참조하십시오.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `ec2` - Amazon RDS가 특정 시점 복원 기능을 제공하는 DB 인스턴스에서 백업 작업을 수행할 수 있도록 허용합니다.
- `secretsmanager` - Amazon RDS가 Amazon RDS로 만든 DB 인스턴스별 암호를 관리할 수 있도록 허용합니다.
- `cloudwatch`, `logs` - Amazon RDS가 CloudWatch 에이전트를 통해 DB 인스턴스 지표와 로그를 CloudWatch에 업로드할 수 있도록 허용합니다.

JSON 정책 문서를 포함하여 이 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 안내서의 AmazonRDSBetaServiceRolePolicy를 참조하세요.

AWS 관리형 정책에 대한 Amazon RDS 업데이트

이 서비스가 이러한 변경 내용을 추적하기 시작한 이후부터 Amazon RDS의 AWS 관리형 정책 업데이트에 대한 세부 정보를 봅니다. 이 페이지의 변경 사항에 대한 자동 알림을 받아보려면 Amazon RDS 문서 기록 페이지에서 RSS 피드를 구독하세요.

변경 사항	설명	날짜
AWS 관리형 정책: AmazonRDSPreviewServiceRolePolicy - 기존 정책에 대한 업데이트	Amazon RDS는 `AWSServiceRoleForRDSPreview` 서비스 연결 역할의 `AmazonRDSPreviewServiceRolePolicy` 에서 `sns:Publish` 권한을 제거했습니다. 자세한 내용은 AWS 관리형 정책: AmazonRDSPreviewServiceRolePolicy 단원을 참조하십시오.	2024년 8월 7일
AWS 관리형 정책: AmazonRDSBetaServiceRolePolicy - 기존 정책에 대한 업데이트	Amazon RDS는 `AWSServiceRoleForRDSBeta` 서비스 연결 역할의 `AmazonRDSBetaServiceRolePolicy` 에서 `sns:Publish` 권한을 제거했습니다. 자세한 내용은 AWS 관리형 정책: AmazonRDSBetaServiceRolePolicy 단원을 참조하십시오.	2024년 8월 7일
Amazon RDS Custom에 대한 서비스 연결 역할 권한 - 기존 정책에 대한 업데이트	Amazon RDS는 `AWSServiceRoleForRDSCustom` 서비스 연결 역할의 `AmazonRDSCustomServiceRolePolicy` 에 새로운 권한을 추가했습니다. 권한을 통해 RDS Custom은 다른 AWS 리전의	2024년 7월 18일

변경 사항	설명	날짜
	Amazon RDS 서비스와 통신하고 EC2 이미지를 복사할 수 있습니다. 자세한 내용은 Amazon RDS Custom에 대한 서비스 연결 역할 권한 단원을 참조하십시오.	
AWS 관리형 정책: AmazonRDSServiceRolePolicy - 기존 정책에 대한 업데이트	Amazon RDS는 `AWSServiceRoleForRDS` 서비스 연결 역할의 `AmazonRDSServiceRolePolicy` 에서 `sns:Publish` 권한을 제거했습니다. 자세한 내용은 AWS 관리형 정책: AmazonRDSServiceRolePolicy 단원을 참조하십시오.	2024년 7월 2일
Amazon RDS Custom에 대한 서비스 연결 역할 권한 - 기존 정책에 대한 업데이트	Amazon RDS는 `AWSServiceRoleForRDSCustom` 서비스 연결 역할의 `AmazonRDSCustomServiceRolePolicy` 에 새로운 권한을 추가했습니다. 이 새 권한을 통해 RDS Custom이 RDS Custom 인스턴스에 인스턴스 프로파일로 서비스 역할을 연결할 수 있습니다. 자세한 내용은 Amazon RDS Custom에 대한 서비스 연결 역할 권한 단원을 참조하십시오.	2024년 4월 19일

변경 사항	설명	날짜
Amazon RDS에 대한 AWS 관리형 정책 - 기존 정책에 대한 업데이트	Amazon RDS는 RDS Custom for SQL Server가 기본 데이터베이스 호스트 인스턴스 유형을 수정할 수 있도록 `AWSServiceRoleForRDSCustom` 서비스 연결 역할의 `AmazonRDSCustomServiceRolePolicy` 에 새 권한을 추가했습니다. 또한 RDS는 데이터베이스 호스트의 인스턴스 유형 정보를 가져올 수 있는 `ec2:DescribeInstanceTypes` 권한을 추가했습니다. 자세한 내용은 Amazon RDS에 대한 AWS 관리형 정책 단원을 참조하십시오.	2024년 4월 8일
Amazon RDS에 대한 AWS 관리형 정책 - 새 정책	Amazon RDS는 RDS Custom이 EC2 인스턴스 프로파일을 통해 자동화 작업 및 데이터베이스 관리 작업을 수행할 수 있도록 `AmazonRDSCustomInstanceProfileRolePolicy` 로 이름이 지정된 새 관리형 정책을 추가했습니다. 자세한 내용은 Amazon RDS에 대한 AWS 관리형 정책 단원을 참조하십시오.	2024년 2월 27일

변경 사항	설명	날짜
Amazon RDS에 대한 서비스 연결 역할 권한 -기존 정책 업데이트	Amazon RDS는 `AWSServiceRoleForRDS` 서비스 연결 역할의 `AmazonRDSServiceRolePolicy` 에 새로운 문 ID를 추가했습니다. 자세한 내용은 Amazon RDS에 대한 서비스 연결 역할 권한 단원을 참조하십시오.	2024년 1월 19일
Amazon RDS에 대한 AWS 관리형 정책 - 기존 정책에 대한 업데이트	`AmazonRDSPerformanceInsightsReadOnly` 및 `AmazonRDSPerformanceInsightsFullAccess` 관리형 정책에서 이제 정책 문에 `Sid`(문 ID)를 식별자로 포함합니다. 자세한 정보는 AWS 관리형 정책: AmazonRDSPerformanceInsightsReadOnly 및 AWS 관리형 정책: AmazonRDSPerformanceInsightsFullAccess 섹션을 참조하세요.	2023년 10월 23일

변경 사항	설명	날짜
Amazon RDS에 대한 서비스 연결 역할 권한 -기존 정책 업데이트	Amazon RDS는 `AWSServiceRoleForRDSCustom` 서비스 연결 역할의 `AmazonRDSCustomServiceRolePolicy` 에 새로운 권한을 추가했습니다. 이 새로운 권한을 통해 RDS Custom for Oracle이 EventBridge 관리형 규칙을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 Amazon RDS Custom에 대한 서비스 연결 역할 권한 단원을 참조하십시오.	2023년 9월 20일
Amazon RDS에 대한 AWS 관리형 정책 - 기존 정책에 대한 업데이트	Amazon RDS에서 `AmazonRDSFullAccess` 관리형 정책에 대한 새로운 권한을 추가했습니다. 이 권한을 통해 일정 기간 동안 성능 분석 보고서를 생성, 확인 및 삭제할 수 있습니다. 성능 개선 도우미의 액세스 정책 구성에 대한 자세한 내용은 Performance Insights에 대한 액세스 정책 구성 섹션을 참조하세요.	2023년 8월 17일

변경 사항	설명	날짜
Amazon RDS에 대한 AWS 관리형 정책 – 신규 정책 및 기존 정책에 대한 업데이트	Amazon RDS에서 `AmazonRDSPerformanceInsightsReadOnly` 관리형 정책 및 `AmazonRDSPerformanceInsightsFullAccess` 라는 신규 관리형 정책에 대한 새로운 권한을 추가했습니다. 이러한 권한을 통해 일정 기간 동안 성능 개선 도우미를 분석하고, 권장 사항과 함께 분석 결과를 보고, 보고서를 삭제할 수 있습니다. 성능 개선 도우미의 액세스 정책 구성에 대한 자세한 내용은 Performance Insights에 대한 액세스 정책 구성 섹션을 참조하세요.	2023년 8월 16일
Amazon RDS에 대한 서비스 연결 역할 권한 -기존 정책 업데이트	Amazon RDS는 `AWSServiceRoleForRDSCustom` 서비스 연결 역할의 `AmazonRDSCustomServiceRolePolicy` 에 새로운 권한을 추가했습니다. 이 새로운 권한은 RDS Custom for Oracle이 DB 스냅샷을 사용하도록 허용합니다. 자세한 내용은 Amazon RDS Custom에 대한 서비스 연결 역할 권한 단원을 참조하십시오.	2023년 6월 23일

변경 사항	설명	날짜
Amazon RDS에 대한 서비스 연결 역할 권한 -기존 정책 업데이트	Amazon RDS는 `AWSServiceRoleForRDSCustom` 서비스 연결 역할의 `AmazonRDSCustomServiceRolePolicy` 에 새로운 권한을 추가했습니다. 이 새로운 권한은 RDS Custom for Oracle이 DB 스냅샷을 사용하도록 허용합니다. 자세한 내용은 Amazon RDS Custom에 대한 서비스 연결 역할 권한 단원을 참조하십시오.	2023년 6월 23일
Amazon RDS에 대한 서비스 연결 역할 권한 -기존 정책 업데이트	Amazon RDS는 `AWSServiceRoleForRDSCustom` 서비스 연결 역할의 `AmazonRDSCustomServiceRolePolicy` 에 새로운 권한을 추가했습니다. 이 새로운 권한은 RDS Custom이 네트워크 인터페이스를 생성하도록 허용합니다. 자세한 내용은 Amazon RDS Custom에 대한 서비스 연결 역할 권한 단원을 참조하십시오.	2023년 5월 30일

변경 사항	설명	날짜
Amazon RDS에 대한 서비스 연결 역할 권한 -기존 정책 업데이트	Amazon RDS는 `AWSServiceRoleForRDSCustom` 서비스 연결 역할의 `AmazonRDSCustomServiceRolePolicy` 에 새로운 권한을 추가했습니다. RDS Custom은 이러한 새 권한을 사용하여 Amazon EBS를 호출해 스토리지 할당량을 확인할 수 있습니다. 자세한 내용은 Amazon RDS Custom에 대한 서비스 연결 역할 권한 단원을 참조하십시오.	2023년 4월 18일

변경 사항	설명	날짜
Amazon RDS에 대한 서비스 연결 역할 권한 -기존 정책 업데이트	Amazon RDS Custom에 Amazon SQS와의 통합을 위해 `AWSServiceRoleForRDSCustom` 서비스 연결 역할의 `AmazonRDSCustomServiceRolePolicy` 에 대한 새로운 권한이 추가되었습니다. RDS Custom을 사용하려면 Amazon SQS와 통합하여 고객 계정의 SQS 대기열을 생성하고 관리해야 합니다. SQS 대기열 이름은 `do-not-delete-rds-custom-[identifier]` 형식을 따르며 `Amazon RDS Custom`이라는 태그가 지정됩니다. RDS Custom이 인스턴스에 연결된 볼륨에 대한 백업을 생성할 수 있도록 `ec2:CreateSnapshot` 에 대한 권한도 추가되었습니다. 자세한 내용은 Amazon RDS Custom에 대한 서비스 연결 역할 권한 단원을 참조하십시오.	2023년 4월 6일

변경 사항	설명	날짜
Amazon RDS에 대한 AWS 관리형 정책 -기존 정책 업데이트	Amazon RDS에서는 새로운 Amazon CloudWatch 네임스페이스 `ListMetrics` 를 `AmazonRDSFullAccess` 및 `AmazonRDSReadOnlyAccess` 에 추가했습니다. 이 네임스페이스는 Amazon RDS가 특정한 리소스 사용량 지표를 나열하는 데 필요합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 CloudWatch 리소스에 대한 액세스 권한 관리 개요를 참조하세요.	2023년 4월 4일
Amazon RDS에 대한 AWS 관리형 정책 -기존 정책 업데이트	Amazon RDS는 RDS 콘솔에서 Amazon DevOps Guru를 사용할 수 있도록 `AmazonRDSFullAccess` 및 `AmazonRDSReadOnlyAccess` 관리형 정책에 새로운 권한을 추가했습니다. 이 권한은 DevOps Guru의 결과를 표시하는 데 필요합니다. 자세한 내용은 AWS 관리형 정책에 대한 Amazon RDS 업데이트를 참조하세요.	2023년 3월 30일

변경 사항	설명	날짜
Amazon RDS에 대한 서비스 연결 역할 권한 -기존 정책 업데이트	Amazon RDS는 AWS Secrets Manager와의 통합을 위해 `AWSServiceRoleForRDS` 서비스 연결 역할의 `AmazonRDSServiceRolePolicy` 에 새로운 권한을 추가했습니다. Secrets Manager에서 마스터 사용자 암호를 관리하려면 RDS와 Secrets Manager의 통합이 필요합니다. 암호는 예약된 명명 규칙을 사용하며 고객 업데이트를 제한합니다. 자세한 내용은 Amazon RDS 및 AWS Secrets Manager를 통한 암호 관리 단원을 참조하십시오.	2022년 12월 22일
Amazon RDS에 대한 서비스 연결 역할 권한 -기존 정책 업데이트	Amazon RDS는 `AWSServiceRoleForRDSCustom` 서비스 연결 역할의 `AmazonRDSCustomServiceRolePolicy` 에 새로운 권한을 추가했습니다. RDS Custom은 DB 클러스터를 지원합니다. 정책의 이러한 새 권한을 통해 RDS Custom이 DB 클러스터를 대신하여 AWS 서비스를 호출할 수 있습니다. 자세한 내용은 Amazon RDS Custom에 대한 서비스 연결 역할 권한 단원을 참조하십시오.	2022년 11월 9일

변경 사항	설명	날짜
Amazon RDS에 대한 서비스 연결 역할 권한 - 기존 정책에 대한 업데이트	Amazon RDS는 AWS Secrets Manager와의 통합을 위해 `AWSServiceRoleForRDS` 서비스 연결 역할에 새로운 권한을 추가했습니다. SSRS(SQL Server Reporting Services) 이메일이 RDS에서 작동하려면 Secrets Manager와 통합해야 합니다. SSRS 이메일은 고객을 대신하여 암호를 생성합니다. 암호는 예약된 명명 규칙을 사용하며 고객 업데이트를 제한합니다. 자세한 정보는 SSRS 이메일을 사용하여 보고서 보내기을 참조하세요.	2022년 8월 26일
Amazon RDS에 대한 서비스 연결 역할 권한 -기존 정책 업데이트	Amazon RDS에서는 `PutMetricData` 용 `AmazonRDSPreviewServiceRolePolicy` 에 새로운 Amazon CloudWatch 네임스페이스를 추가했습니다. 이 네임스페이스는 Amazon RDS가 리소스 사용량 지표를 게시하는 데 필요합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 조건 키를 사용하여 CloudWatch 네임스페이스에 대한 액세스 제한을 참조하세요.	2022년 6월 7일

변경 사항	설명	날짜
Amazon RDS에 대한 서비스 연결 역할 권한 -기존 정책 업데이트	Amazon RDS에서는 `PutMetricData` 용 `AmazonRDSBetaServiceRolePolicy` 에 새로운 Amazon CloudWatch 네임스페이스를 추가했습니다. 이 네임스페이스는 Amazon RDS가 리소스 사용량 지표를 게시하는 데 필요합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 조건 키를 사용하여 CloudWatch 네임스페이스에 대한 액세스 제한을 참조하세요.	2022년 6월 7일
Amazon RDS에 대한 서비스 연결 역할 권한 -기존 정책 업데이트	Amazon RDS에서는 `PutMetricData` 용 `AWSServiceRoleForRDS` 에 새로운 Amazon CloudWatch 네임스페이스를 추가했습니다. 이 네임스페이스는 Amazon RDS가 리소스 사용량 지표를 게시하는 데 필요합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 조건 키를 사용하여 CloudWatch 네임스페이스에 대한 액세스 제한을 참조하세요.	2022년 4월 22일

변경 사항	설명	날짜
Amazon RDS에 대한 서비스 연결 역할 권한 -기존 정책 업데이트	고객 소유 IP 풀 및 로컬 게이트웨이 라우팅 테이블(LGW-RTB)에 대한 권한을 관리하기 위해 Amazon RDS에서 `AWSServiceRoleForRDS` 서비스 연결 역할에 새 권한을 추가했습니다. 이 권한은 RDS on Outposts가 Outposts의 여러 로컬 네트워크에서 다중 AZ 복제를 수행하는 데 필요합니다. 자세한 내용은 AWS Outposts의 Amazon RDS 다중 AZ 배포 작업 단원을 참조하십시오.	2022년 4월 19일
ID 기반 정책 -기존 정책 업데이트	LGW-RTB에 대한 권한을 설명하기 위해 Amazon RDS에서 `AmazonRDSFullAccess` 관리형 정책에 새 권한을 추가했습니다. 이 권한은 RDS on Outposts가 Outposts의 여러 로컬 네트워크에서 다중 AZ 복제를 수행하기 위해 권한을 설명하는 데 필요합니다. 자세한 내용은 AWS Outposts의 Amazon RDS 다중 AZ 배포 작업 단원을 참조하십시오.	2022년 4월 19일

변경 사항	설명	날짜
Amazon RDS에 대한 AWS 관리형 정책 - 새 정책	Amazon RDS에서는 Amazon RDS가 DB 인스턴스를 대신하여 AWS를 호출할 수 있도록 `AmazonRDSPerformanceInsightsReadOnly` 라는 새로운 관리형 정책을 추가했습니다. 성능 개선 도우미의 액세스 정책 구성에 대한 자세한 내용은 Performance Insights에 대한 액세스 정책 구성 섹션을 참조하세요.	2022년 3월 10일
Amazon RDS에 대한 서비스 연결 역할 권한 -기존 정책 업데이트	Amazon RDS에서는 `PutMetricData` 용 `AWSServiceRoleForRDS` 에 새로운 Amazon CloudWatch 네임스페이스를 추가했습니다. Amazon DocumentDB(MongoDB 호환) 및 Amazon Neptune에서 CloudWatch 지표를 게시하려면 이러한 네임스페이스가 필요합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 조건 키를 사용하여 CloudWatch 네임스페이스에 대한 액세스 제한을 참조하세요.	2022년 3월 4일

변경 사항	설명	날짜
Amazon RDS Custom에 대한 서비스 연결 역할 권한 - 새 정책	Amazon RDS에서는 RDS Custom이 DB 인스턴스를 대신하여 AWS 서비스를 호출할 수 있도록 `AWSServiceRoleForRDSCustom` 라는 새로운 서비스 연결 역할을 추가했습니다.	2021년 10월 26일
Amazon RDS에서 변경 사항 추적 시작	Amazon RDS가 AWS 관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2021년 10월 26일

교차 서비스 혼동된 대리자 문제 방지

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에 작업을 수행하도록 강요할 수 있는 보안 문제입니다. AWS에서는 교차 서비스 가장으로 인해 혼동된 대리자 문제가 발생할 수 있습니다.

교차 서비스 가장은 한 서비스(호출하는 서비스)가 다른 서비스(호출되는 서비스)를 직접적으로 호출할 때 발생할 수 있습니다. 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 호출 서비스를 조작할 수 있습니다. 이를 방지하기 위해 AWS에서는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 보안 주체를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 제공합니다. 자세한 내용은 IAM 사용 설명서의 혼동된 대리자 문제를 참조하세요.

Amazon RDS가 다른 서비스에 제공하는 리소스에 대한 권한을 제한하려면 리소스 정책에서 `aws:SourceArn` 및 `aws:SourceAccount` 전역 조건 컨텍스트 키를 사용하는 것이 좋습니다.

경우에 따라 `aws:SourceArn` 값에 계정 ID가 포함되지 않습니다. Amazon S3 버킷에 Amazon 리소스 이름(ARN)을 사용하는 경우를 예로 들 수 있습니다. 이러한 경우 두 전역 조건 컨텍스트 키를 모두 사용하여 권한을 제한해야 합니다. 경우에 따라 두 전역 조건 컨텍스트 키를 모두 사용하고 `aws:SourceArn` 값에 계정 ID가 포함됩니다. 이러한 경우 `aws:SourceAccount` 값과 `aws:SourceArn`의 계정이 동일한 정책 문에서 사용될 때 동일한 계정 ID를 사용하는지 확인합니다. 하나의 리소스만 교차 서비스 액세스와 연결되게 하려는 경우 `aws:SourceArn`을 사용합니다. 지정된 AWS 계정의 모든 리소스가 교차 서비스 사용과 연결되게 하려는 경우 `aws:SourceAccount`를 사용합니다.

`aws:SourceArn`의 값이 Amazon RDS 리소스 유형에 대한 ARN인지 확인하세요. 자세한 내용은 Amazon RDS의 Amazon 리소스 이름(ARN) 섹션을 참조하세요.

혼동된 대리인 문제로부터 보호하는 가장 효과적인 방법은 리소스의 전체 ARN이 포함된 `aws:SourceArn` 글로벌 조건 컨텍스트 키를 사용하는 것입니다. 경우에 따라 리소스의 전체 ARN을 모르거나 여러 리소스를 지정하고 있을 수 있습니다. 이러한 경우 ARN의 알 수 없는 부분에 대해 와일드카드(*)와 함께 `aws:SourceArn` 전역 컨텍스트 조건 키를 사용하세요. 예를 들면, `arn:aws:rds:*:123456789012:*`입니다.

다음 예에서는 Amazon RDS에서 `aws:SourceArn` 및 `aws:SourceAccount` 전역 조건 컨텍스트 키를 사용하여 혼동된 대리자 문제를 방지하는 방법을 보여줍니다.

JSON

```
{
```

```
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "rds.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:rds:us-east-1:123456789012:db:mydbinstance"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

aws:SourceArn 및 aws:SourceAccount 전역 조건 컨텍스트 키를 사용하는 정책의 추가 예는 다음 섹션을 참조하세요.

- Amazon SNS 주제에 알림을 게시할 권한 부여
- 기본 백업 및 복원을 위한 IAM 역할 수동으로 만들기
- SQL Server DB 인스턴스에 대한 Windows 인증 설정
- RDS for SQL Server와 S3를 통합하기 위한 사전 요구 사항
- 수동으로 SQL Server Audit에 대한 IAM 역할 생성
- Amazon S3와 RDS for Oracle 통합을 위한 IAM 권한 구성
- Amazon S3 버킷에 대한 액세스 권한 설정(PostgreSQL 가져오기)
- Amazon S3 버킷에 대한 액세스 권한 설정(PostgreSQL 내보내기)

MariaDB, MySQL 및 PostgreSQL IAM 데이터베이스 인증

AWS Identity and Access Management(IAM) 데이터베이스 인증을 사용하여 DB 인스턴스에 인증할 수 있습니다. IAM 데이터베이스 인증은 MariaDB, MySQL, PostgreSQL과 함께 작동합니다. 이러한 인증 방식은 DB 인스턴스에 연결할 때 암호를 사용할 필요 없습니다. 대신에 인증 토큰을 사용합니다.

인증 토큰이란 요청이 있을 때 Amazon RDS가 생성하는 고유 문자열입니다. 인증 토큰은 AWS 서명 버전 4를 통해 생성됩니다. 각 토큰의 수명은 15분입니다. 인증을 외부에서 IAM을 사용해 관리하기 때문에 사용자 자격 증명을 데이터베이스에 저장할 필요도 없습니다. 또한 표준 데이터베이스 인증 방식도 사용 가능합니다. 토큰은 인증에만 사용되며 설정된 후에는 세션에 영향을 주지 않습니다.

IAM 데이터베이스 인증은 다음과 같은 이점이 있습니다.

- 데이터베이스를 오가는 네트워크 트래픽은 SSL(Secure Sockets Layer) 또는 TLS(Transport Layer Security)를 통해 암호화됩니다. Amazon RDS에서 SSL/TLS를 사용하는 방법에 대한 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 단원을 참조하십시오.
- 데이터베이스 리소스에 대한 액세스는 DB 인스턴스에서 개별적으로 관리할 필요 없이 IAM을 통해 중앙에서 관리할 수 있습니다.
- Amazon EC2에서 실행되는 애플리케이션의 경우, 암호가 아닌 EC2 인스턴스용 프로파일 자격 증명을 사용해 데이터베이스에 액세스하기 때문에 보안을 더욱 강화하는 효과가 있습니다.

일반적으로 애플리케이션에서 초당 200개 미만의 연결을 생성하고 애플리케이션 코드에서 사용자 이름과 암호를 직접 관리하지 않으려는 경우 IAM 데이터베이스 인증을 사용하는 것이 좋습니다.

Amazon Web Services(AWS) JDBC 드라이버는 IAM 데이터베이스 인증을 지원합니다. 자세한 내용은 Amazon Web Services (AWS) JDBC Driver GitHub repository에서 AWS IAM Authentication Plugin을 참조하세요.

Amazon Web Services(AWS) Python 드라이버는 IAM 데이터베이스 인증을 지원합니다. 자세한 내용은 Amazon Web Services (AWS) Python Driver GitHub repository에서 AWS IAM Authentication Plugin을 참조하세요.

다음 주제를 탐색하여 DB 인증을 위한 IAM을 설정하는 프로세스를 알아봅니다.

- IAM 데이터베이스 인증의 활성화 및 비활성화
- IAM 데이터베이스 액세스를 위한 IAM 정책 생성 및 사용
- IAM 인증을 사용하여 데이터베이스 계정 생성

- IAM 인증을 사용하여 DB 인스턴스에 연결

리전 및 버전 사용 가능 여부

기능 가용성 및 해당 지원은 각 데이터베이스 엔진의 특정 버전에 따라 다릅니다. Amazon RDS 및 IAM 데이터베이스 인증을 통한 엔진, 버전 및 리전 가용성에 대한 자세한 내용은 Amazon RDS에서 IAM 데이터베이스 인증을 지원하는 리전 및 DB 엔진 섹션을 참조하세요.

CLI 및 SDK 지원

AWS CLI 및 다음 언어별 AWS SDK에서 IAM 데이터베이스 인증을 사용할 수 있습니다.

- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby

IAM 데이터베이스 인증 방식의 제한 사항

IAM 데이터베이스 인증을 사용하는 경우, 다음 한도가 적용됩니다.

- IAM 데이터베이스 인증은 초당 200개의 연결로 연결을 스로틀링합니다.

  동일한 인증 토큰을 사용하는 연결은 제한되지 않습니다. 가능하면 인증 토큰을 재사용하는 것이 좋습니다.
- 현재 IAM 데이터베이스 인증은 모든 전역 조건 컨텍스트 키를 지원하지 않습니다.

  전역 조건 컨텍스트 키에 대한 자세한 내용은 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.
- PostgreSQL의 경우 IAM 역할(`rds_iam`)이 사용자(마스터 사용자 RDS 포함)에 추가되면 IAM 인증이 암호 인증보다 우선하므로, 사용자는 IAM 사용자로 로그인해야 합니다.

- PostgreSQL의 경우 Amazon RDS는 IAM 및 Kerberos 인증 방법을 모두 동시에 활성화하는 것을 지원하지 않습니다.
- PostgreSQL의 경우 IAM 인증을 사용하여 복제 연결을 설정할 수 없습니다.
- 인증 토큰을 생성할 때는 DB 인스턴스 엔드포인트 대신 사용자 지정 Route 53 DNS 레코드를 사용할 수 없습니다.
- CloudWatch와 CloudTrail은 IAM 인증을 로깅하지 않습니다. 이러한 서비스는 데이터베이스 연결을 활성화하도록 IAM 역할을 승인하는 `generate-db-auth-token` API 직접 호출을 추적하지 않습니다.
- IAM DB 권한 부여에는 데이터베이스 인스턴스의 컴퓨팅 리소스가 필요합니다. 신뢰할 수 있는 연결을 위해 데이터베이스에 300MiB 및 1,000MiB 사이의 추가 메모리가 있어야 합니다. 워크로드에 필요한 메모리를 보려면 IAM DB 인증을 사용하기 전과 후에 확장 모니터링 프로세스 목록에서 RDS 프로세스에 대한 RES 열을 비교합니다. RDS 콘솔에서 OS 지표 보기을(를) 참조하세요.

  버스트 가능 클래스 인스턴스를 사용하는 경우 버퍼 및 캐시와 같은 다른 파라미터에서 사용되는 메모리를 동일한 양만큼 줄여 메모리가 부족하지 않도록 합니다.
- RDS for MySQL의 경우 IAM 인증으로 구성한 데이터베이스 사용자에 암호 기반 인증을 사용할 수 없습니다.
- 엔진에 상관없이 RDS on Outposts에서는 IAM DB 인증이 지원되지 않습니다.

IAM 데이터베이스 인증에 대한 권장 사항

IAM 데이터베이스 인증을 사용할 경우 다음을 따르는 게 좋습니다.

- 애플리케이션에 초당 200개 미만의 새 IAM 데이터베이스 인증 연결이 필요한 경우에는 IAM 데이터베이스 인증 방식을 사용합니다.

  Amazon RDS에서 작동하는 데이터베이스 엔진은 초당 인증 횟수에 제한이 없습니다. 하지만 IAM 데이터베이스 인증 방식을 사용할 때는 애플리케이션이 인증 토큰을 생성해야 합니다. 이렇게 생성된 토큰은 애플리케이션이 DB 인스턴스에 연결하는 데 사용됩니다. 초당 허용되는 새 연결의 최대 수를 초과하면 IAM 데이터베이스 인증에 오버헤드가 추가로 발생하여 연결 병목 현상이 발생할 수 있습니다.

  지속적인 연결 생성을 줄이려면 애플리케이션에서 연결 풀링을 사용하는 것이 좋습니다. 이렇게 하면 IAM DB 인증으로 인한 오버헤드를 줄이고 애플리케이션이 기존 연결을 재사용할 수 있습니다. 또는 이러한 사용 사례에 RDS 프록시를 사용하는 것도 좋습니다. RDS 프록시에는 추가 비용이 부과됩니다. RDS 프록시 요금을 참조하세요.

- IAM 데이터베이스 인증 토큰의 크기는 IAM 태그 수, IAM 서비스 정책, ARN 길이, 기타 IAM 및 데이터베이스 속성 등 여러 요소에 따라 달라집니다. 이 토큰의 최소 크기는 일반적으로 약 1KB지만 더 클 수도 있습니다. 이 토큰은 IAM 인증을 사용하는 데이터베이스에 대한 연결 문자열에서 암호로 사용되므로 데이터베이스 드라이버(예: ODBC) 및/또는 어떤 도구도 크기로 인해 이 토큰을 제한하거나 자르지 않도록 해야 합니다. 토큰이 잘리면 데이터베이스와 IAM에서 수행하는 인증 검증이 실패합니다.
- IAM 데이터베이스 인증 토큰을 생성할 때 임시 보안 인증 정보를 사용하는 경우, IAM 데이터베이스 인증 토큰을 사용하여 연결을 요청할 때 임시 보안 인증 정보가 여전히 유효해야 합니다.

지원되지 않는 AWS 전역 조건 컨텍스트 키

IAM 데이터베이스 인증은 다음과 같은 AWS 전역 조건 컨텍스트 키의 하위 집합을 지원하지 않습니다.

- `aws:Referer`
- `aws:SourceIp`
- `aws:SourceVpc`
- `aws:SourceVpce`
- `aws:UserAgent`
- `aws:VpcSourceIp`

자세한 내용은 IAM 사용 설명서의 AWS 글로벌 조건 컨텍스트 키를 참조하세요.

IAM 데이터베이스 인증의 활성화 및 비활성화

DB 인스턴스에서는 기본적으로 IAM 데이터베이스 인증이 비활성화되어 있습니다. IAM 데이터베이스 인증은 AWS Management Console, AWS CLI 또는 API를 사용하여 활성화하거나 비활성화할 수 있습니다.

다음 작업 중 하나를 수행할 때 IAM 데이터베이스 인증을 활성화할 수 있습니다.

- IAM 데이터베이스 인증이 활성화된 새 DB 인스턴스를 생성하려면 Amazon RDS DB 인스턴스 생성 단원을 참조하세요.
- IAM 데이터베이스 인증을 사용하도록 DB 인스턴스를 수정하려면 Amazon RDS DB 인스턴스 수정 단원을 참조하세요.
- IAM 데이터베이스 인증이 활성화된 스냅샷에서 DB 인스턴스를 복원하려면 DB 인스턴스 복원 섹션을 참조하세요.

- IAM 데이터베이스 인증이 사용 설정된 시점으로 DB 인스턴스를 복원하려면 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 섹션을 참조하세요.

PostgreSQL DB 인스턴스에 대한 IAM 인증을 사용하려면 SSL 값이 1이어야 합니다. SSL 값이 0인 경우 PostgreSQL DB 인스턴스에 대해 IAM 인증을 활성화할 수 없습니다. PostgreSQL DB 인스턴스에 대해 IAM 인증을 활성화할 경우 SSL 값을 0으로 변경할 수 없습니다.

콘솔

각 생성 또는 수정 워크플로에는 IAM 데이터베이스 인증을 활성화하거나 비활성화할 수 있는 데이터베이스 인증(Database authentication) 섹션이 있습니다. 이 섹션에서 암호 및 IAM 데이터베이스 인증(Password and IAM database authentication)을 선택하여 IAM 데이터베이스 인증을 활성화합니다.

기존 DB 인스턴스에서 IAM 데이터베이스 인증을 활성화하거나 비활성화하려면

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 수정하려는 DB 인스턴스를 선택합니다.

   > **Note**
   >
   > DB 인스턴스가 IAM 인증과 호환되는지 확인합니다. 호환성 요구 사항은 리전 및 버전 사용 가능 여부 단원을 참조하십시오.

4. 수정을 선택합니다.
5. 이 [데이터베이스 인증(Database authentication)] 섹션에서 [암호 및 IAM 데이터베이스 인증(Password and IAM database authentication)]을 선택하여 IAM 데이터베이스 인증을 활성화합니다. 암호 인증 또는 암호 및 Kerberos 인증을 선택하여 IAM 인증을 비활성화합니다.
6. 또한 CloudWatch Logs에 IAM DB 인증 로그 게시를 사용하도록 선택할 수 있습니다. 로그 내보내기에서 iam-db-auth-error 로그 옵션을 선택합니다. CloudWatch Logs에 로그를 게시하면 스토리지가 사용되며 해당 스토리지에 대한 요금이 부과됩니다. 더 이상 필요하지 않은 CloudWatch Logs는 삭제해야 합니다.
7. [Continue]를 선택합니다.
8. 변경 사항을 즉시 적용하려면 수정 예약(Scheduling of modifications) 섹션에서 즉시(Immediately)를 선택합니다.
9. DB 인스턴스 수정 을 선택합니다.

AWS CLI

AWS CLI를 사용하여 새로운 DB 인스턴스를 IAM 인증 방식으로 생성하려면 create-db-instance 명령을 사용하십시오. 다음 예제와 같이 --enable-iam-database-authentication 옵션을 지정합니다.

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --db-instance-class db.m3.medium \
    --engine MySQL \
    --allocated-storage 20 \
    --master-username masterawsuser \
    --manage-master-user-password \
    --enable-iam-database-authentication
```

기존 DB 인스턴스를 업데이트하여 IAM 인증을 사용하게 하거나 사용하지 않게 하려면 AWS CLI 명령 modify-db-instance를 사용합니다. 상황에 따라 --enable-iam-database-authentication 또는 --no-enable-iam-database-authentication 옵션을 지정합니다.

> **Note**
>
> DB 인스턴스가 IAM 인증과 호환되는지 확인합니다. 호환성 요구 사항은 리전 및 버전 사용 가능 여부 단원을 참조하십시오.

기본적으로 Amazon RDS는 다음 유지 관리 기간에 수정 작업을 수행합니다. 이러한 기본 설정을 무시하고 IAM DB 인증을 최대한 빠르게 활성화하려면 --apply-immediately 파라미터를 사용합니다.

다음은 기존 DB 인스턴스일 때 IAM 인증을 바로 활성화하는 방법을 설명한 예제입니다.

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --apply-immediately \
    --enable-iam-database-authentication
```

DB 인스턴스를 복원하는 경우에는 다음 AWS CLI 명령 중 하나를 사용하십시오.

- restore-db-instance-to-point-in-time
- restore-db-instance-from-db-snapshot

IAM 데이터베이스 인증은 기본적으로 원본 스냅샷으로 기본 설정됩니다. 이 설정을 변경하려면 상황에 따라 `--enable-iam-database-authentication` 또는 `--no-enable-iam-database-authentication` 옵션을 설정합니다.

RDS API

API를 사용하여 새로운 DB 인스턴스를 IAM 인증 방식으로 생성하려면 API 작업 `CreateDBInstance`를 사용하십시오. `EnableIAMDatabaseAuthentication` 파라미터를 `true`로 설정합니다.

기존 DB 인스턴스를 업데이트하여 IAM 인증을 사용하게 하거나 사용하지 않게 하려면 API 작업 `ModifyDBInstance`를 사용합니다. `EnableIAMDatabaseAuthentication` 파라미터를 `true`로 설정하여 IAM 인증을 활성화하거나, `false`로 설정하여 비활성화합니다.

> **Note**
>
> DB 인스턴스가 IAM 인증과 호환되는지 확인합니다. 호환성 요구 사항은 리전 및 버전 사용 가능 여부 단원을 참조하십시오.

DB 인스턴스를 복원하는 경우에는 다음 API 작업 중 하나를 사용하십시오.

- `RestoreDBInstanceFromDBSnapshot`
- `RestoreDBInstanceToPointInTime`

IAM 데이터베이스 인증은 기본적으로 원본 스냅샷으로 기본 설정됩니다. 이 설정을 변경하려면 `EnableIAMDatabaseAuthentication` 파라미터를 `true`로 설정하여 IAM 인증을 활성화하거나, 혹은 `false`로 설정하여 비활성화합니다.

IAM 데이터베이스 액세스를 위한 IAM 정책 생성 및 사용

사용자 또는 역할이 DB 인스턴스에 연결할 수 있도록 허용하려면 IAM 정책을 생성해야 합니다. 그런 다음 정책을 권한 세트 또는 역할에 연결합니다.

> **Note**
>
> IAM 정책에 대한 자세한 정보는 Amazon RDS의 자격 증명 및 액세스 관리 단원을 참조하십시오.

다음은 사용자가 IAM 데이터베이스 인증 방식을 사용해 DB 인스턴스 에 연결할 수 있도록 허용하는 정책 예제입니다.

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
             "rds-db:connect"
         ],
         "Resource": [
             "arn:aws:rds-db:us-east-2:1234567890:dbuser:db-ABCDEFGHIJKL01234/
db_user"
         ]
      }
   ]
}
```

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
             "rds-db:connect"
         ],
         "Resource": [
             "arn:aws:rds-db:us-east-2:1234567890:dbuser:cluster-
ABCDEFGHIJKL01234/db_user"
         ]
      }
   ]
}
```

> **Important**
>
> 관리자 권한이 있는 사용자는 IAM 정책에서 명시적 권한 없이도 DB 인스턴스에 액세스할 수 있습니다. DB 인스턴스에 대한 관리자 액세스를 제한하고 싶은 경우에는 더 적은 적정 권한을 가진 IAM 역할을 생성하고 이를 관리자에게 할당할 수 있습니다.

> **Note**
>
> `rds-db:` 접두사를 `rds:`로 시작하는 다른 RDS API 작업 접두사와 혼동하지 마십시오. `rds-db:` 접두사와 `rds-db:connect` 작업은 IAM 데이터베이스 인증 전용입니다. 다른 컨텍스트에서는 유효하지 않습니다.

위의 예제 정책에는 다음 요소와 함께 단일 문이 포함되어 있습니다.

- `Effect` – `Allow`를 지정하여 DB 인스턴스에 대한 액세스를 부여합니다. 액세스를 명시적으로 허용하지 않으면 액세스가 기본적으로 거부됩니다.
- `Action` – `rds-db:connect`를 지정하여 DB 인스턴스에 대한 연결을 허용합니다.
- `Resource` – 하나의 DB 인스턴스의 한 데이터베이스 계정을 기술하는 Amazon 리소스 이름(ARN)을 지정합니다. ARN 형식은 다음과 같습니다.

```
arn:aws:rds-db:region:account-id:dbuser:DbiResourceId/db-user-name
```

이 형식에서 다음 항목을 교체합니다.

- *`region`*은 DB 인스턴스의 AWS 리전입니다. 정책 예제에서 사용되는 AWS 리전은 `us-east-2`입니다.
- *`account-id`*은 DB 인스턴스의 AWS 계정 번호입니다. 정책 예제에서 사용되는 계정 번호는 `1234567890`입니다. 사용자는 DB 인스턴스의 계정과 동일한 계정에 있어야 합니다.

  크로스 계정 액세스를 수행하려면 DB 인스턴스의 계정에서 위에 있는 정책을 사용하여 IAM 역할을 생성하고 다른 계정이 해당 역할을 맡도록 허용합니다.
- *`DbiResourceId`*는 DB 인스턴스의 식별자입니다. 이 식별자는 AWS 리전에 고유하며, 절대로 바뀌지 않습니다. 정책 예제에서 사용되는 식별자는 `db-ABCDEFGHIJKL01234`입니다.

Amazon RDS용 AWS Management Console에서 DB 인스턴스 리소스 ID를 찾으려면 DB 인스턴스를 선택하여 세부 정보를 확인하세요. 그런 다음 구성 탭을 선택합니다. 그러면 리소스 ID가 구성 섹션에 표시됩니다.

그 밖에 다음과 같이 AWS CLI 명령을 사용하여 현재 AWS 리전에 속한 모든 DB 인스턴스의 식별자와 리소스 ID 목록을 조회하는 방법도 있습니다.

```
aws rds describe-db-instances --query "DBInstances[*].
[DBInstanceIdentifier,DbiResourceId]"
```

Amazon Aurora를 사용하는 경우 `DbiResourceId` 대신 `DbClusterResourceId`를 지정하세요. 자세한 내용은 Amazon Aurora 사용 설명서에서 IAM 데이터베이스 액세스를 위한 IAM 정책 생성 및 사용을 참조하세요.

> **Note**
>
> RDS 프록시를 통해 데이터베이스에 연결하는 경우, 프록시 리소스 ID(예: `prx-ABCDEFGHIJKL01234`)를 지정합니다. RDS 프록시에서 IAM 데이터베이스 인증을 사용하는 방법에 대한 자세한 내용은 IAM 인증을 사용하여 프록시에 연결 단원을 참조하십시오.

- *`db-user-name`*은 IAM 인증과 연결할 데이터베이스 계정 이름입니다. 정책 예제에서 사용되는 데이터베이스 계정은 `db_user`입니다.

다른 ARN을 구성하여 다양한 액세스 패턴을 지원할 수 있습니다. 다음 정책에서는 DB 인스턴스에서 서로 다른 데이터베이스 계정 2개에 대한 액세스를 허용합니다.

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
             "rds-db:connect"
```

```
            ],
            "Resource": [
                "arn:aws:rds-db:us-east-2:123456789012:dbuser:db-ABCDEFGHIJKL01234/
jane_doe",
                "arn:aws:rds-db:us-east-2:123456789012:dbuser:db-ABCDEFGHIJKL01234/
mary_roe"
            ]
        }
    ]
}
```

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
             "rds-db:connect"
         ],
         "Resource": [
             "arn:aws:rds-db:us-east-2:123456789012:dbuser:cluster-
ABCDEFGHIJKL01234/jane_doe",
             "arn:aws:rds-db:us-east-2:123456789012:dbuser:cluster-
ABCDEFGHIJKL01234/mary_roe"
         ]
      }
   ]
}
```

다음 정책에서는 "*" 문자를 사용하여 특정 AWS 계정과 AWS 리전의 모든 DB 인스턴스 및 데이터베이스 계정을 일치시킵니다.

JSON

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rds-db:connect"
            ],
            "Resource": [
                "arn:aws:rds-db:us-east-2:1234567890:dbuser:*/*"
            ]
        }
    ]
}
```

다음 정책은 특정 AWS 계정과 AWS 리전의 모든 DB 인스턴스를 일치시킵니다. 하지만 정책에 따라 jane_doe 데이터베이스 계정을 가지고 있는 DB 인스턴스에게만 액세스 권한이 부여됩니다.

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
             "rds-db:connect"
         ],
         "Resource": [
             "arn:aws:rds-db:us-east-2:123456789012:dbuser:*/jane_doe"
         ]
      }
   ]
}
```

사용자 또는 역할은 데이터베이스 사용자가 액세스할 수 있는 데이터베이스에만 액세스할 수 있습니다. 예를 들어 DB 인스턴스에 이름이 dev인 데이터베이스와 test인 데이터베이스가 있다고 가정하겠습니다. 이때 데이터베이스 사용자인 jane_doe가 dev에 대한 액세스 권한만 가지고 있다면 사용자

jane_doe와 함께 해당 DB 인스턴스에 액세스하는 모든 사용자 또는 역할도 dev 액세스 권한만 갖게 됩니다. 이러한 액세스 제한은 테이블, 뷰 등 다른 데이터베이스 객체에 대해서도 똑같이 적용됩니다.

관리자는 지정된 필요한 리소스에서 특정 API 작업을 수행할 수 있는 권한을 엔티티에 부여하는 IAM 정책을 생성해야 합니다. 그런 다음 관리자는 해당 권한이 필요한 권한 세트 또는 역할에 이러한 정책을 연결해야 합니다. 정책에 대한 예시는 Amazon RDS 자격 증명 기반 정책 예 단원을 참조하십시오.

권한 세트 또는 역할에 IAM 정책 연결

데이터베이스 인증을 위한 IAM 정책을 생성하였으면 이제 정책을 권한 세트 또는 역할에 연결해야 합니다. 이번 주제에 대한 자습서는 IAM 사용자 안내서의 첫 번째 고객 관리형 정책 생성 및 연결을 참조하십시오.

자습서를 읽어보면 이번 단원에서 소개하는 정책 예제 중 한 가지를 출발점으로 자신만의 요건에 따라 지정하여 사용할 수 있습니다. 자습서를 끝까지 따르다 보면 연결된 정책을 통해 rds-db:connect 작업이 가능한 권한 세트를 얻게 될 것입니다.

> **Note**
>
> 여러 권한 세트 또는 역할을 동일한 데이터베이스 사용자 계정에 매핑할 수 있습니다. 예를 들어 IAM 정책이 다음과 같은 리소스 ARN을 지정하였다고 가정하겠습니다.
>
> ```
> arn:aws:rds-db:us-east-2:123456789012:dbuser:db-12ABC34DEFG5HIJ6KLMNOP78QR/
> jane_doe
> ```
>
> Jane, Bob 및 Diego에게 정책을 연결하면 각 사용자는 jane_doe 데이터베이스 계정을 사용하여 지정된 DB 인스턴스에 연결할 수 있습니다.

IAM 인증을 사용하여 데이터베이스 계정 생성

IAM 데이터베이스 인증 방식에서는 데이터베이스 암호를 사용자 계정에 할당할 필요 없습니다. 데이터베이스 계정에 매핑되어 있는 사용자를 제거할 경우에는 DROP USER 문으로 데이터베이스 계정도 제거해야 합니다.

> **Note**
>
> IAM 인증에 사용되는 사용자 이름은 데이터베이스에 있는 사용자 이름과 대소문자가 일치해야 합니다.

주제

- MariaDB 및 MySQL에서 IAM 인증 사용
- PostgreSQL에서 IAM 인증 사용

MariaDB 및 MySQL에서 IAM 인증 사용

MariaDB 및 MySQL에서는 AWS에서 제공하는 플러그인인 `AWSAuthenticationPlugin`에서 인증을 처리합니다. 이 플러그인은 IAM과 원활하게 연동되어 사용자를 인증합니다. 마스터 사용자 또는 사용자를 생성하고 권한을 부여할 수 있는 다른 사용자로 DB 인스턴스에 연결합니다. 연결한 후 다음 예제와 같이 `CREATE USER` 문을 실행합니다.

```
CREATE USER 'jane_doe' IDENTIFIED WITH AWSAuthenticationPlugin AS 'RDS';
```

`IDENTIFIED WITH` 절을 사용하면 MariaDB 및 MySQL이 `AWSAuthenticationPlugin`을 통해 데이터베이스 계정(`jane_doe`)을 인증할 수 있습니다. `AS 'RDS'` 절은 인증 방법을 참조합니다. 지정한 데이터베이스 사용자 이름이 IAM 데이터베이스 액세스에 대한 IAM 정책의 리소스와 같은지 확인합니다. 자세한 내용은 IAM 데이터베이스 액세스를 위한 IAM 정책 생성 및 사용 섹션을 참조하세요.

> **Note**
>
> 다음과 같은 메시지가 표시되면 현재 DB 인스턴스에서 AWS 제공 플러그인을 사용할 수 없는 것입니다.
>
> `ERROR 1524 (HY000): Plugin 'AWSAuthenticationPlugin' is not loaded`
>
> 위와 같은 오류 문제를 해결하려면 지원되는 구성을 사용하고 있는지, 그리고 DB 인스턴스 에서 IAM 데이터베이스 인증이 활성화되어 있는지 확인하십시오. 자세한 내용은 리전 및 버전 사용 가능 여부 및 IAM 데이터베이스 인증의 활성화 및 비활성화 단원을 참조하십시오.

AWSAuthenticationPlugin을 사용하여 계정을 생성한 이후 계정 관리 방법은 다른 데이터베이스 계정과 동일합니다. 예를 들어 GRANT 및 REVOKE 문으로 계정 권한을 수정하거나, 혹은 ALTER USER 문으로 여러 가지 계정 속성을 변경할 수 있습니다.

IAM을 사용하는 경우 데이터베이스 네트워크 트래픽은 SSL/TLS를 사용하여 암호화됩니다. SSL 연결을 허용하려면 다음 명령을 사용해 사용자 계정을 수정합니다.

```
ALTER USER 'jane_doe'@'%' REQUIRE SSL;
```

PostgreSQL에서 IAM 인증 사용

PostgreSQL에서 IAM 인증을 사용하려면 마스터 사용자 또는 사용자를 생성하고 권한을 부여할 수 있는 다른 사용자로 DB 인스턴스에 연결해야 합니다. 연결한 후 데이터베이스 사용자를 생성하고 다음 예제에 나온 것처럼 사용자에게 rds_iam 역할을 부여합니다.

```
CREATE USER db_userx;
GRANT rds_iam TO db_userx;
```

지정한 데이터베이스 사용자 이름이 IAM 데이터베이스 액세스에 대한 IAM 정책의 리소스와 같은지 확인합니다. 자세한 내용은 IAM 데이터베이스 액세스를 위한 IAM 정책 생성 및 사용 단원을 참조하십시오. IAM 인증을 사용하려면 rds_iam 역할을 부여해야 합니다. 중첩된 멤버십이나 역할의 간접 부여도 사용할 수 있습니다.

IAM 인증을 사용하여 DB 인스턴스에 연결

IAM 데이터베이스 인증 방식에서는 DB 인스턴스 에 연결할 때 인증 토큰을 사용합니다. 인증 토큰이란 암호 대신 사용하는 문자열을 말합니다. 인증 토큰은 생성 후 15분 동안만 유효하며 이 시간이 지나면 만료됩니다. 만료된 토큰을 사용하여 연결하려고 하면 연결 요청이 거부됩니다.

모든 인증 토큰은 AWS 서명 버전 4를 사용하여 유효한 서명이 있어야 합니다. (자세한 내용은 AWS 일반 참조의 서명 버전 4 서명 프로세스를 참조하세요.) AWS SDK for Java 또는 AWS SDK for Python (Boto3)와 같은 AWS CLI 및 AWS SDK는 사용자가 생성한 각 토큰에 자동으로 서명할 수 있습니다.

AWS Lambda와 같은 AWS 서비스에서 Amazon RDS에 연결할 때 인증 토큰을 사용할 수 있습니다. 토큰을 사용하면 코드에 암호를 넣지 않아도 됩니다. 그 밖에 AWS SDK를 사용하여 인증 토큰을 프로그래밍 방식으로 생성하고 프로그래밍 방식으로 서명하는 방법도 있습니다.

IAM 인증 토큰에 서명까지 마쳤으면 이제 Amazon RDS DB 인스턴스에 연결할 수 있습니다. 다음 섹션에서는 명령줄 도구 또는 AWS SDK(예: AWS SDK for Java 또는 AWS SDK for Python (Boto3))를 사용하여 연결하는 방법에 대해 알아보겠습니다.

자세한 내용은 다음 블로그 게시물을 참조하십시오.

- IAM 인증을 사용하여 SQL Workbench/J를 Aurora MySQL 또는 Amazon RDS for MySQL에 연결
- Using IAM authentication to connect with pgAdmin Amazon Aurora PostgreSQL or Amazon RDS for PostgreSQL

사전 조건

다음은 IAM 인증을 사용하여 DB 인스턴스에 연결하기 위한 사전 조건입니다.

- IAM 데이터베이스 인증의 활성화 및 비활성화
- IAM 데이터베이스 액세스를 위한 IAM 정책 생성 및 사용
- IAM 인증을 사용하여 데이터베이스 계정 생성

주제

- AWS 드라이버와 함께 IAM 인증을 사용하여 DB 인스턴스에 연결
- 명령줄에서 IAM 인증을 사용하여 DB 인스턴스에 연결: AWS CLI 및 mysql 클라이언트
- 명령줄: AWS CLI 및 psql Client에서 IAM 인증을 사용하여 DB 인스턴스에 연결
- IAM 인증 및 AWS SDK for .NET를 사용하여 DB 인스턴스에 연결
- IAM 인증 및 AWS SDK for Go를 사용하여 DB 인스턴스에 연결
- IAM 인증 및 AWS SDK for Java를 사용하여 DB 인스턴스에 연결
- IAM 인증 및 AWS SDK for Python (Boto3)를 사용하여 DB 인스턴스에 연결

AWS 드라이버와 함께 IAM 인증을 사용하여 DB 인스턴스에 연결

더 빠른 전환 및 장애 조치 시간, AWS Secrets Manager, AWS Identity and Access Management(IAM) 및 페더레이션 ID를 사용한 인증을 지원하도록 설계된 AWS 드라이버 제품군입니다. AWS 드라이버는 DB 인스턴스 상태 모니터링과 인스턴스 토폴로지 파악을 통해 새 라이터를 결정합니다. 이 접근 방식은 전환 및 장애 조치 시간을 오픈 소스 드라이버의 경우 수십 초였던 것에 비해 10초 미만으로 단축합니다.

AWS 드라이버에 대한 자세한 내용은 RDS for MariaDB, RDS for MySQL 또는 RDS for PostgreSQL DB 인스턴스에 대한 해당 언어 드라이버를 참조하세요.

> **Note**
>
> RDS for MariaDB에서 지원되는 유일한 기능은 AWS Secrets Manager, AWS Identity and Access Management(IAM) 및 페더레이션 ID를 사용한 인증입니다.

명령줄에서 IAM 인증을 사용하여 DB 인스턴스에 연결: AWS CLI 및 mysql 클라이언트

아래 설명과 같이 AWS CLI 및 `mysql` 명령줄 도구를 사용하여 명령줄에서 Amazon RDS DB 인스턴스로 연결할 수 있습니다.

사전 조건

다음은 IAM 인증을 사용하여 DB 인스턴스에 연결하기 위한 사전 조건입니다.

- IAM 데이터베이스 인증의 활성화 및 비활성화
- IAM 데이터베이스 액세스를 위한 IAM 정책 생성 및 사용
- IAM 인증을 사용하여 데이터베이스 계정 생성

> **Note**
>
> IAM 인증을 통한 SQL Workbench/J를 사용하여 데이터베이스에 연결하는 방법에 대한 자세한 내용은 블로그 게시물 IAM 인증을 사용하여 SQL Workbench/J를 Aurora MySQL 또는 Amazon RDS for MySQL에 연결을 참조하세요.

주제

- IAM 인증 토크 생성
- DB 인스턴스에 연결

IAM 인증 토크 생성

다음은 AWS CLI를 사용하여 서명이 되어 있는 인증 토큰을 생성하는 방법을 설명한 예제입니다.

```
aws rds generate-db-auth-token \
   --hostname rdsmysql.123456789012.us-west-2.rds.amazonaws.com \
   --port 3306 \
```

```
  --region us-west-2 \
  --username jane_doe
```

위의 예제에서 각 파라미터는 다음과 같습니다.

- `--hostname` – 액세스할 DB 인스턴스의 호스트 이름입니다.
- `--port` – DB 인스턴스에 연결할 때 사용할 포트 번호입니다.
- `--region` - DB 인스턴스가 실행되는 AWS 리전입니다.
- `--username` – 액세스할 데이터베이스 계정입니다.

토큰에서 가장 앞의 일부 문자는 다음과 같은 모습입니다.

```
rdsmysql.123456789012.us-west-2.rds.amazonaws.com:3306/?
Action=connect&DBUser=jane_doe&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Expires=900...
```

> **Note**
>
> 인증 토큰을 생성할 때는 DB 인스턴스 엔드포인트 대신 사용자 지정 Route 53 DNS 레코드를 사용할 수 없습니다.

DB 인스턴스에 연결

일반적인 연결 형식은 다음과 같습니다.

```
mysql --host=hostName --port=portNumber --ssl-ca=full_path_to_ssl_certificate --enable-
cleartext-plugin --user=userName --password=authToken
```

파라미터는 다음과 같습니다.

- `--host` – 액세스할 DB 인스턴스의 호스트 이름입니다.
- `--port` – DB 인스턴스에 연결할 때 사용할 포트 번호입니다.
- `--ssl-ca` - 퍼블릭 키를 포함하는 SSL 인증서 파일의 전체 경로

  MariaDB용 SSL/TLS 지원 방법에 대한 자세한 내용은 Amazon RDS의 MariaDB DB 인스턴스에 대한 SSL/TLS 지원 섹션을 참조하세요.

MySQL용 SSL/TLS 지원 방법에 대한 자세한 내용은 Amazon RDS의 MySQL DB 인스턴스에 대한 SSL/TLS 지원 섹션을 참조하세요.

SSL 인증서를 다운로드하려면 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 섹션을 참조하세요.

- `--enable-cleartext-plugin` – 현재 연결에서 `AWSAuthenticationPlugin`을 사용하도록 지정하는 값입니다.

  MariaDB 클라이언트를 사용하는 경우 `--enable-cleartext-plugin` 옵션이 필요하지 않습니다.
- `--user` – 액세스할 데이터베이스 계정입니다.
- `--password` – 서명된 IAM 인증 토큰입니다.

인증 토큰은 수백 자의 문자로 구성됩니다. 그렇기 때문에 명령줄에서는 다루지 불편할 수도 있습니다. 이러한 문제를 해결하기 위해 토큰을 환경 변수로 저장한 후 연결할 때 이 변수를 사용하는 것도 한 가지 방법입니다. 다음은 이러한 문제 해결 방법을 설명한 예제입니다. 이 예에서 */sample_dir/*는 퍼블릭 키를 포함하는 SSL 인증서 파일의 전체 경로입니다.

```
RDSHOST="mysqldb.123456789012.us-east-1.rds.amazonaws.com"
TOKEN="$(aws rds generate-db-auth-token --hostname $RDSHOST --port 3306 --region us-
west-2 --username jane_doe )"

mysql --host=$RDSHOST --port=3306 --ssl-ca=/sample_dir/global-bundle.pem --enable-
cleartext-plugin --user=jane_doe --password=$TOKEN
```

`AWSAuthenticationPlugin`을 사용하여 연결할 때는 SSL을 통해 보안을 유지합니다. 이러한 보안 여부를 확인하려면 `mysql>` 명령 프롬프트에 다음과 같이 입력합니다.

```
show status like 'Ssl%';
```

그러면 출력 시 다음과 같이 자세하게 표시됩니다.

```
+---------------+-------------+
| Variable_name | Value
                                                                |
```

```
+---------------+-------------+
| ...           | ...
| Ssl_cipher    | AES256-SHA

                                                                  |
| ...           | ...
| Ssl_version   | TLSv1.1

                                                                  |
| ...           | ...
+-----------------------------+
```

프록시를 통해 DB 인스턴스에 연결하려는 경우 IAM 인증을 사용하여 프록시에 연결을 참조하세요.

명령줄: AWS CLI 및 psql Client에서 IAM 인증을 사용하여 DB 인스턴스에 연결

아래 설명과 같이 AWS CLI 및 psql 명령줄 도구를 사용하여 명령줄에서 Amazon RDS for PostgreSQL DB 인스턴스에 연결할 수 있습니다.

사전 조건

다음은 IAM 인증을 사용하여 DB 인스턴스에 연결하기 위한 사전 조건입니다.

- IAM 데이터베이스 인증의 활성화 및 비활성화
- IAM 데이터베이스 액세스를 위한 IAM 정책 생성 및 사용
- IAM 인증을 사용하여 데이터베이스 계정 생성

> **Note**
>
> IAM 인증을 통한 pgAdmin을 사용하여 데이터베이스에 연결하는 방법에 대한 자세한 내용은 블로그 게시물 Using IAM authentication to connect with pgAdmin Amazon Aurora PostgreSQL or Amazon RDS for PostgreSQL을 참조하세요.

주제

- IAM 인증 토크 생성
- Amazon RDS PostgreSQL 인스턴스에 연결

IAM 인증 토크 생성

인증 토큰은 수백 자의 문자로 구성되므로 명령줄에서는 다루기 불편할 수 있습니다. 이러한 문제를 해결하기 위해 토큰을 환경 변수로 저장한 후 연결할 때 이 변수를 사용하는 것도 한 가지 방법입니다. 다음 예제는 AWS CLI에서 `generate-db-auth-token` 명령을 사용하여 서명된 인증 토큰을 받고 이를 `PGPASSWORD` 환경 변수에 저장하는 방법을 보여 줍니다.

```
export RDSHOST="rdspostgres.123456789012.us-west-2.rds.amazonaws.com"
export PGPASSWORD="$(aws rds generate-db-auth-token --hostname $RDSHOST --port 5432 --
region us-west-2 --username jane_doe )"
```

예제에서 `generate-db-auth-token` 명령에 대한 파라미터는 다음과 같습니다.

- `--hostname` – 액세스할 DB 인스턴스의 호스트 이름입니다.
- `--port` – DB 인스턴스에 연결할 때 사용할 포트 번호입니다.
- `--region` - DB 인스턴스가 실행되는 AWS 리전입니다.
- `--username` – 액세스할 데이터베이스 계정입니다.

생성된 토큰에서 처음 몇 글자는 다음과 같은 모습입니다.

```
rdspostgres.123456789012.us-west-2.rds.amazonaws.com:5432/?
Action=connect&DBUser=jane_doe&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Expires=900...
```

> **Note**
>
> 인증 토큰을 생성할 때는 DB 인스턴스 엔드포인트 대신 사용자 지정 Route 53 DNS 레코드를 사용할 수 없습니다.

Amazon RDS PostgreSQL 인스턴스에 연결

psql을 사용한 일반적인 연결 형식은 다음과 같습니다.

```
psql "host=hostName port=portNumber sslmode=verify-full
 sslrootcert=full_path_to_ssl_certificate dbname=DBName user=userName
 password=authToken"
```

파라미터는 다음과 같습니다.

- `host` – 액세스할 DB 인스턴스의 호스트 이름입니다.
- `port` – DB 인스턴스에 연결할 때 사용할 포트 번호입니다.
- `sslmode` – 사용할 SSL 모드입니다.

  `sslmode=verify-full`을 사용하면 SSL 연결에서 SSL 인증서의 엔드포인트와 비교하여 DB 인스턴스 엔드포인트를 확인합니다.
- `sslrootcert` - 퍼블릭 키를 포함하는 SSL 인증서 파일의 전체 경로

  자세한 내용은 PostgreSQL DB 인스턴스와 함께 SSL 사용 단원을 참조하십시오.

  SSL 인증서를 다운로드하려면 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 섹션을 참조하세요.
- `dbname` – 액세스할 데이터베이스입니다.
- `user` – 액세스할 데이터베이스 계정입니다.
- `password` – 서명된 IAM 인증 토큰입니다.

> **Note**
>
> 인증 토큰을 생성할 때는 DB 인스턴스 엔드포인트 대신 사용자 지정 Route 53 DNS 레코드를 사용할 수 없습니다.

다음 예제는 psql을 사용하여 연결하는 방법을 보여줍니다. 예에서 psql은 호스트에 대해 환경 변수 RDSHOST를 사용하고 생성된 토큰에 대해 환경 변수 PGPASSWORD를 사용합니다. 또한 */sample_dir/*은 퍼블릭 키를 포함하는 SSL 인증서 파일의 전체 경로입니다.

```
export RDSHOST="rdspostgres.123456789012.us-west-2.rds.amazonaws.com"
export PGPASSWORD="$(aws rds generate-db-auth-token --hostname $RDSHOST --port 5432 --
region us-west-2 --username jane_doe )"

psql "host=$RDSHOST port=5432 sslmode=verify-full sslrootcert=/sample_dir/global-
bundle.pem dbname=DBName user=jane_doe password=$PGPASSWORD"
```

프록시를 통해 DB 인스턴스에 연결하려는 경우 IAM 인증을 사용하여 프록시에 연결을 참조하세요.

IAM 인증 및 AWS SDK for .NET를 사용하여 DB 인스턴스에 연결

아래 설명과 같이 AWS SDK for .NET를 사용하여 RDS for MariaDB, MySQL 또는 PostgreSQL DB 인스턴스에 연결할 수 있습니다.

사전 조건

다음은 IAM 인증을 사용하여 DB 인스턴스에 연결하기 위한 사전 조건입니다.

- IAM 데이터베이스 인증의 활성화 및 비활성화
- IAM 데이터베이스 액세스를 위한 IAM 정책 생성 및 사용
- IAM 인증을 사용하여 데이터베이스 계정 생성

예시

다음 코드 예제는 인증 토큰을 생성한 다음 이 토큰을 사용하여 DB 인스턴스에 연결하는 방법을 보여 줍니다.

이 코드 예제를 실행하려면 AWS SDK for .NET가 필요하며 이는 AWS 사이트에서 받을 수 있습니다. `AWSSDK.CORE` 및 `AWSSDK.RDS` 패키지가 필요합니다. DB 인스턴스에 연결하려면 DB 엔진용 .NET 데이터베이스 커넥터(예: MariaDB 또는 MySQL용 MySQL커넥터 또는 PostgreSQL용 Npgsql)를 사용합니다.

이 코드는 MariaDB 또는 MySQL DB 인스턴스에 연결됩니다. 필요하다면 다음 변수 값을 변경합니다.

- `server` - 액세스할 DB 인스턴스의 엔드포인트입니다.
- `user` – 액세스할 데이터베이스 계정입니다.
- `database` – 액세스할 데이터베이스입니다.
- `port` – DB 인스턴스에 연결할 때 사용할 포트 번호입니다.
- `SslMode` – 사용할 SSL 모드입니다.

  `SslMode=Required`을 사용하면 SSL 연결에서 SSL 인증서의 엔드포인트와 비교하여 DB 인스턴스 엔드포인트를 확인합니다.
- `SslCa` - Amazon RDS에 대한 SSL 인증서의 전체 경로입니다.

  인증서를 다운로드하려면 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 섹션을 참조하세요.

> **Note**
>
> 인증 토큰을 생성할 때는 DB 인스턴스 엔드포인트 대신 사용자 지정 Route 53 DNS 레코드를 사용할 수 없습니다.

```
using System;
using System.Data;
using MySql.Data;
using MySql.Data.MySqlClient;
using Amazon;

namespace ubuntu
{
  class Program
  {
    static void Main(string[] args)
    {
      var pwd =
 Amazon.RDS.Util.RDSAuthTokenGenerator.GenerateAuthToken(RegionEndpoint.USEast1,
 "mysqldb.123456789012.us-east-1.rds.amazonaws.com", 3306, "jane_doe");
      // for debug only Console.Write("{0}\n", pwd);  //this verifies the token is
 generated

      MySqlConnection conn = new MySqlConnection($"server=mysqldb.123456789012.us-
east-1.rds.amazonaws.com;user=jane_doe;database=mydB;port=3306;password={pwd};SslMode=Required;
      conn.Open();

      // Define a query
      MySqlCommand sampleCommand = new MySqlCommand("SHOW DATABASES;", conn);

      // Execute a query
      MySqlDataReader mysqlDataRdr = sampleCommand.ExecuteReader();

      // Read all rows and output the first column in each row
      while (mysqlDataRdr.Read())
        Console.WriteLine(mysqlDataRdr[0]);

      mysqlDataRdr.Close();
      // Close connection
      conn.Close();
    }
```

```
  }
}
```

이 코드는 PostgreSQL DB 인스턴스에 연결됩니다.

필요하다면 다음 변수 값을 변경합니다.

- `Server` - 액세스할 DB 인스턴스의 엔드포인트입니다.
- `User ID` – 액세스할 데이터베이스 계정입니다.
- `Database` – 액세스할 데이터베이스입니다.
- `Port` – DB 인스턴스에 연결할 때 사용할 포트 번호입니다.
- `SSL Mode` – 사용할 SSL 모드입니다.

  `SSL Mode=Required`을 사용하면 SSL 연결에서 SSL 인증서의 엔드포인트와 비교하여 DB 인스턴스 엔드포인트를 확인합니다.
- `Root Certificate` - Amazon RDS에 대한 SSL 인증서의 전체 경로입니다.

  인증서를 다운로드하려면 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 섹션을 참조하세요.

> **Note**
>
> 인증 토큰을 생성할 때는 DB 인스턴스 엔드포인트 대신 사용자 지정 Route 53 DNS 레코드를 사용할 수 없습니다.

```
using System;
using Npgsql;
using Amazon.RDS.Util;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            var pwd =
 RDSAuthTokenGenerator.GenerateAuthToken("postgresmydb.123456789012.us-
east-1.rds.amazonaws.com", 5432, "jane_doe");
```

```
// for debug only Console.Write("{0}\n", pwd);  //this verifies the token is generated

            NpgsqlConnection conn = new
 NpgsqlConnection($"Server=postgresmydb.123456789012.us-east-1.rds.amazonaws.com;User
 Id=jane_doe;Password={pwd};Database=mydb;SSL Mode=Require;Root
 Certificate=full_path_to_ssl_certificate");
            conn.Open();

            // Define a query
                   NpgsqlCommand cmd = new NpgsqlCommand("select count(*) FROM
 pg_user", conn);

            // Execute a query
            NpgsqlDataReader dr = cmd.ExecuteReader();

            // Read all rows and output the first column in each row
            while (dr.Read())
                Console.Write("{0}\n", dr[0]);

            // Close connection
            conn.Close();
        }
    }
}
```

프록시를 통해 DB 인스턴스에 연결하려는 경우 IAM 인증을 사용하여 프록시에 연결을 참조하세요.

IAM 인증 및 AWS SDK for Go를 사용하여 DB 인스턴스에 연결

아래 설명과 같이 AWS SDK for Go를 사용하여 RDS for MariaDB, MySQL 또는 PostgreSQL DB 인스턴스에 연결할 수 있습니다.

사전 조건

다음은 IAM 인증을 사용하여 DB 인스턴스에 연결하기 위한 사전 조건입니다.

- IAM 데이터베이스 인증의 활성화 및 비활성화
- IAM 데이터베이스 액세스를 위한 IAM 정책 생성 및 사용
- IAM 인증을 사용하여 데이터베이스 계정 생성

예시

이 코드 예제를 실행하려면 AWS SDK for Go가 필요하며 이는 AWS 사이트에서 받을 수 있습니다.

필요하다면 다음 변수 값을 변경합니다.

- `dbName` – 액세스할 데이터베이스입니다.
- `dbUser` – 액세스할 데이터베이스 계정입니다.
- `dbHost` - 액세스할 DB 인스턴스의 엔드포인트입니다.

> **Note**
>
> 인증 토큰을 생성할 때는 DB 인스턴스 엔드포인트 대신 사용자 지정 Route 53 DNS 레코드를 사용할 수 없습니다.

- `dbPort` – DB 인스턴스에 연결할 때 사용할 포트 번호입니다.
- `region` - DB 인스턴스가 실행되는 AWS 리전입니다.

또한 샘플 코드에서 가져온 라이브러리가 시스템에 있는지 확인합니다.

> **Important**
>
> 이 섹션의 예에서는 다음 코드를 사용하여 로컬 환경에서 데이터베이스에 액세스하는 자격 증명을 제공합니다.
> `creds := credentials.NewEnvCredentials()`
> Amazon EC2 또는 Amazon ECS 같은 AWS 서비스에서 데이터베이스에 액세스하는 경우 코드를 다음 코드로 바꿀 수 있습니다.
> `sess := session.Must(session.NewSession())`
> `creds := sess.Config.Credentials`
> 이 변경 작업을 수행하는 경우 다음 가져오기를 추가해야 합니다.
> `"github.com/aws/aws-sdk-go/aws/session"`

주제

- IAM 인증 및 AWS SDK for Go V2를 사용하여 연결
- IAM 인증 및 AWS SDK for Go V1을 사용하여 연결

IAM 인증 및 AWS SDK for Go V2를 사용하여 연결

IAM 인증 및 AWS SDK for Go V2를 사용하여 DB 인스턴스에 연결할 수 있습니다.

다음 코드 예제는 인증 토큰을 생성한 다음 이 토큰을 사용하여 DB 인스턴스에 연결하는 방법을 보여 줍니다.

이 코드는 MariaDB 또는 MySQL DB 인스턴스에 연결됩니다.

```
package main

import (
     "context"
     "database/sql"
     "fmt"

     "github.com/aws/aws-sdk-go-v2/config"
     "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
     _ "github.com/go-sql-driver/mysql"
)

func main() {

     var dbName string = "DatabaseName"
     var dbUser string = "DatabaseUser"
     var dbHost string = "mysqldb.123456789012.us-east-1.rds.amazonaws.com"
     var dbPort int = 3306
     var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
     var region string = "us-east-1"

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
     panic("configuration error: " + err.Error())
    }

    authenticationToken, err := auth.BuildAuthToken(
     context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
    if err != nil {
     panic("failed to create authentication token: " + err.Error())
    }

    dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
        dbUser, authenticationToken, dbEndpoint, dbName,
    )
```

```
    db, err := sql.Open("mysql", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

이 코드는 PostgreSQL DB 인스턴스에 연결됩니다.

```
package main

import (
     "context"
     "database/sql"
     "fmt"

     "github.com/aws/aws-sdk-go-v2/config"
     "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
     _ "github.com/lib/pq"
)

func main() {

     var dbName string = "DatabaseName"
     var dbUser string = "DatabaseUser"
     var dbHost string = "postgresmydb.123456789012.us-east-1.rds.amazonaws.com"
     var dbPort int = 5432
     var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
     var region string = "us-east-1"

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
     panic("configuration error: " + err.Error())
    }

    authenticationToken, err := auth.BuildAuthToken(
     context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
    if err != nil {
```

```
     panic("failed to create authentication token: " + err.Error())
    }

    dsn := fmt.Sprintf("host=%s port=%d user=%s password=%s dbname=%s",
        dbHost, dbPort, dbUser, authenticationToken, dbName,
    )

    db, err := sql.Open("postgres", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

프록시를 통해 DB 인스턴스에 연결하려는 경우 IAM 인증을 사용하여 프록시에 연결을 참조하세요.

IAM 인증 및 AWS SDK for Go V1을 사용하여 연결

IAM 인증 및 AWS SDK for Go V1을 사용하여 DB 인스턴스에 연결할 수 있습니다.

다음 코드 예제는 인증 토큰을 생성한 다음 이 토큰을 사용하여 DB 인스턴스에 연결하는 방법을 보여 줍니다.

이 코드는 MariaDB 또는 MySQL DB 인스턴스에 연결됩니다.

```
package main

import (
    "database/sql"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go/aws/credentials"
    "github.com/aws/aws-sdk-go/service/rds/rdsutils"
    _ "github.com/go-sql-driver/mysql"
)

func main() {
    dbName := "app"
```

```
    dbUser := "jane_doe"
    dbHost := "mysqldb.123456789012.us-east-1.rds.amazonaws.com"
    dbPort := 3306
    dbEndpoint := fmt.Sprintf("%s:%d", dbHost, dbPort)
    region := "us-east-1"

    creds := credentials.NewEnvCredentials()
    authToken, err := rdsutils.BuildAuthToken(dbEndpoint, region, dbUser, creds)
    if err != nil {
        panic(err)
    }

    dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
        dbUser, authToken, dbEndpoint, dbName,
    )

    db, err := sql.Open("mysql", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

이 코드는 PostgreSQL DB 인스턴스에 연결됩니다.

```
package main

import (
 "database/sql"
 "fmt"

 "github.com/aws/aws-sdk-go/aws/credentials"
 "github.com/aws/aws-sdk-go/service/rds/rdsutils"
 _ "github.com/lib/pq"
)

func main() {
    dbName := "app"
    dbUser := "jane_doe"
```

```
    dbHost := "postgresmydb.123456789012.us-east-1.rds.amazonaws.com"
    dbPort := 5432
    dbEndpoint := fmt.Sprintf("%s:%d", dbHost, dbPort)
    region := "us-east-1"

    creds := credentials.NewEnvCredentials()
    authToken, err := rdsutils.BuildAuthToken(dbEndpoint, region, dbUser, creds)
    if err != nil {
        panic(err)
    }

    dsn := fmt.Sprintf("host=%s port=%d user=%s password=%s dbname=%s",
        dbHost, dbPort, dbUser, authToken, dbName,
    )

    db, err := sql.Open("postgres", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

프록시를 통해 DB 인스턴스에 연결하려는 경우 IAM 인증을 사용하여 프록시에 연결을 참조하세요.

IAM 인증 및 AWS SDK for Java를 사용하여 DB 인스턴스에 연결

아래 설명과 같이 AWS SDK for Java를 사용하여 RDS for MariaDB, MySQL 또는 PostgreSQL DB 인스턴스에 연결할 수 있습니다.

사전 조건

다음은 IAM 인증을 사용하여 DB 인스턴스에 연결하기 위한 사전 조건입니다.

- IAM 데이터베이스 인증의 활성화 및 비활성화
- IAM 데이터베이스 액세스를 위한 IAM 정책 생성 및 사용
- IAM 인증을 사용하여 데이터베이스 계정 생성
- AWS SDK for Java 설정

Java 2.x용 SDK를 사용하는 방법에 대한 예제는 Java 2.x용 SDK를 사용한 Amazon RDS 예제를 참조하세요. 또한 AWS 고급 JDBC 래퍼를 사용할 수 있습니다. AWS Advanced JDBC Wrapper documentation을 참조하세요.

주제

- IAM 인증 토크 생성
- IAM 인증 토큰 수동 구성
- DB 인스턴스에 연결

IAM 인증 토크 생성

AWS SDK for Java로 프로그램을 개발할 때는 `RdsIamAuthTokenGenerator` 클래스를 사용하여 서명된 인증 토큰을 가져올 수 있습니다. 이 클래스를 사용하려면 AWS 자격 증명을 입력해야 합니다. 이렇게 하려면 `DefaultAWSCredentialsProviderChain` 클래스의 인스턴스를 생성합니다. `DefaultAWSCredentialsProviderChain`은 기본 자격 증명 공급자 체인에서 찾은 첫 번째 AWS 액세스 키와 보안 키를 사용합니다. AWS 액세스 키에 대한 자세한 내용은 사용자의 액세스 키 관리를 참조하세요.

> **Note**
>
> 인증 토큰을 생성할 때는 DB 인스턴스 엔드포인트 대신 사용자 지정 Route 53 DNS 레코드를 사용할 수 없습니다.

`RdsIamAuthTokenGenerator` 인스턴스를 생성한 후에는 `getAuthToken` 메서드를 호출하여 서명된 토큰을 가져올 수 있습니다. 이때 AWS 리전, 호스트 이름, 포트 이름 및 사용자 이름을 입력합니다. 다음은 각 정보의 입력 방법을 설명한 코드 예제입니다.

```
package com.amazonaws.codesamples;

import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.services.rds.auth.GetIamAuthTokenRequest;
import com.amazonaws.services.rds.auth.RdsIamAuthTokenGenerator;

public class GenerateRDSAuthToken {

    public static void main(String[] args) {

     String region = "us-west-2";
```

```
     String hostname = "rdsmysql.123456789012.us-west-2.rds.amazonaws.com";
     String port = "3306";
     String username = "jane_doe";

     System.out.println(generateAuthToken(region, hostname, port, username));
    }

    static String generateAuthToken(String region, String hostName, String port, String
 username) {

     RdsIamAuthTokenGenerator generator = RdsIamAuthTokenGenerator.builder()
      .credentials(new DefaultAWSCredentialsProviderChain())
      .region(region)
      .build();

     String authToken = generator.getAuthToken(
      GetIamAuthTokenRequest.builder()
      .hostname(hostName)
      .port(Integer.parseInt(port))
      .userName(username)
      .build());

     return authToken;
    }

}
```

IAM 인증 토큰 수동 구성

Java에서 인증 토큰을 가장 쉽게 생성할 수 있는 방법은 `RdsIamAuthTokenGenerator`를 사용하는 것입니다. 이 클래스는 인증 토큰을 생성한 후 AWS 서명 버전 4를 사용해 서명까지 마칩니다. (자세한 내용은 AWS 일반 참조의 서명 버전 4 서명 프로세스를 참조하세요.)

그 밖에 다음 코드 예제와 같이 인증 토큰을 수동으로 구성하여 서명하는 방법도 있습니다.

```
package com.amazonaws.codesamples;

import com.amazonaws.SdkClientException;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.auth.SigningAlgorithm;
import com.amazonaws.util.BinaryUtils;
import org.apache.commons.lang3.StringUtils;
```

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.Charset;
import java.security.MessageDigest;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.SortedMap;
import java.util.TreeMap;

import static com.amazonaws.auth.internal.SignerConstants.AWS4_TERMINATOR;
import static com.amazonaws.util.StringUtils.UTF8;

public class CreateRDSAuthTokenManually {
    public static String httpMethod = "GET";
    public static String action = "connect";
    public static String canonicalURIParameter = "/";
    public static SortedMap<String, String> canonicalQueryParameters = new TreeMap();
    public static String payload = StringUtils.EMPTY;
    public static String signedHeader = "host";
    public static String algorithm = "AWS4-HMAC-SHA256";
    public static String serviceName = "rds-db";
    public static String requestWithoutSignature;

    public static void main(String[] args) throws Exception {

        String region = "us-west-2";
        String instanceName = "rdsmysql.123456789012.us-west-2.rds.amazonaws.com";
        String port = "3306";
        String username = "jane_doe";

        Date now = new Date();
        String date = new SimpleDateFormat("yyyyMMdd").format(now);
        String dateTimeStamp = new
 SimpleDateFormat("yyyyMMdd'T'HHmmss'Z'").format(now);
        DefaultAWSCredentialsProviderChain creds = new
 DefaultAWSCredentialsProviderChain();
     String awsAccessKey = creds.getCredentials().getAWSAccessKeyId();
     String awsSecretKey = creds.getCredentials().getAWSSecretKey();
        String expiryMinutes = "900";

        System.out.println("Step 1:  Create a canonical request:");
        String canonicalString = createCanonicalString(username, awsAccessKey, date,
 dateTimeStamp, region, expiryMinutes, instanceName, port);
        System.out.println(canonicalString);
```

```
        System.out.println();

        System.out.println("Step 2:  Create a string to sign:");
        String stringToSign = createStringToSign(dateTimeStamp, canonicalString,
awsAccessKey, date, region);
        System.out.println(stringToSign);
        System.out.println();

        System.out.println("Step 3:  Calculate the signature:");
        String signature = BinaryUtils.toHex(calculateSignature(stringToSign,
newSigningKey(awsSecretKey, date, region, serviceName)));
        System.out.println(signature);
        System.out.println();

        System.out.println("Step 4:  Add the signing info to the request");

        System.out.println(appendSignature(signature));
        System.out.println();

    }

    //Step 1: Create a canonical request date should be in format YYYYMMDD and dateTime
should be in format YYYYMMDDTHHMMSSZ
    public static String createCanonicalString(String user, String accessKey, String
date, String dateTime, String region, String expiryPeriod, String hostName, String
port) throws Exception {
        canonicalQueryParameters.put("Action", action);
        canonicalQueryParameters.put("DBUser", user);
        canonicalQueryParameters.put("X-Amz-Algorithm", "AWS4-HMAC-SHA256");
        canonicalQueryParameters.put("X-Amz-Credential", accessKey + "%2F" + date +
"%2F" + region + "%2F" + serviceName + "%2Faws4_request");
        canonicalQueryParameters.put("X-Amz-Date", dateTime);
        canonicalQueryParameters.put("X-Amz-Expires", expiryPeriod);
        canonicalQueryParameters.put("X-Amz-SignedHeaders", signedHeader);
        String canonicalQueryString = "";
        while(!canonicalQueryParameters.isEmpty()) {
            String currentQueryParameter = canonicalQueryParameters.firstKey();
            String currentQueryParameterValue =
canonicalQueryParameters.remove(currentQueryParameter);
            canonicalQueryString = canonicalQueryString + currentQueryParameter + "=" +
currentQueryParameterValue;
            if (!currentQueryParameter.equals("X-Amz-SignedHeaders")) {
                canonicalQueryString += "&";
            }
```

```
        }
        String canonicalHeaders = "host:" + hostName + ":" + port + '\n';
        requestWithoutSignature = hostName + ":" + port + "/?" + canonicalQueryString;

        String hashedPayload = BinaryUtils.toHex(hash(payload));
        return httpMethod + '\n' + canonicalURIParameter + '\n' + canonicalQueryString
 + '\n' + canonicalHeaders + '\n' + signedHeader + '\n' + hashedPayload;

    }

    //Step 2: Create a string to sign using sig v4
    public static String createStringToSign(String dateTime, String canonicalRequest,
 String accessKey, String date, String region) throws Exception {
        String credentialScope = date + "/" + region + "/" + serviceName + "/
aws4_request";
        return algorithm + '\n' + dateTime + '\n' + credentialScope + '\n' +
 BinaryUtils.toHex(hash(canonicalRequest));

    }

    //Step 3: Calculate signature
    /**
     * Step 3 of the &AWS; Signature version 4 calculation. It involves deriving
     * the signing key and computing the signature. Refer to
     * http://docs.aws.amazon
     * .com/general/latest/gr/sigv4-calculate-signature.html
     */
    public static byte[] calculateSignature(String stringToSign,
                                            byte[] signingKey) {
        return sign(stringToSign.getBytes(Charset.forName("UTF-8")), signingKey,
                SigningAlgorithm.HmacSHA256);
    }

    public static byte[] sign(byte[] data, byte[] key,
                          SigningAlgorithm algorithm) throws SdkClientException {
        try {
            Mac mac = algorithm.getMac();
            mac.init(new SecretKeySpec(key, algorithm.toString()));
            return mac.doFinal(data);
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to calculate a request signature: "
                            + e.getMessage(), e);
        }
```

```
    }

    public static byte[] newSigningKey(String secretKey,
                                    String dateStamp, String regionName, String
 serviceName) {
        byte[] kSecret = ("AWS4" + secretKey).getBytes(Charset.forName("UTF-8"));
        byte[] kDate = sign(dateStamp, kSecret, SigningAlgorithm.HmacSHA256);
        byte[] kRegion = sign(regionName, kDate, SigningAlgorithm.HmacSHA256);
        byte[] kService = sign(serviceName, kRegion,
                SigningAlgorithm.HmacSHA256);
        return sign(AWS4_TERMINATOR, kService, SigningAlgorithm.HmacSHA256);
    }

    public static byte[] sign(String stringData, byte[] key,
                        SigningAlgorithm algorithm) throws SdkClientException {
        try {
            byte[] data = stringData.getBytes(UTF8);
            return sign(data, key, algorithm);
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to calculate a request signature: "
                            + e.getMessage(), e);
        }
    }

    //Step 4: append the signature
    public static String appendSignature(String signature) {
        return requestWithoutSignature + "&X-Amz-Signature=" + signature;
    }

    public static byte[] hash(String s) throws Exception {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            md.update(s.getBytes(UTF8));
            return md.digest();
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to compute hash while signing request: "
                            + e.getMessage(), e);
        }
    }
}
```

DB 인스턴스에 연결

다음은 인증 토큰을 생성하고, 이를 사용하여 MariaDB 또는 MySQL을 실행하는 인스턴스에 연결하는 방법을 보여주는 코드 예입니다.

이 코드 예제를 실행하려면 AWS SDK for Java가 필요하며 이는 AWS 사이트에서 받을 수 있습니다. 또한 다음이 필요합니다.

- MySQL Connector/J. 이 코드 예제는 `mysql-connector-java-5.1.33-bin.jar`로 테스트됩니다.
- AWS 리전에 고유한 Amazon RDS에 대한 중간 인증서입니다. (자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 섹션을 참조하세요.) 예제가 실행되면 클래스 로더가 쉽게 찾을 수 있도록 이 Java 코드 예제와 동일한 디렉터리에서 인증서를 찾기 시작합니다.
- 필요하다면 다음 변수 값을 변경합니다.
  - `RDS_INSTANCE_HOSTNAME` – 액세스할 DB 인스턴스의 호스트 이름입니다.
  - `RDS_INSTANCE_PORT` - PostgreSQL DB 인스턴스에 연결할 때 사용할 포트 이름입니다.
  - `REGION_NAME` - DB 인스턴스가 실행되는 AWS 리전입니다.
  - `DB_USER` – 액세스할 데이터베이스 계정입니다.
  - `SSL_CERTIFICATE` - AWS 리전에 고유한 Amazon RDS에 대한 SSL 인증서입니다.

    AWS 리전에 대한 인증서를 다운로드하려면 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 섹션을 참조하세요. SSL 인증서는 예제 실행 시 클래스 로더가 인증서를 찾을 수 있도록 이 Java 프로그램 파일과 동일한 디렉터리에 설치합니다.

다음은 기본 자격 증명 공급자 체인에서 AWS 자격 증명을 가져오는 코드 예제입니다.

> **Note**
>
> 보안 모범 사례로 여기에 표시된 프롬프트 이외에 `DEFAULT_KEY_STORE_PASSWORD`에 대한 암호를 지정하는 것이 좋습니다.

```
package com.amazonaws.samples;

import com.amazonaws.services.rds.auth.RdsIamAuthTokenGenerator;
import com.amazonaws.services.rds.auth.GetIamAuthTokenRequest;
import com.amazonaws.auth.BasicAWSCredentials;
```

```
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.security.KeyStore;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Properties;

import java.net.URL;

public class IAMDatabaseAuthenticationTester {
    //&AWS; Credentials of the IAM user with policy enabling IAM Database Authenticated
 access to the db by the db user.
    private static final DefaultAWSCredentialsProviderChain creds = new
 DefaultAWSCredentialsProviderChain();
    private static final String AWS_ACCESS_KEY =
 creds.getCredentials().getAWSAccessKeyId();
    private static final String AWS_SECRET_KEY =
 creds.getCredentials().getAWSSecretKey();

    //Configuration parameters for the generation of the IAM Database Authentication
 token
    private static final String RDS_INSTANCE_HOSTNAME = "rdsmysql.123456789012.us-
west-2.rds.amazonaws.com";
    private static final int RDS_INSTANCE_PORT = 3306;
    private static final String REGION_NAME = "us-west-2";
    private static final String DB_USER = "jane_doe";
    private static final String JDBC_URL = "jdbc:mysql://" + RDS_INSTANCE_HOSTNAME +
 ":" + RDS_INSTANCE_PORT;

    private static final String SSL_CERTIFICATE = "rds-ca-2019-us-west-2.pem";

    private static final String KEY_STORE_TYPE = "JKS";
    private static final String KEY_STORE_PROVIDER = "SUN";
    private static final String KEY_STORE_FILE_PREFIX = "sys-connect-via-ssl-test-
cacerts";
```

```
    private static final String KEY_STORE_FILE_SUFFIX = ".jks";
    private static final String DEFAULT_KEY_STORE_PASSWORD = "changeit";

    public static void main(String[] args) throws Exception {
        //get the connection
        Connection connection = getDBConnectionUsingIam();

        //verify the connection is successful
        Statement stmt= connection.createStatement();
        ResultSet rs=stmt.executeQuery("SELECT 'Success!' FROM DUAL;");
        while (rs.next()) {
             String id = rs.getString(1);
            System.out.println(id); //Should print "Success!"
        }

        //close the connection
        stmt.close();
        connection.close();

        clearSslProperties();

    }

    /**
     * This method returns a connection to the db instance authenticated using IAM
Database Authentication
     * @return
     * @throws Exception
     */
    private static Connection getDBConnectionUsingIam() throws Exception {
        setSslProperties();
        return DriverManager.getConnection(JDBC_URL, setMySqlConnectionProperties());
    }

    /**
     * This method sets the mysql connection properties which includes the IAM Database
Authentication token
     * as the password. It also specifies that SSL verification is required.
     * @return
     */
    private static Properties setMySqlConnectionProperties() {
        Properties mysqlConnectionProperties = new Properties();
        mysqlConnectionProperties.setProperty("verifyServerCertificate","true");
        mysqlConnectionProperties.setProperty("useSSL", "true");
```

```
        mysqlConnectionProperties.setProperty("user",DB_USER);
        mysqlConnectionProperties.setProperty("password",generateAuthToken());
        return mysqlConnectionProperties;
    }

    /**
     * This method generates the IAM Auth Token.
     * An example IAM Auth Token would look like follows:
     * btusi123.cmz7kenwo2ye.rds.cn-north-1.amazonaws.com.cn:3306/?
Action=connect&DBUser=iamtestuser&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Date=20171003T010726Z&X-Amz-SignedHeaders=host&X-Amz-Expires=899&X-Amz-
Credential=AKIAPFXHGVDI5RNFO4AQ%2F20171003%2Fcn-north-1%2Frds-db%2Faws4_request&X-Amz-
Signature=f9f45ef96c1f770cdad11a53e33ffa4c3730bc03fdee820cfdf1322eed15483b
     * @return
     */
    private static String generateAuthToken() {
        BasicAWSCredentials awsCredentials = new BasicAWSCredentials(AWS_ACCESS_KEY,
 AWS_SECRET_KEY);

        RdsIamAuthTokenGenerator generator = RdsIamAuthTokenGenerator.builder()
                .credentials(new
 AWSStaticCredentialsProvider(awsCredentials)).region(REGION_NAME).build();
        return generator.getAuthToken(GetIamAuthTokenRequest.builder()

 .hostname(RDS_INSTANCE_HOSTNAME).port(RDS_INSTANCE_PORT).userName(DB_USER).build());
    }

    /**
     * This method sets the SSL properties which specify the key store file, its type
 and password:
     * @throws Exception
     */
    private static void setSslProperties() throws Exception {
        System.setProperty("javax.net.ssl.trustStore", createKeyStoreFile());
        System.setProperty("javax.net.ssl.trustStoreType", KEY_STORE_TYPE);
        System.setProperty("javax.net.ssl.trustStorePassword",
 DEFAULT_KEY_STORE_PASSWORD);
    }

    /**
     * This method returns the path of the Key Store File needed for the SSL
 verification during the IAM Database Authentication to
     * the db instance.
     * @return
```

```
     * @throws Exception
     */
    private static String createKeyStoreFile() throws Exception {
        return createKeyStoreFile(createCertificate()).getPath();
    }

    /**
     *  This method generates the SSL certificate
     * @return
     * @throws Exception
     */
    private static X509Certificate createCertificate() throws Exception {
        CertificateFactory certFactory = CertificateFactory.getInstance("X.509");
        URL url = new File(SSL_CERTIFICATE).toURI().toURL();
        if (url == null) {
            throw new Exception();
        }
        try (InputStream certInputStream = url.openStream()) {
            return (X509Certificate) certFactory.generateCertificate(certInputStream);
        }
    }

    /**
     * This method creates the Key Store File
     * @param rootX509Certificate - the SSL certificate to be stored in the KeyStore
     * @return
     * @throws Exception
     */
    private static File createKeyStoreFile(X509Certificate rootX509Certificate) throws
Exception {
        File keyStoreFile = File.createTempFile(KEY_STORE_FILE_PREFIX,
KEY_STORE_FILE_SUFFIX);
        try (FileOutputStream fos = new FileOutputStream(keyStoreFile.getPath())) {
            KeyStore ks = KeyStore.getInstance(KEY_STORE_TYPE, KEY_STORE_PROVIDER);
            ks.load(null);
            ks.setCertificateEntry("rootCaCertificate", rootX509Certificate);
            ks.store(fos, DEFAULT_KEY_STORE_PASSWORD.toCharArray());
        }
        return keyStoreFile;
    }

    /**
     * This method clears the SSL properties.
     * @throws Exception
```

```
     */
    private static void clearSslProperties() throws Exception {
           System.clearProperty("javax.net.ssl.trustStore");
           System.clearProperty("javax.net.ssl.trustStoreType");
           System.clearProperty("javax.net.ssl.trustStorePassword");
    }

}
```

프록시를 통해 DB 인스턴스에 연결하려는 경우 IAM 인증을 사용하여 프록시에 연결을 참조하세요.

IAM 인증 및 AWS SDK for Python (Boto3)를 사용하여 DB 인스턴스에 연결

아래 설명과 같이 AWS SDK for Python (Boto3)를 사용하여 RDS for MariaDB, MySQL 또는 PostgreSQL DB 인스턴스에 연결할 수 있습니다.

사전 조건

다음은 IAM 인증을 사용하여 DB 인스턴스에 연결하기 위한 사전 조건입니다.

- IAM 데이터베이스 인증의 활성화 및 비활성화
- IAM 데이터베이스 액세스를 위한 IAM 정책 생성 및 사용
- IAM 인증을 사용하여 데이터베이스 계정 생성

또한 샘플 코드에서 가져온 라이브러리가 시스템에 있는지 확인합니다.

예시

코드 예제에서는 공유 자격 증명에 프로필을 사용합니다. 자격 증명 지정에 대한 자세한 내용은 AWS SDK for Python (Boto3) 설명서의 자격 증명을 참조하십시오.

다음 코드 예제는 인증 토큰을 생성한 다음 이 토큰을 사용하여 DB 인스턴스에 연결하는 방법을 보여 줍니다.

이 코드 예제를 실행하려면 AWS SDK for Python (Boto3)가 필요하며 이는 AWS 사이트에서 받을 수 있습니다.

필요하다면 다음 변수 값을 변경합니다.

- `ENDPOINT` - 액세스할 DB 인스턴스의 엔드포인트입니다.
- `PORT` – DB 인스턴스에 연결할 때 사용할 포트 번호입니다.
- `USER` – 액세스할 데이터베이스 계정입니다.

- REGION - DB 인스턴스가 실행되는 AWS 리전입니다.
- DBNAME – 액세스할 데이터베이스입니다.
- SSLCERTIFICATE - Amazon RDS에 대한 SSL 인증서의 전체 경로입니다.

  ssl_ca의 경우 SSL 인증서를 지정합니다. SSL 인증서를 다운로드하려면 SSL/TLS를 사용하여 DB 인스턴스 또는 클러스터에 대한 연결 암호화 섹션을 참조하세요.

> **Note**
>
> 인증 토큰을 생성할 때는 DB 인스턴스 엔드포인트 대신 사용자 지정 Route 53 DNS 레코드를 사용할 수 없습니다.

이 코드는 MariaDB 또는 MySQL DB 인스턴스에 연결됩니다.

이 코드를 실행하기 전에 Python Package 색인에 있는 지침에 따라 PyMySQL 드라이버를 설치하세요.

```
import pymysql
import sys
import boto3
import os

ENDPOINT="mysqldb.123456789012.us-east-1.rds.amazonaws.com"
PORT="3306"
USER="jane_doe"
REGION="us-east-1"
DBNAME="mydb"
os.environ['LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN'] = '1'

#gets the credentials from .aws/credentials
session = boto3.Session(profile_name='default')
client = session.client('rds')

token = client.generate_db_auth_token(DBHostname=ENDPOINT, Port=PORT, DBUsername=USER,
 Region=REGION)

try:
    conn =
 pymysql.connect(auth_plugin_map={'mysql_clear_password':None},host=ENDPOINT,
```

```
 user=USER, password=token, port=PORT, database=DBNAME, ssl_ca='SSLCERTIFICATE',
 ssl_verify_identity=True, ssl_verify_cert=True)
    cur = conn.cursor()
    cur.execute("""SELECT now()""")
    query_results = cur.fetchall()
    print(query_results)
except Exception as e:
    print("Database connection failed due to {}".format(e))
```

이 코드는 PostgreSQL DB 인스턴스에 연결됩니다.

이 코드를 실행하기 전에 Psycopg 설명서의 지침에 따라 psycopg2를 설치하세요.

```
import psycopg2
import sys
import boto3
import os

ENDPOINT="postgresmydb.123456789012.us-east-1.rds.amazonaws.com"
PORT="5432"
USER="jane_doe"
REGION="us-east-1"
DBNAME="mydb"

#gets the credentials from .aws/credentials
session = boto3.Session(profile_name='RDSCreds')
client = session.client('rds')

token = client.generate_db_auth_token(DBHostname=ENDPOINT, Port=PORT, DBUsername=USER,
 Region=REGION)

try:
    conn = psycopg2.connect(host=ENDPOINT, port=PORT, database=DBNAME, user=USER,
 password=token, sslrootcert="SSLCERTIFICATE")
    cur = conn.cursor()
    cur.execute("""SELECT now()""")
    query_results = cur.fetchall()
    print(query_results)
except Exception as e:
    print("Database connection failed due to {}".format(e))
```

프록시를 통해 DB 인스턴스에 연결하려는 경우 IAM 인증을 사용하여 프록시에 연결을 참조하세요.

IAM DB 인증 문제 해결

여기에서는 몇 가지 일반적인 IAM DB 인증 문제에 대한 문제 해결 아이디어와 IAM DB 인증에 대한 CloudWatch 로그 관련 정보를 확인할 수 있습니다.

IAM DB 인증 오류 로그를 CloudWatch Logs로 내보내기

IAM DB 인증 오류 로그는 데이터베이스 호스트에 저장되며 이러한 로그를 CloudWatch Logs 계정으로 내보낼 수 있습니다. 이 페이지의 로그 및 문제 해결 방법을 사용하여 IAM DB 인증 문제를 해결합니다.

콘솔, AWS CLI 및 RDS API에서 CloudWatch Logs로 로그 내보내기를 사용 설정할 수 있습니다. 콘솔 지침은 Amazon CloudWatch Logs에 데이터베이스 로그 게시(을)를 참조하세요.

AWS CLI에서 DB 인스턴스 를 만들 때 IAM DB 인증 오류 로그를 CloudWatch Logs로 내보내려면 다음 명령을 사용합니다.

```
aws rds create-db-instance --db-instance-identifier mydbinstance \
--region us-east-1 \
--db-instance-class db.t3.large \
--allocated-storage 50 \
--engine postgres \
--engine-version 16 \
--port 5432 \
--master-username master \
--master-user-password password \
--publicly-accessible \
--enable-iam-database-authentication \
--enable-cloudwatch-logs-exports=iam-db-auth-error
```

AWS CLI에서 DB 인스턴스 를 수정할 때 IAM DB 인증 오류 로그를 CloudWatch Logs로 내보내려면 다음 명령을 사용합니다.

```
aws rds modify-db-instance --db-instance-identifier mydbinstance \
--region us-east-1 \
--cloudwatch-logs-export-configuration '{"EnableLogTypes":["iam-db-auth-error"]}'
```

DB 인스턴스 가 IAM DB 인증 로그를 CloudWatch Logs로 내보내고 있는지 확인하려면 `describe-db-instances` 명령의 출력에서 `EnabledCloudwatchLogsExports` 파라미터가 `iam-db-auth-error`로 설정되어 있는지 확인합니다.

```
aws rds describe-db-instances --region us-east-1 --db-instance-identifier mydbinstance
            ...

             "EnabledCloudwatchLogsExports": [
                "iam-db-auth-error"
            ],
            ...
```

IAM DB 인증 CloudWatch 지표

Amazon RDS 는 IAM DB 인증에 대한 실시간에 가까운 지표를 Amazon CloudWatch 계정에 제공합니다. 다음 표에는 CloudWatch를 사용하여 사용할 수 있는 IAM DB 인증 지표가 나열되어 있습니다.

지표	설명
IamDbAuthConnectionRequests	IAM DB 인증으로 구성된 총 연결 요청 개수입니다.
IamDbAuthConnectionSuccess	성공한 IAM DB 인증 요청의 총 개수입니다.
IamDbAuthConnectionFailure	실패한 IAM DB 인증 요청의 총 개수입니다.
IamDbAuthConnectionFailureInvalidToken	유효하지 않은 토큰으로 인해 실패한 IAM DB 인증 요청의 총 개수입니다.
IamDbAuthConnectionFailureInsufficientPermissions	잘못된 정책 또는 권한으로 인해 실패한 IAM DB 인증 요청의 총 개수입니다.
IamDbAuthConnectionFailureThrottling	IAM DB 인증 스로틀링으로 인해 실패한 IAM DB 인증 요청의 총 개수입니다.
IamDbAuthConnectionFailureServerError	IAM DB 인증 기능의 내부 서버 오류로 인해 실패한 IAM DB 인증 요청의 총 개수입니다.

일반적인 문제 및 해결 방법

IAM DB 인증 사용 시 다음과 같은 문제가 발생할 수 있습니다. 표에 나와 있는 문제 해결 단계를 사용하여 문제를 해결합니다.

오류	지표	원인	솔루션
[ERROR] Failed to authenticate the connection request for user *db_user* because the provided token is malformed or otherwise invalid. (Status Code: 400, Error Code: InvalidToken)	IamDbAuthConnectionFailure IamDbAuthConnectionFailureInvalidToken	연결 요청의 IAM DB 인증 토큰이 유효한 SigV4a 토큰이 아니거나 올바른 형식이 아닙니다.	애플리케이션에서 토큰 생성 전략을 확인합니다. 경우에 따라 토큰을 유효한 형식으로 전달해야 합니다. 토큰을 잘라내거나 잘못된 문자열 형식으로 지정하면 토큰이 무효화됩니다.
[ERROR] Failed to authenticate the connection request for user *db_user* because the token age is longer than 15 minutes. (Status Code: 400, Error Code:ExpiredToken)	IamDbAuthConnectionFailure IamDbAuthConnectionFailureInvalidToken	IAM DB 인증 토큰이 만료되었습니다. 토큰은 15분 동안만 유효합니다.	애플리케이션에서 토큰 캐싱 및/또는 토큰 재사용 로직을 확인합니다. 15분보다 오래된 토큰은 재사용해서는 안 됩니다.
[ERROR] Failed to authorize the connection request for user *db_user* because the IAM policy assumed by the caller 'arn:aws:sts::1234	IamDbAuthConnectionFailure IamDbAuthConnectionFailureInsufficientPermissions	이 오류는 다음과 같은 이유 때문일 수 있습니다. • 애플리케이션에서 수임하는 IAM 정책은 rds-db:connect 작업을 승인하지 않습니다.	애플리케이션에서 수임하는 IAM 역할 및/또는 정책을 확인합니다. DB에 연결할 때와 동일한 정책을 수임하여 토큰을 생성해야 합니다.

오류	지표	원인	솔루션
56789012: assumed-role/ <RoleName>/ <RoleSession>' is not authorized to perform `rds-db:connect` on the DB instance. (Status Code: 403, Error Code:NotAuthorized)		• *db_user*가 데이터베이스에 연결하는 데 잘못된 역할/정책을 수임하고 있습니다. • *db_user*에 대해 올바른 정책을 수임하고 있지만 올바른 데이터베이스에 연결되어 있지 않습니다.	
[ERROR] Failed to authorize the connection request for user *db_user* due to IAM DB authentication throttling. (Status Code: 429, Error Code: ThrottlingException)	IamDbAuthConnectionFailure IamDbAuthConnectionFailureThrottling	짧은 시간 내에 DB에 너무 많은 연결 요청을 하고 있습니다. IAM DB 인증 스로틀링 한도는 초당 200개의 연결입니다.	IAM 인증으로 새 연결을 설정하는 속도를 줄입니다. 애플리케이션에서 설정된 연결을 재사용하려면 RDS 프록시를 사용하여 연결 풀링을 구현하는 것이 좋습니다.

오류	지표	원인	솔루션

`[ERROR] Failed to authorize the connection request for user db_user due to an internal IAM DB authentication error. (Status Code: 500, Error Code: InternalError)`	`IamDbAuthConnectionFailure`
	
	`IamDbAuthConnectionFailureThrottling`
		IAM DB 인증으로 DB 연결을 승인하는 동안 내부 오류가 발생했습니다.	문제를 조사하려면 https://aws.amazon.com/premiumsupport/를 통해 문의하시기 바랍니다.

Amazon RDS 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 Amazon RDS 및 IAM에서 발생할 수 있는 공통적인 문제를 진단하고 수정할 수 있습니다.

주제

- Amazon RDS에서 작업을 수행할 권한이 없음
- iam:PassRole을 수행하도록 인증되지 않음
- 내 AWS 계정 외부의 사용자가 내 Amazon RDS 리소스에 액세스할 수 있도록 하려고 합니다.

Amazon RDS에서 작업을 수행할 권한이 없음

AWS Management Console에서 작업을 수행할 권한이 없다는 메시지가 나타나는 경우 관리자에게 문의하여 도움을 받아야 합니다. 관리자는 로그인 보안 인증 정보를 제공한 사람입니다.

다음 예제 오류는 `mateojackson` 사용자가 콘솔을 사용하여 *##*에 대한 세부 정보를 보려고 하지만 `rds:GetWidget` 권한이 없는 경우에 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 rds:GetWidget on resource: my-example-widget
```

이 경우 Mateo는 *my-example-widget* 태스크를 사용하여 rds:*GetWidget* 리소스에 액세스하도록 허용하는 정책을 업데이트하라고 관리자에게 요청합니다.

iam:PassRole을 수행하도록 인증되지 않음

iam:PassRole 태스크를 수행할 권한이 없다는 오류가 수신되면 관리자에게 문의하여 도움을 받아야 합니다. 관리자는 로그인 보안 인증 정보를 제공한 사람입니다. 역할을 Amazon RDS로 전달하도록 허용하는 정책을 업데이트하라고 관리자에게 요청합니다.

일부 AWS 서비스에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신, 해당 서비스에 기존 역할을 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 이름이 marymajor인 사용자가 콘솔을 사용하여 Amazon RDS에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 태스크를 수행하려면 서비스에 서비스 역할이 부여한 권한이 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우 Mary는 iam:PassRole 태스크를 수행하도록 허용하는 정책을 업데이트하라고 관리자에게 요청합니다.

내 AWS 계정 외부의 사용자가 내 Amazon RDS 리소스에 액세스할 수 있도록 하려고 합니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하십시오.

- Amazon RDS에서 이러한 기능을 지원하는지 여부를 알아보려면 Amazon RDS에서 IAM을 사용하는 방법 단원을 참조하십시오.
- 소유하고 있는 AWS 계정의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 자신이 소유한 다른 AWS 계정의 IAM 사용자에 대한 액세스 권한 제공을 참조하십시오.
- 리소스에 대한 액세스 권한을 서드 파티 AWS 계정에게 제공하는 방법을 알아보려면 IAM 사용 설명서의 서드 파티가 소유한 AWS 계정에 대한 액세스 제공을참조하십시오.

- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하십시오.
- 교차 계정 액세스를 위한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하십시오.

Amazon RDS의 로깅 및 모니터링

모니터링은 Amazon RDS 와 AWS 솔루션의 신뢰성, 가용성 및 성능을 유지하는 중요한 역할을 합니다. 다중 지점 실패가 발생할 경우 보다 쉽게 디버깅할 수 있도록 AWS 솔루션의 모든 부분으로부터 모니터링 데이터를 수집해야 합니다. AWS는 Amazon RDS 리소스를 모니터링하고 잠재적 인시던트에 대응하기 위한 여러 도구를 제공합니다.

Amazon CloudWatch 경보

Amazon CloudWatch 경보를 사용하면 지정한 기간 동안 단일 지표를 감시합니다. 지표가 지정된 임계값을 초과하면 Amazon SNS 주제 또는 AWS Auto Scaling 정책으로 알림이 전송됩니다. CloudWatch 경보는 단순히 특정 상태에 있다고 해서 작업을 호출하지 않습니다. 대신, 상태가 변경되어 지정된 기간 동안 유지되어야 합니다.

AWS CloudTrail 로그

CloudTrail은 Amazon RDS 에서 사용자, 역할 또는 AWS 서비스가 수행한 작업의 기록을 제공합니다. CloudTrail은 콘솔의 호출과 Amazon RDS API 작업에 대한 코드 호출을 포함하여 Amazon RDS 에 대한 모든 API 직접 호출을 이벤트로 캡처합니다. CloudTrail에서 수집하는 정보를 사용하여 Amazon RDS 에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다. 자세한 정보는 AWS CloudTrail에서 Amazon RDS API 호출 모니터링을 참조하세요.

확장 모니터링

Amazon RDS 는 DB 인스턴스 가 실행되는 운영 체제(OS)에 대한 측정치를 실시간으로 제공합니다. 콘솔을 사용하여 DB 인스턴스 에 대한 측정치를 보거나, 선택한 모니터링 시스템의 Amazon CloudWatch Logs에서 확장 모니터링 JSON 출력을 사용할 수 있습니다. 자세한 정보는 Enhanced Monitoring을 사용하여 OS 지표 모니터링을 참조하세요.

Amazon RDS 성능 개선 도우미

성능 개선 도우미는 기존 Amazon RDS 모니터링 기능을 확장한 것으로서 데이터베이스 성능을 표시하여 영향을 미치는 성능 문제를 분석하는 데 효과적입니다. 성능 개선 도우미 대시보드가 데이터베이스 부하를 시각화하여 대기 시간, SQL 문, 호스트 또는 사용자를 기준으로 부하를 필터링합니다. 자세한 정보는 성능 개선 도우미를 통한 Amazon RDS 모니터링을 참조하세요.

데이터베이스 로그

AWS Management Console 콘솔, AWS CLI 또는 RDS API를 사용하여 데이터베이스 로그를 보고 다운로드하고 조사할 수 있습니다. 자세한 정보는 Amazon RDS 로그 파일 모니터링을 참조하세요.

Amazon RDS 권장 사항

Amazon RDS 에서 데이터베이스 리소스에 대한 자동 권장 사항을 제공합니다. 이러한 권장 사항은 DB 인스턴스 구성, 사용량 및 성능 데이터 분석을 통해 모범 사례 지침을 제공합니다. 자세한 정보는 Amazon RDS의 권장 사항을 참조하세요.

Amazon RDS 이벤트 알림

Amazon RDS 는 Amazon RDS 이벤트 발생 시 Amazon Simple Notification Service(Amazon SNS)를 사용하여 알림 서비스를 제공합니다. 이 서비스는 AWS 리전에 따라 Amazon SNS가 지원하는 알림 메시지 형식에 따라 이메일, 문자 또는 HTTP 엔드포인트 호출 등이 될 수 있습니다. 자세한 정보는 Amazon RDS 이벤트 알림 작업을 참조하세요.

AWS Trusted Advisor

Trusted Advisor는 수십만 명의 AWS 고객에게 서비스를 제공하면서 익힌 모범 사례를 활용합니다. Trusted Advisor는 AWS 환경을 검사한 후 비용 절감, 시스템 가용성 및 성능 향상 또는 보안 격차를 해결할 기회가 있을 때 권장 사항을 제시합니다. 모든 AWS 고객은 5개의 Trusted Advisor 점검 항목에 액세스할 수 있습니다. Business 또는 Enterprise Support 플랜을 보유한 고객은 모든 Trusted Advisor 점검 항목을 볼 수 있습니다.

Trusted Advisor에는 다음과 같이 Amazon RDS 관련 검사가 있습니다.

- Amazon RDS 유휴 DB 인스턴스
- Amazon RDS 보안 그룹 액세스 위험
- Amazon RDS 백업
- Amazon RDS 다중 AZ

이러한 사항에 대한 자세한 정보를 알고 싶다면 Trusted Advisor Best Practices (Checks) 단원을 참조하십시오.

Amazon RDS 모니터링에 대한 자세한 내용은 Amazon RDS 인스턴스에서 지표 모니터링 섹션을 참조하세요.

Amazon RDS의 규정 준수 확인

서드 파티 감사자는 여러 AWS 규정 준수 프로그램의 일환으로 Amazon RDS의 보안 및 규정 준수를 평가합니다. 여기에는 SOC, PCI, FedRAMP, HIPAA 등이 포함됩니다.

특정 규정 준수 프로그램 범위에 속하는 AWS 서비스의 목록은 규정 준수 프로그램 제공 AWS 범위 내 서비스를 참조하세요. 일반 정보는 AWS 규정 준수 프로그램을 참조하세요.

AWS Artifact를 사용하여 제3자 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 AWS Artifact의 보고서 다운로드를 참조하십시오.

Amazon RDS 사용 시 규정 준수 책임은 데이터의 민감도, 조직의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다. AWS에서는 규정 준수를 지원할 다음과 같은 리소스를 제공합니다.

- 보안 및 규정 준수 빠른 시작 안내서 – 이 배포 안내서에서는 아키텍처 고려 사항에 대해 설명하고 보안 및 규정 준수에 중점을 둔 기본 AWS 환경을 배포하기 위한 단계를 제공합니다.
- Architecting for HIPAA Security and Compliance on Amazon Web Services(Amazon Web Services에서 HIPAA 보안 및 규정 준수를 위한 설계) - 이 백서는 기업에서 AWS를 사용하여 HIPAA를 준수하는 애플리케이션을 생성하는 방법을 설명합니다.
- AWS 규정 준수 리소스 이 워크북 및 안내서는 귀사의 산업 및 위치에 적용될 수 있습니다.
- AWS Config – 이 AWS 서비스로 리소스 구성이 내부 관행, 업계 지침 및 규정을 준수하는 정도를 평가할 수 있습니다.
- AWS Security Hub – 이 AWS 서비스(은)는 AWS 내의 보안 상태에 대한 포괄적인 보기를 제공합니다. Security Hub는 보안 제어를 사용하여 AWS리소스를 평가하고 보안 업계 표준 및 모범 사례에 대한 규정 준수를 확인합니다. 지원되는 서비스 및 제어 목록은 Security Hub 제어 참조를 참조하세요.

Amazon RDS의 복원성

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다. AWS 리전은 물리적으로 분리되고 격리된 다수의 가용 리전을 제공하며 이러한 가용 리전은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크에 연결되어 있습니다. 가용 영역을 사용하면 중단 없이 가용 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하십시오.

AWS 글로벌 인프라 외에 Amazon RDS도 데이터 복원성과 백업 요구 사항을 지원하는 여러 가지 기능을 제공합니다.

백업 및 복원

Amazon RDS는 DB 인스턴스의 자동 백업을 생성하고 저장합니다. Amazon RDS는 개별 데이터베이스가 아닌 전체 DB 인스턴스를 백업하여 DB 인스턴스의 스토리지 볼륨 스냅샷을 생성합니다.

Amazon RDS는 DB 인스턴스 백업 기간 동안 DB 인스턴스의 자동 백업을 생성합니다. Amazon RDS는 사용자가 지정한 백업 보존 기간에 따라 DB 인스턴스의 자동 백업을 저장합니다. 필요할 경우 백업 보존 기간 중 어느 특정 시점으로든 데이터베이스를 복구할 수 있습니다. 또한 수동으로 DB 스냅샷을 생성하여 DB 인스턴스를 백업할 수도 있습니다.

원본 DB 인스턴스가 장애로 중단될 경우 재해 복구 솔루션인 이 DB 스냅샷에서 복원하여 DB 인스턴스를 생성할 수 있습니다.

자세한 내용은 데이터 백업, 복원 및 내보내기 섹션을 참조하세요.

복제

Amazon RDS는 MariaDB, MySQL, Oracle 및 PostgreSQL DB 엔진의 기본 복제 기능을 사용하여 원본 DB 인스턴스의 읽기 전용 복제본이라고 하는 특수한 유형의 DB 인스턴스를 생성할 수 있습니다. 원본 DB 인스턴스에 적용된 업데이트는 읽기 전용 복제본에 비동기식으로 적용됩니다. 애플리케이션에서 읽기 전용 복제본으로 읽기 쿼리를 라우팅하여 원본 DB 인스턴스의 로드를 줄일 수 있습니다. 읽기 전용 복제본을 사용하면 읽기 중심의 데이터베이스 워크로드에 대한 단일 DB 인스턴스의 용량 제한에서 벗어나 탄력적으로 늘릴 수 있습니다. 원본 DB 인스턴스가 실패할 경우 재해 복구 솔루션으로 읽기 전용 복제본을 독립형 인스턴스로 승격할 수 있습니다. 일부 DB 엔진의 경우, Amazon RDS에서도 다른 복제 옵션이 지원됩니다.

자세한 내용은 DB 인스턴스 읽기 전용 복제본 작업 섹션을 참조하세요.

Failover

Amazon RDS는 다중 AZ 배포를 사용해 DB 인스턴스에 고가용성과 장애 조치 기능을 지원합니다. Amazon RDS는 다양한 기술을 이용해 장애 조치 지원을 제공합니다. Oracle, PostgreSQL, MySQL 및 MariaDB DB 인스턴스용 다중 AZ 배포는 Amazon의 장애 조치 기술을 사용합니다. SQL Server DB 인스턴스는 SQL Server 데이터베이스 미러링(DBM)을 사용합니다.

자세한 내용은 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 섹션을 참조하세요.

Amazon RDS의 인프라 보안

관리형 서비스인 Amazon Relational Database Service는 AWS 전역 네트워크 보안으로 보호됩니다. AWS 보안 서비스와 AWS의 인프라 보호 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하세요. 인프라 보안에 대한 모범 사례를 사용하여 AWS 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 인프라 보호를 참조하세요.

AWS에서 게시한 API 호출을 사용하여 네트워크를 통해 Amazon RDS에 액세스합니다. 고객은 다음을 지원해야 합니다.

- Transport Layer Security(TLS) TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군 Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 주체와 관련된 비밀 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 인증을 생성하여 요청에 서명할 수 있습니다.

또한 Amazon RDS는 인프라 보안을 지원하는 기능을 제공합니다.

보안 그룹

보안 그룹은 DB 인스턴스에서 송수신되는 트래픽에 대한 액세스를 제어합니다. 기본적으로 DB 인스턴스에 대한 네트워크 액세스는 해제되어 있습니다. IP 주소 범위, 포트 또는 보안 그룹에서 액세스를 허용하는 보안 그룹의 규칙을 지정할 수 있습니다. 수신 규칙이 설정되면 동일한 규칙이 해당 보안 그룹과 연결된 모든 DB 인스턴스에 적용됩니다.

자세한 내용은 보안 그룹을 통한 액세스 제어 단원을 참조하세요.

퍼블릭 액세스 가능성

Amazon VPC 서비스 기반 Virtual Private Cloud(VPC)에서 DB 인스턴스를 시작할 때 해당 DB 인스턴스의 퍼블릭 액세스를 켜거나 끌 수 있습니다. 퍼블릭 액세스 가능성 파라미터를 사용하여 사용자가 생성한 DB 인스턴스가 퍼블릭 IP 주소로 확인되는 DNS 이름을 가지도록 지정할 수 있습니다. 이 파라미터를 사용하여 DB 인스턴스에 대한 퍼블릭 액세스 여부를 지정할 수 있습니다. Public accessibility 파라미터를 수정하여 퍼블릭 액세스 가능성을 켜거나 끄도록 DB 인스턴스를 수정할 수 있습니다.

자세한 내용은 VPC에 있는 DB 인스턴스를 인터넷에서 숨기기 섹션을 참조하세요.

> **Note**
>
> DB 인스턴스가 VPC에 있지만 공개적으로 액세스할 수 없는 경우 AWS Site-to-Site VPN 연결 또는 AWS Direct Connect 연결을 사용하여 프라이빗 네트워크에서 액세스할 수도 있습니다. 자세한 내용은 인터네트워크 트래픽 개인 정보 단원을 참조하세요.

Amazon RDS API 및 인터페이스 VPC 엔드포인트(AWS PrivateLink)

인터페이스VPC 엔드포인트를 생성하여 VPC와 Amazon RDS API 엔드포인트 간에 프라이빗 연결을 설정할 수 있습니다. 인터페이스 엔드포인트는 로 구동됩니다AWS PrivateLink

AWS PrivateLink를 사용하면 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 AWS Direct Connect 연결 없이 Amazon RDS API 작업에 비공개로 액세스할 수 있습니다. VPC의 DB 인스턴스는 DB 인스턴스 를 시작, 수정 또는 종료하기 위해 Amazon RDS API 엔드포인트와 통신하는 데 퍼블릭 IP 주소가 필요하지 않습니다. 또한 DB 인스턴스가 사용 가능한 RDS API 작업을 사용하기 위해 퍼블릭 IP 주소가 필요하지 않습니다. VPC와 Amazon RDS 간의 트래픽은 Amazon 네트워크를 벗어나지 않습니다.

각 인터페이스 엔드포인트는 서브넷에서 하나 이상의 탄력적 네트워크 인터페이스로 표현됩니다. 탄력적 네트워크 인터페이스에 대한 자세한 내용은 Amazon EC2 사용 설명서의 탄력적 네트워크 인터페이스를 참조하세요.

VPC 엔드포인트에 대한 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 VPC 엔드포인트(AWS PrivateLink)를 참조하세요. RDS API 작업에 대한 자세한 내용은 Amazon RDS API 참조를 참조하세요.

DB 인스턴스에 연결하기 위해 인터페이스 VPC 엔드포인트가 필요하지 않습니다. 자세한 내용은 VPC에서 DB 인스턴스에 액세스하는 시나리오을 참조하세요.

VPC 엔드포인트에 대한 고려 사항

Amazon RDS API 엔드포인트에 대한 인터페이스 VPC 엔드포인트를 설정하기 전에 Amazon VPC 사용 설명서에서 인터페이스 엔드포인트 속성 및 제한 사항을 검토해야 합니다.

Amazon RDS 리소스 관리와 관련된 모든 RDS API 작업은 AWS PrivateLink를 사용하여 VPC에서 사용할 수 있습니다.

VPC 엔드포인트 정책은 RDS API 엔드포인트에 대해 지원됩니다. 기본적으로, 엔드포인트를 통해 RDS API 작업에 대한 전체 액세스가 허용됩니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하세요.

가용성

현재 Amazon RDS API가 VPC 엔드포인트를 지원하는 AWS 리전은 다음과 같습니다.

- 미국 동부(오하이오)
- 미국 동부(버지니아 북부)
- 미국 서부(캘리포니아 북부)
- 미국 서부(오레곤)
- 아프리카(케이프타운)
- 아시아 태평양(홍콩)
- 아시아 태평양(뭄바이)
- 아시아 태평양(오사카)
- 아시아 태평양(서울)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(도쿄)
- 캐나다(중부)
- 캐나다 서부(캘거리)
- 중국(베이징)
- 중국(닝샤)
- 유럽(프랑크푸르트)
- 유럽(취리히)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(파리)
- 유럽(스톡홀름)
- 유럽(밀라노)
- 이스라엘(텔아비브)
- 중동(바레인)
- 남아메리카(상파울루)
- AWS GovCloud(미국 동부)
- AWS GovCloud(미국 서부)

Amazon RDS API에 대한 인터페이스 VPC 엔드포인트 생성

Amazon VPC 콘솔 또는 AWS Command Line Interface(AWS CLI)를 사용하여 Amazon RDS API에 대한 VPC 엔드포인트를 생성할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 생성을 참조하십시오.

서비스 이름 `com.amazonaws.region.rds`을 사용하여 Amazon RDS API에 대한 VPC 엔드포인트를 생성합니다.

중국의 AWS 리전을 제외하고, 엔드포인트에 프라이빗 DNS를 활성화한 경우 AWS 리전에 대한 기본 DNS 이름(예:`rds.us-east-1.amazonaws.com`)을 사용하여 VPC 엔드포인트를 통해 Amazon RDS에 API 요청을 수행할 수 있습니다. 중국(베이징) 및 중국(닝샤) AWS 리전의 경우 각각 `rds-api.cn-north-1.amazonaws.com.cn` 및 `rds-api.cn-northwest-1.amazonaws.com.cn`을 사용하여 VPC 엔드포인트를 통해 API 요청을 수행할 수 있습니다.

자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트를 통해 서비스 액세스를 참조하십시오.

Amazon RDS API에 대한 VPC 엔드포인트 정책 생성

Amazon RDS API에 대한 액세스를 제어하는 VPC 엔드포인트에 엔드포인트 정책을 연결할 수 있습니다. 이 정책은 다음 정보를 지정합니다.

- 작업을 수행할 수 있는 보안 주체.
- 수행할 수 있는 작업.
- 작업을 수행할 수 있는 리소스.

자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하십시오.

예제: Amazon RDS API 작업에 대한 VPC 엔드포인트 정책

다음은 Amazon RDS API에 대한 엔드포인트 정책의 예입니다. 이 정책은 엔드포인트에 연결될 때 모든 리소스의 모든 보안 주체에 대한 액세스 권한을 나열된 Amazon RDS API 작업에 부여합니다.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
```

```
         "Action":[
            "rds:CreateDBInstance",
            "rds:ModifyDBInstance",
            "rds:CreateDBSnapshot"
         ],
         "Resource":"*"
      }
   ]
}
```

예제: 지정된 AWS 계정의 모든 액세스를 거부하는 VPC 엔드포인트 정책

다음 VPC 엔드포인트 정책은 AWS 계정 123456789012가 엔드포인트를 사용하는 리소스에 대한 모든 액세스를 거부합니다. 이 정책은 다른 계정의 모든 작업을 허용합니다.

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": "*",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": { "AWS": [ "123456789012" ] }
     }
   ]
}
```

Amazon RDS의 보안 모범 사례

AWS Identity and Access Management(IAM) 계정을 사용해 Amazon RDS API 작업, 특히 Amazon RDS 리소스를 만들거나, 수정하거나, 삭제하는 작업에 대한 액세스를 제어합니다. 이러한 리소스 중에는 DB 인스턴스 , 보안 그룹 및 파라미터 그룹이 있습니다. 또한 IAM 을 사용해 DB 인스턴스 백업 및 복구 같은 공통 관리 작업을 수행하는 작업을 제어합니다.

- 본인을 포함하여 Amazon RDS 리소스를 관리하는 각 개인에 대해 개별 사용자를 만듭니다. AWS 루트 자격 증명을 사용하여 Amazon RDS 리소스를 관리하지 마세요.

- 각 사용자에게 각자의 임무를 수행하는 데 필요한 최소 권한 집합을 부여합니다.
- IAM 그룹을 사용해 여러 사용자에 대한 권한을 효과적으로 관리합니다.
- IAM 자격 증명을 정기적으로 순환합니다.
- Amazon RDS 보안 암호를 자동으로 교체할 수 있도록 AWS Secrets Manager를 구성합니다. 자세한 내용은 AWS Secrets Manager 사용 설명서에서 AWS Secrets Manager 비밀 교체를 참조하세요. AWS Secrets Manager 프로그래밍 방식에서 자격 증명을 검색할 수도 있습니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 비밀 값 검색을 참조하세요.

Amazon RDS 보안에 대한 자세한 내용은 Amazon RDS의 보안 섹션을 참조하세요. IAM에 대한 자세한 내용은 AWS Identity and Access Management 단원을 참조하십시오. IAM 모범 사례에 대한 자세한 내용은 IAM 모범 사례 단원을 참조하십시오.

AWS Security Hub는 보안 제어를 사용하여 리소스 구성 및 보안 표준을 평가하여 다양한 규정 준수 프레임워크를 준수할 수 있도록 지원합니다. Security Hub를 사용하여 RDS 리소스를 평가하는 방법에 대한 자세한 내용은 AWS Security Hub 사용 설명서의 Amazon Relational Database Service 제어를 참조하세요.

Security Hub를 사용하여 보안 모범 사례와 관련된 RDS의 사용량을 모니터링할 수 있습니다. 자세한 내용은 AWS Security Hub란 무엇인가요?를 참조하세요.

AWS Management Console, AWS CLI 또는 RDS API를 사용하여 마스터 사용자의 암호를 변경합니다. SQL 클라이언트 등과 같은 다른 도구를 사용하여 마스터 사용자 암호를 변경할 경우 의도치 않게 사용자에 대해 권한이 취소될 수 있습니다.

보안 그룹을 통한 액세스 제어

VPC 보안 그룹은 DB 인스턴스 에서 송수신되는 트래픽에 대한 액세스를 제어합니다. 기본적으로 DB 인스턴스 에 대한 네트워크 액세스는 해제되어 있습니다. IP 주소 범위, 포트 또는 보안 그룹에서 액세스를 허용하는 보안 그룹의 규칙을 지정할 수 있습니다. 수신 규칙이 설정되면 동일한 규칙이 해당 보안 그룹과 연결된 모든 DB 인스턴스 에 적용됩니다. 보안 그룹에서 최대 20개의 규칙을 지정할 수 있습니다.

VPC 보안 그룹 개요

각 VPC 보안 그룹 규칙을 설정하면 특정 소스가 해당 VPC 보안 그룹과 연결되어 있는 VPC의 DB 인스턴스 에 액세스할 수 있습니다. 소스는 주소 범위(예: 203.0.113.0/24) 또는 다른 VPC 보안 그룹일 수 있습니다. VPC 보안 그룹을 소스로 지정하면 소스 VPC 보안 그룹을 사용하는 모든 인스턴스(일반적

으로 애플리케이션 서버)에서 수신 트래픽이 허용됩니다. VPC 보안 그룹에는 인바운드 및 아웃바운드 트래픽을 모두 제어하는 규칙이 있을 수 있습니다. 그러나 아웃바운드 트래픽 규칙은 일반적으로 DB 인스턴스 에 적용되지 않습니다. 아웃바운드 트래픽 규칙은 DB 인스턴스 가 클라이언트 역할을 하는 경우에만 적용됩니다. 예를 들면 아웃바운드 트래픽 규칙은 아웃바운드 데이터베이스 링크가 있는 Oracle DB 인스턴스에 적용됩니다. Amazon EC2 API를 사용하거나, 혹은 VPC 콘솔에서 보안 그룹 옵션을 선택하여 VPC 보안 그룹을 생성해야 합니다.

VPC의 인스턴스 에 대한 액세스를 허용하는 VPC 보안 그룹과 관련된 규칙을 만들 때 그 규칙이 액세스를 허용하는 주소들의 각 범위에 대해 포트를 지정해야 합니다. 예를 들어, VPC의 인스턴스에 대한 Secure Shell(SSH) 액세스를 활성화하고 싶다면 지정된 주소 범위와 관련해 TCP 포트 22에 대한 액세스를 허용하는 규칙을 생성합니다.

VPC의 서로 다른 인스턴스에게 서로 다른 포트에 대한 액세스를 허용하는 여러 개의 VPC 보안 그룹을 구성할 수 있습니다. 예를 들어 VPC의 웹 서버에 대해서는 TCP 포트 80으로 액세스가 가능하도록 VPC 보안 그룹을 생성합니다. 그런 다음 VPC의 RDS for MySQL DB 인스턴스에 대해서는 TCP 포트 3306으로 액세스할 수 있도록 다른 VPC 보안 그룹을 만들면 됩니다.

VPC 보안 그룹에 관한 자세한 내용은 Amazon Virtual Private Cloud 사용 설명서의 보안 그룹을 참조하십시오.

> **Note**
>
> DB 인스턴스 가 VPC에 있지만 공개적으로 액세스할 수 없는 경우 AWS Site-to-Site VPN 연결 또는 AWS Direct Connect 연결을 사용하여 프라이빗 네트워크에서 액세스할 수도 있습니다. 자세한 정보는 인터네트워크 트래픽 개인 정보을 참조하세요.

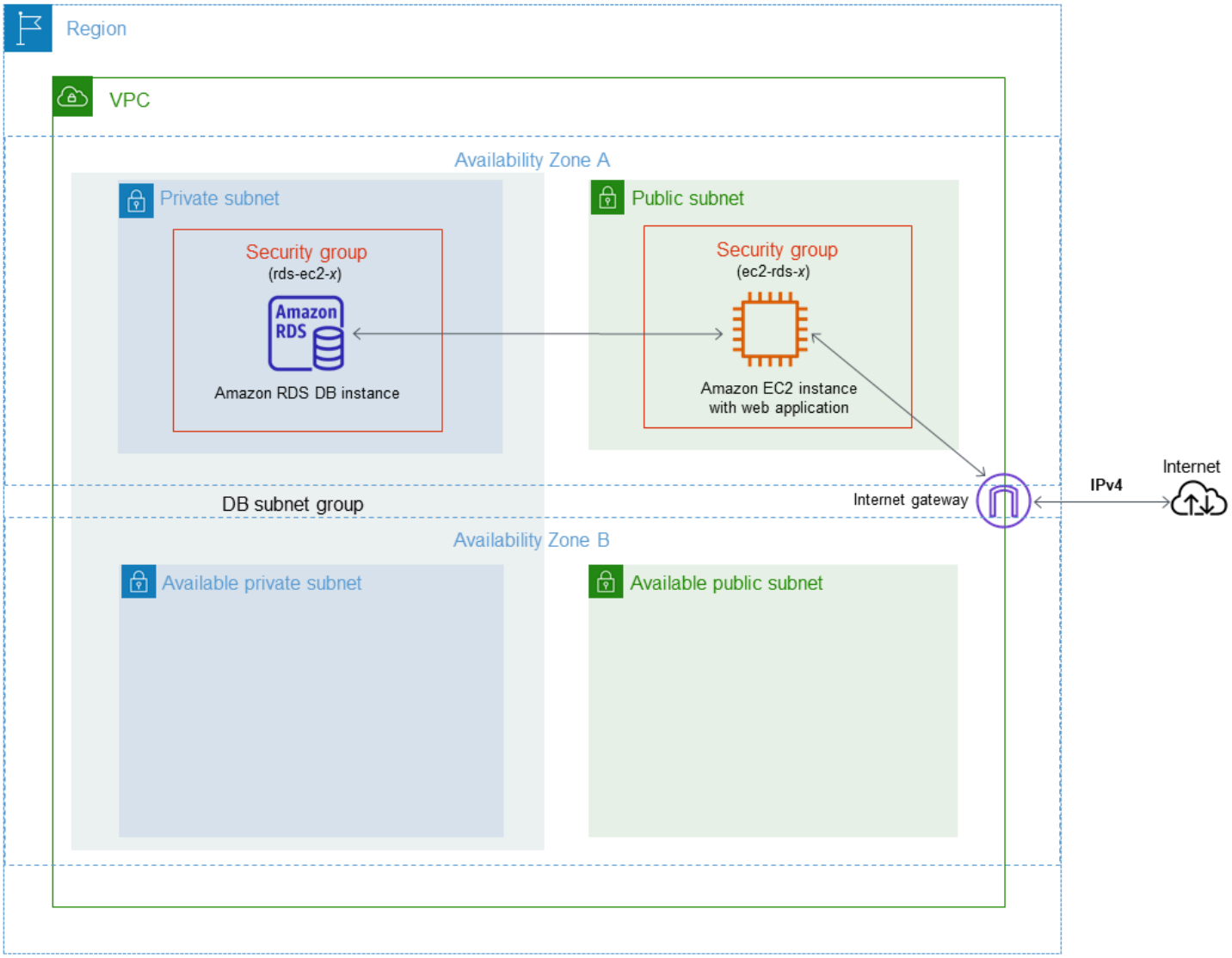
보안 그룹 시나리오

VPC에서 DB 인스턴스 를 사용하는 일반적인 사례는 동일한 VPC의 Amazon EC2 인스턴스에서 실행 중이며 VPC 외부의 클라이언트 애플리케이션에서 액세스한 애플리케이션 서버를 사용하여 데이터를 공유하는 것입니다. 이러한 시나리오에서는 AWS Management Console의 RDS 및 VPC 페이지를 사용하거나, 혹은 RDS 및 EC2 API 작업을 사용하여 필요한 인스턴스와 보안 그룹을 생성합니다.

1. VPC 보안 그룹(예: `sg-0123ec2example`)을 생성하고 클라이언트 애플리케이션의 IP 주소를 소스로 사용하는 인바운드 규칙을 생성합니다. 이 보안 그룹에서는 클라이언트 애플리케이션이 이 보안 그룹을 사용하는 VPC의 EC2 인스턴스에 연결할 수 있습니다.
2. 애플리케이션에 대한 EC2 인스턴스를 생성하고 이전 단계에서 생성한 VPC 보안 그룹(`sg-0123ec2example`)에 EC2 인스턴스를 추가합니다.

3. 두 번째 VPC 보안 그룹(예: `sg-6789rdsexample`)을 생성하고 1단계에서 만든 VPC 보안 그룹(`sg-0123ec2example`)을 소스로 지정해 새 규칙을 생성합니다.
4. 새 DB 인스턴스 를 만들고 이전 단계에서 만든 VPC 보안 그룹(`sg-6789rdsexample`)에 DB 인스턴스 를 추가합니다. DB 인스턴스 를 만들 때 3단계에서 생성한 VPC 보안 그룹(`sg-6789rdsexample`) 규칙에 대해 지정한 것과 동일한 보안 포트 번호를 사용합니다.

다음 다이어그램은 이 시나리오를 보여 줍니다.



이 시나리오의 VPC 구성에 대한 자세한 지침은 자습서: DB 인스턴스에 사용할 Amazon VPC 생성 (IPv4 전용) 섹션을 참조하세요. VPC 사용에 대한 자세한 내용은 Amazon VPC 및 Amazon RDS 섹션을 참조하세요.

VPC 보안 그룹 생성

VPC 콘솔을 사용하여 DB 인스턴스에 대한 VPC 보안 그룹을 생성할 수 있습니다. 보안 그룹 생성에 대한 자세한 내용은 보안 그룹을 생성하여 VPC 내부의 DB 인스턴스에 대한 액세스를 제공 및 Amazon Virtual Private Cloud 사용 설명서의 보안 그룹을 참조하십시오.

보안 그룹과 DB 인스턴스의 연결

보안 그룹은 RDS 콘솔의 [수정(Modify)] 옵션을 사용하거나, `ModifyDBInstance` Amazon RDS API 또는 `modify-db-instance` AWS CLI 명령을 사용해 DB 인스턴스와 연결할 수 있습니다.

다음 CLI 예제는 특정 VPC 보안 그룹을 연결하고 DB 인스턴스에서 DB 보안 그룹을 제거합니다.

```
aws rds modify-db-instance --db-instance-identifier dbName --vpc-security-group-ids sg-ID
```

DB 인스턴스 수정에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요. DB 인스턴스를 DB 스냅샷에서 복원할 때 보안 그룹에서 고려해야 할 사항은 보안 그룹 고려 사항 섹션을 참조하세요.

> **Note**
>
> 포트 값이 기본값이 아닌 값으로 구성된 경우 RDS 콘솔은 데이터베이스에 대해 다른 보안 그룹 규칙 이름을 표시합니다.

Oracle DB 인스턴스용 RDS의 경우 Oracle Enterprise Manager Database Express(OEM), Oracle Management Agent for Enterprise Manager Cloud Control(OEM 에이전트) 및 Oracle Secure Sockets Layer 옵션에 대한 보안 그룹 옵션 설정을 채워 추가 보안 그룹을 연결할 수 있습니다. 이 경우 DB 인스턴스와 연결된 보안 그룹과 옵션 설정이 모두 DB 인스턴스에 적용됩니다. 이러한 옵션 그룹에 대한 자세한 내용은 Oracle Enterprise Manager, Oracle Management Agent for Enterprise Manager Cloud Control 및 Oracle 보안 소켓 Layer 섹션을 참조하세요.

마스터 사용자 계정 권한

새로운 DB 인스턴스 를 만들 때 사용되는 기본 마스터 사용자는 해당 DB 인스턴스 에 대한 특정 권한을 갖습니다. DB 인스턴스 가 만들어진 후에는 마스터 사용자 이름을 변경할 수 없습니다.

Important

애플리케이션에서 직접 마스터 사용자를 사용하지 않는 것이 좋습니다. 대신에 애플리케이션에 필요한 최소 권한으로 생성한 데이터베이스 사용자를 사용하는 모범 사례를 준수하십시오.

Note

마스터 사용자의 권한을 실수로 삭제한 경우, DB 인스턴스 를 수정하고 새 마스터 사용자 암호를 설정하여 복원할 수 있습니다. DB 인스턴스 를 수정하는 것에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.

다음 표에는 마스터 사용자가 각 데이터베이스 엔진에 대해 갖는 권한 및 데이터베이스 역할이 나와 있습니다.

데이터베이스 엔진	시스템 권한	데이터베이스 역할
RDS for Db2	마스터 사용자는 `masterdba` 그룹에 할당되고 `master_user_role` 을 받습니다. SYSMON,DBADM과 DATAACCESS 및 ACCCESSCTRL , BINDADD,CONNECT, CREATETAB ,CREATE_SECURE_OBJECT , EXPLAIN,IMPLICIT_SCHEMA , LOAD,SQLADM,WLMADM	DBA,DBA_RESTRICTED , DEVELOPER ,ROLE_NULLID_PACKAGES , ROLE_PROCEDURES ,ROLE_TABLESPACES 자세한 정보는 Amazon RDS for Db2 기본 역할을 참조하세요.
RDS for MariaDB	SELECT,INSERT,UPDATE,DELETE, CREATE,DROP,RELOAD, PROCESS,REFERENCES ,INDEX, ALTER,SHOW DATABASES ,CREATE TEMPORARY TABLES ,LOCK TABLES, EXECUTE,REPLICATION CLIENT,CREATE VIEW ,SHOW VIEW,CREATE	—

데이터베이스 엔진	시스템 권한	데이터베이스 역할
	ROUTINE, ALTER ROUTINE,CREATE USER, EVENT,TRIGGER,REPLICATION SLAVE RDS for MariaDB 버전 11.4부터 마스터 사용자에게도 SHOW CREATE ROUTINE 권한이 부여됩니다.	
RDS for MySQL 8.0.36 이상	SELECT,INSERT,UPDATE, DELETE,CREATE,DROP, RELOAD,PROCESS, REFERENCES ,INDEX,ALTER, SHOW DATABASES ,CREATE TEMPORARY TABLES,LOCK TABLES,EXECUTE, REPLICATION SLAVE ,REPLICATION CLIENT , CREATE VIEW,SHOW VIEW,CREATE ROUTINE,ALTER ROUTINE,CREATE USER,EVENT,TRIGGER, CREATE ROLE,DROP ROLE, APPLICATION_PASSWORD_ADMIN , ROLE_ADMIN ,SET_USER_ID , XA_RECOVER_ADMIN	rds_superuser_role rds_superuser_role 에 대한 자세한 정보는 RDS for MySQL에 대한 역할 기반 권한 모델 섹션을 참조하세요.
RDS for MySQL 버전 8.0.36 미만	SELECT,INSERT,UPDATE, DELETE,CREATE,DROP, RELOAD,PROCESS, REFERENCES ,INDEX,ALTER, SHOW DATABASES ,CREATE TEMPORARY TABLES,LOCK TABLES,EXECUTE, REPLICATION CLIENT ,CREATE VIEW, SHOW VIEW,CREATE ROUTINE,ALTER ROUTINE,CREATE USER,EVENT, TRIGGER,REPLICATION SLAVE	—

데이터베이스 엔진	시스템 권한	데이터베이스 역할
RDS for PostgreSQL	CREATE ROLE,CREATE DB, PASSWORD VALID UNTIL INFINITY,CREATE EXTENSION ,ALTER EXTENSION ,DROP EXTENSION ,CREATE TABLESPACE ,ALTER <OBJECT> OWNER,CHECKPOINT , PG_CANCEL_BACKEND( ) , PG_TERMINATE_BACKEND() ,SELECT PG_STAT_REPLICATION ,EXECUTE PG_STAT_STATEMENTS_RESET() ,OWN POSTGRES_FDW_HANDLER() ,OWN POSTGRES_FDW_VALIDATOR() ,OWN POSTGRES_FDW , EXECUTE PG_BUFFERCACHE_PAGES() ,SELECT PG_BUFFERCACHE	RDS_SUPERUSER RDS_SUPERUSER에 대한 자세한 내용은 PostgreSQL 역할 및 권한 이해 섹션을 참조하세요.
RDS for Oracle	ADMINISTER DATABASE TRIGGER ,ALTER DATABASE LINK,ALTER PUBLIC DATABASE LINK, AUDIT SYSTEM,CHANGE NOTIFICATION , DROP ANY DIRECTORY ,EXEMPT ACCESS POLICY,EXEMPT IDENTITY POLICY,EXEMPT REDACTION POLICY,FLASHBACK ANY TABLE, GRANT ANY OBJECT PRIVILEGE ,RESTRICTED SESSION ,SELECT ANY TABLE,UNLIMITED TABLESPACE	DBA **Note** DBA 역할은 다음 권한에서 제외됩니다. ALTER DATABASE,ALTER SYSTEM, CREATE ANY DIRECTORY ,CREATE EXTERNAL JOB,CREATE PLUGGABLE DATABASE, GRANT ANY PRIVILEGE ,GRANT ANY ROLE,READ ANY FILE GROUP

데이터베이스 엔진	시스템 권한	데이터베이스 역할
Amazon RDS for Microsoft SQL Server	ADMINISTER BULK OPERATIONS ,ALTER ANY CONNECTION ,ALTER ANY CREDENTIAL , ALTER ANY EVENT SESSION,ALTER ANY LINKED SERVER,ALTER ANY LOGIN,ALTER ANY SERVER AUDIT,ALTER ANY SERVER ROLE, ALTER SERVER STATE,ALTER TRACE, CONNECT SQL,CREATE ANY DATABASE, VIEW ANY DATABASE,VIEW ANY DEFINITION ,VIEW SERVER STATE,ALTER ON ROLE SQLAgentOperatorRole	DB_OWNER(데이터베이스 수준 역할), PROCESSADMIN (서버 수준 역할), SETUPADMIN (서버 수준 역할), SQLAgentUserRole (데이터베이스 수준 역할)

Amazon RDS에 서비스 연결 역할 사용

Amazon RDS는 AWS Identity and Access Management(IAM) 서비스 연결 역할 사용 서비스 연결 역할은 Amazon RDS에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 Amazon RDS에서 사전 정의하며 서비스에서 다른 AWS 서비스를 자동으로 호출하기 위해 필요한 모든 권한을 포함합니다.

필요한 권한을 수동으로 추가할 필요가 없으므로 서비스 연결 역할은 Amazon RDS를 더 쉽게 설정할 수 있습니다. Amazon RDS에서 서비스 연결 역할의 권한을 정의하므로 다르게 정의되지 않으면, Amazon RDS만 해당 역할을 수임할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며, 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

먼저 역할의 관련 리소스를 삭제해야만 역할을 삭제할 수 있습니다. 이렇게 하면 리소스에 대한 액세스 권한을 부주의로 삭제할 수 없기 때문에 Amazon RDS 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는 AWS 서비스를 참조하고 서비스 연결 역할(Service-Linked Role) 열에 예(Yes)가 표시된 서비스를 찾으세요. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 링크가 있는 예를 선택합니다.

Amazon RDS에 대한 서비스 연결 역할 권한

Amazon RDS에서는 AWSServiceRoleForRDS라는 서비스 연결 역할을 사용하여 Amazon RDS가 DB 인스턴스를 대신하여 AWS 서비스를 호출하도록 허용합니다.

AWSServiceRoleForRDS 서비스 연결 역할은 역할을 수임하기 위해 다음 서비스를 신뢰합니다.

- `rds.amazonaws.com`

이 서비스 연결 역할에는 계정에서 운영할 수 있는 권한을 부여하는 `AmazonRDSServiceRolePolicy`라는 권한 정책이 연결되어 있습니다.

JSON 정책 문서를 포함하여 이 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 안내서의 AmazonRDSServiceRolePolicy를 참조하세요.

> **Note**
>
> IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 링크 역할을 생성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 다음 오류 메시지가 표시되는 경우:
> 리소스를 만들 수 없습니다. 서비스 연결 역할을 생성할 권한이 있는지 확인하십시오. 그렇지 않은 경우 기다렸다가 나중에 다시 시도하십시오.

다음 권한이 활성화되어 있는지 확인하십시오.

```
{
    "Action": "iam:CreateServiceLinkedRole",
    "Effect": "Allow",
    "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
    "Condition": {
        "StringLike": {
            "iam:AWSServiceName":"rds.amazonaws.com"
        }
    }
}
```

자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하세요.

Amazon RDS에 대한 서비스 연결 역할 생성

서비스 연결 역할은 수동으로 생성할 필요가 없습니다. DB 인스턴스를 생성하면 Amazon RDS에서 서비스 연결 역할을 생성합니다.

> **Important**
>
> 서비스 연결 역할을 지원하기 시작한 2017년 12월 1일 이전에 Amazon RDS 서비스를 사용하고 있었다면, Amazon RDS가 계정에 AWSServiceRoleForRDS 역할을 생성했습니다. 자세한 내용은 내 AWS 계정에 표시되는 새 역할을 참조하세요.

이 서비스 연결 역할을 삭제한 다음 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. DB 인스턴스를 생성하면 Amazon RDS에서 서비스 연결 역할을 다시 생성합니다.

Amazon RDS에 대한 서비스 연결 역할 편집

Amazon RDS는 AWSServiceRoleForRDS 서비스 연결 역할을 편집하도록 허용하지 않습니다. 서비스 연결 역할을 생성한 후에는 다양한 개체가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하세요.

Amazon RDS에 대한 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제할 것을 권합니다. 이렇게 하면 적극적으로 모니터링하거나 유지 관리하지 않은 미사용 엔터티가 없습니다. 그러나 서비스 연결 역할을 삭제하려면 먼저 모든 DB 인스턴스 를 삭제해야 합니다.

서비스 연결 역할 정리

IAM을 사용하여 서비스 연결 역할을 삭제하기 전에 먼저 역할에 활성 세션이 없는지 확인하고 역할에서 사용되는 리소스를 모두 제거해야 합니다.

IAM 콘솔에서 서비스 연결 역할에 활성 세션이 있는지 확인하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. IAM 콘솔의 탐색 창에서 역할(Roles)을 선택합니다. 그런 다음 AWSServiceRoleForRDS 역할의 이름(확인란 아님)을 선택합니다.
3. 선택한 역할의 요약 페이지에서 마지막 액세스 탭을 선택합니다.
4. 마지막 액세스 탭에서 서비스 연결 역할의 최근 활동을 검토합니다.

> **Note**
>
> Amazon RDS에서 AWSServiceRoleForRDS 역할을 사용하는지 잘 모를 경우에는 역할을 삭제할 수 있습니다. 서비스에서 역할을 사용하는 경우에는 삭제가 안 되어 역할이 사용 중인 AWS 리전을 볼 수 있습니다. 역할이 사용 중인 경우에는 세션이 종료될 때까지 기다렸다가 역할을 삭제해야 합니다. 서비스 연결 역할에 대한 세션은 취소할 수 없습니다.

AWSServiceRoleForRDS 역할을 제거하려면 먼저 모든 DB 인스턴스 를 삭제해야 합니다.

모든 인스턴스 삭제

이러한 절차 중 하나에 따라 각 인스턴스를 삭제합니다.

인스턴스를 삭제하려면(콘솔)

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 데이터베이스를 선택합니다.
3. 삭제하려는 인스턴스를 선택합니다.

4. [Actions]에 대해 [Delete]를 선택합니다.
5. 최종 스냅샷 생성? 메시지가 표시되는 경우 예 또는 아니요를 선택합니다.
6. 이전 단계에서 예를 선택한 경우 최종 스냅샷 이름에 최종 스냅샷 이름을 입력합니다.
7. Delete(삭제)를 선택합니다.

인스턴스를 삭제하려면(CLI)

AWS CLI 명령 참조에서 `delete-db-instance` 섹션을 참조하세요.

인스턴스를 삭제하려면(API)

Amazon RDS API Reference의 `DeleteDBInstance` 참조.

IAM 콘솔, IAM CLI 또는 IAM API를 사용하여 AWSServiceRoleForRDS 서비스 연결 역할을 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 삭제를 참조하십시오.

Amazon RDS Custom에 대한 서비스 연결 역할 권한

Amazon RDS Custom은 `AWSServiceRoleForRDSCustom`이라는 서비스 연결 역할을 사용하여 RDS Custom이 RDS DB 리소스를 대신하여 AWS 서비스를 직접 호출하도록 허용합니다.

AWSServiceRoleForRDSCustom 서비스 연결 역할은 역할을 수임하기 위해 다음 서비스를 신뢰합니다.

- `custom.rds.amazonaws.com`

이 서비스 연결 역할에는 계정에서 운영할 수 있는 권한을 부여하는 `AmazonRDSCustomServiceRolePolicy`라는 권한 정책이 연결되어 있습니다.

RDS Custom에 대한 서비스 연결 역할을 생성, 편집 또는 삭제 작업은 Amazon RDS와 동일하게 작동합니다. 자세한 내용은 AWS 관리형 정책: AmazonRDSCustomServiceRolePolicy 섹션을 참조하세요.

> **Note**
>
> IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 링크 역할을 생성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 다음 오류 메시지가 표시되는 경우:
> 리소스를 만들 수 없습니다. 서비스 연결 역할을 생성할 권한이 있는지 확인하십시오. 그렇지 않은 경우 기다렸다가 나중에 다시 시도하십시오.

다음 권한이 활성화되어 있는지 확인하십시오.

```
{
    "Action": "iam:CreateServiceLinkedRole",
    "Effect": "Allow",
    "Resource": "arn:aws:iam::*:role/aws-service-role/custom.rds.amazonaws.com/
AmazonRDSCustomServiceRolePolicy",
    "Condition": {
        "StringLike": {
            "iam:AWSServiceName":"custom.rds.amazonaws.com"
        }
    }
}
```

자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하세요.

Amazon RDS 베타에 대한 서비스 연결 역할 권한

Amazon RDS는 AWSServiceRoleForRDSBeta라는 서비스 연결 역할을 사용하여 Amazon RDS가 RDS DB 리소스를 대신하여 AWS 서비스를 직접 호출할 수 있도록 지원합니다.

AWSServiceRoleForRDSBeta 서비스 연결 역할은 역할을 수임하기 위해 다음 서비스를 신뢰합니다.

- rds.amazonaws.com

이 서비스 연결 역할에는 계정에서 운영할 수 있는 권한을 부여하는 AmazonRDSBetaServiceRolePolicy라는 권한 정책이 연결되어 있습니다. 자세한 내용은 AWS 관리형 정책: AmazonRDSBetaServiceRolePolicy 섹션을 참조하세요.

> **Note**
>
> IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 링크 역할을 생성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 다음 오류 메시지가 표시되는 경우:
> 리소스를 만들 수 없습니다. 서비스 연결 역할을 생성할 권한이 있는지 확인하십시오. 그렇지 않은 경우 기다렸다가 나중에 다시 시도하십시오.
> 다음 권한이 활성화되어 있는지 확인하십시오.
>
> ```
> {
> ```

```
    "Action": "iam:CreateServiceLinkedRole",
    "Effect": "Allow",
    "Resource": "arn:aws:iam::*:role/aws-service-role/custom.rds.amazonaws.com/
AmazonRDSBetaServiceRolePolicy",
    "Condition": {
        "StringLike": {
            "iam:AWSServiceName":"custom.rds.amazonaws.com"
        }
    }
}
```

자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하세요.

Amazon RDS 미리 보기에 대한 서비스 연결 역할

Amazon RDS는 AWSServiceRoleForRDSPreview라는 서비스 연결 역할을 사용하여 Amazon RDS가 RDS DB 리소스를 대신하여 AWS 서비스를 직접 호출할 수 있도록 지원합니다.

AWSServiceRoleForRDSPreview 서비스 연결 역할은 역할을 수임하기 위해 다음 서비스를 신뢰합니다.

- rds.amazonaws.com

이 서비스 연결 역할에는 계정에서 운영할 수 있는 권한을 부여하는 AmazonRDSPreviewServiceRolePolicy라는 권한 정책이 연결되어 있습니다. 자세한 내용은 AWS 관리형 정책: AmazonRDSPreviewServiceRolePolicy 섹션을 참조하세요.

> **Note**
>
> IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 링크 역할을 생성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 다음 오류 메시지가 표시되는 경우:
> 리소스를 만들 수 없습니다. 서비스 연결 역할을 생성할 권한이 있는지 확인하십시오. 그렇지 않은 경우 기다렸다가 나중에 다시 시도하십시오.
> 다음 권한이 활성화되어 있는지 확인하십시오.
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
> ```

```
    "Resource": "arn:aws:iam::*:role/aws-service-role/custom.rds.amazonaws.com/
AmazonRDSPreviewServiceRolePolicy",
    "Condition": {
        "StringLike": {
            "iam:AWSServiceName":"custom.rds.amazonaws.com"
        }
    }
}
```

자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하세요.

Amazon VPC 및 Amazon RDS

Amazon Virtual Private Cloud(Amazon VPC)를 사용하면 Amazon RDS DB 인스턴스와 같은 AWS 리소스를 Virtual Private Cloud(VPC)로 시작할 수 있습니다.

VPC를 사용하면 가상 네트워킹 환경을 완벽하게 제어할 수 있습니다. 자기만의 IP 주소 범위를 선택하고, 서브넷을 생성하고, 라우팅 및 액세스 제어 목록을 구성할 수 있습니다. DB 인스턴스를 VPC에서 실행하는 데는 추가 비용이 들지 않습니다.

계정에는 기본 VPC가 있습니다. 달리 지정하지 않는 한 새 DB 인스턴스는 모두 기본 VPC에서 생성됩니다.

주제

- VPC에서 DB 인스턴스를 사용한 작업
- DB 인스턴스에 대한 VPC 업데이트
- VPC에서 DB 인스턴스에 액세스하는 시나리오
- 자습서: DB 인스턴스에 사용할 Amazon VPC 생성(IPv4 전용)
- 자습서: DB 인스턴스(듀얼 스택 모드)에 사용할 VPC 생성
- VPC에 있지 않은 DB 인스턴스를 VPC로 이동

다음에서는 Amazon RDS DB 인스턴스와 관련된 VPC 기능에 대한 토론을 찾을 수 있습니다. Amazon VPC 대한 자세한 내용은 Amazon VPC 시작 안내서 및 Amazon VPC 사용 설명서를 참조하세요.

VPC에서 DB 인스턴스를 사용한 작업

DB 인스턴스는 Virtual Private Cloud(VPC) 내에 있어야 합니다. VPC 는 AWS 클라우드의 다른 가상 네트워크에서 논리적으로 격리된 가상 네트워크입니다. Amazon VPC를 사용하면 Amazon RDS DB 인스턴스 또는 Amazon EC2 인스턴스와 같은 AWS 리소스를 VPC로 시작할 수 있습니다. VPC는 계정과 함께 제공되는 기본 VPC일 수도 있고, 사용자가 만들 수도 있습니다. 모든 VPC는 AWS 계정과 연결됩니다.

기본 VPC에는 VPC 내의 리소스를 격리하는 데 사용할 수 있는 3개의 서브넷이 있습니다. 또한 VPC 외부에서 VPC 내부의 리소스에 액세스할 수 있도록 하는 데 사용할 수 있는 인터넷 게이트웨이도 있습니다.

VPC 내부와 외부의 Amazon RDS DB 인스턴스를 포함하는 시나리오 목록은 VPC에서 DB 인스턴스에 액세스하는 시나리오 단원을 참조하세요.

주제

- VPC에서 DB 인스턴스를 사용한 작업
- DB 서브넷 그룹을 사용한 작업
- 공유 서브넷
- Amazon RDS IP 주소 지정
- VPC에 있는 DB 인스턴스를 인터넷에서 숨기기
- VPC에 DB 인스턴스 만들기

다음 자습서에서는 일반적인 Amazon RDS 시나리오에 사용할 수 있는 VPC를 생성하는 방법을 배울 수 있습니다.

- 자습서: DB 인스턴스에 사용할 Amazon VPC 생성(IPv4 전용)
- 자습서: DB 인스턴스(듀얼 스택 모드)에 사용할 VPC 생성

VPC에서 DB 인스턴스를 사용한 작업

다음은 VPC에서 DB 인스턴스를 사용하여 작업할 때 유용한 몇 가지 팁입니다.

- VPC는 최소 2개 이상의 서브넷을 보유해야 합니다. 이러한 서브넷은 DB 인스턴스를 배포하고자 하는 AWS 리전에 있는 2개의 다른 가용 영역에 있어야 합니다. 서브넷은 사용자가 지정할 수 있고 보안 및 운영상의 필요를 바탕으로 인스턴스를 그룹화할 수 있게 해주는 VPC IP 주소 범위의 한 부분입니다.

  다중 AZ 배포의 경우, AWS 리전에 있는 두 개 이상의 가용 영역에 대한 서브넷을 정의하면 Amazon RDS가 필요에 따라 다른 가용 영역에 새로운 예비 복제본을 만들 수 있습니다. 단일 AZ 배포의 경우도 어느 시점에 단일 AZ 배포를 다중 AZ 배포로 변환하려면 이 작업을 수행해야 합니다.

  > **Note**
  >
  > 로컬 영역의 DB 서브넷 그룹에는 서브넷이 하나만 있을 수 있습니다.

- VPC에서 DB 인스턴스에 공개적으로 액세스할 수 있도록 하려면 VPC 속성인 DNS 호스트 이름과 DNS 확인을 활성화해야 합니다.
- VPC에는 사용자가 만드는 DB 서브넷 그룹이 있어야 합니다. 생성한 서브넷을 지정하여 DB 서브넷 그룹을 생성합니다. Amazon RDS는 DB 인스턴스와 연결할 서브넷 그룹 내의 서브넷과 IP 주소를 선택합니다. DB 인스턴스는 서브넷이 포함된 가용 영역을 사용합니다.

- VPC에는 DB 인스턴스에 대한 액세스를 허용하는 VPC 보안 그룹이 있어야 합니다.

  자세한 내용은 VPC에서 DB 인스턴스에 액세스하는 시나리오 단원을 참조하세요.
- 각 서브넷의 CIDR 블록은 장애 조치와 컴퓨팅 조정을 포함한 유지 관리 활동 중에 Amazon RDS가 사용할 예비 IP 주소를 수용할 만큼 충분히 커야 합니다. 예를 들어, 일반적으로 10.0.0.0/24 및 10.0.1.0/24와 같은 범위는 충분히 큽니다.
- VPC의 인스턴스 테넌시 속성 값은 기본 또는 전용 중 하나일 수 있습니다. 모든 기본 VPC에서는 인스턴스 테넌시 속성이 기본값으로 설정되어 있으며, 기본 VPC는 어떤 DB 인스턴스 클래스라도 지원할 수 있습니다.

  인스턴스 테넌시 속성이 전용으로 설정된 전용 VPC에 DB 인스턴스를 사용하도록 선택한 경우 DB 인스턴스의 DB 인스턴스 클래스는 승인된 Amazon EC2 전용 인스턴스 유형 중 하나여야 합니다. 예를 들어 r5.large EC2 전용 인스턴스는 db.r5.large DB 인스턴스 클래스에 해당합니다. VPC의 인스턴스 테넌시에 대한 자세한 내용은 Amazon Elastic Compute Cloud 사용 설명서의 전용 인스턴스를 참조하세요.

  전용 인스턴스에 존재할 수 있는 인스턴스 유형에 대한 자세한 내용은 Amazon EC2 요금 페이지에서 Amazon EC2 전용 인스턴스를 참조하세요.

  > **Note**
  >
  > 인스턴스 테넌시 속성을 DB 인스턴스에 대해 전용으로 설정하면 DB 인스턴스가 전용 호스트에서 실행된다는 보장이 없습니다.

- 옵션 그룹이 DB 인스턴스에 할당되면 DB 인스턴스의 VPC와 연결됩니다. 이 연결은 DB 인스턴스를 다른 VPC로 복원하려는 경우 DB 인스턴스에 할당된 옵션 그룹을 사용할 수 없음을 의미합니다.
- DB 인스턴스를 다른 VPC로 복원하는 경우 기본 옵션 그룹을 DB 인스턴스에 할당하거나, 해당 VPC에 연결된 옵션 그룹을 할당하거나, 새 옵션 그룹을 생성하여 DB 인스턴스에 할당해야 합니다. Oracle TDE와 같은 지속적 또는 영구적 옵션의 경우 DB 인스턴스를 다른 VPC로 복구할 때 지속적 또는 영구적 옵션을 포함하는 새 옵션 그룹을 생성해야 합니다.

DB 서브넷 그룹을 사용한 작업

서브넷은 사용자가 보안 및 운영상의 필요를 바탕으로 리소스를 그룹화하기 위해 지정하는 VPC IP 주소 범위의 특정 부분입니다. DB 서브넷 그룹은 사용자가 VPC에서 만든 다음 DB 인스턴스에 대해 지정하는 서브넷(일반적으로 프라이빗)의 모음입니다. DB 서브넷 그룹을 사용하면 AWS CLI 또는 API

를 사용하여 DB 인스턴스를 생성할 때 특정 VPC를 지정할 수 있습니다. 콘솔을 사용하는 경우 사용할 VPC와 서브넷 그룹을 선택할 수 있습니다.

각 DB 서브넷 그룹은 지정된 AWS 리전에서 두 개 이상의 가용 영역에 서브넷이 있어야 합니다. VPC에서 DB 인스턴스를 생성할 때 이에 대한 DB 서브넷 그룹을 선택합니다. DB 서브넷 그룹에서 Amazon RDS는 의 DB 인스턴스와 연결할 서브넷과 해당 서브넷 내의 IP 주소를 선택합니다. DB는 서브넷이 포함된 가용 영역을 사용합니다. Amazon RDS는 항상 사용 가능한 IP 주소 공간이 있는 서브넷에서 IP 주소를 할당합니다.

다중 AZ 배포의 기본 DB 인스턴스에 오류가 있을 경우 Amazon RDS는 해당 스탠바이를 승격한 후 나중에 다른 가용 영역 중 하나에서 서브넷의 IP 주소를 사용하여 새 스탠바이를 만들 수 있습니다.

DB 서브넷 그룹의 서브넷은 퍼블릭 또는 프라이빗입니다. 서브넷은 네트워크 액세스 제어 목록(ACL) 및 라우팅 테이블에 대해 설정한 구성에 따라 퍼블릭 또는 프라이빗입니다. DB 인스턴스에 공개적으로 액세스할 수 있도록 하려면 DB 서브넷 그룹의 모든 서브넷이 퍼블릭이어야 합니다. 공개적으로 액세스할 수 있는 DB 인스턴스와 연결된 서브넷이 퍼블릭에서 프라이빗으로 변경되면 DB 인스턴스 가용성에 영향을 줄 수 있습니다.

듀얼 스택 모드를 지원하는 DB 서브넷 그룹을 만들려면 DB 서브넷 그룹에 추가하는 각 서브넷에 인터넷 프로토콜 버전 6(IPv6) CIDR 블록이 연결되어 있는지 확인합니다. 자세한 내용은 Amazon VPC 사용 설명서의 Amazon RDS IP 주소 지정 및 IPv6로 마이그레이션을 참조하세요.

> **Note**
>
> 로컬 영역의 DB 서브넷 그룹에는 서브넷이 하나만 있을 수 있습니다.

Amazon RDS가 VPC에 DB 인스턴스를 생성할 때는 DB 서브넷 그룹의 IP 주소를 사용하여 DB 인스턴스에 네트워크 인터페이스를 할당합니다. 하지만 기본 IP 주소가 장애 조치 중에 바뀌기 때문에 DB 인스턴스에 연결할 때 반드시 도메인 이름 시스템(DNS) 이름을 사용하는 것이 좋습니다. 장애 조치 중에 기본 IP 주소가 변경되므로 이 방법을 사용하는 것이 좋습니다.

> **Note**
>
> VPC에서 실행하는 각각의 DB 인스턴스에 대해 Amazon RDS가 복구 작업에 사용할 수 있도록 DB 서브넷 그룹의 각 서브넷에 한 개 이상의 주소를 예약해야 합니다.

공유 서브넷

공유 VPC에서 DB 인스턴스를 생성할 수 있습니다.

공유 VPC를 사용할 때 염두에 두어야 할 몇 가지 고려 사항은 다음과 같습니다.

- DB 인스턴스를 공유 VPC 서브넷에서 비공유 VPC 서브넷으로 또는 그 반대로 이동할 수 있습니다.
- 공유 VPC의 참여자는 VPC에 보안 그룹을 만들어야 DB 인스턴스를 생성할 수 있습니다.
- 공유 VPC의 소유자와 참여자는 SQL 쿼리를 사용하여 데이터베이스에 액세스할 수 있습니다. 그러나 리소스 생성자만 리소스에 대한 API 호출을 수행할 수 있습니다.

Amazon RDS IP 주소 지정

IP 주소가 있으면 VPC 내의 리소스가 서로 통신하고, 인터넷을 통해 다른 리소스와 통신할 수 있습니다. Amazon RDS는 IPv4와 IPv6 주소 지정 프로토콜을 모두 지원합니다. 기본적으로 Amazon RDS와 Amazon VPC는 IPv4 주소 지정 프로토콜을 사용합니다. 이 동작은 끌 수 없습니다. VPC를 생성할 때 VPC에 IPv4 CIDR 블록(프라이빗 IPv4 주소)를 지정해야 합니다. IPv6 CIDR 블록을 VPC와 서브넷에 할당하고 그 블록에 속한 IPv6 주소를 서브넷의 DB 인스턴스에 할당할 수도 있습니다.

IPv6 프로토콜을 지원하면 지원되는 IP 주소 수가 늘어납니다. IPv6 프로토콜을 사용하면 향후에 인터넷이 성장할 때를 대비해 사용 가능한 주소를 충분히 확보할 수 있습니다. 신규 및 기존 RDS 리소스는 VPC 내에서 IPv4 및 IPv6 주소를 사용할 수 있습니다. 애플리케이션의 다른 부분에서 사용되는 두 프로토콜 간의 네트워크 트래픽을 구성, 보안 및 변환하면 운영 오버헤드가 발생할 수 있습니다. Amazon RDS 리소스의 IPv6 프로토콜을 표준화하여 네트워크 구성을 간소화할 수 있습니다.

주제

- IPv4 주소
- IPv6 주소
- 듀얼 스택 모드

IPv4 주소

VPC를 생성할 때 VPC의 IPv4 주소 범위를 `10.0.0.0/16`와 같은 CIDR 블록 형태로 지정해야 합니다. DB 서브넷 그룹은 이 CIDR 블록에서 DB 인스턴스가 사용할 수 있는 IP 주소의 범위를 정의합니다. 이 IP 주소는 프라이빗 또는 퍼블릭일 수 있습니다.

프라이빗 IPv4 주소는 인터넷을 통해 연결할 수 없는 IP 주소입니다. 프라이빗 IPv4 주소는 DB인스턴스 및 같은 VPC 내의 Amazon EC2 인스턴스와 같은 기타 리소스 간의 통신을 위해 사용될 수 있습니다. 각 DB 인스턴스에는 VPC 내 통신을 위한 프라이빗 IP 주소가 있습니다.

퍼블릭 IP 주소는 인터넷을 통해 연결할 수 있는 IPv4 주소입니다. 퍼블릭 주소는 DB 인스턴스 및 SQL 클라이언트와 같이 인터넷상 리소스 간의 통신을 위해 사용될 수 있습니다. DB 인스턴스 가 퍼블릭 IP 주소를 수신할지 여부를 제어할 수 있습니다.

일반 Amazon RDS 시나리오에서 사용할 수 있는 IPv4 주소만을 사용하여 VPC를 생성하는 방법을 보여주는 자습서는 자습서: DB 인스턴스에 사용할 Amazon VPC 생성(IPv4 전용) 섹션을 참조하세요.

IPv6 주소

IPv6 CIDR 블록을 VPC와 서브넷에 연결하고 그 블록에 속한 IPv6 주소를 VPC의 리소스에 할당할 수도 있습니다. 각 IPv6 주소는 전역적으로 고유합니다.

VPC에 대한 IPv6 CIDR 블록은 Amazon의 IPv6 주소 풀에서 자동으로 할당되므로 범위를 직접 선택할 수는 없습니다.

IPv6 주소에 연결할 때 다음 조건이 충족되는지 확인하세요.

- IPv6를 통한 클라이언트에서 데이터베이스로의 트래픽이 허용되도록 클라이언트가 구성됩니다.
- DB 인스턴스에서 사용하는 RDS 보안 그룹은 IPv6를 통한 클라이언트에서 데이터베이스로의 트래픽이 허용되도록 올바르게 구성됩니다.
- 클라이언트 운영 체제 스택은 IPv6 주소의 트래픽을 허용하며 운영 체제 드라이버 및 라이브러리는 올바른 기본 DB 인스턴스 엔드포인트(IPv4 또는 IPv6)를 선택하도록 구성됩니다.

IPv6에 대한 자세한 내용은 Amazon VPC 사용 설명서의 IP 주소 지정 섹션을 참조하세요.

듀얼 스택 모드

DB 인스턴스가 IPv4 및 IPv6 주소 지정 프로토콜 모두를 통해 통신할 수 있을 때 듀얼 스택 모드로 실행됩니다. 따라서 리소스는 IPv4, IPv6 또는 모두를 통해 DB 인스턴스와 통신할 수 있습니다. RDS는 프라이빗 듀얼 스택 모드 DB 인스턴스의 IPv6 엔드포인트에 대한 인터넷 게이트웨이 액세스를 비활성화합니다. RDS는 IPv6 엔드포인트가 프라이빗이며 VPC 내에서만 액세스할 수 있도록 하기 위해 이 작업을 수행합니다.

주제

- 듀얼 스택 모드 및 DB 서브넷 그룹

- 듀얼 스택 모드 DB 인스턴스 작업
- 듀얼 스택 모드를 사용하도록 IPv4 전용 DB 인스턴스 수정
- 리전 및 버전 사용 가능 여부
- 듀얼 스택 네트워크 DB 인스턴스에 대한 제한 사항

일반 Amazon RDS 시나리오에서 사용할 수 있는 IPv4와 IPv6 주소 모두를 사용하여 VPC를 생성하는 방법을 보여주는 자습서는 자습서: DB 인스턴스(듀얼 스택 모드)에 사용할 VPC 생성 섹션을 참조하세요.

듀얼 스택 모드 및 DB 서브넷 그룹

듀얼 스택 모드를 사용하려면 DB 인스턴스와 연결하는 DB 서브넷 그룹의 각 서브넷에 IPv6 CIDR 블록이 연결되어 있어야 합니다. 새 DB 서브넷 그룹을 생성하거나 기존 DB 서브넷 그룹을 수정하여 이 요구 사항을 충족하도록 수정할 수 있습니다. DB 인스턴스 가 듀얼 스택 모드가 된 후에는 클라이언트가 정상적으로 연결할 수 있습니다. 클라이언트 보안 방화벽과 RDS DB 인스턴스 보안 그룹이 IPv6을 통한 트래픽을 허용하도록 정확하게 구성되어 있는지 확인합니다. 연결하기 위해 클라이언트는 DB 인스턴스의 엔드포인트를 사용합니다. 클라이언트 애플리케이션은 데이터베이스에 연결할 때 기본 프로토콜을 지정할 수 있습니다. 듀얼 스택 모드에서 DB 인스턴스는 클라이언트의 기본 네트워크 프로토콜(IPv4 또는 IPv6)을 감지하고 연결에 해당 프로토콜을 사용합니다.

서브넷 삭제 또는 CIDR 연결 해제로 인해 DB 서브넷 그룹이 듀얼 스택 모드 지원을 중지하는 경우 DB 서브넷 그룹과 연결된 DB 인스턴스에 대해 네트워크 상태가 호환되지 않을 위험이 있습니다. 또한 새 듀얼 스택 모드 DB 인스턴스를 생성할 때 DB 서브넷 그룹을 사용할 수 없습니다.

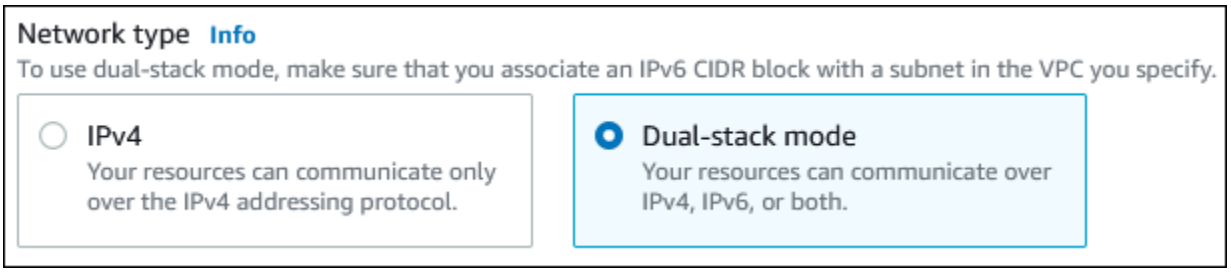
AWS Management Console을 사용하여 DB 서브넷 그룹이 듀얼 스택 모드를 지원하는지 알아보려면 DB 서브넷 그룹의 세부 정보 페이지에서 네트워크 유형(Network type)을 확인합니다. AWS CLI를 사용하여 DB 서브넷 그룹이 듀얼 스택 모드를 지원하는지 알아보려면 describe-db-subnet-groups 명령을 실행하고 출력에서 `SupportedNetworkTypes`를 확인합니다.

읽기 전용 복제본은 독립적인 DB 인스턴스로 취급되며 기본 DB 인스턴스와 다른 네트워크 유형을 가질 수 있습니다. 읽기 전용 복제본의 기본 DB 인스턴스의 네트워크 유형을 변경하는 경우 읽기 전용 복제본은 영향을 받지 않습니다. DB 인스턴스를 복원할 때 지원되는 모든 네트워크 유형으로 복원할 수 있습니다.

듀얼 스택 모드 DB 인스턴스 작업

DB 인스턴스를 생성하거나 수정할 때 리소스가 DB 인스턴스와 IPv4, IPv6 또는 둘 다를 통해 통신할 수 있도록 허용하기 위해 듀얼 스택 모드를 지정할 수 있습니다.

AWS Management Console을 사용하여 DB 인스턴스를 생성하거나 수정할 때 네트워크 유형 섹션에서 듀얼 스택 모드를 지정할 수 있습니다. 다음 이미지는 콘솔의 네트워크 유형 섹션을 보여줍니다.



AWS CLI를 사용하여 DB 인스턴스를 생성하거나 수정할 때 듀얼 스택 모드를 사용하려면 `--network-type` 옵션을 `DUAL`로 설정합니다. 를 사용하여 DB 인스턴스를 생성하거나 수정할 때 듀얼 스택 모드를 사용하려면 `NetworkType` 옵션을 `DUAL`로 설정합니다. DB 인스턴스의 네트워크 유형을 수정하는 경우 가동 중지가 발생할 수 있습니다. 지정된 DB 엔진 버전 또는 DB 서브넷 그룹에서 듀얼 스택 모드가 지원되지 않는 경우 `NetworkTypeNotSupported` 오류가 반환됩니다.

DB 인스턴스를 생성하는 방법에 대한 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요. DB 인스턴스 수정에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하세요.

콘솔을 사용하여 DB 인스턴스가 이중 스택 모드인지 확인하려면 DB 인스턴스의 연결 및 보안 탭에서 네트워크 유형을 확인하세요.

듀얼 스택 모드를 사용하도록 IPv4 전용 DB 인스턴스 수정

듀얼 스택 모드를 사용하도록 IPv4 전용 DB 인스턴스를 수정할 수 있습니다. 그러려면 DB 인스턴스의 네트워크 유형을 변경합니다. 수정으로 인해 가동 중지가 발생할 수 있습니다.

유지 관리 기간 동안 Amazon RDS DB 인스턴스 의 네트워크 유형을 변경하는 것이 좋습니다. 새 인스턴스의 네트워크 유형을 듀얼 스택 모드로 설정하는 것은 현재 지원되지 않습니다. `modify-db-instance` 명령을 사용하여 네트워크 유형을 수동으로 설정할 수 있습니다.

듀얼 스택 모드를 사용하도록 DB 인스턴스를 수정하기 전에 DB 서브넷 그룹이 듀얼 스택 모드를 지원하는지 확인하세요. DB 인스턴스와 연결된 DB 서브넷 그룹이 듀얼 스택 모드를 지원하지 않는 경우 DB 인스턴스를 수정할 때 이를 지원하는 다른 DB 서브넷 그룹을 지정하세요. DB 인스턴스 의 DB 서브넷 그룹을 수정하면 가동 중지가 발생할 수 있습니다.

듀얼 스택 모드를 사용하도록 DB 인스턴스 를 변경하기 전에 DB 인스턴스 의 DB 서브넷 그룹을 수정할 경우, DB 서브넷 그룹이 변경 전후에 DB 인스턴스에 대해 유효한지 확인하세요.

RDS for PostgreSQL, RDS for MySQL, RDS for Oracle, RDS for MariaDB 단일 AZ 인스턴스의 경우 `DUAL`로 설정된 `--network-type` 파라미터만 사용해 modify-db-instance 명령을 실행하여 네트워

크를 듀얼 스택 모드로 변경하는 것이 좋습니다. 동일한 API 호출에서 다른 파라미터를 `--network-type` 파라미터와 함께 추가하면 가동 중지가 발생할 수 있습니다. 여러 파라미터를 수정하려면 다른 파라미터를 사용하여 다른 `modify-db-instance` 요청을 보내기 전에 네트워크 유형 수정을 성공적으로 완료했는지 확인합니다.

RDS for PostgreSQL, RDS for MySQL, RDS for Oracle 및 RDS for MariaDB 다중 AZ DB 인스턴스용의 네트워크 유형을 수정하면 `--network-type` 파라미터만 사용하거나, modify-db-instance 명령 내에 파라미터를 결합할 경우 잠시 가동 중지가 발생하고 장애 조치가 트리거됩니다.

RDS for SQL Server 단일 AZ 또는 다중 AZ DB 인스턴스에서 네트워크 유형을 수정하면 `--network-type` 파라미터만 사용하거나, `modify-db-instance` 명령 내에 파라미터를 결합할 경우 가동 중지가 발생합니다. 네트워크 유형을 수정하면 SQL Server 다중 AZ 인스턴스에서 장애 조치가 발생합니다.

변경 후 DB 인스턴스에 연결할 수 없는 경우 선택한 네트워크(IPv4 또는 IPv6)의 데이터베이스로의 트래픽을 허용하도록 클라이언트 및 데이터베이스 보안 방화벽과 라우팅 테이블이 정확하게 구성되어 있는지 확인하세요. IPv6 주소를 사용하여 연결하려면 운영 체제 파라미터, 라이브러리 또는 드라이버를 수정해야 할 수도 있습니다.

듀얼 스택 모드를 사용하도록 DB 인스턴스를 수정하는 경우, 단일 AZ 배포에서 다중 AZ 배포로 또는 다중 AZ 배포에서 단일 AZ 배포로의 보류 중인 변경이 있어서는 안 됩니다.

듀얼 스택 모드를 사용하도록 IPv4 전용 DB 인스턴스 수정

1. 듀얼 스택 모드를 지원하도록 DB 서브넷 그룹을 수정하거나 듀얼 스택 모드를 지원하는 DB 서브넷 그룹을 생성합니다.

   a. IPv6 CIDR 블록을 VPC와 연결

      자세한 내용은 Amazon VPC 사용 설명서의 VPC에 IPv6 CIDR 블록 추가를 참조하세요.

   b. IPv6 CIDR 블록을 DB 서브넷 그룹의 모든 서브넷에 연결합니다.

      자세한 내용은 Amazon VPC 사용 설명서의 서브넷에 IPv6 CIDR 블록 추가를 참조하세요.

   c. DB 서브넷 그룹이 듀얼 스택 모드를 지원하는지 확인합니다.

      AWS Management Console을 사용 중인 경우 DB 서브넷 그룹을 선택하고 지원되는 네트워크 유형(Supported network types) 값을 듀얼, IPv4(Dual, IPv4)로 설정하세요.

      AWS CLI를 사용 중인 경우 describe-db-subnet-groups 명령을 실행하고 DB 인스턴스의 `SupportedNetworkType` 값을 `Dual, IPv4`로 설정하세요.

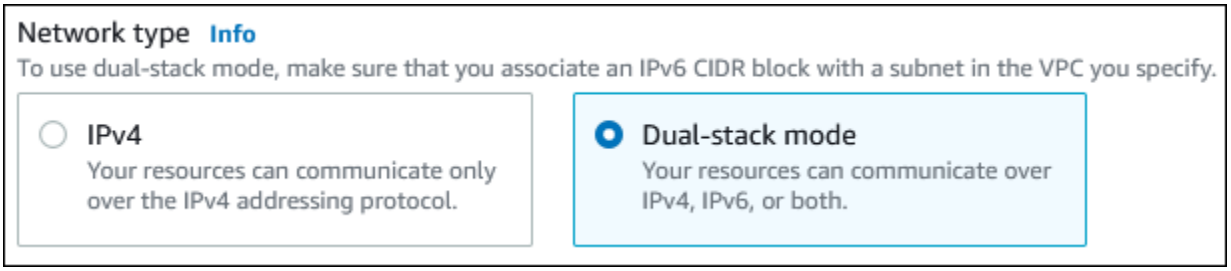
2. 데이터베이스에 대한 IPv6 연결을 허용하도록 DB 인스턴스와 연결된 보안 그룹을 수정하거나 IPv6 연결을 허용하는 새 보안 그룹을 생성합니다.

   자세한 내용은 Amazon VPC 사용 설명서의 보안 그룹 규칙을 참조하세요.

3. 듀얼 스택 모드를 지원하도록 DB 인스턴스를 수정합니다. 이렇게 하려면 네트워크 유형을 듀얼 스택 모드로 설정하세요.

   콘솔을 사용 중인 경우 다음 설정이 올바른지 확인합니다.

   - 네트워크 유형 – 듀얼 스택 모드

   - DB 서브넷 그룹 - 이전 단계에서 구성한 DB 서브넷 그룹
   - 보안 그룹 - 이전 단계에서 구성한 보안

   AWS CLI를 사용 중인 경우 다음 설정이 올바른지 확인합니다.

   - `--network-type` – `dual`
   - `--db-subnet-group-name` - 이전 단계에서 구성한 DB 서브넷 그룹
   - `--vpc-security-group-ids` - 이전 단계에서 구성한 VPC 보안 그룹

   예시:

   ```
   aws rds modify-db-instance --db-instance-identifier my-instance --network-type
    "DUAL"
   ```

4. DB 인스턴스가 듀얼 스택 모드를 지원하는지 확인합니다.

   콘솔을 사용하는 경우 DB 인스턴스에 대한 연결 및 보안 탭을 선택합니다. 해당 탭에서 네트워크 유형 값이 듀얼 스택 모드인지 확인합니다.

   AWS CLI를 사용 중인 경우 describe-db-instances 명령을 실행하고 DB 인스턴스의 `NetworkType` 값이 `dual`인지 확인합니다.

DB 인스턴스 엔드포인트에서 `dig` 명령을 실행하여 연결된 IPv6 주소를 식별합니다.

```
dig db-instance-endpoint AAAA
```

IPv6 주소가 아닌 DB 인스턴스 엔드포인트를 사용하여 DB 인스턴스에 연결합니다.

리전 및 버전 사용 가능 여부

기능 가용성 및 해당 지원은 각 데이터베이스 엔진의 특정 버전 및 AWS 리전 리전에 따라 다릅니다. 듀얼 스택 모드의 버전 및 리전 가용성에 대한 자세한 내용은 Amazon RDS의 이중 스택 모드를 지원하는 리전 및 DB 엔진 섹션을 참조하세요.

듀얼 스택 네트워크 DB 인스턴스에 대한 제한 사항

듀얼 스택 네트워크 DB 인스턴스에는 다음과 같은 제한이 적용됩니다.

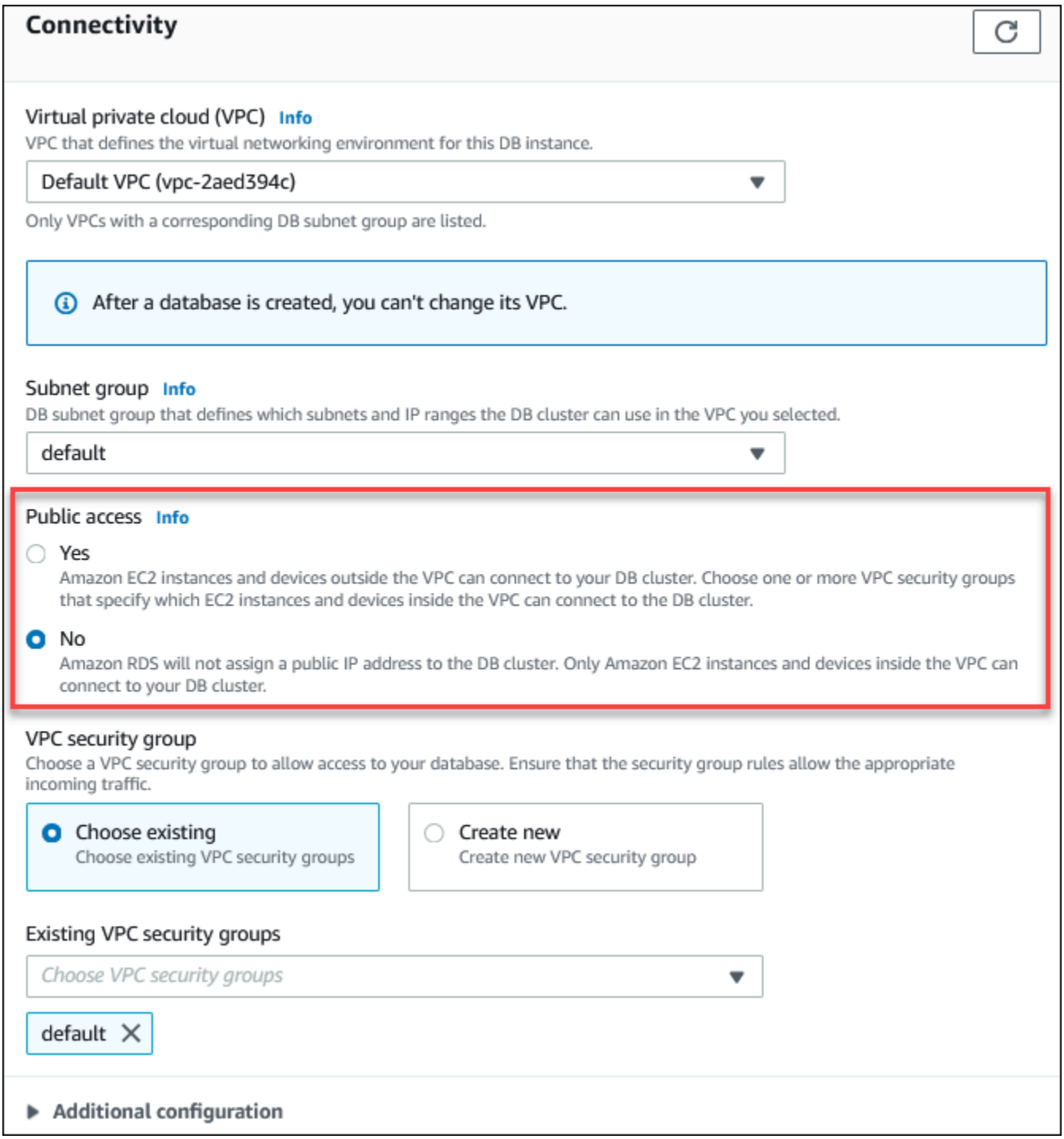
- DB 인스턴스는 IPv6 프로토콜을 단독으로 사용할 수 없습니다. IPv4를 단독으로 사용하거나 IPv4 및 IPv6 프로토콜(듀얼 스택 모드)을 사용할 수 있습니다.
- Amazon RDS에서는 네이티브 IPv6 서브넷을 지원하지 않습니다.
- 듀얼 스택 모드를 사용하는 DB 인스턴스는 프라이빗이어야 합니다. 공개적으로 액세스할 수 없습니다.
- RDS for SQL Server의 경우 Always On AG 가용성 그룹 리스너 엔드포인트를 사용하는 듀얼 스택 모드 DB 인스턴스는 IPv4 주소만 제공합니다.
- 듀얼 스택 모드 DB 인스턴스에는 RDS 프록시를 사용할 수 없습니다.
- AWS Outposts DB 인스턴스에서 RDS와 함께 듀얼 스택 모드를 사용할 수 없습니다.
- 로컬 영역의 DB 인스턴스에서 RDS와 함께 듀얼 스택 모드를 사용할 수 없습니다.

VPC에 있는 DB 인스턴스를 인터넷에서 숨기기

한 가지 일반적인 Amazon RDS 시나리오는 일반에 공개되어 있는 웹 애플리케이션을 포함한 Amazon EC2 인스턴스와 공개적으로 액세스할 수 없는 데이터베이스를 포함한 DB 인스턴스가 있는 VPC를 사용하는 경우입니다. 예를 들어, 퍼블릭 서브넷과 프라이빗 서브넷이 있는 VPC를 생성할 수 있습니다. 웹 서버로 작동하는 EC2 인스턴스는 퍼블릭 서브넷에 배포할 수 있습니다. DB 인스턴스는 프라이빗 서브넷에 배포됩니다. 이런 배포 환경에서는 웹 서버만 DB 인스턴스에 액세스할 수 있습니다. 이 시나리오에 대한 그림은 동일한 VPC에 있는 Amazon EC2 인스턴스가 VPC 내에 있는 DB 인스턴스에 액세스 단원을 참조하세요.

VPC 내에서 DB 인스턴스를 시작하면 DB 인스턴스는 VPC 내부 트래픽에 대한 프라이빗 IP 주소를 가집니다. 이 프라이빗 IP 주소는 공개적으로 액세스할 수 없습니다. 퍼블릭 액세스 옵션을 사용하여 DB 인스턴스에 프라이빗 IP 주소 외에 퍼블릭 IP 주소가 있는지 여부를 지정할 수 있습니다. DB 인스턴스가 공개적으로 액세스 가능한 것으로 지정된 경우 해당 DNS 엔드포인트는 VPC 내에서 프라이빗 IP 주소로 확인됩니다. VPC 외부에서 퍼블릭 IP 주소로 확인됩니다. DB 인스턴스에 대한 액세스는 궁극적으로 사용하는 보안 그룹에 의해 제어됩니다. DB 인스턴스에 할당된 보안 그룹에 이를 허용하는 인바운드 규칙이 포함되어 있지 않으면 해당 퍼블릭 액세스가 허용되지 않습니다. 또한 DB 인스턴스에 공개적으로 액세스할 수 있으려면 해당 DB 서브넷 그룹의 서브넷에 인터넷 게이트웨이가 있어야 합니다. 자세한 내용은 Amazon RDS DB 인스턴스에 연결할 수 없음 섹션을 참조하세요.

퍼블릭 액세스 옵션을 수정하여 퍼블릭 액세스를 켜거나 끄도록 DB 인스턴스를 수정할 수 있습니다. 다음 그림은 추가 연결 구성 섹션의 퍼블릭 액세스 옵션을 보여줍니다. 옵션을 설정하려면 연결(Connectivity) 섹션에서 추가 연결 구성(Additional connectivity configuration) 섹션을 엽니다.



퍼블릭 액세스(Public access) 옵션을 설정하기 위해 DB 인스턴스를 수정하는 방법에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하세요.

VPC에 DB 인스턴스 만들기

다음 절차에 따라 VPC에서 DB 인스턴스를 생성할 수 있습니다. 기본 VPC를 사용하려면 2단계부터 시작하고 이미 생성된 VPC 및 DB 서브넷 그룹을 사용할 수 있습니다. 추가 VPC를 만들려면 새 VPC를 만들 수 있습니다.

> **Note**
>
> VPC의 DB 인스턴스에 공개적으로 액세스할 수 있도록 하려면 VPC 속성 DNS 호스트 이름 및 DNS 확인을 활성화하여 VPC에 대한 DNS 정보를 업데이트해야 합니다. VPC 인스턴스의 DNS 정보 업데이트에 대한 자세한 내용은 VPC에 대한 DNS 지원 업데이트를 참조하세요.

다음 단계에 따라 VPC에서 DB 인스턴스를 만들 수 있습니다:

- 1단계: VPC 생성
- 2단계: DB 서브넷 그룹 만들기
- 3단계: VPC 보안 그룹 만들기
- 4단계: VPC에서 DB 인스턴스 만들기

1단계: VPC 생성

최소 2개의 가용 영역에 서브넷이 있는 VPC를 생성합니다. DB 서브넷 그룹을 만들 때 이들 서브넷이 사용됩니다. 기본 VPC가 있는 경우에는 해당 AWS 리전의 각 가용 영역에 서브넷이 자동으로 생성됩니다.

자세한 내용은 프라이빗 서브넷과 퍼블릭 서브넷을 포함하는 VPC 생성 단원을 참조하거나 Amazon VPC 사용 설명서의 VPC 생성을 참조하세요.

2단계: DB 서브넷 그룹 만들기

DB 서브넷 그룹은 사용자가 VPC에 대해 만든 다음 DB 인스턴스에 대해 지정하는 서브넷(일반적으로 프라이빗)의 모음입니다. DB 서브넷 그룹을 사용하면 AWS CLI 또는 RDS API를 사용하여 DB 인스턴스를 생성할 때 특정 VPC를 지정할 수 있습니다. 콘솔을 사용하는 경우 사용할 VPC와 서브넷만 선택할 수 있습니다. 각 DB 서브넷 그룹은 해당 AWS 리전에서 두 개 이상의 가용 영역에 한 개 이상의 서브넷이 있어야 합니다. 모범 사례로서, 각 DB 서브넷 그룹에는 AWS 리전의 가용 영역마다 하나 이상의 서브넷이 있어야 합니다.

다중 AZ 배포의 경우, AWS 리전에 있는 모든 가용 영역에 대한 서브넷을 정의하면 Amazon RDS가 필요한 경우 다른 가용 영역에 새로운 예비 복제본을 만들 수 있습니다. 나중에 다중 AZ 배포로 변환할 수 있으므로, 단일 AZ 배포의 경우에도 이 모범 사례를 따를 수 있습니다.

DB 인스턴스에 공개적으로 액세스할 수 있도록 하려면 DB 서브넷 그룹의 서브넷에 인터넷 게이트웨이가 있어야 합니다. 서브넷용 인터넷 게이트웨이에 대한 자세한 내용은 Amazon VPC 사용 설명서의 인터넷 게이트웨이를 사용하여 인터넷에 연결을 참조하세요.

> **Note**
>
> 로컬 영역의 DB 서브넷 그룹에는 서브넷이 하나만 있을 수 있습니다.

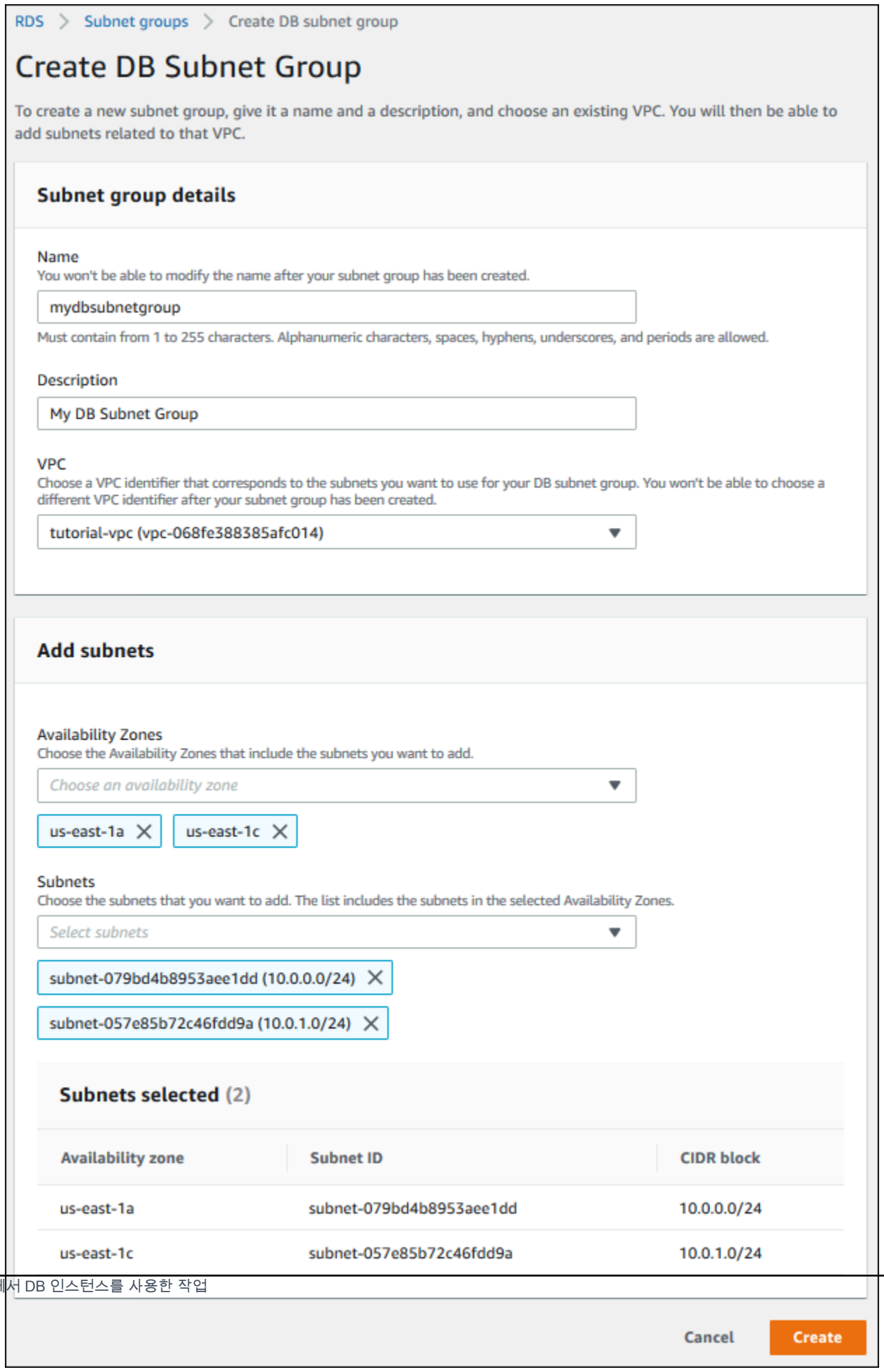
VPC에서 DB 인스턴스를 생성할 때 이에 대한 DB 서브넷 그룹을 선택합니다. Amazon RDSA는 DB 인스턴스와 연결할 서브넷 내의 서브넷과 IP 주소를 선택합니다. DB 서브넷 그룹이 없는 경우 Amazon RDS는 DB 인스턴스를 생성할 때 기본 서브넷 그룹을 생성합니다. Amazon RDS에서 탄력적 네트워크 인터페이스를 생성하고 해당 IP 주소로 DB 인스턴스에 연결합니다. DB 인스턴스는 서브넷이 포함된 가용 영역을 사용합니다.

다중 AZ 배포의 경우, AWS 리전에 있는 두 개 이상의 가용 영역에 대한 서브넷을 정의하면 Amazon RDS가 필요한 경우 다른 가용 영역에 새로운 예비 복제본을 만들 수 있습니다. 단일 AZ 배포의 경우도 어느 시점에 단일 AZ 배포를 다중 AZ 배포로 변환하려면 이 작업을 수행해야 합니다.

이 단계에서는 DB 서브넷 그룹을 만들고 VPC용으로 만든 서브넷을 추가합니다.

DB 서브넷 그룹을 만드는 방법

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 [Subnet groups]를 선택합니다.
3. [Create DB Subnet Group]을 선택합니다.
4. [Name]에 DB 서브넷 그룹의 이름을 입력합니다.
5. 설명에 DB 서브넷 그룹에 대한 설명을 입력합니다.
6. VPC에서 기본 VPC 또는 생성한 VPC를 선택합니다.
7. 서브넷 추가 섹션의 가용 영역에서 해당 서브넷을 포함하는 가용 영역을 선택한 다음 서브넷에서 서브넷을 선택합니다.


RDS > Subnet groups > Create DB subnet group
Create DB Subnet Group
To create a new subnet group, give it a name and a description, and choose an existing VPC. You will then be able to add subnets related to that VPC.
Subnet group details
Name
You won't be able to modify the name after your subnet group has been created.
mydbsubnetgroup
Must contain from 1 to 255 characters. Alphanumeric characters, spaces, hyphens, underscores, and periods are allowed.
Description
My DB Subnet Group
VPC
Choose a VPC identifier that corresponds to the subnets you want to use for your DB subnet group. You won't be able to choose a different VPC identifier after your subnet group has been created.
tutorial-vpc (vpc-068fe388385afc014)
Add subnets
Availability Zones
Choose the Availability Zones that include the subnets you want to add.
Choose an availability zone
us-east-1a
us-east-1c
Subnets
Choose the subnets that you want to add. The list includes the subnets in the selected Availability Zones.
Select subnets
subnet-079bd4b8953aee1dd (10.0.0.0/24)
subnet-057e85b72c46fdd9a (10.0.1.0/24)
Subnets selected (2)
Availability zone
Subnet ID
CIDR block
us-east-1a
subnet-079bd4b8953aee1dd
10.0.0.0/24
us-east-1c
subnet-057e85b72c46fdd9a
10.0.1.0/24
Cancel
Create

> **Note**
>
> 로컬 영역을 활성화한 경우 Create DB subnet group(DB 서브넷 그룹 생성) 페이지에서 가용 영역 그룹을 선택할 수 있습니다. 이 경우 가용 영역 그룹, 가용 영역 및 서브넷을 선택합니다.

8. 생성(Create)을 선택합니다.

   새 DB 서브넷 그룹은 RDS 콘솔의 DB 서브넷 그룹 목록에 나타납니다. DB 서브넷 그룹을 선택하면 창 하단에 있는 세부 정보 창에서 그룹과 연결된 모든 서브넷을 포함한 세부 정보를 확인할 수 있습니다.

3단계: VPC 보안 그룹 만들기

DB 인스턴스를 만들기 전에 DB 인스턴스와 연결할 VPC 보안 그룹을 만들어야 합니다. VPC 보안 그룹을 생성하지 않는 경우 DB 인스턴스를 생성할 때 기본 보안 그룹을 사용할 수 있습니다. DB 인스턴스에 대한 보안 그룹을 생성하는 방법에 대한 지침은 프라이빗 DB 인스턴스에 대한 VPC 보안 그룹 생성 단원을 참조하거나 Amazon VPC 사용 설명서의 보안 그룹을 사용하여 리소스에 대한 트래픽 제어를 참조하세요.

4단계: VPC에서 DB 인스턴스 만들기

이 단계에서는 DB 인스턴스를 만들고 이전 단계에서 만든 VPC 이름, DB 서브넷 그룹 및 VPC 보안 그룹을 사용합니다.

> **Note**
>
> VPC의 DB 인스턴스에 공개적으로 액세스할 수 있도록 하려면 VPC 속성 DNS 호스트 이름 및 DNS 확인을 활성화해야 합니다. 자세한 내용을 알아보려면 Amazon VPC 사용 설명서의 VPC에 대한 DNS 속성을 참조하세요.

DB 인스턴스 생성 방법에 대한 자세한 내용은 Amazon RDS DB 인스턴스 생성 단원을 참조하세요.

연결 섹션에 메시지가 표시되면 VPC 이름, DB 서브넷 그룹 및 VPC 보안 그룹을 입력합니다.

DB 인스턴스에 대한 VPC 업데이트

AWS Management Console을 사용하여 DB 인스턴스를 다른 VPC로 이동할 수 있습니다.

DB 인스턴스 수정에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하세요. 수정 페이지의 연결 섹션에서 다음과 같이 DB 서브넷 그룹에 새 DB 서브넷 그룹을 입력합니다. 새 서브넷 그룹은 새 VPC에 있는 서브넷 그룹이어야 합니다.



다음 조건이 적용되는 경우 DB 인스턴스의 VPC를 변경할 수 없습니다.

- DB 인스턴스는 여러 가용 영역에 있습니다. DB 인스턴스를 단일 가용 영역으로 변환하고, 새 VPC로 이동한 다음, 다중 AZ DB 인스턴스로 다시 변환할 수 있습니다. 자세한 내용은 Amazon RDS에 대한 다중 AZ 배포 구성 및 관리 단원을 참조하세요.
- DB 인스턴스에는 하나 이상의 읽기 전용 복제본이 있습니다. 읽기 전용 복제본을 제거하고, DB 인스턴스를 새 VPC로 이동한 다음, 읽기 전용 복제본을 다시 추가할 수 있습니다. 자세한 내용은 DB 인스턴스 읽기 전용 복제본 작업 단원을 참조하세요.
- DB 인스턴스는 읽기 전용 복제본입니다. 읽기 전용 복제본을 승격한 다음 독립 실행형 DB 인스턴스를 새 VPC로 이동할 수 있습니다. 자세한 내용은 읽기 전용 복제본을 독립 DB 인스턴스로 승격 단원을 참조하세요.
- 대상 VPC의 서브넷 그룹은 DB 인스턴스의 가용 영역에 서브넷이 없습니다. DB 인스턴스의 가용 영역에 있는 서브넷을 DB 서브넷 그룹에 추가한 다음 DB 인스턴스를 새 VPC로 이동할 수 있습니다. 자세한 내용은 DB 서브넷 그룹을 사용한 작업 단원을 참조하세요.

VPC에서 DB 인스턴스에 액세스하는 시나리오

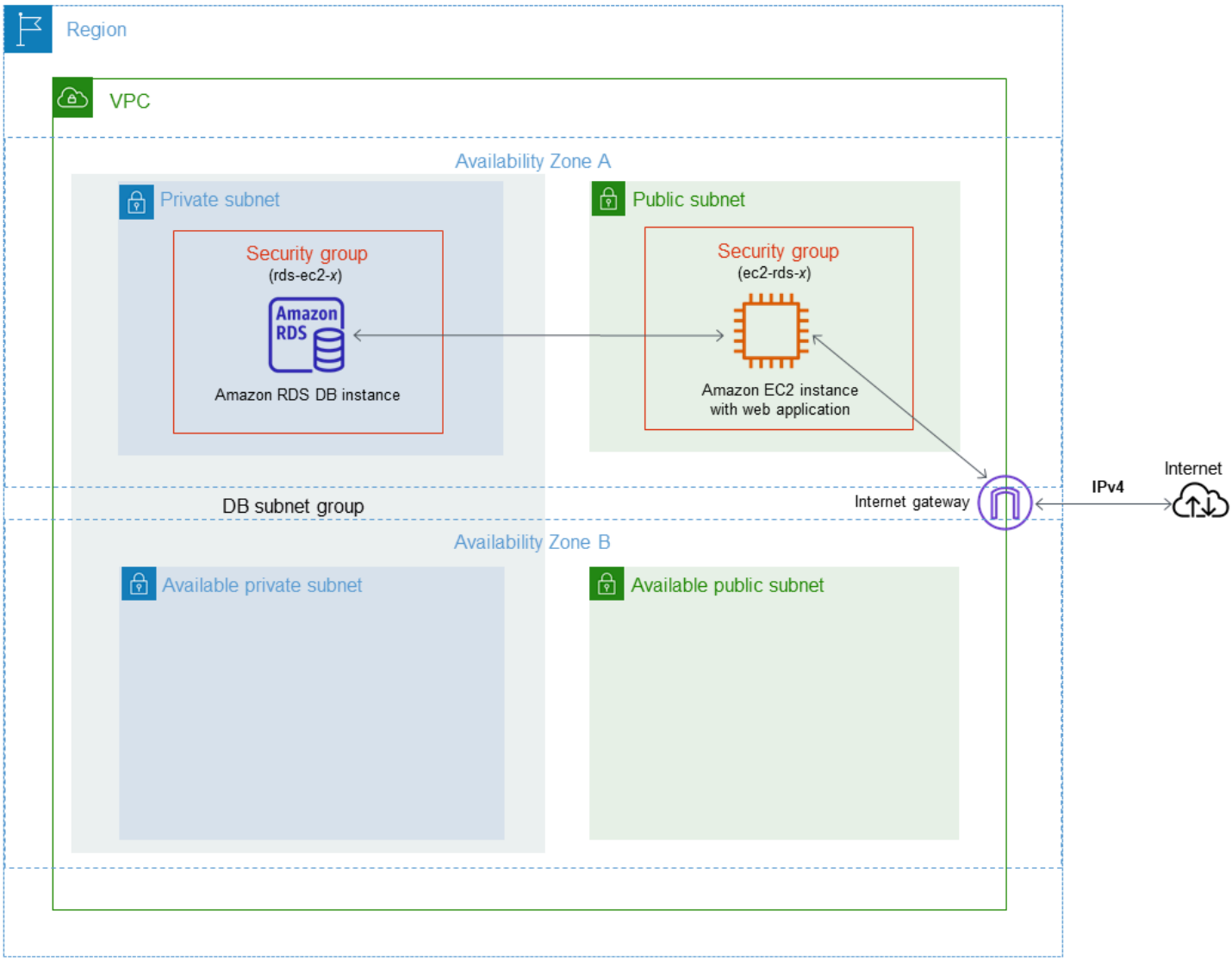
Amazon RDS에서는 VPC의 DB 인스턴스에 액세스하는 다음 시나리오를 지원합니다.

- 동일한 VPC에 있는 Amazon EC2 인스턴스
- 다른 VPC에 있는 EC2 인스턴스
- 클라이언트 애플리케이션이 인터넷을 통해
- 프라이빗 네트워크

동일한 VPC에 있는 Amazon EC2 인스턴스가 VPC 내에 있는 DB 인스턴스에 액세스

VPC에서 DB 인스턴스를 사용하는 일반적인 사례는 동일한 VPC의 Amazon EC2 인스턴스에서 실행 중인 애플리케이션 서버와 데이터를 공유하는 것입니다.

다음 다이어그램은 이 시나리오를 보여 줍니다.



동일한 VPC에서 EC2 인스턴스와 DB 인스턴스 간 액세스를 관리하는 가장 간단한 방법은 다음과 같습니다.

- DB 인스턴스가 포함될 VPC 보안 그룹을 생성합니다. 이 보안 그룹을 사용해 DB 인스턴스에 대한 액세스를 제한할 수 있습니다. 예를 들어 이 보안 그룹에 대한 사용자 지정 규칙을 생성할 수 있습니다. 이렇게 하면 DB 인스턴스를 생성할 때 할당한 포트와 개발 또는 기타 목적으로 DB 인스턴스에 액세스하는 데 사용하는 IP 주소를 사용하여 TCP 액세스를 허용할 수 있습니다.

- EC2 인스턴스(웹 서버 및 클라이언트)가 포함될 VPC 보안 그룹을 생성합니다. 이 보안 그룹은 필요할 경우 VPC의 라우팅 테이블을 사용해 인터넷에서 EC2 인스턴스 액세스를 허용할 수 있습니다. 예를 들어, 이 보안 그룹에서 TCP가 포트 22를 통해 EC2 인스턴스에 액세스하도록 허용하는 규칙을 설정할 수 있습니다.
- DB 인스턴스에 대한 보안 그룹에서 EC2 인스턴스에 대해 생성한 보안 그룹으로부터의 연결을 허용하는 사용자 지정 규칙을 만듭니다. 이러한 규칙을 통해 보안 그룹의 모든 구성원은 DB 인스턴스에 액세스할 수 있게 됩니다.

별도의 가용 영역에서 구성된 추가 퍼블릭 및 프라이빗 서브넷이 있습니다. RDS DB 서브넷 그룹은 최소 2개의 가용 영역에 한 개 이상의 서브넷이 필요합니다. 추가 서브넷을 사용하면 향후 다중 AZ DB 인스턴스 배포로 쉽게 전환할 수 있습니다.

이 시나리오에 대해 퍼블릭 서브넷과 프라이빗 서브넷 모두에서 VPC를 생성하는 방법을 보여주는 자습서는 자습서: DB 인스턴스에 사용할 Amazon VPC 생성(IPv4 전용) 단원을 참조하세요.

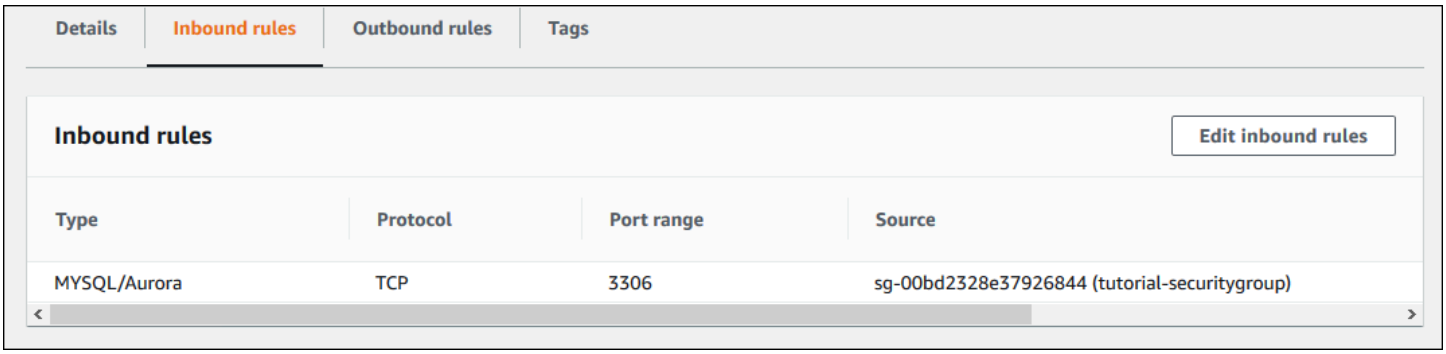
> **Tip**
>
> DB 인스턴스를 생성할 때 Amazon EC2 인스턴스와 DB 인스턴스 간의 네트워크 연결을 자동으로 설정할 수 있습니다. 자세한 내용은 EC2 인스턴스와의 자동 네트워크 연결 구성 단원을 참조하세요.

VPC 보안 그룹에서 다른 보안 그룹으로부터의 연결을 허용하는 규칙을 만들려면 다음을 수행합니다.

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/vpc에서 Amazon VPC 콘솔을 엽니다.
2. 탐색 창에서 Security groups를 선택합니다.
3. 다른 보안 그룹의 구성원에 대한 액세스를 허용할 보안 그룹을 선택하거나 생성합니다. 위 시나리오에서 이 보안 그룹을 DB 인스턴스에 사용합니다. 인바운드 규칙 탭을 선택한 후 인바운드 규칙 편집을 선택합니다.
4. 인바운드 규칙 편집 페이지에서 규칙 추가]를 선택합니다.
5. 유형에서 MYSQL/Aurora와 같이 DB 인스턴스를 생성할 때 사용한 포트에 해당하는 항목을 선택합니다.
6. 소스 상자에 일치하는 보안 그룹을 나열하는 보안 그룹의 ID를 입력합니다. 이 보안 그룹에서 보호 중인 리소스에 대한 액세스를 허용할 구성원이 포함된 보안 그룹을 선택합니다. 위 시나리오에서 이 보안 그룹을 EC2 인스턴스에 사용합니다.

7. 필요한 경우 유형으로 모든 TCP를 지정하고 소스 상자에 보안 그룹을 입력한 규칙을 생성하여 TCP 프로토콜에 대해 단계를 반복합니다. UDP 프로토콜을 사용하려는 경우 유형이 모든 UDP이고 소스에 보안 그룹이 있는 규칙을 생성합니다.
8. 규칙 저장을 선택합니다.

다음 화면은 소스에 대한 보안 그룹이 포함된 인바운드 규칙을 보여 줍니다.

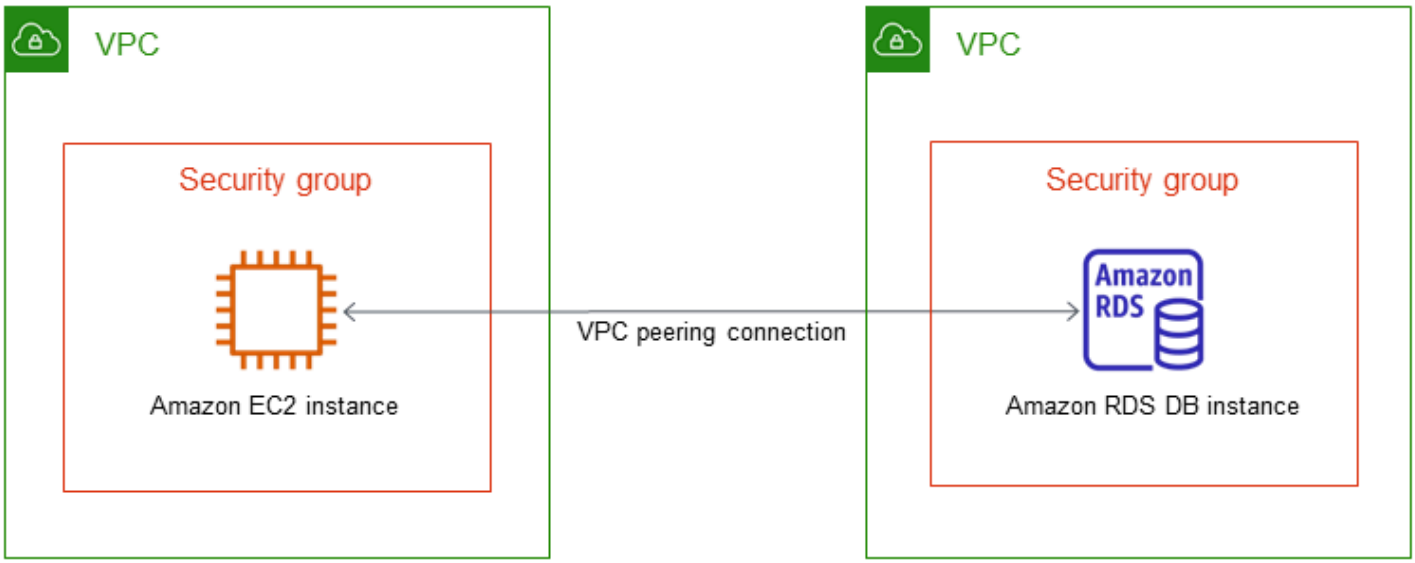


EC2 인스턴스에서 DB 인스턴스에 연결하는 방법에 대한 자세한 내용은 Amazon RDS DB 인스턴스에 연결 단원을 참조하세요.

VPC에 있는 DB 인스턴스에 다른 VPC에 있는 EC2 인스턴스가 액세스

DB 인스턴스가 액세스 시 사용할 EC2 인스턴스와 다른 VPC에 있는 경우 DB 인스턴스에 액세스하기 위해 VPC 피어링을 사용할 수 있습니다.

다음 다이어그램은 이 시나리오를 보여 줍니다.

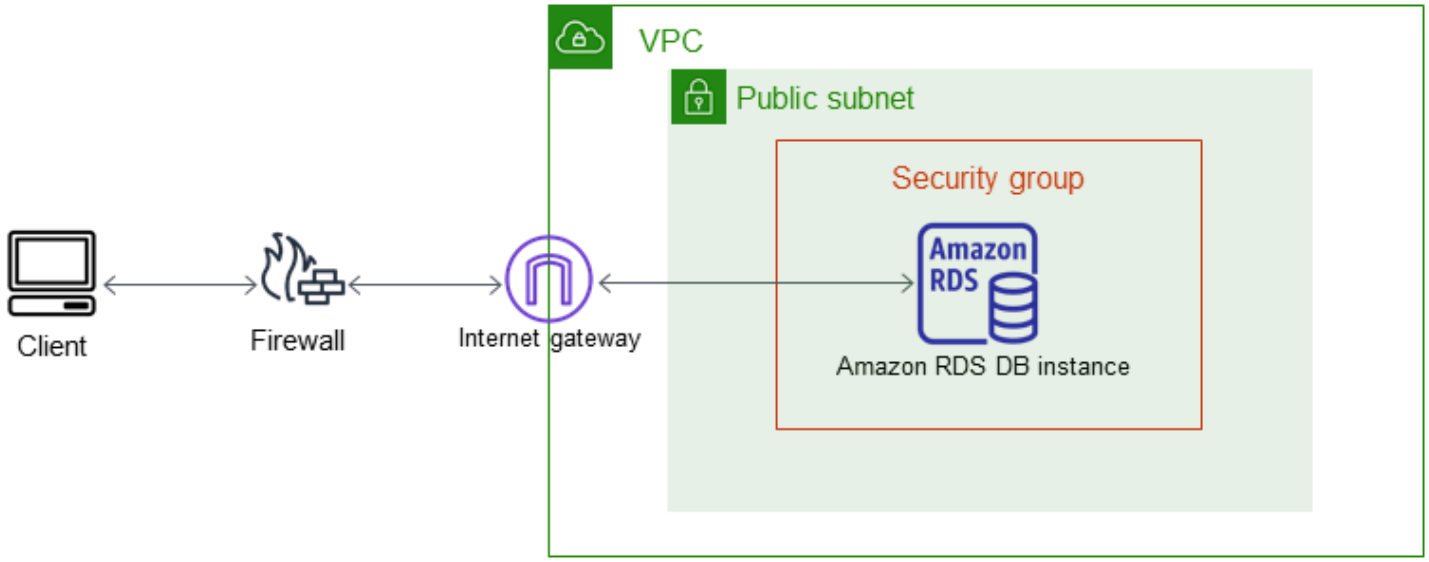


VPC 피어링 연결은 프라이빗 IP 주소를 사용하여 두 VPC 간에 트래픽을 라우팅할 수 있도록 하기 위한 두 VPC 사이의 네트워킹 연결입니다. 동일한 네트워크에 속하는 경우와 같이 VPC의 리소스가 서로 통신할 수 있습니다. 자체 VPC 간, 다른 AWS 계정에서 VPC를 사용하여 또는 다른 AWS 리전에서 VPC를 사용하여 VPC 피어링 연결을 만들 수 있습니다. VPC 피어링에 대한 자세한 내용은 Amazon Virtual Private Cloud 사용 설명서의 VPC 피어링을 참조하세요.

클라이언트 애플리케이션이 인터넷을 통해 VPC에 있는 DB 인스턴스에 액세스

인터넷을 통해 클라이언트 애플리케이션에서 VPC에 있는 DB 인스턴스에 액세스하려면, 단일 퍼블릭 서브넷이 있는 VPC와 인터넷을 통한 통신을 지원하는 인터넷 게이트웨이를 구성합니다.

다음 다이어그램은 이 시나리오를 보여 줍니다.



다음 구성을 권장합니다.

- 크기가 /16인 VPC(예: CIDR: 10.0.0.0/16). 이 크기는 65,536개의 프라이빗 IP 주소를 제공합니다.
- 크기가 /24인 서브넷(예: CIDR: 10.0.0.0/24). 이 크기는 256개의 프라이빗 IP 주소를 제공합니다.
- Amazon RDS DB 인스턴스는 VPC 및 서브넷과 연결됩니다. Amazon RDS는 서브넷 내의 IP 주소를 DB 인스턴스에 할당합니다.
- VPC를 인터넷 및 다른 AWS 제품과 연결하는 인터넷 게이트웨이.
- DB 인스턴스와 연결된 보안 그룹입니다. 보안 그룹의 인바운드 규칙은 클라이언트 애플리케이션이 사용자의 DB 인스턴스에 액세스할 수 있도록 해줍니다.

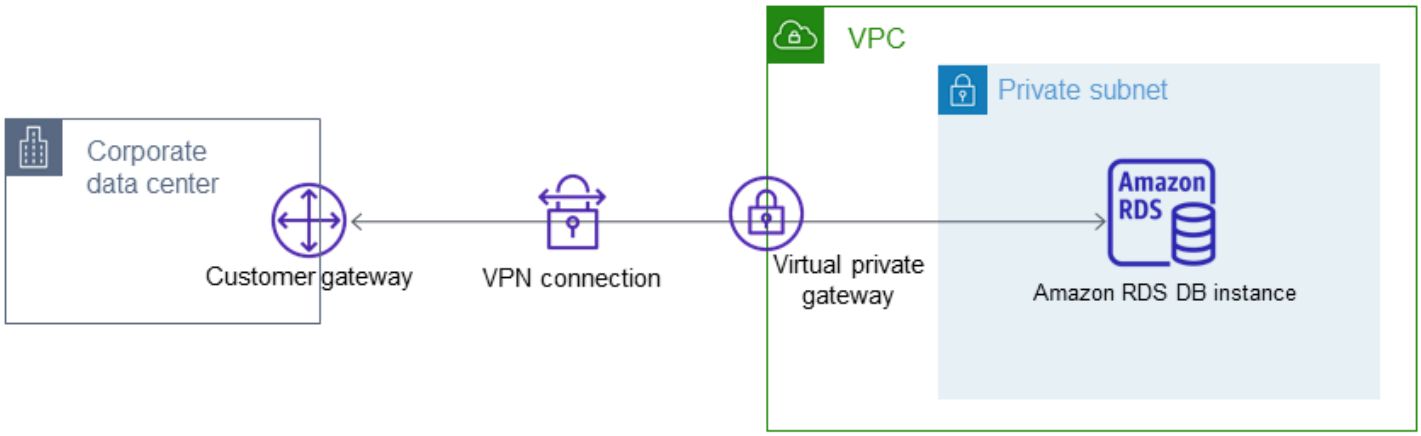
VPC에서 DB 인스턴스 생성에 대한 자세한 내용은 VPC에 DB 인스턴스 만들기 단원을 참조하세요.

프라이빗 네트워크에서 액세스하는 VPC의 DB 인스턴스

DB 인스턴스에 공개적으로 액세스할 수 없는 경우 프라이빗 네트워크에서 액세스할 수 있는 옵션은 다음과 같습니다.

- AWS Site-to-Site VPN 연결. 자세한 내용은 AWS Site-to-Site VPN란 무엇입니까?를 참조하십시오.
- AWS Direct Connect 연결. 자세한 내용은 AWS Direct Connect란 무엇인가요?를 참조하세요.
- AWS Client VPN 연결. 자세한 내용은 AWS Client VPN란 무엇입니까?를 참조하세요.

다음 다이어그램은 AWS Site-to-Site VPN 연결 시나리오를 보여줍니다.

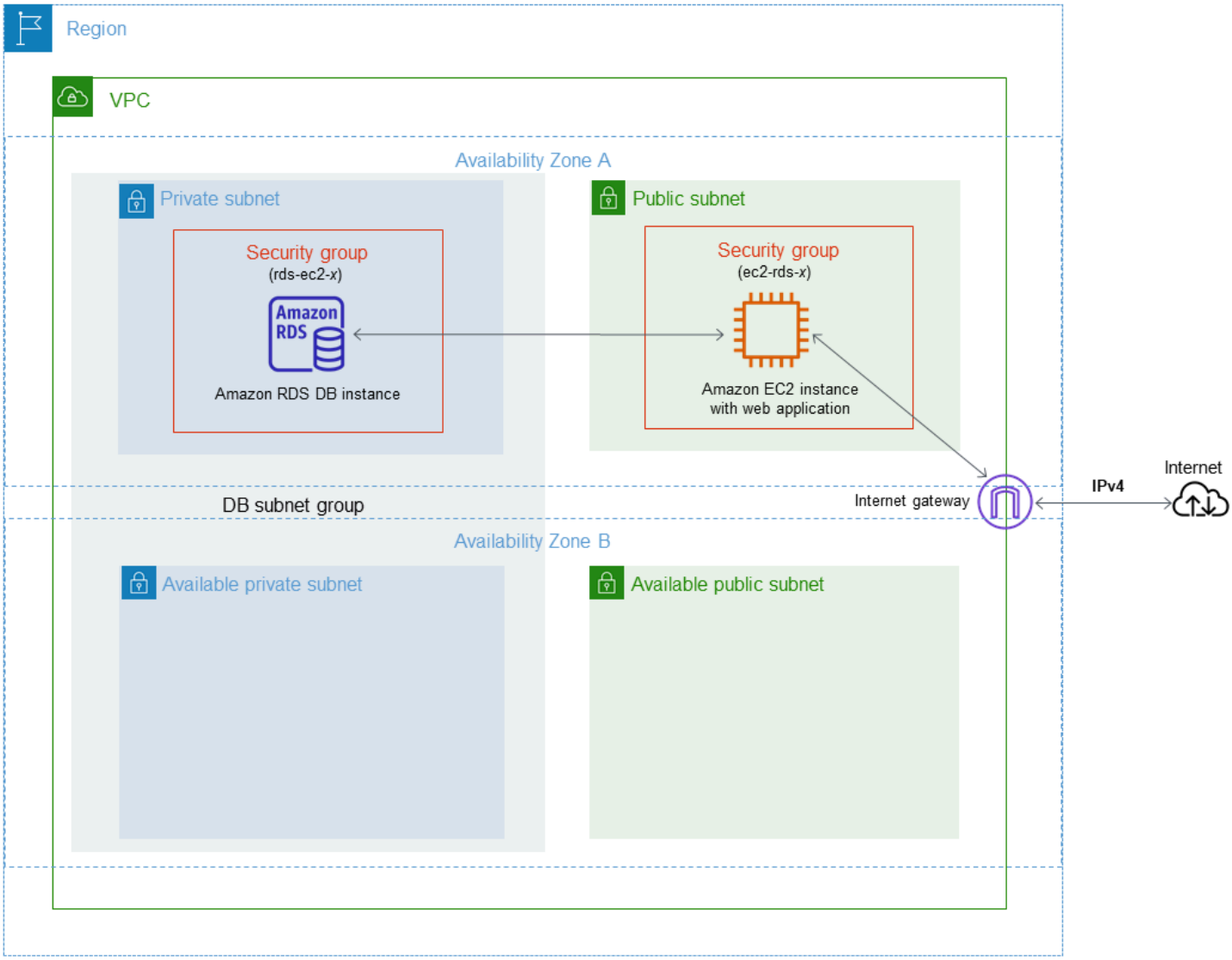


자세한 내용은 인터네트워크 트래픽 개인 정보 단원을 참조하십시오.

자습서: DB 인스턴스에 사용할 Amazon VPC 생성(IPv4 전용)

일반적인 시나리오에는 Amazon VPC 서비스를 기반으로 하는 Virtual Private Cloud(VPC)의 DB 인스턴스가 포함됩니다. 이 VPC는 동일한 VPC에서 실행 중인 웹 서버와 데이터를 공유합니다. 이 자습에서는 이 시나리오를 위한 VPC를 생성합니다.

다음 다이어그램은 이 시나리오를 보여 줍니다. 다른 시나리오에 대한 자세한 내용은 VPC에서 DB 인스턴스에 액세스하는 시나리오을(를) 참조하십시오.



퍼블릭 인터넷이 아닌 웹 서버에서만 DB 인스턴스를 사용할 수 있어야 합니다. 따라서 퍼블릭 서브넷과 프라이빗 서브넷을 모두 포함하여 VPC를 생성합니다. 퍼블릭 서브넷에서 웹 서버를 호스팅하므로 웹 서버에서 퍼블릭 인터넷에 액세스할 수 있습니다. DB 인스턴스는 프라이빗 서브넷에서 호스팅됩니다. Amazon EC2 인스턴스는 동일한 VPC 내에서 호스팅되므로 DB 인스턴스에 연결할 수 있습니다. 하지만 퍼블릭 인터넷에서는 DB 인스턴스를 사용할 수 없으므로 보안이 강화됩니다.

이 자습서에서는 별도의 가용 영역에서 추가 퍼블릭 및 프라이빗 서브넷을 구성합니다. 이러한 서브넷은 자습서에서 사용하지 않습니다. RDS DB 서브넷 그룹은 최소 2개의 가용 영역에 한 개 이상의 서브넷이 필요합니다. 추가 서브넷이 있으면 future 다중 AZ DB 인스턴스 배포로 쉽게 전환할 수 있습니다.

이 자습서에서는 Amazon RDS DB 인스턴스용 VPC 구성에 대해 설명합니다. 이 VPC 시나리오에 대한 웹 서버를 생성하는 방법을 보여 주는 자습서는 자습서: 웹 서버 및 Amazon RDS DB 인스턴스 생성 단원을 참조하세요. Amazon VPC 대한 자세한 내용은 Amazon VPC 시작 안내서 및 Amazon VPC 사용 설명서를 참조하세요.

> **Tip**
>
> Amazon EC2 인스턴스와 DB 간에 네트워크 연결을 설정할 수 있습니다.예DB를 생성할 때 자동으로예. 네트워크 구성은 이 자습서에서 설명한 것과 비슷합니다. 자세한 내용은 EC2 인스턴스와의 자동 네트워크 연결 구성. 섹션을 참조하세요.

프라이빗 서브넷과 퍼블릭 서브넷을 포함하는 VPC 생성

다음은 퍼블릭 서브넷과 프라이빗 서브넷을 모두 포함하는 VPC를 생성하는 절차입니다.

VPC 및 서브넷을 생성하는 방법

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. AWS Management Console의 오른쪽 상단에서 VPC를 생성할 리전을 선택합니다. 이 예에서는 미국 서부(오레곤) 리전을 사용합니다.
3. 왼쪽 상단 모서리에서 VPC 대시보드를 선택합니다. VPC 생성을 시작하려면 VPC 생성을 선택합니다.
4. VPC 설정의 생성할 리소스에서 VPC 등을 선택합니다.
5. VPC 설정을 위해 다음 값을 설정합니다.
   - 네임 태그 자동 생성 - **`tutorial`**
   - IPv4 CIDR block: - **`10.0.0.0/16`**
   - IPv6 CIDR 블록 - No IPv6 CIDR Block(IPv6 CIDR 블록 없음)
   - 테넌시 - 기본값
   - 가용 영역(AZ)의 수 - 2
   - Customize AZs(AZ 사용자 지정) - 기본값을 유지합니다.

- 퍼블릭 서브넷 수 – 2
- 프라이빗 서브넷 수 – 2
- Customize subnets CIDR blocks(서브넷 CIDR 블록 사용자 지정) - 기본값을 유지합니다.
- NAT 게이트웨이($) – 없음
- VPC 엔드포인트 – 없음
- DNS options(DNS 옵션) - 기본값을 유지합니다.

> **Note**
>
> Amazon RDS에서는 다중 AZ DB 인스턴스 배포를 지원하려면 서로 다른 두 가용 영역에 있는 Amazon RDS가 필요합니다. 이 자습서에서는 단일 AZ 배포를 생성하지만 요구 사항에 따라 향후 다중 AZ DB 인스턴스 배포로 더욱 쉽게 전환할 수 있습니다.

6. VPC 생성을 선택합니다.

퍼블릭 웹 서버에 대해 VPC 보안 그룹 생성

이제 퍼블릭 액세스를 위한 보안 그룹을 생성합니다. VPC의 퍼블릭 EC2 인스턴스에 연결하려면 VPC 보안 그룹에 인바운드 규칙을 추가해야 합니다. 이를 통해 인터넷에서 트래픽을 연결할 수 있습니다.

VPC 보안 그룹의 생성 방법

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. VPC 대시보드, 보안 그룹, 보안 그룹 생성을 차례대로 선택합니다.
3. 보안 그룹 생성 페이지에서 다음 값을 설정합니다.
   - 보안 그룹 이름: **`tutorial-securitygroup`**
   - 설명: **`Tutorial Security Group`**
   - VPC: 이전에 생성한 VPC를 선택합니다(예: vpc-***identifier***(tutorial-vpc)).
4. 인바운드 규칙을 보안 그룹에 추가합니다.
   a. Secure Shell(SSH)을 사용하여 VPC의 EC2 인스턴스에 연결하는 데 사용할 IP 주소를 지정합니다. 퍼블릭 IP 주소를 확인하려면 다른 브라우저 창 또는 탭에서 https://checkip.amazonaws.com의 서비스를 사용합니다. IP 주소의 예는 `203.0.113.25/32`입니다.

대부분의 경우 고정 IP 주소가 없는 방화벽 뒤나 인터넷 서비스 제공업체(ISP)를 통해 연결하는 경우가 많습니다. 그렇다면 클라이언트 컴퓨터에서 사용하는 IP 주소 범위를 찾습니다.

> **Warning**
>
> SSH 액세스에 `0.0.0.0/0`을 사용하는 경우 모든 IP 주소가 SSH를 사용하여 퍼블릭 인스턴스에 액세스할 수 있도록 활성화합니다. 이 방법은 테스트 환경에서 잠시 사용하는 것은 괜찮지만 프로덕션 환경에서는 안전하지 않습니다. 프로덕션에서는 SSH를 사용하여 인스턴스에 액세스할 수 있는 특정 IP 주소 또는 주소 범위만 인증합니다.

b. 인바운드 규칙 섹션에서 규칙 추가를 선택합니다.

c. 새로운 인바운드 규칙으로 다음 값을 설정하여 SSH에서 Amazon EC2 인스턴스에 액세스하도록 허용합니다. 이렇게 하면 Amazon EC2 인스턴스에 연결하여 웹 서버 및 다른 유틸리티를 설치할 수 있습니다. 또한 EC2 인스턴스에 연결하여 웹 서버의 콘텐츠를 업로드할 수도 있습니다.

- Type: **SSH**
- 소스: a단계의 IP 주소 또는 범위(예: **203.0.113.25/32**)

d. 규칙 추가를 선택합니다.

e. 새 인바운드 규칙에 다음 값을 설정하여 웹 서버에 대한 HTTP 액세스를 허용합니다.

- 유형: **HTTP**
- 소스: **0.0.0.0/0**

5. 보안 그룹을 생성하려면 보안 그룹 생성(Create security group)을 선택합니다.

   이 자습서에서 나중에 필요하므로 보안 그룹 ID를 적어 둡니다.

프라이빗 DB 인스턴스에 대한 VPC 보안 그룹 생성

DB 인스턴스를 프라이빗으로 유지하려면 프라이빗 액세스를 위한 보조 보안 그룹을 생성합니다. VPC의 프라이빗 DB 인스턴스에 연결하려면 웹 서버로부터의 트래픽만 허용하는 인바운드 규칙을 VPC 보안 그룹에 추가해야 합니다.

VPC 보안 그룹의 생성 방법

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.

2. VPC 대시보드, 보안 그룹, 보안 그룹 생성을 차례대로 선택합니다.
3. 보안 그룹 생성 페이지에서 다음 값을 설정합니다.

   - 보안 그룹 이름: **tutorial-db-securitygroup**
   - 설명: **Tutorial DB Instance Security Group**
   - VPC: 이전에 생성한 VPC를 선택합니다(예: vpc-***identifier***(tutorial-vpc)).
4. 인바운드 규칙을 보안 그룹에 추가합니다.

   a. 인바운드 규칙 섹션에서 규칙 추가를 선택합니다.

   b. 새로운 인바운드 규칙으로 다음 값을 설정하여 Amazon EC2 인스턴스의 포트 3306에서 MySQL 트래픽을 허용합니다. 이렇게 하면 웹 서버에서 DB 인스턴스에 연결할 수 있습니다. 그러면 웹 애플리케이션의 데이터를 데이터베이스에 저장하고 검색할 수 있습니다.

      - 유형: **MySQL/Aurora**
      - 소스: 이 자습서에서 이전에 생성한 tutorial-securitygroup 보안 그룹의 식별자(예: sg-9edd5cfb)입니다.
5. 보안 그룹을 생성하려면 보안 그룹 생성을 선택합니다.

DB 서브넷 그룹 만들기

DB 서브넷 그룹은 사용자가 VPC에서 만든 다음 DB 인스턴스에 대해 지정하는 서브넷의 모음입니다. DB 서브넷 그룹에서 DB 인스턴스를 생성할 때 특정 VPC를 지정할 수 있습니다.

DB 서브넷 그룹을 만들려면

1. VPC에서 데이터베이스의 프라이빗 서브넷을 식별합니다.

   a. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.

   b. VPC 대시보드를 선택한 다음 서브넷을 선택합니다.

   c. tutorial-subnet-private1-us-west-2a 및 tutorial-subnet-private2-us-west-2b라는 서브넷의 서브넷 ID를 기록해 둡니다.

      DB 서브넷 그룹을 생성할 때 서브넷 ID가 필요합니다.
2. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.

   Amazon VPC 콘솔이 아닌 Amazon RDS 콘솔에 연결해야 합니다.
3. 탐색 창에서 [Subnet groups]를 선택합니다.

4. DB 서브넷 그룹 생성을 선택합니다.
5. DB 서브넷 그룹 생성 페이지에서 서브넷 그룹 세부 정보에 이들 값을 설정합니다.
   - 이름: **`tutorial-db-subnet-group`**
   - 설명: **`Tutorial DB Subnet Group`**
   - VPC: tutorial-vpc(vpc-***`identifier`***)
6. 서브넷 추가 섹션에서 가용 영역 및 서브넷을 선택합니다.

   이 자습서에서는 us-west-2a와 us-west-2b를 가용 영역으로 선택합니다. 서브넷에서 이전 단계에서 식별한 프라이빗 서브넷을 선택합니다.
7. 생성(Create)을 선택합니다.

   새 DB 서브넷 그룹은 RDS 콘솔의 DB 서브넷 그룹 목록에 나타납니다. DB 서브넷 그룹을 선택하여 창 하단의 세부 정보 창에서 세부 정보를 볼 수 있습니다. 이러한 세부 정보에는 그룹과 연결된 모든 서브넷이 포함됩니다.

> **Note**
>
> 자습서: 웹 서버 및 Amazon RDS DB 인스턴스 생성을 완료하기 위해 이 VPC를 생성한 경우 Amazon RDS DB 인스턴스 생성 의 지침에 따라 DB 인스턴스를 생성합니다.

VPC 삭제

이 자습서에 대한 VPC 및 기타 리소스를 생성한 후, 더 이상 필요하지 않은 경우 삭제할 수 있습니다.

> **Note**
>
> 이 자습서를 위해 생성한 VPC에 리소스를 추가한 경우 VPC를 삭제하기 전에 이러한 리소스를 삭제해야 할 수 있습니다. 예를 들어 이러한 리소스에는 Amazon EC2 인스턴스 또는 Amazon RDS DB 인스턴스가 포함될 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC의 삭제를 참조하세요.

VPC 및 관련 리소스 삭제하기

1. DB 서브넷 그룹을 삭제합니다.

a. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
b. 탐색 창에서 서브넷 그룹을 선택합니다.
c. 삭제하려는 DB 서브넷 그룹을 선택합니다(예: tutorial-db-subnet-group).
d. 삭제를 선택한 다음 확인 창에서 삭제를 선택합니다.

2. VPC ID를 기록합니다.

a. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
b. VPC 대시보드를 선택한 다음 VPC를 선택합니다.
c. 목록에서 생성한 VPC를 식별합니다(예:tutorial-vpc).
d. 생성한 VPC의 VPC ID를 기록합니다. 이후 단계에서 해당 VPC ID가 필요합니다.

3. 보안 그룹을 삭제합니다.

a. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
b. VPC 대시보드를 선택한 다음 보안 그룹을 선택합니다.
c. Amazon RDS DB 인스턴스에 대한 보안 그룹을 선택합니다(예:tutorial-db-securitygroup).
d. 작업(Actions)에서 보안 그룹 삭제를 선택한 다음 확인 페이지에서 삭제(Delete)를 선택합니다.
e. 보안 그룹 페이지에서 Amazon EC2 인스턴스에 대한 보안 그룹을 선택합니다(예:tutorial-securitygroup).
f. 작업(Actions)에서 보안 그룹 삭제를 선택한 다음 확인 페이지에서 삭제(Delete)를 선택합니다.

4. VPC를 삭제합니다.

a. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
b. VPC 대시보드를 선택한 다음 VPC를 선택합니다.
c. 삭제하려는 VPC 선택합니다(예:tutorial-vpc).
d. 작업에서 VPC 삭제를 선택합니다.

   확인 페이지에는 VPC와 연결된 서브넷을 포함하여 삭제될 VPC와 연결된 다른 리소스가 표시됩니다.

e. 확인 페이지에서 **delete**를 입력하고 삭제를 선택합니다.

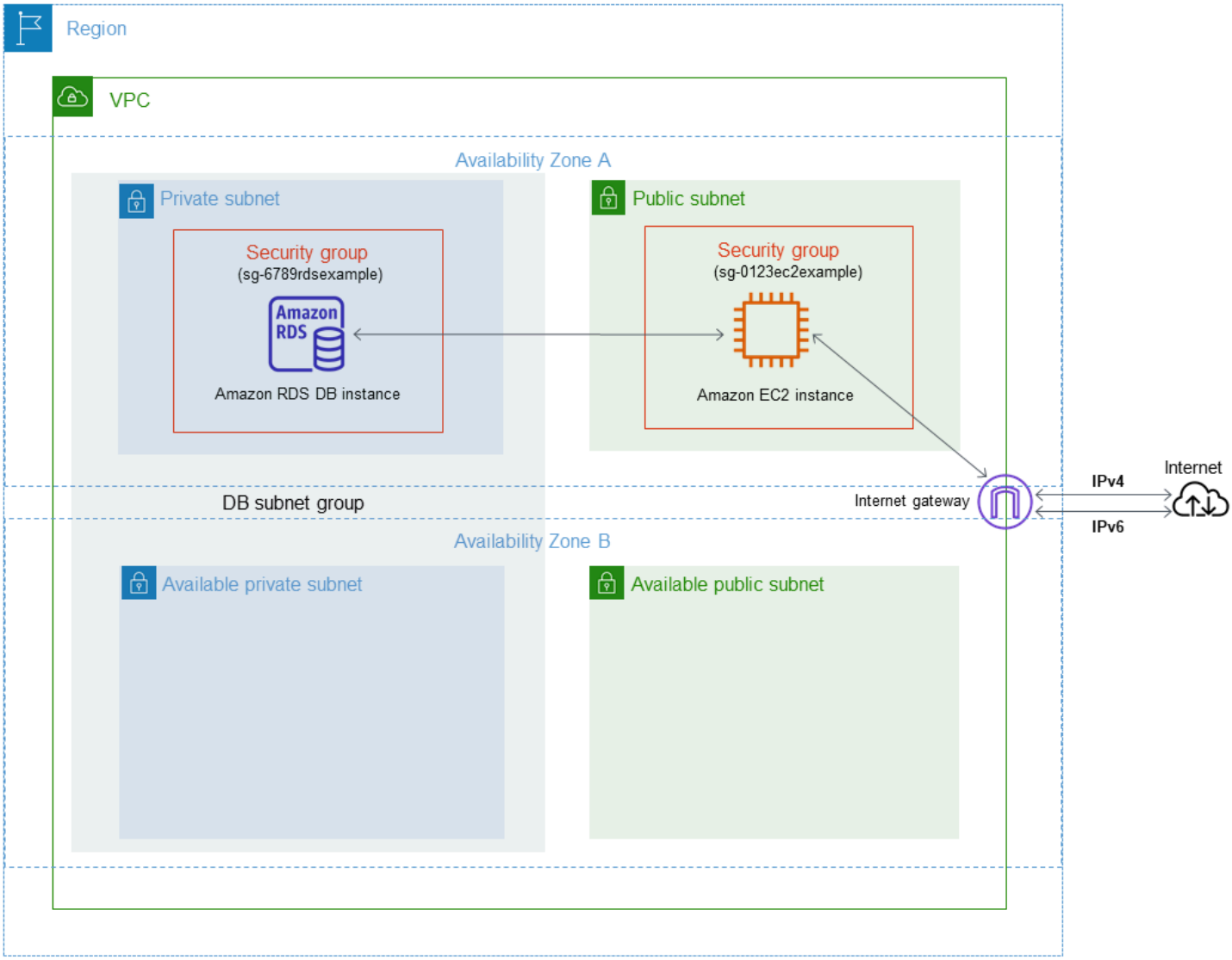
자습서: DB 인스턴스(듀얼 스택 모드)에 사용할 VPC 생성

일반적인 시나리오에는 Amazon VPC 서비스를 기반으로 하는 Virtual Private Cloud(VPC)의 DB 인스턴스가 포함됩니다. 이 VPC는 동일한 VPC에서 실행 중인 퍼블릭 Amazon EC2 인스턴스와 데이터를 공유합니다.

이 자습서에서는 듀얼 스택 모드에서 실행되는 데이터베이스와 함께 작동하는 이 시나리오의 VPC를 생성합니다. IPv6 주소 지정 프로토콜을 통한 연결을 가능하게 하는 듀얼 스택 모드입니다. IP 주소에 대한 자세한 내용은 Amazon RDS IP 주소 지정 단원을 참조하십시오.

이중 스택 네트워크 인스턴스 대부분의 리전에서 지원됩니다. 자세한 정보는 리전 및 버전 사용 가능 여부 섹션을 참조하세요. 이중 스택 모드의 제한 사항을 보려면 듀얼 스택 네트워크 DB 인스턴스에 대한 제한 사항을 참조하세요.

다음 다이어그램은 이 시나리오를 보여 줍니다.



다른 시나리오에 대한 자세한 내용은 VPC에서 DB 인스턴스에 액세스하는 시나리오을(를) 참조하세요.

퍼블릭 인터넷이 아닌 Amazon EC2 인스턴스에서만 DB 인스턴스를 사용할 수 있어야 합니다. 따라서 퍼블릭 서브넷과 프라이빗 서브넷을 모두 포함하여 VPC를 생성합니다. 퍼블릭 서브넷에서 Amazon EC2 인스턴스를 호스팅하므로 Amazon EC2 인스턴스에서 퍼블릭 인터넷에 액세스할 수 있습니다. DB 인스턴스는 프라이빗 서브넷에서 호스팅됩니다. Amazon EC2 인스턴스는 동일한 VPC 내에서 호스팅되므로 DB 인스턴스에 연결할 수 있습니다. 하지만 퍼블릭 인터넷에서는 DB 인스턴스를 사용할 수 없으므로 보안이 강화됩니다.

이 자습서에서는 별도의 가용 영역에서 추가 퍼블릭 및 프라이빗 서브넷을 구성합니다. 이러한 서브넷은 자습서에서 사용하지 않습니다. RDS DB 서브넷 그룹은 최소 2개의 가용 영역에 한 개 이상의 서브

넷이 필요합니다. 추가 서브넷을 사용하면 향후 다중 AZ DB 인스턴스 배포로 쉽게 전환할 수 있습니다.

듀얼 스택 모드를 사용하는 DB 인스턴스를 생성하려면 네트워크 유형(Network type) 설정에 듀얼 스택 모드(Dual-stack mode)를 지정하세요. 동일한 설정으로 DB 인스턴스를 수정할 수도 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 및 Amazon RDS DB 인스턴스 수정 단원을 참조하세요.

이 자습서에서는 Amazon RDS DB 인스턴스용 VPC 구성에 대해 설명합니다. Amazon VPC에 대한 자세한 내용은 Amazon VPC 사용 설명서를 참조하세요.

프라이빗 서브넷과 퍼블릭 서브넷을 포함하는 VPC 생성

다음은 퍼블릭 서브넷과 프라이빗 서브넷을 모두 포함하는 VPC를 생성하는 절차입니다.

VPC 및 서브넷을 생성하는 방법

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. AWS Management Console의 오른쪽 상단에서 VPC를 생성할 리전을 선택합니다. 이 예에서는 미국 동부(오하이오) 리전을 사용합니다.
3. 왼쪽 상단 모서리에서 VPC 대시보드를 선택합니다. VPC 생성을 시작하려면 VPC 생성을 선택합니다.
4. VPC 설정의 생성할 리소스에서 VPC 등을 선택합니다.
5. 남은 VPC 설정에서는 다음 값들을 설정합니다.

   - 네임 태그 자동 생성 - **`tutorial-dual-stack`**
   - IPv4 CIDR block: - **`10.0.0.0/16`**
   - IPv6 CIDR 블록(IPv6 CIDR block) - Amazon 제공 IPv6 CIDR 블록(Amazon-provided IPv6 CIDR block)
   - 테넌시 - 기본값
   - 가용 영역(AZ)의 수 - 2
   - Customize AZs(AZ 사용자 지정) - 기본값을 유지합니다.
   - 퍼블릭 서브넷 수 – 2
   - 프라이빗 서브넷 수 – 2
   - Customize subnets CIDR blocks(서브넷 CIDR 블록 사용자 지정) - 기본값을 유지합니다.
   - NAT 게이트웨이($) – 없음
   - 외부 전용 인터넷 게이트웨이 - 아니요

- VPC 엔드포인트 – 없음
- DNS options(DNS 옵션) - 기본값을 유지합니다.

> **Note**
>
> Amazon RDS에서는 다중 AZ DB 인스턴스 배포를 지원하려면 서로 다른 두 가용 영역에 있는 Amazon RDS가 필요합니다. 이 자습서에서는 단일 AZ 배포를 생성하지만 요구 사항에 따라 향후 다중 AZ DB 인스턴스 배포로 쉽게 전환할 수 있습니다.

6. VPC 생성을 선택합니다.

퍼블릭 Amazon EC2 인스턴스에 대한 VPC 보안 그룹 생성

이제 퍼블릭 액세스를 위한 보안 그룹을 생성합니다. VPC의 퍼블릭 EC2 인스턴스에 연결하려면 인터넷으로부터의 트래픽 연결을 허용하는 VPC 보안 그룹에 인바운드 규칙을 추가해야 합니다.

VPC 보안 그룹의 생성 방법

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. VPC 대시보드, 보안 그룹, 보안 그룹 생성을 차례대로 선택합니다.
3. 보안 그룹 생성 페이지에서 다음 값을 설정합니다.
   - 보안 그룹 이름: **`tutorial-dual-stack-securitygroup`**
   - 설명: **`Tutorial Dual-Stack Security Group`**
   - VPC: 이전에 생성한 VPC 선택(예: vpc-**###**(tutorial-dual-stack-vpc)
4. 인바운드 규칙을 보안 그룹에 추가합니다.
   a. Secure Shell(SSH)을 사용하여 VPC의 EC2 인스턴스에 연결하는 데 사용할 IP 주소를 지정합니다.

      IPv4(인터넷 프로토콜 버전 4) 주소의 예는 `203.0.113.25/32`입니다. IPv6(인터넷 프로토콜 버전 6) 주소 범위의 예는 `2001:db8:1234:1a00::/64`입니다.

      대부분의 경우 고정 IP 주소가 없는 방화벽 뒤나 인터넷 서비스 제공업체(ISP)를 통해 연결하는 경우가 많습니다. 그렇다면 클라이언트 컴퓨터에서 사용하는 IP 주소 범위를 찾습니다.

> **Warning**
>
> IPv4에 `0.0.0.0/0`, IPv6에 `::0`을 사용하면 SSH를 통해 모든 IP 주소가 퍼블릭 인스턴스에 액세스하도록 설정할 수 있습니다. 이 방법은 테스트 환경에서 잠시 사용하는 것은 괜찮지만 프로덕션 환경에서는 안전하지 않습니다. 프로덕션에서는 특정 IP 주소나 주소 범위만 인스턴스에 액세스하도록 허용하세요.

b. 인바운드 규칙 섹션에서 규칙 추가를 선택합니다.

c. 새로운 인바운드 규칙으로 다음 값을 설정하여 SSH(Secure Shell)에서 Amazon EC2 인스턴스에 액세스하도록 허용합니다. 이렇게 하면 EC2 인스턴스에 연결하여 SQL 클라이언트 및 다른 애플리케이션을 설치할 수 있습니다. EC2 인스턴스에 액세스할 수 있도록 IP 주소를 지정합니다.

- 유형: **SSH**
- 소스(Source): a단계의 IP 주소 또는 범위입니다. IPv4 IP 주소의 예는 **203.0.113.25/32**입니다. IPv6 IP 주소의 예는 **2001:DB8::/32**입니다.

5. 보안 그룹을 생성하려면 보안 그룹 생성(Create security group)을 선택합니다.

   이 자습서에서 나중에 필요하므로 보안 그룹 ID를 적어 둡니다.

프라이빗 DB 인스턴스에 대한 VPC 보안 그룹 생성

DB 인스턴스를 프라이빗으로 유지하려면 프라이빗 액세스를 위한 보조 보안 그룹을 생성합니다. VPC의 프라이빗 DB 인스턴스에 연결하려면 VPC 보안 그룹에 인바운드 규칙을 추가해야 합니다. 이 규칙은 Amazon EC2 인스턴스로부터의 트래픽만 허용합니다.

VPC 보안 그룹의 생성 방법

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. VPC 대시보드, 보안 그룹, 보안 그룹 생성을 차례대로 선택합니다.
3. 보안 그룹 생성 페이지에서 다음 값을 설정합니다.

- 보안 그룹 이름: **tutorial-dual-stack-db-securitygroup**
- 설명: **Tutorial Dual-Stack DB Instance Security Group**
- VPC: 이전에 생성한 VPC 선택(예: vpc-**###**(tutorial-dual-stack-vpc)

4. 인바운드 규칙을 보안 그룹에 추가합니다.

a. 인바운드 규칙 섹션에서 규칙 추가를 선택합니다.

b. 새로운 인바운드 규칙으로 다음 값을 설정하여 Amazon EC2 인스턴스의 포트 3306에서 MySQL 트래픽을 허용합니다. 이렇게 하면 EC2 인스턴스에서 DB 인스턴스에 연결할 수 있습니다. 이는 EC2 인스턴스의 데이터를 데이터베이스에 전송할 수 있음을 의미합니다.

- 유형: MySQL/Aurora
- 소스: 이 자습서에서 이전에 생성한 tutorial-dual-stack-securitygroup 보안 그룹의 식별자(예: sg-9edd5cfb)입니다.

5. 보안 그룹을 생성하려면 보안 그룹 생성을 선택합니다.

DB 서브넷 그룹 만들기

DB 서브넷 그룹은 사용자가 VPC에서 만든 다음 DB 인스턴스에 대해 지정하는 서브넷의 모음입니다. DB 서브넷 그룹을 사용하면 DB 인스턴스를 생성할 때 특정 VPC를 지정할 수 있습니다. DUAL과 호환 가능한 DB 서브넷 그룹을 생성하려면 모든 서브넷이 DUAL과 호환 가능해야 합니다. DUAL과 호환 가능하려면 서브넷에 IPv6 CIDR이 연결되어 있어야 합니다.

DB 서브넷 그룹을 만들려면

1. VPC에서 데이터베이스의 프라이빗 서브넷을 식별합니다.

   a. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.

   b. VPC 대시보드를 선택한 다음 서브넷을 선택합니다.

   c. tutorial-dual-stack-subnet-private1-us-west-2a 및 tutorial-dual-stack-subnet-private2-us-west-2b 서브넷의 서브넷 ID를 기록해 둡니다.

   DB 서브넷 그룹을 생성할 때 서브넷 ID가 필요합니다.

2. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.

   Amazon VPC 콘솔이 아닌 Amazon RDS 콘솔에 연결해야 합니다.

3. 탐색 창에서 [Subnet groups]를 선택합니다.

4. DB 서브넷 그룹 생성을 선택합니다.

5. DB 서브넷 그룹 생성 페이지에서 서브넷 그룹 세부 정보에 이들 값을 설정합니다.

- 이름: **tutorial-dual-stack-db-subnet-group**
- 설명: **Tutorial Dual-Stack DB Subnet Group**

- VPC: tutorial-dual-stack-vpc(vpc-***identifier***)

6. 서브넷 추가(Add subnets) 섹션에서 가용 영역(Availability Zones) 및 서브넷(Subnets)의 값을 선택합니다.

   이 자습서에서는 us-east-2a와 us-east-2b를 가용 영역으로 선택합니다. 서브넷에서 이전 단계에서 식별한 프라이빗 서브넷을 선택합니다.

7. 생성(Create)을 선택합니다.

새 DB 서브넷 그룹은 RDS 콘솔의 DB 서브넷 그룹 목록에 나타납니다. DB 서브넷 그룹을 선택하여 세부 정보를 확인할 수 있습니다. 여기에는 지원되는 주소 지정 프로토콜과 그룹과 연결된 모든 서브넷 및 DB 서브넷 그룹에서 지원하는 네트워크 유형이 포함됩니다.

듀얼 스택 모드로 Amazon EC2 인스턴스 생성

Amazon EC2 인스턴스를 생성하려면 Amazon EC2 사용 설명서의 새 시작 인스턴스 마법사를 사용하여 인스턴스 시작의 지침을 따르세요.

인스턴스 세부 정보 구성 페이지에서 이러한 값을 설정하고 다른 값은 기본값으로 유지합니다.

- 네트워크 - 프라이빗 서브넷과 퍼블릭 서브넷을 포함하는 VPC 생성에서 생성한 tutorial-dual-stack-vpc(vpc-***identifier***)와 같이 퍼블릭 및 프라이빗 서브넷이 있는 기존 VPC를 선택합니다.
- 서브넷 - 퍼블릭 Amazon EC2 인스턴스에 대한 VPC 보안 그룹 생성에서 생성한 subnet-***identifier*** | tutorial-dual-stack-subnet-public1-us-east-2a | us-east-2a와 같은 기존 퍼블릭 서브넷을 선택합니다.
- 퍼블릭 IP 자동 할당 - 활성화를 선택합니다.
- Auto-assign IPv6 IP - 활성화를 선택합니다.
- 방화벽(보안 그룹) - 기존 보안 그룹 선택을 선택합니다.
- 일반 보안 그룹 - 퍼블릭 Amazon EC2 인스턴스에 대한 VPC 보안 그룹 생성에서 생성한 `tutorial-securitygroup`과 같이 기존 보안 그룹을 선택합니다. 선택한 보안 그룹에 SSH(Secure Shell) 및 HTTP 액세스에 대한 인바운드 규칙이 포함되어 있는지 확인합니다.

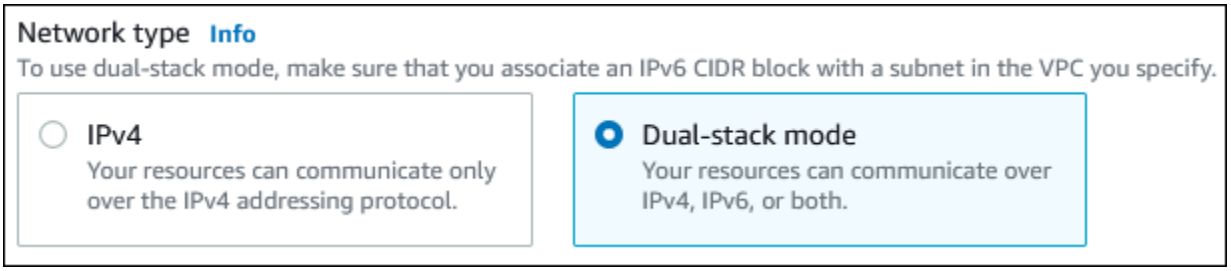
듀얼 스택 모드로 DB 인스턴스 생성

이 단계에서는 듀얼 스택 모드로 실행되는 Amazon RDS DB 인스턴스를 생성합니다.

DB 인스턴스를 생성하려면

1. https://console.aws.amazon.com/rds/에서 AWS Management Console에 로그인한 후 Amazon RDS 콘솔을 엽니다.
2. 콘솔의 오른쪽 상단에서 DB 인스턴스를 생성하려는 을 선택합니다. 이 예에서는 미국 동부(오하이오) 리전을 사용합니다.
3. 탐색 창에서 데이터베이스를 선택합니다.
4. 데이터베이스 생성을 선택합니다.
5. [데이터베이스 생성(Create database)] 페이지에서 다음과 같이 [표준 생성(Standard create)] 옵션이 선택되어 있는지 확인하고 []을 선택합니다.
6. [연결(Connectivity)] 섹션에서 다음 값을 설정합니다.

   - 네트워크 유형(Network type) – 듀얼 스택 모드(Dual-stack mode) 선택



   - 가상 사설 클라우드(VPC) - 프라이빗 서브넷과 퍼블릭 서브넷을 포함하는 VPC 생성에서 생성한 tutorial-dual-stack-vpc(vpc-*identifier*)와 같이 퍼블릭 및 프라이빗 서브넷이 있는 기존 VPC를 선택합니다.

     VPC는 서로 다른 가용 영역에 서브넷이 있어야 합니다.
   - 서브넷 그룹(Subnet Group) - DB 서브넷 그룹 만들기에서 생성한 tutorial-dual-stack-db-subnet-group과 같은 VPC용 DB 서브넷 그룹을 선택합니다.
   - 퍼블릭 액세스— 선택아니요.
   - VPC 보안 그룹 (방화벽)— 선택기존 항목 선택.
   - 기존 VPC 보안 그룹(Existing VPC security groups) – 프라이빗 DB 인스턴스에 대한 VPC 보안 그룹 생성에서 생성한 tutorial-dual-stack-db-securitygroup과 같이 프라이빗 액세스에 맞게 구성된 기존 VPC 보안 그룹을 선택합니다.

     각각에 연결된 X를 선택해 기본 보안 그룹 같은 다른 보안 그룹을 제거합니다.
   - [Availability Zone]에서 [us-west-2a]를 선택합니다.

     AZ 간 트래픽을 피하려면 DB 인스턴스와 EC2 인스턴스가 동일한 가용 영역에 있어야 합니다.

7. 나머지 섹션에서 DB 인스턴스 설정을 지정합니다. 각 설정에 대한 자세한 내용은 DB 인스턴스에 대한 설정 단원을 참조하세요.

Amazon EC2 인스턴스 및 DB 인스턴스에 연결

Amazon EC2 인스턴스와 DB 인스턴스가 듀얼 스택 모드로 생성된 후에는 IPv6 프로토콜을 사용하여 각 인스턴스에 연결할 수 있습니다. IPv6 프로토콜을 사용하여 Amazon EC2 인스턴스에 연결하려면 Amazon EC2 사용 설명서의 Linux 인스턴스에 연결에 있는 지침을 따르세요.

Amazon EC2 인스턴스에서 RDS for MySQL DB 인스턴스에 연결하려면 MySQL DB 인스턴스에 연결의 지침을 따르세요.

VPC 삭제

이 자습서에 대한 VPC 및 기타 리소스를 생성한 후, 더 이상 필요하지 않은 경우 삭제할 수 있습니다.

이 자습서를 위해 생성한 VPC에 리소스를 추가한 경우 VPC를 삭제하기 전에 이러한 리소스를 삭제해야 할 수 있습니다. 리소스의 예로는 Amazon EC2 인스턴스 또는 DB 인스턴스가 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC의 삭제를 참조하세요.

VPC 및 관련 리소스 삭제하기

1. DB 서브넷 그룹을 삭제합니다.
    a. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
    b. 탐색 창에서 서브넷 그룹을 선택합니다.
    c. 삭제할 DB 서브넷 그룹을 선택합니다(예: tutorial-db-subnet-group).
    d. 삭제를 선택한 다음 확인 창에서 삭제를 선택합니다.
2. VPC ID를 기록합니다.
    a. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
    b. VPC 대시보드를 선택한 다음 VPC를 선택합니다.
    c. 목록에서 생성한 VPC를 식별합니다(예:tutorial-dual-stack-vpc).
    d. 생성한 VPC의 VPC ID를 기록합니다. 이후 단계에서 해당 VPC ID가 필요합니다.
3. 보안 그룹을 삭제합니다.
    a. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
    b. VPC 대시보드를 선택한 다음 보안 그룹을 선택합니다.

c. Amazon RDS DB 인스턴스에 대한 보안 그룹을 선택합니다(예:utorial-dual-stack-db-securitygroup).

d. 작업(Actions)에서 보안 그룹 삭제(Delete security groups)를 선택한 다음 확인 페이지에서 삭제(Delete)를 선택합니다.

e. 보안 그룹(Security Groups) 페이지에서 Amazon EC2 인스턴스에 대한 보안 그룹을 선택합니다(예:tutorial-dual-stack-securitygroup).

f. 작업(Actions)에서 보안 그룹 삭제(Delete security groups)를 선택한 다음 확인 페이지에서 삭제(Delete)를 선택합니다.

4. NAT 게이트웨이를 삭제합니다.

a. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.

b. VPC 대시보드를 선택한 다음 NAT 게이트웨이를 선택합니다.

c. 생성한 VPC의 NAT 게이트웨이를 선택합니다. VPC ID를 사용하여 올바른 NAT 게이트웨이를 식별합니다.

d. 작업에서 NAT 게이트웨이 삭제(Delete NAT gateway)를 선택합니다.

e. 확인 페이지에서 **delete**를 입력하고 삭제를 선택합니다.

5. VPC를 삭제합니다.

a. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.

b. VPC 대시보드를 선택한 다음 VPC를 선택합니다.

c. 삭제하려는 VPC를 선택합니다(예:tutorial-dual-stack-vpc).

d. 작업에서 VPC 삭제를 선택합니다.

확인 페이지에는 VPC와 연결된 서브넷을 포함하여 삭제될 VPC와 연결된 다른 리소스가 표시됩니다.

e. 확인 페이지에서 **delete**를 입력하고 삭제를 선택합니다.

6. 탄력적 IP 주소를 릴리스합니다.

a. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.

b. EC2 대시보드를 선택한 다음 탄력적 IP 주소를 선택합니다.

c. 릴리스하려는 탄력적 IP 주소를 선택합니다.

d. 작업(Actions)에서 탄력적 IP 주소 릴리스(Release Elastic IP addresses)를 선택합니다.

e. 확인 페이지에서 릴리스를 선택합니다.

VPC에 있지 않은 DB 인스턴스를 VPC로 이동

EC2-Classic 플랫폼 상의 일부 레거시 DB 인스턴스는 VPC에서 실행되지 않습니다. DB 인스턴스가 VPC 내에 있지 않을 경우 AWS Management Console을 사용하여 DB 인스턴스를 VPC 내로 손쉽게 이동할 수 있습니다. VPC에 있지 않은 DB 인스턴스를 VPC 내로 이동하려면 먼저 VPC를 만들어야 합니다.

EC2-Classic은 2022년 8월 15일에 사용 중지되었습니다. EC2-Classic에서 VPC로 마이그레이션하지 않은 경우 가능한 한 빨리 마이그레이션하는 것이 좋습니다. 자세한 내용은 Amazon EC2 사용 설명서의 EC2-Classic에서 VPC로 마이그레이션 및 블로그 EC2-Classic Networking is Retiring – Here's How to Prepare를 참조하세요.

Important

신규 Amazon RDS 고객이거나, 전에는 DB 인스턴스를 생성한 적이 전혀 없거나, 전에 사용한 적 없는 AWS 리전에서 DB 인스턴스를 생성하는 경우, 거의 모든 경우에 EC2-VPC 플랫폼에 있고 기본 VPC가 생성됩니다. VPC의 DB 인스턴스 작업에 대한 자세한 내용은 VPC에서 DB 인스턴스를 사용한 작업을(를) 참조하세요.

다음 단계에 따라 DB 인스턴스에 대한 VPC를 만듭니다.

- 1단계: VPC 생성
- 2단계: DB 서브넷 그룹 만들기
- 3단계: VPC 보안 그룹 만들기

VPC를 만든 후 다음 단계에 따라 DB 인스턴스를 VPC 내로 이동합니다.

- DB 인스턴스에 대한 VPC 업데이트

마이그레이션 직전에 DB 인스턴스의 백업을 생성하는 것이 좋습니다. 이렇게 하면 마이그레이션이 실패할 경우 데이터를 복원할 수 있습니다. 자세한 내용은 데이터 백업, 복원 및 내보내기 섹션을 참조하세요.

다음은 DB 인스턴스를 VPC로 이동하는 데 따른 몇 가지 제한 사항입니다.

- 이전 세대 DB 인스턴스 클래스 – 이전 세대 DB 인스턴스 클래스는 VPC 플랫폼에서 지원되지 않을 수 있습니다. DB 인스턴스를 VPC로 이동할 때 db.m3 또는 db.r3 DB 인스턴스 클래스를 선택합니다. DB 인스턴스를 VPC로 이동한 후 이후 DB 인스턴스 클래스를 사용하도록 DB 인스턴스를 확장할 수 있습니다. VPC 지원 인스턴스 클래스의 전체 목록은 Amazon RDS 인스턴스 유형을 참조하세요.
- 다중 AZ – VPC에 있지 않은 다중 AZ DB 인스턴스를 VPC로 이동하는 것은 현재 지원되지 않습니다. DB 인스턴스를 VPC로 이동하려면 단일 AZ 배포가 되도록 먼저 DB 인스턴스를 수정합니다. 다중 AZ 배포 설정을 [아니오(No)]로 변경합니다. DB 인스턴스를 VPC로 이동한 후 이를 다시 수정하여 다중 AZ 배포로 만듭니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.
- 읽기 전용 복제본 – VPC에 있지 않은 읽기 전용 복제본을 포함한 DB 인스턴스를 VPC로 이동하는 것은 현재 지원되지 않습니다. DB 인스턴스를 VPC로 이동하려면 먼저 읽기 전용 복제본을 모두 삭제합니다. DB 인스턴스를 VPC로 이동한 후 읽기 전용 복제본을 다시 생성합니다. 자세한 내용은 DB 인스턴스 읽기 전용 복제본 작업 섹션을 참조하세요.
- 옵션 그룹 – DB 인스턴스를 VPC로 이동하고 DB 인스턴스에서 사용자 지정 옵션 그룹을 사용하는 경우 DB 인스턴스와 연결된 옵션 그룹을 변경합니다. 옵션 그룹은 플랫폼마다 특정하며, VPC로 이동하는 것은 플랫폼의 변경에 해당합니다. 이런 경우에서 사용자 지정 옵션 그룹을 사용하려면 기본 VPC 옵션 그룹을 DB 인스턴스에 할당하거나, 이동하려는 대상 VPC에 있는 다른 DB 인스턴스에서 사용되는 옵션 그룹을 할당하거나, 새 옵션 그룹을 만들어 DB 인스턴스에 할당합니다. 자세한 내용은 옵션 그룹 작업 섹션을 참조하세요.

가동 중지를 최소화하면서 VPC에 없는 DB 인스턴스를 VPC로 이동하는 방법

다음 방법을 사용하여 가동 중지를 최소화하면서 VPC에 없는 DB 인스턴스를 VPC로 이동할 수 있습니다. 이러한 방법은 원본 DB 인스턴스의 중단을 최소화하고 마이그레이션 중에 사용자 트래픽을 처리하도록 허용합니다. 그러나 VPC로 마이그레이션하는 데 필요한 시간은 데이터베이스 크기 및 라이브 워크로드 특성에 따라 달라집니다.

- AWS Database Migration Service(AWS DMS) - AWS DMS를 사용하면 소스 DB 인스턴스가 완전히 작동하는 동안 데이터의 라이브 마이그레이션을 수행할 수 있지만 제한된 DDL 문 세트만 복제합니다. AWS DMS는 인덱스, 사용자, 권한, 저장 프로시저, 테이블 데이터와 직접 관련되지 않은 기타 데이터베이스 변경 사항과 같은 항목을 전파하지 않습니다. 또한 AWS DMS에서는 초기 DB 인스턴스 생성에 RDS 스냅샷을 자동으로 사용하지 않으므로 마이그레이션 시간이 늘어날 수 있습니다. 자세한 내용은 AWS Database Migration Service을 참조하세요.
- DB 스냅샷 복원 또는 특정 시점 복구 – DB 인스턴스의 스냅샷을 복원하거나 DB 인스턴스를 특정 시점으로 복원하여 DB 인스턴스를 VPC로 이동할 수 있습니다. 자세한 내용은 DB 인스턴스 복원 및 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 단원을 참조하세요.

Amazon RDS에 대한 할당량 및 제약 조건

다음에는 Amazon RDS에 대한 리소스 할당량 및 명명 제약 조건에 대한 설명을 찾을 수 있습니다.

주제

- Amazon RDS의 할당량
- Amazon RDS의 명명 제약 조건
- 최대 데이터베이스 연결 수
- Amazon RDS의 파일 크기 제한

Amazon RDS의 할당량

각 AWS 계정에는 AWS 리전마다 생성할 수 있는 Amazon RDS 리소스 수에 할당량이 있습니다. 리소스 할당량에 도달하면 해당 리소스 생성을 위한 추가 호출이 예외와 함께 실패합니다.

다음 표에는 AWS 리전별 리소스 및 그 할당량이 나열되어 있습니다.

이름	기본값	조정 가능	설명
DB 보안 그룹당 권한 부여	지원되는 각 리전: 20	아니요	DB 보안 그룹당 보안 그룹 인증 수
사용자 지정 엔진 버전	지원되는 각 리전: 40	예	현재 리전에서 이 계정에 허용된 커스텀 엔진 버전의 최대 개수
DB 클러스터 파라미터 그룹	지원되는 각 리전: 50	아니요	DB 클러스터 파라미터 그룹의 최대 개수

이름	기본값	조정 가능	설명
DB 클러스터	지원되는 각 리전: 40	예	현재 리전에서 이 계정에 허용된 클러스터의 최대 개수
DB 인스턴스	지원되는 각 리전: 40	예	현재 리전에서 이 계정에 허용된 DB 인스턴스의 최대 개수
DB 서브넷 그룹	지원되는 각 리전: 50	예	DB 서브넷 그룹의 최대 개수
데이터 API HTTP 요청 본문 크기	지원되는 각 리전: 4MB	아니요	HTTP 요청 본문에 허용된 최대 크기입니다.
데이터 API 최대 동시 클러스터-암호 페어 수	지원되는 각 리전: 30개	아니요	현재 AWS 리전의 이 계정에 대한 동시 데이터 API 요청에서 Aurora Serverless v1 DB 클러스터 및 암호의 최대 고유한 페어 개수입니다.
데이터 API 최대 동시 요청 수	지원되는 각 리전: 500	아니요	Aurora Serverless v1 DB 클러스터에 대해 동일한 암호를 사용하고 동시에 처리될 수 있는 최대 데이터 API 요청 개수입니다. 추가 요청은 대기열에 추가되고, 처리 중인 요청이 완료되면 처리됩니다.

이름	기본값	조정 가능	설명
데이터 API 최대 결과 집합 크기	지원되는 각 리전: 1MB	아니요	데이터 API에서 반환할 수 있는 데이터베이스 결과 세트의 최대 크기입니다.
JSON 응답 문자열의 데이터 API 최대 크기	지원되는 각 리전: 10MB	아니요	RDS 데이터 API에서 반환하는 단순화된 JSON 응답 문자열의 최대 크기입니다.
초당 데이터 API 요청 수	지원되는 각 리전: 초당 1,000개	아니요	현재 AWS 리전에서 이 계정에 허용되는 초당 데이터 API에 대한 최대 요청 수입니다. 이 할당량은 Amazon Aurora Serverless v1 클러스터에만 적용됩니다.
이벤트 구독	지원되는 각 리전: 20	예	이벤트 구독의 최대 개수
DB 클러스터당 IAM 역할	지원되는 각 리전: 5	예	DB 클러스터와 연결된 IAM 역할의 최대 개수
DB 인스턴스당 IAM 역할	지원되는 각 리전: 5	예	DB 인스턴스와 연결된 IAM 역할의 최대 개수
통합	지원되는 각 리전: 100	아니요	현재 AWS 리전에서 이 계정에 허용된 최대 통합 수
수동 DB 클러스터 스냅샷	지원되는 각 리전: 100	예	수동 DB 클러스터 스냅샷의 최대 수

이름	기본값	조정 가능	설명
수동 DB 인스턴스 스냅샷 수	지원되는 각 리전: 100	예	수동 DB 인스턴스 스냅샷의 최대 수
옵션 그룹 수	지원되는 각 리전: 20	예	옵션 그룹의 최대 개수
파라미터 그룹	지원되는 각 리전: 50	예	파라미터 그룹의 최대 개수
프록시	지원되는 각 리전: 20	예	현재 AWS 리전에서 이 계정에 허용된 프록시의 최대 개수
기본당 읽기 전용 복제본	지원되는 각 리전: 15개	예	프라이머리 DB 인스턴스당 읽기 전용 복제본의 최대 개수 Amazon Aurora에서는 이 할당량을 조정할 수 없습니다.
예약 DB 인스턴스	지원되는 각 리전: 40	예	현재 AWS 리전에서 이 계정에 허용된 예약 DB 인스턴스의 최대 개수
보안 그룹당 규칙	지원되는 각 리전: 20	아니요	DB 보안 그룹당 규칙의 최대 개수
보안 그룹	지원되는 각 리전: 25	예	DB 보안 그룹의 최대 개수
보안 그룹(VPC)	지원되는 각 리전: 5	아니요	Amazon VPC당 DB 보안 그룹의 최대 개수

이름	기본값	조정 가능	설명
DB 서브넷 그룹당 서브넷 수	지원되는 각 리전: 20	아니요	DB 서브넷 그룹당 서브넷의 최대 개수
리소스당 태그	지원되는 각 리전: 50	아니요	Amazon RDS 리소스당 태그의 최대 개수
모든 DB 인스턴스의 총 스토리지	지원되는 각 리전: 100,000기가바이트	예	함께 추가된 모든 Amazon RDS DB 인스턴스에 대한 EBS 볼륨의 최대 총 스토리지(GB)입니다. 이 할당량은 각 DB 클러스터에 대해 최대 클러스터 볼륨이 128TiB인 Amazon Aurora에는 적용되지 않습니다.

Note

기본적으로 최대 총 40개의 DB 인스턴스를 실행할 수 있습니다. RDS DB 인스턴스, Aurora DB 인스턴스, Amazon Neptune 인스턴스 및 Amazon DocumentDB 인스턴스가 이 할당량에 적용됩니다.
Amazon RDS DB 인스턴스에는 다음과 같은 제한이 적용됩니다.

- '라이선스 포함' 모델에서 각 SQL Server 에디션(Enterprise, Standard, Web, 및 Express)의 경우 10개
- '라이선스 포함' 모델에서 Oracle의 경우 10개
- 'BYOL(bring-your-own-license)' 모델에서 Db2의 경우 40개
- MySQL, MariaDB 또는 PostgreSQL의 경우 40개
- 'BYOL4(bring-your-own-license)' 라이선싱 모델에서 Oracle의 경우 40개

애플리케이션에 DB 인스턴스가 더 필요한 경우 Service Quotas 콘솔을 열어 추가 DB 인스턴스를 요청할 수 있습니다. 탐색 창에서 AWS 서비스를 선택합니다. Amazon Relational Database Service(Amazon RDS)를 선택하고, 할당량을 선택한 다음, 지침에 따라 할당량 증가를 요청합니다. 자세한 내용은 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하십시오.

RDS for Oracle과 RDS for SQL Server의 경우 읽기 전용 복제본의 한도는 각 리전의 소스 데이터베이스당 5개입니다.

AWS Backup에서 관리하는 백업은 수동 DB 스냅샷으로 간주되지만 수동 스냅샷 할당량에 포함되지는 않습니다. AWS Backup에 대한 자세한 내용은 AWS Backup 개발자 가이드를 참조하십시오.

RDS API 작업을 사용하고 초당 호출 수의 기본 할당량을 초과하는 경우, Amazon RDS API는 다음과 같이 오류를 생성합니다.

ClientError: *API_name* 작업을 호출하는 동안 오류 발생(ThrottlingException): 속도 초과됨.

이 경우 초당 호출 수를 줄입니다. 할당량은 대부분의 사용 사례를 다루기 위한 것입니다. 더 높은 할당량이 필요한 경우 다음 옵션 중 하나를 사용하여 할당량 증가를 요청할 수 있습니다.

- 콘솔에서 Service Quotas 콘솔을 엽니다.
- AWS CLI에서 request-service-quota-increase AWS CLI 명령을 사용합니다.

자세한 내용은 Service Quotas 사용 설명서를 참조하세요.

Amazon RDS의 명명 제약 조건

다음은 Amazon RDS의 명명 제약 조건을 설명한 것입니다.

- DB 인스턴스 식별자:
  - 1–63자의 영숫자 또는 하이픈으로 구성되어야 합니다.
  - 첫 번째 문자는 글자이어야 합니다.
  - 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.
  - 각 AWS 리전별로 AWS 계정 1개의 모든 DB 인스턴스는 고유해야 합니다.
- 초기 데이터베이스 이름:

  - 데이터베이스 이름 제약 조건은 각 데이터베이스 엔진에 따라 다릅니다. 자세한 내용은 각 DB 인스턴스를 생성할 때 사용 가능한 설정을 참조하세요.
  - SQL Server – DB 인스턴스를 만든 후 데이터베이스를 만듭니다.
- 마스터 사용자 이름 - 마스터 사용자 이름 제약 조건은 각 데이터베이스 엔진에 따라 다릅니다. 자세한 내용은 DB 인스턴스를 생성할 때 사용 가능한 설정을 참조하세요.
- 마스터 암호:
  - 마스터 데이터베이스 사용자의 암호에는 /, ', ", @ 또는 공백을 제외한 모든 인쇄 가능한 ASCII 문자가 포함될 수 있습니다.

    Oracle의 경우 &는 추가적인 문자 제한 사항입니다.
  - 암호는 DB 엔진에 따라 다음과 같은 수의 인쇄 가능한 ASCII 문자를 포함할 수 있습니다.
    - Db2: 8–255
    - MariaDB 및 MySQL: 8–41
    - Oracle: 8–30
    - SQL Server 및 PostgreSQL: 8–128
- DB 파라미터 그룹:
  - 1–255자의 영숫자로 구성되어야 합니다.
  - 첫 번째 문자는 글자이어야 합니다.
  - 하이픈은 허용되지만 이름은 하이픈으로 끝나거나 하이픈이 2개 연속으로 이어져서는 안 됩니다.
- DB 서브넷 그룹:
  - 1–255자로 구성되어야 합니다.
  - 영숫자, 스페이스, 하이픈, 밑줄, 마침표를 사용할 수 있습니다.

최대 데이터베이스 연결 수

최대 동시 데이터베이스 연결 수는 DB 엔진 유형과 DB 인스턴스 클래스의 메모리 할당에 따라 다릅니다. 최대 연결 수는 일반적으로 DB 인스턴스에 연결된 파라미터 그룹에서 설정됩니다. 예외적으로, Microsoft SQL Server의 경우 SQL Server Mangment Studio(SSMS)의 DB 인스턴스에 대한 서버 속성에서 설정됩니다.

데이터베이스 연결은 메모리를 사용합니다. 이러한 파라미터 중 하나를 너무 높게 설정하면 메모리 부족 상태가 발생하여 DB 인스턴스가 파라미터 호환 장애(incompatible-parameters) 상태가 될 수 있습

니다. 자세한 내용은 메모리 제한에 대한 호환되지 않는 파라미터 상태 진단 및 해결 단원을 참조하십시오.

애플리케이션이 연결을 자주 열고 닫거나, 수명이 긴 연결을 많이 열어 두는 경우, Amazon RDS 프록시를 사용하는 것이 좋습니다. RDS 프록시는 데이터베이스 연결을 효율적으로 안전하게 공유하기 위해 연결 풀링을 사용하는 완전관리형 고가용성 데이터베이스 프록시입니다. RDS 프록시에 대한 자세한 내용은 Amazon RDS Proxy 섹션을 참조하세요.

> **Note**
>
> Oracle의 경우 최대 사용자 프로세스 수와 사용자 및 시스템 세션을 설정합니다.
> Db2의 경우 최대 연결 수를 설정할 수 없습니다. 한도는 64,000입니다.

다음 표에는 여러 DB 엔진의 최대 데이터베이스 연결 수에 대한 정보가 나와 있습니다.

DB 엔진	파라미터	허용된 값	기본값	설명
MariaDB 및 MySQL	`max_connections`	1–100000	MariaDB 버전 10.5 및 10.6을 제외한 모든 MariaDB 및 MySQL 버전의 기본값: {DBInstanceClassMemory/12582880} 공식은 MB/12와 사실상 동일합니다. MariaDB 버전 10.5 및 10.6의 기본값: LEAST({DBInstanceClassMemory/25165760},12000) 공식은 MB/25와 사실상 동일합니다.	허용되는 동시 클라이언트 연결 수

DB 엔진	파라미터	허용된 값	기본값	설명
			두 경우 모두 기본값 계산으로 인해 값이 16,000보다 큰 경우 Amazon RDS는 MariaDB 및 MySQL DB 인스턴스에 대한 제한을 16,000으로 설정합니다.	
Oracle	`processes`	80–20000	LEAST({DBInstanceClassMemory/9868951}, 20000)	사용자 프로세스
Oracle	`sessions`	100–65535	해당 사항 없음	사용자 및 시스템 세션
PostgreSQL	`max_connections`	6–8388607	LEAST({DBInstanceClassMemory/9531392}, 5000)	최대 동시 연결 수
SQL 서버	최대 동시 연결 수	0–32767	0(무제한)	최대 동시 연결 수

`DBInstanceClassMemory`는 바이트 단위입니다. 이 값이 계산되는 방법에 대한 자세한 내용은 DB 파라미터 지정 섹션을 참조하세요. 운영 체제 및 RDS 관리 프로세스용으로 예약된 메모리로 인해 이 메모리 크기는 에 대한 DB 인스턴스 클래스의 하드웨어 사양에 표시된 기비바이트(GiB) 값보다 작습니다.

예를 들어 일부 DB 인스턴스 클래스는 메모리 크기가 8GiB(8,589,934,592바이트)입니다. 메모리가 8GiB인 DB 인스턴스 클래스(예: db.m7g.large)에서 실행되는 MySQL DB 인스턴스의 경우 총 메모리를 사용하는 방정식은 `8589934592/12582880=683`입니다. 그러나 변수 `DBInstanceClassMemory`는 운영 체제에 예약된 양과 DB 인스턴스를 관리하는 RDS 프로세스에 예약된 양을 자동으로 뺍니다. 그런 다음 빼고 남은 값을 12,582,880으로 나눕니다. 이렇게 계산하면 `max_connections`의 값이 683이 아닌 약 630이 됩니다. 이 값은 DB 인스턴스 클래스 및 DB 엔진에 따라 달라집니다.

MariaDB 또는 MySQL DB 인스턴스가 db.t3.micro 또는 db.t3.small과 같은 소규모 DB 인스턴스 클래스에서 실행 중인 경우 사용 가능한 총 메모리가 부족합니다. 이러한 DB 인스턴스 클래스의 경우

RDS는 사용 가능한 메모리의 상당 부분을 예약하며, 이는 max_connections 값에 영향을 줍니다. 예를 들어 db.t3.micro DB 인스턴스 클래스에서 실행되는 MySQL DB 인스턴스의 기본 최대 연결 수는 약 60개입니다. DB MariaDB 또는 MySQL DB 인스턴스에 연결하고 다음 SQL 명령을 실행하여 DB MariaDB 또는 MySQL DB 인스턴스의 max_connections 값을 확인할 수 있습니다.

```
SHOW GLOBAL VARIABLES LIKE 'max_connections';
```

Amazon RDS의 파일 크기 제한

파일 크기 제한은 특정 Amazon RDS DB 인스턴스에 적용됩니다. 자세한 내용은 다음의 엔진별 제한을 참조하십시오.

- Amazon RDS의 MariaDB 파일 크기 제한
- Amazon RDS의 MySQL 파일 크기 제한
- Amazon RDS의 Oracle 파일 크기 제한

Amazon RDS 문제 해결

다음 섹션을 바탕으로 Amazon RDS 및 Amazon Aurora의 DB 인스턴스와 관련하여 발생하는 문제를 해결할 수 있습니다.

주제

- Amazon RDS DB 인스턴스에 연결할 수 없음
- Amazon RDS 보안 문제
- 호환되지 않는 네트워크 상태 문제 해결
- DB 인스턴스 소유자 암호 재설정
- Amazon RDS DB 인스턴스 중단 또는 재부팅
- Amazon RDS DB 파라미터 변경 사항이 적용 안 됨
- Amazon RDS DB 인스턴스 스토리지 부족
- Amazon RDS 사용 가능한 DB 인스턴스 부족
- Amazon RDS의 여유 메모리 부족
- MySQL 및 MariaDB 문제
- 백업 보존 기간을 0으로 설정할 수 없음

Amazon RDS API를 사용해 문제를 디버깅하는 방법에 대한 자세한 내용은 Amazon RDS에서 애플리케이션 문제 해결 단원을 참조하세요.

Amazon RDS DB 인스턴스에 연결할 수 없음

DB 인스턴스에 연결할 수 없는 경우 공통적인 원인은 다음과 같습니다.

- 인바운드 규칙 – 로컬 방화벽에서 적용되는 액세스 규칙과 DB 인스턴스에 액세스할 수 있는 권한이 부여된 IP 주소가 일치하지 않을 수 있습니다. 보안 그룹의 인바운드 규칙에 문제가 있을 가능성이 매우 높습니다.

  기본적으로 DB 인스턴스는 액세스를 허용하지 않습니다. 액세스 권한은 VPC와 연결된 보안 그룹을 통해 부여되며, 이는 DB 인스턴스로 들어오고 나가는 트래픽을 허용합니다. 필요한 경우 특정 상황에 대한 인바운드 및 아웃바운드 규칙을 보안 그룹에 추가합니다. IP 주소, IP 주소의 범위 또는 다른 VPC 보안 그룹을 지정할 수 있습니다.

> **Note**
>
> 새 인바운드 규칙을 추가할 때 원본 의 내 IP를 선택하여 브라우저에서 감지된 IP 주소에서 DB 인스턴스에 액세스하도록 허용할 수 있습니다.

보안 그룹 설정에 대한 자세한 내용은 보안 그룹을 생성하여 VPC 내부의 DB 인스턴스에 대한 액세스를 제공 단원을 참조하십시오.

> **Note**
>
> 169.254.0.0/16 범위의 IP 주소에서 클라이언트 연결은 허용되지 않습니다. 이는 로컬 링크 주소 지정에 사용되는 APIPA(Automatic Private IP Addressing) 범위입니다.

- 퍼블릭 액세스 가능성 – 클라이언트 애플리케이션을 사용하는 등 VPC 외부에서 DB 인스턴스에 연결하려면 인스턴스에 퍼블릭 IP 주소가 할당되어 있어야 합니다.

  인스턴스에 공개적으로 액세스할 수 있도록 하려면 인스턴스를 수정하고 Public accessibility(퍼블릭 액세스 가능성)에서 예를 선택합니다. 자세한 내용은 VPC에 있는 DB 인스턴스를 인터넷에서 숨기기 단원을 참조하십시오.
- 포트 – 로컬 방화벽 제한 때문에 DB 인스턴스를 만들 때 지정한 포트를 사용하여 통신을 주고받을 수 없습니다. 이 경우 네트워크에서 지정한 포트를 인바운드 및 아웃바운드 통신에 사용할 수 있는지 여부를 네트워크 관리자에게 확인하십시오.
- 가용성 – 새로 생성한 DB 인스턴스의 경우 DB 인스턴스를 사용할 준비가 될 때까지 DB 인스턴스의 상태는 `creating`입니다. 상태가 `available`로 변경되면 DB 인스턴스에 연결할 수 있습니다. DB 인스턴스의 크기에 따라, 인스턴스를 사용할 수 있을 때까지 최장 20분까지 걸릴 수 있습니다.
- 인터넷 게이트웨이 – DB 인스턴스에 공개적으로 액세스할 수 있도록 하려면 DB 서브넷 그룹의 서브넷에 인터넷 게이트웨이가 있어야 합니다.

  서브넷에 인터넷 게이트웨이를 구성하려면

  1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
  2. 탐색 창에서 데이터베이스를 선택한 다음 DB 인스턴스의 이름을 선택합니다.
  3. 연결&보안 탭에서, VPC에서의 VPC ID 값과 서브넷에서의 서브넷 ID 값을 적어둡니다.
  4. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.

5. 탐색 창에서 [Internet Gateways]를 선택합니다. VPC에 인터넷 게이트웨이가 연결되어 있는지 확인합니다. 또는 인터넷 게이트웨이 생성을 선택하여 인터넷 게이트웨이를 만듭니다. 인터넷 게이트웨이를 선택한 후 VPC에 연결을 선택하고 지침에 따라 VPC에 연결합니다.
6. 탐색 창에서 서브넷을 선택한 후 해당 서브넷을 선택합니다.
7. [라우팅 테이블(Route table)] 탭에서 대상 위치로 `0.0.0.0/0` 경로가 있으며, VPC의 대상으로 해당 인터넷 게이트웨이가 있는지 확인합니다.

   IPv6 주소를 이용해 인스턴스에 연결하는 경우 인터넷 게이트웨이를 가리키는 모든 IPv6 트래픽(`::/0`)에 대한 경로가 있는지 확인합니다. 그렇지 않으면 다음을 수행하십시오.

   a. 라우팅 테이블의 ID(rtb-xxxxxxxx)를 선택해 해당 라우팅 테이블로 이동합니다.
   b. 라우팅 탭에서 라우팅 편집을 선택합니다. 라우팅 추가를 선택하고 대상 위치로 `0.0.0.0/0`을, 대상으로 인터넷 게이트웨이를 사용합니다.

      IPv6의 경우 라우팅 추가를 선택하고 대상 위치로 `::/0`을, 대상으로 인터넷 게이트웨이를 사용합니다.
   c. 라우팅 저장을 선택합니다.

   또한 IPv6 엔드포인트에 연결하려는 경우 클라이언트 IPv6 주소 범위가 DB 인스턴스에 연결할 수 있는 권한이 있는지 확인합니다.

자세한 내용은 VPC에서 DB 인스턴스를 사용한 작업 단원을 참조하세요.

엔진별 연결 문제는 다음 주제를 참조하십시오.

- SQL Server DB 인스턴스에 대한 연결 문제 해결
- Oracle DB 인스턴스에 대한 연결 문제 해결
- RDS for PostgreSQL 인스턴스에 대한 연결 문제 해결
- 최대 MySQL 및 MariaDB 연결 수

DB 인스턴스 연결 테스트

공통 Linux 또는 Microsoft Windows 도구를 사용하여 DB 인스턴스에 대한 연결을 테스트할 수 있습니다.

Linux 또는 Unix 터미널에서 다음을 입력하여 연결을 테스트할 수 있습니다. *DB-instance-endpoint*를 엔드포인트로 대체하고 *port*를 DB 인스턴스의 포트로 대체합니다.

```
nc -zv DB-instance-endpoint port
```

예를 들어, 다음은 샘플 명령과 반환 값을 보여 줍니다.

```
nc -zv postgresql1.c6c8mn7fake0.us-west-2.rds.amazonaws.com 8299

  Connection to postgresql1.c6c8mn7fake0.us-west-2.rds.amazonaws.com 8299 port [tcp/
vvr-data] succeeded!
```

Windows 사용자는 Telnet을 사용하여 DB 인스턴스에 대한 연결을 테스트할 수 있습니다. Telnet 작업은 연결 테스트 이외의 목적으로는 지원되지 않습니다. 연결에 성공한 경우 이 작업을 수행할 때 아무런 메시지도 반환되지 않습니다. 연결에 실패한 경우 다음과 같은 오류 메시지가 수신됩니다.

```
C:\>telnet sg-postgresql1.c6c8mntfake0.us-west-2.rds.amazonaws.com 819

  Connecting To sg-postgresql1.c6c8mntfake0.us-west-2.rds.amazonaws.com...Could not
 open
  connection to the host, on port 819: Connect failed
```

Telnet 작업으로 성공 메시지가 반환되면 보안 그룹이 올바로 구성된 것입니다.

> **Note**
>
> Amazon RDS는 ping을 포함하여 ICMP(Internet Control Message Protocol) 트래픽을 수락하지 않습니다.

연결 인증 문제 해결

경우에 따라서는 DB 인스턴스에 연결할 수 있지만 인증 오류가 발생하는 경우가 있습니다. 이러한 경우 DB 인스턴스에 대한 마스터 사용자 암호를 재설정하는 것이 좋습니다. RDS 인스턴스를 수정하여 이 작업을 수행할 수 있습니다.

DB 인스턴스 변경에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

Amazon RDS 보안 문제

보안 문제를 방지하려면 사용자 계정에 AWS 계정 루트 사용자 이메일 주소와 암호를 사용하지 마세요. 모범 사례에 따라 루트 사용자로 사용자를 생성하고 이런 사용자를 DB 사용자 계정에 할당하는 것이 좋습니다. 필요한 경우 루트 사용자로 다른 사용자 계정을 만들 수도 있습니다.

사용자 생성에 대한 자세한 정보는 AWS 계정에서 IAM 사용자 생성을 참조하세요. AWS IAM Identity Center에서의 사용자 생성에 대한 자세한 정보는 IAM Identity Center에서 ID 관리 섹션을 참조하십시오.

오류 메시지 "계정 속성을 불러오지 못했습니다. 일부 콘솔 기능이 손상되었을 수 있습니다."

여러 가지 이유로 이 오류가 발생할 수 있습니다. 계정에 권한이 없거나 계정이 제대로 설정되지 않았기 때문일 수 있습니다. 새 계정인 경우 계정이 준비되기까지 충분한 시간이 지나지 않았을 수도 있습니다. 기존 계정인 경우 DB 인스턴스 생성과 같은 특정 작업을 수행하기 위한 액세스 정책에 권한이 없을 수 있습니다. 이 문제를 해결하려면 관리자가 필요한 역할을 해당 계정에 제공해야 합니다. 자세한 내용은 IAM 설명서를 참조하세요.

호환되지 않는 네트워크 상태 문제 해결

호환되지 않는 네트워크 상태란 데이터베이스 수준에서는 데이터베이스에 계속 액세스할 수 있지만 수정하거나 재부팅할 수는 없는 상태를 의미합니다.

원인

DB 인스턴스의 네트워크가 호환되지 않는 상태는 다음 작업 중 하나로 인해 발생할 수 있습니다.

- DB 인스턴스 클래스 수정
- 다중 AZ DB 클러스터 배포를 사용하도록 DB 인스턴스 수정
- 유지 관리 이벤트로 인해 호스트 교체
- 대체 DB 인스턴스 시작
- 스냅샷 백업에서 복원
- 중지된 DB 인스턴스 시작

해결 방법

start-db-instance 명령 사용

네트워크가 호환되지 않는 상태에 있는 데이터베이스를 수정하려면 다음 지침을 따릅니다.

1. https://console.aws.amazon.com/rds/ 페이지를 열고 탐색 창에서 데이터베이스를 선택합니다.
2. 호환되지 않는 네트워크 상태에 있는 DB 인스턴스를 선택하고 연결 및 보안 탭에서 DB 인스턴스 식별자, VPC ID, 서브넷 ID를 메모해 둡니다.
3. AWS CLI를 사용하여 `start-db-instance` 명령을 실행합니다. 값은 `--db-instance-identifier`로 지정합니다.

> **Note**
>
> 데이터베이스가 호환되지 않는 모드일 때 이 명령을 실행하면 가동 중지 시간이 발생할 수 있습니다.
> `start-db-instance` 명령을 실행해도 RDS for SQL Server DB 인스턴스에서는 이 문제가 해결되지 않습니다.

명령이 성공적으로 실행되면 데이터베이스 상태가 사용 가능으로 변경됩니다.

데이터베이스가 재시작되는 경우 호환되지 않는 네트워크 상태로 이동하기 전에 DB 인스턴스가 인스턴스에서 마지막 작업을 실행했기 때문일 수 있습니다. 이로 인해 인스턴스가 호환되지 않는 네트워크 상태로 돌아갈 수 있습니다.

`start-db-instance` 명령이 실패하거나 인스턴스가 호환되지 않는 네트워크 상태로 다시 전환되면 RDS 콘솔에서 데이터베이스 페이지를 열고 데이터베이스를 선택합니다. 로그 및 이벤트 섹션으로 이동합니다. 최근 이벤트 섹션에는 다음으로 수행해야 할 추가 해결 단계가 표시됩니다. 메시지는 다음과 같이 분류됩니다.

- 내부 리소스 확인: 내부 리소스에 문제가 있을 수 있습니다.
- DNS 확인: VPC 콘솔에서 VPC의 DNS 확인 및 호스트 이름을 확인합니다.
- ENI 확인: 데이터베이스의 탄력적 네트워크 인터페이스(ENI)가 없을 수 있습니다.
- 게이트웨이 확인: 공개적으로 사용 가능한 데이터베이스의 인터넷 게이트웨이는 VPC에 연결되어 있지 않습니다.
- IP 확인: 서브넷에는 무료 IP 주소가 없습니다.

- 보안 그룹 확인: 데이터베이스와 연결된 보안 그룹이 없거나 보안 그룹이 유효하지 않습니다.
- 서브넷 확인: DB 서브넷 그룹에 유효한 서브넷이 없거나 서브넷에 문제가 있습니다.
- VPC 확인: 데이터베이스와 연결된 VPC가 유효하지 않습니다.

특정 시점으로 복원 수행

데이터베이스가 호환되지 않는 네트워크 상태가 되는 경우에 대비하여 백업(스냅샷 또는 논리적)을 보관하는 것이 좋습니다. 백업 소개 섹션을 참조하세요. 자동 백업을 사용 설정한 경우 데이터베이스에 대한 모든 쓰기를 일시적으로 중지하고 시점 복구를 수행할 수 있습니다.

> **Note**
>
> 인스턴스가 호환되지 않는 네트워크 상태가 된 후에는 DB 인스턴스에 액세스하여 논리적 백업을 수행하지 못할 수 있습니다.

자동 백업을 사용 설정하지 않았다면 새 DB 인스턴스를 만드세요. 그런 다음 AWS Database Migration Service(AWS DMS) 또는 백업 및 복원 도구를 사용하여 데이터를 마이그레이션합니다.

이 방법으로도 문제가 해결되지 않으면 지원로 문의하여 추가 지원을 받으세요.

DB 인스턴스 소유자 암호 재설정

DB 인스턴스 가 잠긴 잠긴 경우 마스터 사용자로 로그인할 수 있습니다. 그런 다음 다른 관리 사용자 또는 역할에 대한 자격 증명을 재설정할 수 있습니다. 마스터 사용자로 로그인할 수 없는 경우 AWS 계정 소유자가 마스터 사용자 암호를 재설정할 수 있습니다. 재설정해야 할 관리 계정 또는 역할에 대한 자세한 내용은 마스터 사용자 계정 권한 단원을 참조하십시오.

Amazon RDS 콘솔, AWS CLI 명령 modify-db-instance 또는 ModifyDBInstance API 작업을 사용하여 DB 인스턴스 암호를 변경할 수 있습니다. DB 인스턴스 변경에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

Amazon RDS DB 인스턴스 중단 또는 재부팅

DB 인스턴스가 재부팅되면 DB 인스턴스가 중단될 수 있습니다. 이는 DB 인스턴스가 액세스할 수 없는 상태로 전환되거나 데이터베이스가 다시 시작될 때도 발생할 수 있습니다. DB 인스턴스를 수동으로

재부팅하면 재부팅이 수행될 수 있습니다. DB 인스턴스 설정을 변경하여 이 변경 사항을 적용하기 위해 재부팅해야 할 때 재부팅이 수행될 수 있습니다.

DB 인스턴스 재부팅은 설정을 변경하여 재부팅해야 할 때 또는 수동으로 재부팅할 때 이 발생합니다. 설정을 변경하고 변경 사항을 즉시 적용할 것을 요청하는 경우 재부팅이 수행될 수 있습니다. 또는 DB 인스턴스의 유지 관리 기간 중에 재부팅이 수행될 수 있습니다.

다음 중 한 가지가 발생할 때는 그 즉시 DB 인스턴스가 재부팅됩니다.

- DB 인스턴스에 대한 백업 보존 기간을 0에서 0이 아닌 값으로 변경하거나 0이 아닌 값에서 0으로 변경합니다. 그런 다음 즉시 적용을 `true`로 설정합니다.
- DB 인스턴스 클래스를 변경하고 즉시 적용이 `true`로 설정된 경우
- 스토리지 유형을 마그네틱(표준)에서 범용(SSD) 또는 프로비저닝된 IOPS(SSD)로 변경하거나 프로비저닝된 IOPS(SSD) 또는 범용(SSD)에서 마그네틱(표준)으로 변경합니다.

유지 관리 기간 중에 다음 중 한 가지가 발생할 때 DB 인스턴스가 재부팅됩니다.

- DB 인스턴스에 대한 백업 보존 기간을 0에서 0이 아닌 값으로 변경하거나 0이 아닌 값에서 0으로 변경하고 즉시 적용이 `false`로 설정된 경우
- DB 인스턴스 클래스를 변경하고 즉시 적용이 `false`로 설정된 경우

DB 파라미터 그룹에서 정적 파라미터를 변경하면 해당 변경 사항은 파라미터 그룹과 연결된 DB 인스턴스를 재부팅해야 적용됩니다. 변경 작업을 수행하려면 수동 재부팅이 필요합니다. 유지 관리 기간 중에는 DB 인스턴스가 자동으로 재부팅되지 않습니다.

DB 인스턴스 작업과 Apply Immediately(즉시 적용) 값 설정이 미치는 효과를 보여주는 표를 보려면 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.

Amazon RDS DB 파라미터 변경 사항이 적용 안 됨

경우에 따라 DB 파라미터 그룹에서 파라미터를 변경할 수 있지만 해당 변경 사항이 적용되지 않을 수 있습니다. 이 경우 DB 파라미터 그룹과 연결된 DB 인스턴스를 재부팅해야 할 수 있습니다. 동적 파라미터를 변경하면 해당 변경 사항이 즉시 적용됩니다. 정적 파라미터를 변경하면 해당 변경 사항은 파라미터 그룹과 연결된 DB 인스턴스를 재부팅해야 적용됩니다.

RDS 콘솔을 사용하여 DB 인스턴스를 재부팅할 수 있습니다. 또는 `RebootDBInstance` API 작업을 명시적으로 호출할 수 있습니다. DB 인스턴스를 다중 AZ 배포로 생성한 경우에는 장애 조치 없이 재부

팅할 수 있습니다. 정적 파라미터 변경 후 연결된 DB 인스턴스를 재부팅하도록 하면 잘못된 파라미터 구성이 API 호출에 영향을 주는 위험을 완화할 수 있습니다. 예를 들면 ModifyDBInstance를 호출하여 DB 인스턴스 클래스를 변경하는 경우입니다. 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정 단원을 참조하십시오.

Amazon RDS DB 인스턴스 스토리지 부족

DB 인스턴스에 스토리지 공간이 부족할 경우 DB 인스턴스를 더 이상 사용하지 못할 수 있습니다. CloudWatch에 게시되는 FreeStorageSpace 지표를 계속 모니터링하여 DB 인스턴스에 충분한 스토리지 여유 공간이 있는지 확인해야 합니다.

데이터베이스 인스턴스의 스토리지가 부족하면 상태가 storage-full로 변경됩니다. 예를 들어, 스토리지를 모두 소진한 DB 인스턴스에 대해 DescribeDBInstances API 작업을 호출하면 다음과 같이 출력됩니다.

```
aws rds describe-db-instances --db-instance-identifier mydbinstance

DBINSTANCE  mydbinstance  2009-12-22T23:06:11.915Z  db.m5.large  mysql8.0  50  sa
storage-full  mydbinstance.clla4j4jgyph.us-east-1.rds.amazonaws.com  3306
us-east-1b  3
 SECGROUP  default  active
 PARAMGRP  default.mysql8.0  in-sync
```

이 상황에서 벗어나려면 ModifyDBInstance API 작업이나 다음 AWS CLI 명령을 사용하여 인스턴스에 스토리지 공간을 더 추가하십시오.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --allocated-storage 60 \
    --apply-immediately
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --allocated-storage 60 ^
    --apply-immediately
```

```
DBINSTANCE  mydbinstance  2009-12-22T23:06:11.915Z  db.m5.large  mysql8.0  50  sa
storage-full  mydbinstance.clla4j4jgyph.us-east-1.rds.amazonaws.com  3306
us-east-1b  3  60
 SECGROUP  default  active
 PARAMGRP  default.mysql8.0  in-sync
```

이제 DB 인스턴스를 설명할 때 DB 인스턴스의 상태가 `modifying`으로 변경되어 스토리지가 확장되고 있는 중임을 알 수 있습니다.

```
aws rds describe-db-instances --db-instance-identifier mydbinstance
```

```
DBINSTANCE  mydbinstance  2009-12-22T23:06:11.915Z  db.m5.large  mysql8.0  50  sa
modifying  mydbinstance.clla4j4jgyph.us-east-1.rds.amazonaws.com
3306  us-east-1b  3  60
 SECGROUP  default  active
 PARAMGRP  default.mysql8.0  in-sync
```

스토리지 확장이 완료되면 DB 인스턴스 상태가 `available`로 변경됩니다.

```
aws rds describe-db-instances --db-instance-identifier mydbinstance
```

```
DBINSTANCE  mydbinstance  2009-12-22T23:06:11.915Z  db.m5.large  mysql8.0  60  sa
available  mydbinstance.clla4j4jgyph.us-east-1.rds.amazonaws.com  3306
us-east-1b  3
 SECGROUP  default  active
 PARAMGRP  default.mysql8.0  in-sync
```

`DescribeEvents` 작업을 사용하면 스토리지 공간이 소진될 때 알림을 받을 수 있습니다. 예를 들어, 이 시나리오에서 이러한 작업 후에 `DescribeEvents` 호출을 실행하면 다음과 같이 출력됩니다.

```
aws rds describe-events --source-type db-instance --source-identifier mydbinstance
```

```
2009-12-22T23:44:14.374Z  mydbinstance  Allocated storage has been exhausted db-
instance
2009-12-23T00:14:02.737Z  mydbinstance  Applying modification to allocated storage db-
instance
2009-12-23T00:31:54.764Z  mydbinstance  Finished applying modification to allocated
 storage
```

Amazon RDS 사용 가능한 DB 인스턴스 부족

DB 인스턴스를 생성, 시작 또는 수정하려고 시도하면 `InsufficientDBInstanceCapacity` 오류가 반환될 수 있습니다. DB 스냅샷에서 DB 인스턴스를 복원하려고 시도할 때도 오류가 반환될 수 있습니다. 이 오류가 반환되는 일반적인 원인은 특정 DB 인스턴스 클래스를 요청된 가용 영역에서 사용할 수 없기 때문입니다. 문제 해결을 위해 다음 중 하나를 시도할 수 있습니다.

- 다른 DB 인스턴스 클래스에서 요청을 재시도합니다.
- 다른 가용 영역에서 요청을 재시도합니다.
- 명시적인 가용 영역을 지정하지 않고 요청을 재시도합니다.

Amazon EC2에서 인스턴스 용량 문제를 해결하는 방법은 Amazon Elastic Compute Cloud 사용 설명서의 불충분한 인스턴스 용량을 참조하십시오.

DB 인스턴스 수정에 대한 자세한 내용은 Amazon RDS DB 인스턴스 수정 단원을 참조하세요.

Amazon RDS의 여유 메모리 부족

여유 메모리는 데이터베이스 엔진에서 사용할 수 있는 DB 인스턴스의 총 랜덤 액세스 메모리(RAM)입니다. 이는 여유 운영 체제(OS) 메모리와 사용 가능한 버퍼 및 페이지 캐시 메모리의 합계입니다. 데이터베이스 엔진이 호스트의 메모리 대부분을 사용하지만, OS 프로세스도 일부 RAM을 사용합니다. 현재 데이터베이스 엔진에 할당되거나 OS 프로세스에서 사용하는 메모리는 여유 메모리에 포함되지 않습니다. 데이터베이스 엔진의 메모리가 부족하면 DB 인스턴스는 버퍼링 및 캐싱에 일반적으로 사용되는 임시 공간을 이용하게 됩니다. 앞서 언급했듯이 이 임시 공간은 여유 메모리에 포함됩니다.

Amazon CloudWatch의 `FreeableMemory` 지표를 사용하여 여유 메모리를 모니터링할 수 있습니다. 자세한 내용은 Amazon RDS의 모니터링 도구 단원을 참조하십시오.

DB 인스턴스에서 사용 가능한 메모리가 계속해서 부족하거나 스왑 공간을 사용하는 경우 더 큰 DB 인스턴스 클래스로 스케일 업하는 것을 고려하세요. 자세한 내용은 DB 인스턴스 클래스 단원을 참조하십시오.

메모리 설정을 변경할 수도 있습니다. 예를 들어, RDS for MySQL에서는 `innodb_buffer_pool_size` 파라미터의 크기를 조정할 수 있습니다. 이 파라미터는 기본적으로 실제 메모리의 75%로 설정됩니다. MySQL 문제 해결 팁을 자세히 알아보려면 Amazon RDS for MySQL 데이터베이스에서 여유 메모리가 부족한 문제를 어떻게 해결할 수 있습니까?를 참조하세요.

MySQL 및 MariaDB 문제

MySQL 및 MariaDB DB 인스턴스의 문제를 진단하고 수정할 수 있습니다.

주제

- 최대 MySQL 및 MariaDB 연결 수
- 메모리 제한에 대한 호환되지 않는 파라미터 상태 진단 및 해결
- 읽기 전용 복제본 간 지연 문제 진단 및 해결
- MySQL 또는 MariaDB 읽기 복제 오류 진단 및 해결
- 이진 로깅이 활성화된 상태에서 트리거를 생성하려면 SUPER 권한 필요
- 특정 시점으로 복원 오류 진단 및 해결
- 복제 중지 오류
- 읽기 전용 복제본 만들기 실패 또는 치명적 오류 1236으로 복제 중단
- 읽기 전용 복제본 복제가 메타데이터 구조를 초기화하지 못함

최대 MySQL 및 MariaDB 연결 수

RDS for MySQL 또는 RDS for MariaDB DB 인스턴스에 허용되는 최대 연결 수는 DB 인스턴스 클래스에 사용 가능한 메모리의 양에 따라 결정됩니다. 사용 가능한 메모리가 많은 DB 인스턴스 클래스는 가능한 연결 수가 더 많아집니다. DB 인스턴스 클래스에 대한 자세한 내용은 DB 인스턴스 클래스 단원을 참조하십시오.

DB 인스턴스의 연결 제한은 기본적으로 DB 인스턴스 클래스의 최대값으로 설정됩니다. 동시 연결 수를 허용된 최대 연결 수까지 임의의 값으로 제한할 수 있습니다. DB 인스턴스의 파라미터 그룹에서 `max_connections` 파라미터를 사용합니다. 자세한 내용은 최대 데이터베이스 연결 수 및 Amazon RDS의 파라미터 그룹 단원을 참조하십시오.

다음 쿼리를 실행하여 MySQL 또는 MariaDB DB 인스턴스에 허용되는 최대 연결 수를 검색할 수 있습니다.

```
SELECT @@max_connections;
```

다음 쿼리를 실행하여 MySQL 또는 MariaDB DB 인스턴스에 대한 활성 연결 수를 검색할 수 있습니다.

```
SHOW STATUS WHERE `variable_name` = 'Threads_connected';
```

메모리 제한에 대한 호환되지 않는 파라미터 상태 진단 및 해결

다음 조건이 충족되는 경우 MariaDB 또는 MySQL DB 인스턴스가 메모리 제한에 대해 파라미터 호환 장애 상태가 될 수 있습니다.

- DB 인스턴스 상태가 사용 가능일 때 DB 인스턴스가 한 시간에 세 번 이상 또는 하루에 다섯 번 이상 다시 시작됩니다.
- 유지 관리 작업 또는 모니터링 프로세스에서 DB 인스턴스를 다시 시작할 수 없기 때문에 DB 인스턴스를 다시 시작하려는 시도가 실패합니다.
- DB 인스턴스의 잠재적인 메모리 사용량이 DB 인스턴스 클래스에 할당된 메모리의 1.2배를 초과합니다.

DB 인스턴스가 1시간에 세 번째로 다시 시작되거나 하루에 다섯 번째로 다시 시작되면 메모리 사용량 검사를 수행합니다. 이 검사에서는 DB 인스턴스의 잠재적인 메모리 사용량을 계산합니다. 이 계산에서 반환되는 값은 다음 값의 합계입니다.

- 값 1 – 다음 파라미터의 합계입니다.
  - `innodb_additional_mem_pool_size`
  - `innodb_buffer_pool_size`

    `innodb_buffer_pool_size`의 값을 수정할 수 있습니다. 그러나 값이 입력한 내용과 항상 일치하지는 않습니다. 여러 가지 이유로 이 불일치가 발생합니다. 첫째, DB 인스턴스가 마이크로 DB 인스턴스인 경우 기본값을 무시하고 256MB로 설정됩니다. 자세한 내용은 innodb_buffer_pool_size 재정의 단원을 참조하십시오.

    둘째, 호스트 관리자, 엔진, 운영 체제, 커널을 위해 DB 인스턴스에 500MB의 메모리가 예약됩니다.

    마지막으로, 단위로 나누어 `innodb_buffer_pool_size`가 최적화됩니다. 호스트 관리자는 해당 단위 중 가장 가까운 배수로 반올림합니다. 단위는 `innodb_buffer_pool_chunk_size`와 `innodb_buffer_pool_instances`를 곱하여 계산합니다. 자세한 내용은 MySQL 설명서의 Configuring InnoDB Buffer Pool Size를 참조하세요.

    `innodb_buffer_pool_instances`의 기본값은 `innodb_buffer_pool_size`가 1GB 미만인 경우를 제외하고 8입니다. `innodb_buffer_pool_size`가 1GB 미만인 경우 `innodb_buffer_pool_instances`의 기본값은 1입니다. `innodb_buffer_pool_chunk_size`의 기본값은 128MB입니다.

  - innodb_log_buffer_size
  - key_buffer_size
  - query_cache_size(MySQL 버전 5.7만 해당)
  - tmp_table_size
- 값 2 – max_connections 파라미터에 다음 파라미터의 합을 곱한 값입니다.
  - binlog_cache_size
  - join_buffer_size
  - read_buffer_size
  - read_rnd_buffer_size
  - sort_buffer_size
  - thread_stack
- 값 3 – performance_schema 파라미터가 활성화된 경우 max_connections 파라미터에 429498을 곱합니다.

  performance_schema 파라미터가 비활성화된 경우 이 값은 0입니다.

따라서 계산에서 반환되는 값은 다음과 같습니다.

```
Value 1 + Value 2 + Value 3
```

이 값이 DB 인스턴스에 사용되는 DB 인스턴스 클래스에 할당된 메모리의 1.2배를 초과하면 DB 인스턴스가 파라미터 호환 장애 상태가 됩니다. DB 인스턴스 클래스에 할당된 메모리에 대한 자세한 내용은 에 대한 DB 인스턴스 클래스의 하드웨어 사양을 참조하십시오.

이 계산에서는 max_connections 파라미터의 값에 여러 파라미터의 합을 곱합니다. max_connections 파라미터가 큰 값으로 설정되어 있는 경우, 검사에서 DB 인스턴스의 잠재적인 메모리 사용량의 값이 지나치게 높게 반환될 수 있습니다. 이 경우 max_connections 파라미터의 값을 낮추는 것이 좋습니다.

이 문제를 해결하려면 다음 단계를 수행합니다.

1. DB 인스턴스와 연결된 DB 파라미터 그룹의 메모리 파라미터를 조정합니다. 잠재적인 메모리 사용량이 DB 인스턴스 클래스에 할당된 메모리의 1.2배보다 낮도록 조정합니다.

   파라미터 설정에 대한 자세한 내용은 Amazon RDS에서 DB 파라미터 그룹의 파라미터 수정을 참조하세요.

2. DB 인스턴스를 다시 시작합니다.

   파라미터 설정에 대한 자세한 내용은 이전에 중지된 Amazon RDS DB 인스턴스 시작을 참조하세요.

읽기 전용 복제본 간 지연 문제 진단 및 해결

MySQL 또는 MariaDB 읽기 전용 복제본을 생성하고 읽기 전용 복제본을 사용할 수 있게 된 후 Amazon RDS는 우선 읽기 전용 복제본 생성 작업이 시작된 시간부터 원본 DB 인스턴스에서 변경된 사항을 복제합니다. 이 단계에서 읽기 전용 복제본의 복제 지연 시간은 0보다 큽니다. Amazon RDS `ReplicaLag` 지표를 보고 Amazon CloudWatch에서 이 지연 시간을 모니터링할 수 있습니다.

`ReplicaLag` 지표는 MariaDB 또는 MySQL `Seconds_Behind_Master` 명령의 `SHOW REPLICA STATUS` 필드 값을 보고합니다. 자세한 내용은 MySQL 설명서의 SHOW REPLICA STATUS 문을 참조하세요.

`ReplicaLag` 지표가 0에 도달하면 복제본이 원본 DB 인스턴스를 따라잡은 것입니다. `ReplicaLag` 메트릭이 -1을 반환하는 경우에는 복제가 활성 상태가 아닐 수 있습니다. 복제 오류 문제를 해결하는 방법은 MySQL 또는 MariaDB 읽기 복제 오류 진단 및 해결 단원을 참조하십시오. `ReplicaLag` 값이 -1인 경우 `Seconds_Behind_Master` 값을 결정할 수 없거나 이 값이 `NULL`이라는 의미일 수도 있습니다.

> **Note**
>
> 이전 버전의 MariaDB에는 `SHOW SLAVE STATUS` 대신 `SHOW REPLICA STATUS`가 사용되었습니다. 10.5 이하의 MariaDB 버전을 사용하는 경우 `SHOW SLAVE STATUS`를 사용합니다.

네트워크가 중단된 기간 동안이나 유지 관리 기간 중에 패치가 적용될 때 `ReplicaLag` 지표는 -1을 반환합니다. 이 경우에는 네트워크 연결이 복원되거나 유지 관리 기간이 종료되기를 기다린 후 `ReplicaLag` 지표를 다시 확인합니다.

MySQL 및 MariaDB 읽기 전용 복제 기술은 비동기식입니다. 따라서 소스 DB 인스턴스의 `BinLogDiskUsage` 지표와 읽기 전용 복제본의 `ReplicaLag` 지표가 가끔 증가할 수도 있습니다. 예를 들어, 원본 DB 인스턴스에 대량의 쓰기 작업이 병렬로 발생하는 경우를 생각해 보십시오. 동시에 읽기 전용 복제본에 대한 쓰기 작업은 단일 I/O 스레드를 사용하여 직렬화됩니다. 이러한 상황으로 인해 원본 인스턴스와 읽기 전용 복제본 사이에 지연 시간이 발생할 수 있습니다.

읽기 전용 복제본과 MySQL에 대한 자세한 내용은 MySQL 설명서의 복제 구현 세부 정보를 참조하십시오. 읽기 전용 복제본과 MariaDB에 대한 자세한 내용은 MariaDB 설명서의 복제 개요를 참조하세요.

다음을 수행하여 원본 DB 인스턴스에 대한 업데이트와 읽기 전용 복제본에 대한 후속 업데이트 사이의 지연 시간을 줄일 수 있습니다.

- 읽기 전용 복제본의 DB 인스턴스 클래스를 원본 DB 인스턴스와 비슷한 스토리지 크기로 설정합니다.
- 원본 DB 인스턴스와 읽기 전용 복제본에 사용되는 DB 파라미터 그룹의 파라미터 설정이 호환되는지 확인합니다. 자세한 정보와 예는 다음 섹션에서 `max_allowed_packet` 파라미터에 대해 설명한 내용을 참조하십시오.
- 쿼리 캐시를 비활성화합니다. 자주 수정되는 테이블의 경우, 쿼리 캐시를 사용하면 캐시가 자주 잠기고 새로 고쳐지기 때문에 복제 지연이 늘어날 수 있습니다. 이럴 경우 쿼리 캐시를 비활성화하면 복제 지연이 줄어드는 효과를 볼 수도 있습니다. DB 인스턴스에 대한 DB 파라미터 그룹에서 `query_cache_type parameter`를 0으로 설정하여 쿼리 캐시를 비활성화할 수 있습니다. 쿼리 캐시에 대한 자세한 정보는 쿼리 캐시 구성을 참조하십시오.
- MySQL 또는 MariaDB용 InnoDB의 읽기 전용 복제본에서 버퍼 풀을 워밍합니다. 예를 들어, 자주 업데이트되는 작은 테이블 집합이 있고 InnoDB 또는 XtraDB 테이블 스키마를 사용하고 있다고 가정해 보십시오. 이 경우 읽기 전용 복제본에 해당 테이블을 덤프합니다. 그러면 데이터베이스 엔진이 디스크에서 해당 테이블의 행을 전체적으로 검사한 다음 버퍼 풀에 캐시합니다. 이 방법을 사용하면 복제본 지연 시간을 줄일 수 있습니다. 다음은 그 한 예입니다.

  대상 LinuxmacOS, 또는Unix:

```
PROMPT> mysqldump \
    -h <endpoint> \
    --port=<port> \
    -u=<username> \
    -p <password> \
    database_name table1 table2 > /dev/null
```

  Windows의 경우:

```
PROMPT> mysqldump ^
    -h <endpoint> ^
    --port=<port> ^
    -u=<username> ^
    -p <password> ^
    database_name table1 table2 > /dev/null
```

MySQL 또는 MariaDB 읽기 복제 오류 진단 및 해결

Amazon RDS는 읽기 전용 복제본의 복제 상태를 모니터링합니다. RDS는 어떤 이유로든 복제가 중지되는 경우 읽기 전용 복제본 인스턴스의 복제 상태 필드를 `Error`로 업데이트합니다. 복제 오류 필드를 확인하여 MySQL 또는 MariaDB 엔진에서 발생한 관련 오류의 세부 정보를 검토할 수 있습니다. RDS-EVENT-0045, RDS-EVENT-0046 및 RDS-EVENT-0057을 포함하여 읽기 전용 복제본의 상태를 나타내는 이벤트도 생성됩니다. 이벤트와 이벤트 구독에 대한 자세한 내용은 Amazon RDS 이벤트 알림 작업 단원을 참조하십시오. MySQL 오류 메시지가 반환되는 경우 MySQL 오류 메시지 설명서에 있는 오류를 확인하세요. MariaDB 오류 메시지가 반환되는 경우 MariaDB 오류 메시지 설명서에 있는 오류를 확인하세요.

복제 오류의 원인이 되는 공통적인 상황은 다음과 같습니다.

- 읽기 전용 복제본에 대한 `max_allowed_packet` 파라미터의 값은 원본 DB 인스턴스에 대한 `max_allowed_packet` 파라미터보다 작습니다.

  `max_allowed_packet` 파라미터는 DB 파라미터 그룹에서 설정할 수 있는 사용자 지정 파라미터입니다. `max_allowed_packet` 파라미터는 데이터베이스에서 실행할 수 있는 데이터 조작 언어(DML)의 최대 크기를 지정하는 데 사용됩니다. 경우에 따라서는 원본 DB 인스턴스의 `max_allowed_packet` 값이 읽기 전용 복제본에 대한 `max_allowed_packet` 값보다 클 수도 있습니다. 이러한 경우 복제 프로세스에서 오류가 발생하여 복제가 중지될 수도 있습니다. 가장 흔한 오류는 `packet bigger than 'max_allowed_packet' bytes`입니다. 원본 및 읽기 전용 복제본이 동일한 `max_allowed_packet` 파라미터 값을 가진 DB 파라미터 그룹을 사용하도록 하여 이 오류를 해결할 수 있습니다.
- 읽기 전용 복제본의 테이블에 쓰기 작업 중일 때. 읽기 전용 복제본에서 인덱스를 생성할 경우 `read_only` 파라미터를 0으로 설정하여 인덱스를 생성해야 합니다. 읽기 전용 복제본에 있는 테이블에 데이터를 쓰면 복제가 중단될 수 있습니다.
- MyISAM과 같은 비트랜잭션 스토리지 엔진 사용. 읽기 전용 복제본에는 트랜잭션 스토리지 엔진이 필요합니다. 복제는 MySQL 또는 MariaDB용 InnoDB에 대해서만 지원됩니다.

  다음 명령으로 MyISAM 테이블을 InnoDB로 변환할 수 있습니다.

  ```
  alter table <schema>.<table_name> engine=innodb;
  ```
- `SYSDATE()`와 같이 안전하지 않은 비결정적 쿼리를 사용하는 경우. 자세한 내용은 MySQL 설명서의 이진 로깅에서 안전한 문과 안전하지 않은 문 결정을 참조하십시오.

다음 단계를 통해 복제 오류를 해결할 수 있습니다.

- 논리적 오류가 발생했는데 이 오류를 건너뛰어도 안전할 경우에는 RDS for MySQL에 대한 현재 복제 오류 건너뛰기에 설명되어 있는 단계를 따르십시오. MySQL 또는 MariaDB DB 인스턴스에서는 `mysql_rds_skip_repl_error` 프로시저를 포함한 버전이 실행 중이어야 합니다. 자세한 내용은 mysql.rds_skip_repl_error 단원을 참조하십시오.
- 이진 로그(binlog) 위치 문제가 발생하는 경우 mysql.rds_next_source_log(RDS for MySQL 메이저 버전 8.4 이상) 또는 mysql.rds_next_master_log(RDS for MariaDB 및 RDS for MySQL 메이저 버전 8.0 이하) 명령으로 복제본 재생 위치를 변경할 수 있습니다.
- 높은 DML 로드로 인해 일시적인 성능 문제가 발생할 수 있습니다. 그러한 경우 읽기 전용 복제본의 DB 파라미터 그룹에서 `innodb_flush_log_at_trx_commit` 파라미터를 2로 설정할 수 있습니다. 그러면 일시적으로 원자성, 일관성, 격리성 및 내구성(ACID)이 감소하지만 읽기 전용 복제본이 변화를 따라잡는 데 도움이 될 수 있습니다.
- 읽기 전용 복제본을 삭제하고 동일한 DB 인스턴스 식별자를 사용하여 인스턴스를 생성할 수 있습니다. 이렇게 하면 엔드포인트는 이전 읽기 전용 복제본의 엔드포인트와 동일하게 유지됩니다.

복제 오류가 해결되면 Replication State가 replicating으로 변경됩니다. 자세한 내용은 MySQL 읽기 전용 복제본의 문제 해결 섹션을 참조하세요.

이진 로깅이 활성화된 상태에서 트리거를 생성하려면 SUPER 권한 필요

RDS for MySQL 또는 RDS for MariaDB DB 인스턴스에서 트리거 생성을 시도할 때 다음 오류가 발생할 수 있습니다.

```
"You do not have the SUPER privilege and binary logging is enabled"
```

이진 로깅이 활성화된 상태에서 트리거를 사용하려면 RDS for MySQL 및 RDS for MariaDB DB 인스턴스로 제한된 SUPER 권한이 필요합니다. `log_bin_trust_function_creators` 파라미터를 true로 설정하면 SUPER 권한 없이 이진 로깅이 활성화된 상태에서 트리거를 생성할 수 있습니다. `log_bin_trust_function_creators`를 true로 설정하려면 새 DB 파라미터 그룹을 생성하거나 기존 DB 파라미터 그룹을 수정해야 합니다.

이진 로깅이 활성화된 상태에서 RDS for MySQL 또는 RDS for MariaDB DB 인스턴스에서 트리거를 생성할 수 있도록 새 DB 파라미터 그룹을 생성할 수 있습니다. 그러기 위해서는 다음 CLI 명령을 사용합니다. 기존 파라미터 그룹을 수정하려면 2단계부터 시작합니다.

CLI를 사용하여 이진 로깅이 활성화된 상태에서 트리거를 허용하는 새 파라미터 그룹을 생성하려면

1. 새 파라미터 그룹을 생성해야 합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds create-db-parameter-group \
    --db-parameter-group-name allow-triggers \
    --db-parameter-group-family mysql8.0 \
    --description "parameter group allowing triggers"
```

Windows의 경우:

```
aws rds create-db-parameter-group ^
    --db-parameter-group-name allow-triggers ^
    --db-parameter-group-family mysql8.0 ^
    --description "parameter group allowing triggers"
```

2. DB 파라미터 그룹이 트리거를 허용하도록 수정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name allow-triggers \
    --parameters "ParameterName=log_bin_trust_function_creators,
 ParameterValue=true, ApplyMethod=pending-reboot"
```

Windows의 경우:

```
aws rds modify-db-parameter-group ^
    --db-parameter-group-name allow-triggers ^
    --parameters "ParameterName=log_bin_trust_function_creators,
 ParameterValue=true, ApplyMethod=pending-reboot"
```

3. DB 인스턴스가 새 DB 파라미터 그룹을 사용하도록 수정합니다.

대상 LinuxmacOS, 또는Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --db-parameter-group-name allow-triggers \
    --apply-immediately
```

Windows의 경우:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --db-parameter-group-name allow-triggers ^
    --apply-immediately
```

4. 변경 사항을 적용하려면 DB 인스턴스를 수동으로 재부팅합니다.

```
aws rds reboot-db-instance --db-instance-identifier mydbinstance
```

특정 시점으로 복원 오류 진단 및 해결

임시 테이블을 포함한 DB 인스턴스 복원

MySQL 또는 MariaDB DB 인스턴스의 특정 시점으로 복원(PITR)을 시도할 때 다음 오류가 발생할 수 있습니다.

```
Database instance could not be restored because there has been incompatible database
 activity for restore
functionality. Common examples of incompatible activity include using temporary tables,
 in-memory tables,
or using MyISAM tables. In this case, use of Temporary table was detected.
```

PITR은 DB 인스턴스를 특정 시점으로 복원하기 위해 MySQL 또는 MariaDB의 백업 스냅샷과 이진 로그(binlog)에 모두 의존합니다. binlog에서는 임시 테이블 정보를 신뢰할 수 없어 PITR 오류가 발생할 수 있습니다. MySQL 또는 MariaDB DB 인스턴스에서 임시 테이블을 사용하면 백업을 더 자주 수행하여 PITR 오류 발생 가능성을 줄일 수 있습니다. PITR 오류는 임시 테이블 생성과 다음 백업 스냅샷 생성 사이의 시간에 발생할 가능성이 가장 높습니다.

인 메모리 테이블을 포함한 DB 인스턴스 복원

인 메모리 테이블이 있는 데이터베이스를 복원할 때 문제가 발생할 수 있습니다. 인 메모리 테이블은 다시 시작하는 동안 제거됩니다. 따라서 재부팅 후 인 메모리 테이블이 비어 있을 수 있습니다. 인 메모리 테이블을 사용할 때는 다시 시작할 경우에 대비하여 빈 테이블을 처리하기 위한 솔루션을 설계하는 것이 좋습니다. 복제된 DB 인스턴스와 함께 인 메모리 테이블을 사용하는 경우 재시작 후 읽기 전용 복제본을 재생성해야 할 수 있습니다. 읽기 전용 복제본이 재부팅되고 비어 있는 인 메모리 테이블에서 데이터를 복구할 수 없는 경우 이 작업이 필요할 수 있습니다.

백업과 PITR에 대한 자세한 정보는 백업 소개 및 Amazon RDS에서 DB 인스턴스를 지정된 시간으로 복원 단원을 참조하십시오.

복제 중지 오류

`mysql.rds_skip_repl_error` 명령을 호출할 때 복제본이 중단되었거나 사용 중지되었다는 오류 메시지가 표시될 수 있습니다.

이 오류 메시지는 복제가 중지되었고 재시작할 수 없기 때문에 표시됩니다.

많은 수의 오류를 건너뛰어야 하는 경우, 복제 지연이 이진 로그 파일의 기본 보관 기간 이상으로 늘어날 수 있습니다. 이 경우 이진 로그 파일이 복제본에서 재생되기 전에 지워지기 때문에 치명적 오류가 발생할 수 있습니다. 이 제거는 복제를 중지시키며, 복제 오류를 건너뛰기 위해 더 이상 `mysql.rds_skip_repl_error` 명령을 호출할 수 없습니다.

이 문제는 복제 원본에서 이진 로그 파일이 보관되는 시간을 늘림으로써 완화할 수 있습니다. binlog 보관 시간을 늘린 후에 복제를 재시작하고 필요에 따라 `mysql.rds_skip_repl_error` 명령을 호출할 수 있습니다.

binlog 보관 시간을 설정하려면 mysql.rds_set_configuration 프로시저를 따르십시오. 'binlog 보관 시간' 구성 파라미터와 DB 클러스터에 binlog 파일을 보관할 시간(최대 720시간(30일))을 함께 지정합니다. 다음 예제에서는 binlog 파일의 보관 기간을 48시간으로 설정합니다.

```
CALL mysql.rds_set_configuration('binlog retention hours', 48);
```

읽기 전용 복제본 만들기 실패 또는 치명적 오류 1236으로 복제 중단

MySQL 또는 MariaDB의 DB 인스턴스에 대한 기본 파라미터 값을 변경한 후 다음과 같은 문제가 발생할 수 있습니다.

- DB 인스턴스의 읽기 전용 복제본을 생성할 수 없습니다.
- `fatal error 1236`로 인해 복제가 실패합니다.

MySQL 및 MariaDB DB 인스턴스에 대한 일부 기본 파라미터 값은 데이터베이스가 ACID를 준수하고 읽기 전용 복제본이 충돌 시 안전한지 확인하는 데 도움이 됩니다. 커밋되기 전에 이진 로그에 트랜잭션을 기록하여 각 커밋이 완전히 동기화되도록 하여 이를 수행합니다. 성능 향상을 위해 이러한 파라미터를 기본값에서 다른 값으로 변경하면 이진 로그에 기록되지 않은 트랜잭션의 경우 복제에 실패할 수 있습니다.

이 문제를 해결하려면 다음 파라미터 값을 설정하십시오.

- `sync_binlog = 1`

- innodb_support_xa = 1
- innodb_flush_log_at_trx_commit = 1

읽기 전용 복제본 복제가 메타데이터 구조를 초기화하지 못함

복제를 시작하려고 할 때 다음과 같은 오류 메시지가 표시됩니다.

```
Read Replica Replication Error - SQLError: 13124, reason: Replica failed to initialize
 applier metadata structure from the repository
```

이 오류는 복제본의 메타데이터 구조에 문제가 있는 경우에 발생합니다. 메타데이터 구조를 수정하려면 새 복제본을 생성해야 합니다.

향후 이 문제가 발생하지 않도록 하려면 다음 작업 중 하나를 수행합니다.

- 가능하면 복제본에서 멀티스레딩을 비활성화합니다. MySQL 8.0.27부터 멀티스레딩이 기본적으로 활성화됩니다.
- 복제본에 멀티스레딩을 사용해야 하는 경우 GTID 기반 복제를 사용하는 것이 좋습니다. 자세한 내용은 GTID 기반 복제 사용 단원을 참조하십시오.

백업 보존 기간을 0으로 설정할 수 없음

몇 가지 이유로 백업 보존 기간을 0으로 설정해야 할 수 있습니다. 예를 들어 보존 기간을 0으로 설정하면 자동 백업을 즉시 비활성화할 수 있습니다.

경우에 따라 값을 0으로 설정하고 보존 기간이 1에서 35 사이여야 한다는 메시지가 표시될 수 있습니다. 이러한 경우 인스턴스에 대해 읽기 전용 복제본을 설정하지 않았는지 확인합니다. 읽기 전용 복제본에는 읽기 전용 복제본 로그를 관리하기 위한 백업이 필요하므로 보존 기간을 0으로 설정할 수 없습니다.

Amazon RDS API 참조

Amazon RDS는 AWS Management Console 및 AWS Command Line Interface(AWS CLI) 외에 API도 제공합니다. API를 사용하여 Amazon RDS의 DB 인스턴스 및 기타 객체 관리 작업을 자동화할 수 있습니다.

- API 작업의 알파벳순 목록은 작업을 참조하십시오.
- 데이터 형식에 대한 알파벳순 목록은 데이터 형식을 참조하십시오.
- 공통 쿼리 파라미터 목록은 공통 파라미터를 참조하십시오.
- 오류 코드에 대한 설명은 공통 오류를 참조하십시오.

AWS CLI에 대한 자세한 내용은 Amazon RDS용 AWS Command Line Interface 참조를 참조하세요.

주제

- 쿼리 API 사용
- Amazon RDS에서 애플리케이션 문제 해결

쿼리 API 사용

다음 단원에서는 쿼리 API와 함께 사용되는 파라미터 및 요청 인증에 대해 간략하게 설명합니다.

쿼리 API의 작동 방식에 대한 일반적인 내용은 Amazon EC2 API Reference의 쿼리 요청을 참조하십시오.

쿼리 파라미터

HTTP 쿼리 기반 요청은 GET 또는 POST와 같은 HTTP 동사와 `Action` 쿼리 매개 변수를 사용하는 HTTP 요청입니다.

각 쿼리 요청은 인증 및 작업을 처리할 수 있도록 일부 공통 파라미터를 포함해야 합니다.

일부 작업은 파라미터의 목록을 허용합니다. 이러한 목록은 `param.`*`n`* 표기법을 사용하여 지정됩니다. *n*의 값은 1부터 시작하는 정수입니다.

Amazon RDS 리전과 엔드포인트에 대한 자세한 내용은 Amazon Web Services 일반 참조의 리전 및 엔드포인트 섹션에서 Amazon Relational Database Service(RDS)를 참조하세요.

쿼리 요청 인증

HTTPS를 통해서만 쿼리 요청을 보낼 수 있으며 모든 쿼리 요청에는 서명이 포함되어야 합니다. AWS 서명 버전 4 또는 서명 버전 2를 사용해야 합니다. 자세한 정보는 서명 버전 4 서명 프로세스 및 서명 버전 2 서명 프로세스를 참조하십시오.

Amazon RDS에서 애플리케이션 문제 해결

Amazon RDS는 Amazon RDS API와 상호 작용하는 동안 발생하는 문제를 해결할 때 도움이 되도록 구체적이고 서술적인 오류를 제공합니다.

주제

- 오류 검색
- 문제 해결 팁

Amazon RDS DB 인스턴스 문제 해결에 대한 자세한 내용은 Amazon RDS 문제 해결 단원을 참조하십시오.

오류 검색

일반적으로 사용자는 시간을 소비하여 결과를 처리하기 전에 애플리케이션이 먼저 해당 요청으로 오류가 발생되는지 여부를 확인하려고 합니다. 오류 발생 여부를 확인하는 가장 쉬운 방법은 Amazon RDS API의 응답에서 `Error` 노드를 찾는 것입니다.

XPath 구문은 `Error` 노드의 존재 여부를 검색하는 간단한 방법을 제공합니다. 또한 오류 코드와 메시지를 비교적 쉽게 검색할 수 있는 방법을 제공합니다. 다음 코드 조각에서는 요청 중에 오류가 발생했는지 여부를 파악하기 위해 Perl 및 XML::Xpath 모듈을 사용합니다. 오류가 발생되면 코드는 응답에 첫 번째 오류 코드와 메시지를 인쇄합니다.

```
use XML::XPath;
    my $xp = XML::XPath->new(xml =>$response);
    if ( $xp->find("//Error") )
    {print "There was an error processing your request:\n", " Error code: ",
    $xp->findvalue("//Error[1]/Code"), "\n", " ",
    $xp->findvalue("//Error[1]/Message"), "\n\n"; }
```

문제 해결 팁

다음 절차를 통해 Amazon RDS API의 문제를 진단하고 해결하는 것이 좋습니다.

- 타겟팅하는 AWS 리전에서 Amazon RDS가 정상적으로 작동하는지 http://status.aws.amazon.com에서 확인합니다.
- 요청 구조 확인.

  각 Amazon RDS 작업에 대한 참조 페이지는 Amazon RDS API 참조에 있습니다. 파라미터를 올바르게 사용하고 있는지 여부를 다시 확인합니다. 어떤 문제가 발생할 수 있을지 알아보려면 샘플 요청이나 사용자 시나리오를 살펴보고 이러한 예시가 유사한 작업을 하는지 확인하세요.
- AWS re:Post 확인

  Amazon RDS와 관련하여 다른 사람들이 경험한 문제에 대한 해결책을 검색할 수 있는 개발 커뮤니티가 있습니다. 주제를 보려면 AWS re:Post로 이동하세요.

문서 기록

현재 API 버전: 2014-10-31

다음 표에서는 2018년 5월 이후 Amazon RDS 사용 설명서의 각 릴리스에서 변경된 중요 사항에 대해 설명합니다. 이 설명서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

> **Note**
>
> 데이터베이스 관련 새로운 소식 페이지에서 새로운 Amazon RDS 기능을 필터링할 수 있습니다. [제품(Products)]에서 [Amazon RDS]를 선택합니다. 그런 다음 **RDS Proxy** 또는 **Oracle 2023**와 같은 키워드를 사용하여 검색합니다.

변경 사항	설명	날짜
Amazon RDS에서 MySQL 8.0.39 지원	이제 MySQL 버전 8.0.39를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하세요.	2024년 8월 12일
기존 정책 업데이트	Amazon RDS는 `AWSServiceRoleForRDSPreview` 서비스 연결 역할의 `AmazonRDSPreviewServiceRolePolicy` 에서 `sns:Publish` 권한을 제거했습니다. 자세한 내용은 AWS 관리형 정책에 대한 Amazon RDS 업데이트를 참조하세요.	2024년 8월 7일
기존 정책 업데이트	Amazon RDS는 `AWSServiceRoleForRDSBeta` 서비스 연결 역할의 `AmazonRDSBetaServiceRolePolicy` 에서 `sns:Publish` 권	2024년 8월 7일

	한을 제거했습니다. 자세한 내용은 AWS 관리형 정책에 대한 Amazon RDS 업데이트를 참조하세요.	
Amazon RDS, 데이터베이스 미리 보기 환경에서 MySQL 8.4 지원	이제 MySQL 8.4를 미국 동부(오하이오) AWS 리전의 데이터베이스 미리 보기 환경에서 사용할 수 있습니다. 자세한 내용은 데이터베이스 미리 보기 환경의 MySQL 버전 8.4를 참조하세요.	2024년 8월 1일
IAM 서비스 연결 역할 권한 업데이트	이 `AmazonRDSCustomServiceRolePolicy` 정책은 이제 다른 AWS 리전의 Amazon RDS 서비스와 통신하고 EC2 이미지를 복사할 수 있는 추가 권한을 부여합니다. 자세한 내용은 AWS 관리형 정책에 대한 Amazon RDS 업데이트를 참조하세요.	2024년 7월 18일
Amazon RDS, 데이터베이스 미리 보기 환경에서 MariaDB 11.4 지원	이제 MariaDB 11.4를 미국 동부(오하이오) AWS 리전의 데이터베이스 미리 보기 환경에서 사용할 수 있습니다. 자세한 내용은 데이터베이스 미리 보기 환경의 MariaDB 11.4를 참조하세요.	2024년 7월 18일

MySQL용 AWS ODBC 드라이버 일반 공개	MySQL용 Amazon Web Services(AWS) ODBC 드라이버는 RDS for MySQL의 고가용성을 위해 설계된 클라이언트 드라이버입니다. 자세한 내용은 MySQL용 Amazon Web Services(AWS) ODBC 드라이버로 RDS for MySQL에 연결을 참조하세요.	2024년 7월 18일
기존 정책 업데이트	Amazon RDS는 `AWSServiceRoleForRDS` 서비스 연결 역할의 `AmazonRDSServiceRolePolicy` 에서 `sns:Publish` 권한을 제거했습니다. 자세한 내용은 AWS 관리형 정책: Amazon RDSServiceRolePolicy를 참조하세요.	2024년 7월 2일
Db2를 위한 AWS Marketplace 비공개 제안	AWS Marketplace가 이제 Db2용 Amazon RDS에 대한 AWS Marketplace를 통해 Db2 라이선스에 대한 비공개 제안을 지원합니다. 자세한 내용은 비공개 제안 받기를 참조하세요.	2024년 7월 1일
Amazon S3로 다중 AZ DB 클러스터 스냅샷 데이터 내보내기	이제 Amazon S3로 다중 AZ DB 클러스터 스냅샷 데이터를 내보낼 수 있습니다. 자세한 내용은 Amazon S3로 DB 스냅샷 데이터 내보내기를 참조하세요.	2024년 6월 27일

Amazon RDS for Oracle에서 사전 구성된 r6i 메모리 최적화 인스턴스 클래스 지원	db.r6i Oracle DB 인스턴스 클래스는 vCPU 당 추가 메모리, 스토리지 및 I/O가 필요한 워크로드에 최적화되어 있습니다. 예를 들어, db.r6i.8xlarge.tpc2.mem4x는 멀티스레딩이 켜져 있으며 db.r6i.8xlarge보다 4배 더 많은 메모리를 제공합니다. 자세한 내용은 RDS for Oracle 인스턴스 클래스를 참조하세요.	2024년 6월 21일
RDS for MySQL 버전 5.7.44-RDS.20240529에 대한 RDS 추가 지원	RDS for MySQL 버전 5.7.44-RDS.20240529에 대한 RDS 추가 지원을 사용할 수 있습니다. 자세한 내용은 Amazon RDS Extended Support versions for RDS for MySQL을 참조하세요.	2024년 6월 20일
Amazon RDS에서 MySQL 8.0.37 지원	이제 MySQL 버전 8.0.37을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하세요.	2024년 6월 18일
Amazon RDS는 MariaDB 10.11.8, 10.6.18, 10.5.25 및 10.4.34를 지원	이제 MariaDB 버전 10.11.8, 10.6.18, 10.5.25, 10.4.34를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하십시오.	2024년 6월 14일

Amazon RDS는 db.m4, db.r4 및 db.t2 DB 인스턴스 클래스에 대한 지원 종료	RDS for MariaDB, RDS for MySQL 및 RDS for PostgreSQL DB 엔진의 경우 더 이상 db.m4, db.r4 및 db.t2 인스턴스 클래스를 사용하는 DB 인스턴스를 생성할 수 없습니다. RDS는 이러한 클래스를 사용하는 기존 DB 인스턴스를 최신 세대로 자동 업그레이드합니다. 자세한 내용은 DB 인스턴스 클래스 유형 단원을 참조하십시오.	2024년 6월 4일
추가 AWS 리전에서 사용 가능한 다중 AZ DB 클러스터	더 많은 AWS 리전에서 다중 AZ DB 클러스터를 생성할 수 있습니다. 지원되는 모든 리전을 보여주는 표는 Amazon RDS에서 다중 AZ DB 클러스터에 대해 지원되는 리전 및 DB 엔진을 참조하세요.	2024년 5월 29일
AWS Python 드라이버 정식 출시	Amazon Web Services(AWS) Python 드라이버는 고급 Python 래퍼로 설계되었습니다. 이 래퍼는 오픈 소스 Psycopg 드라이버의 기능을 보완하고 확장합니다. 자세한 내용은 Connecting to DB instances with the AWS drivers를 참조하세요.	2024년 5월 23일

더 많은 리전에서 RDS 프록시 사용 가능	이제 RDS Proxy를 아시아 태평양(하이데라바드), 아시아 태평양(멜버른), 중동(UAE), 이스라엘(텔아비브), 캐나다 서부(캘거리), 유럽(취리히) 리전에서 사용할 수 있습니다. RDS 프록시에 대한 자세한 내용은 Amazon RDS 프록시 사용을 참조하세요.	2024년 5월 21일
AWS Marketplace를 통한 Db2 라이선스	AWS Marketplace를 통한 Db2 라이선스를 사용하여 이제 시간당 요금을 지불하고 Amazon RDS for Db2용 Db2 라이선스를 구독할 수 있습니다. 자세한 내용은 Amazon RDS for Db2 라이선스 옵션을 참조하세요.	2024년 5월 21일
Amazon RDS에서 성능 개선 도우미에 대한 세분화된 액세스 지원	이제 성능 개선 도우미에서 개별 차원에 대한 액세스를 허용하거나 거부할 수 있습니다. 이 세분화된 액세스는 `GetResourceMetrics` , `DescribeDimensionKeys` , `GetDimensionKeyDetails` 작업에 사용할 수 있습니다. 자세한 내용은 성능 개선 도우미에 대한 세분화된 액세스 권한 부여를 참조하세요.	2024년 5월 21일
Amazon RDS for MySQL에 대한 Amazon RDS 추가 지원 버전	RDS for MySQL에 대한 RDS 추가 지원 버전의 모든 릴리스를 볼 수 있습니다. 자세한 내용은 Amazon RDS Extended Support versions for RDS for MySQL을 참조하세요.	2024년 5월 16일

Amazon RDS, 데이터베이스 미리 보기 환경에서 MySQL 8.3 지원	이제 MySQL 8.3을 미국 동부(오하이오) AWS 리전의 데이터베이스 미리 보기 환경에서 사용할 수 있습니다. 자세한 내용은 MySQL version 8.3 in the Database Preview environment를 참조하세요.	2024년 4월 30일
Amazon RDS for Db2 시간대 지원	이제 RDS for Db2는 새 RDS for Db2 DB 인스턴스에 대한 현지 시간대 설정을 지원합니다. 자세한 내용은 Amazon RDS for Db2 DB 인스턴스의 현지 시간대를 참조하세요.	2024년 4월 25일
IAM 서비스 연결 역할 권한 업데이트	이제 `AmazonRDSCustomServiceRolePolicy` 정책에서는 추가 권한을 부여하여 RDS Custom 인스턴스에 인스턴스 프로파일로 서비스 역할을 연결합니다. 자세한 내용은 AWS 관리형 정책에 대한 Amazon RDS 업데이트를 참조하세요.	2024년 4월 19일
모든 AWS 리전에서 Amazon RDS for Oracle이 Oracle Data Guard 전환 지원	이제 지원되는 모든 리전에서 Oracle Data Guard 전환을 사용할 수 있습니다. 자세한 내용을 알아보려면 Oracle Data Postom 전환 개요를 참조하세요.	2024년 4월 16일

RDS Custom for Oracle이 Oracle Standard Edition 2 지원	이제 Oracle Database 12c 릴리스 1(12.1), 12c 릴리스 2(12.2), 18c 및 19c에서 Standard Edition 2를 사용하여 DB 인스턴스를 생성할 수 있습니다. CDB와 비CDB를 모두 생성할 수 있습니다. 자세한 내용을 알아보려면 RDS Custom for Oracle에 대한 에디션 및 라이선스 지원을 참조하세요.	2024년 4월 11일
Amazon RDS for Oracle이 Oracle APEX 버전 23.2.v1 지원	Oracle Database 19c 이상에서 APEX 23.2.v1을 사용할 수 있습니다. 자세한 내용은 Oracle Application Express 단원을 참조하세요.	2024년 4월 11일
RDS Custom 서비스 연결 역할 권한 업데이트	이제 `AmazonRDSCustomServiceRolePolicy` 는 RDS Custom for SQL Server가 EC2 인스턴스 유형 정보를 가져오고 DB 호스트 인스턴스 유형을 수정할 수 있는 추가 권한을 부여합니다. 자세한 내용은 AWS 관리형 정책에 대한 업데이트를 참조하세요.	2024년 4월 8일
Amazon RDS Custom for Oracle에서 db.x2iezn DB 인스턴스 클래스 지원	이제 RDS Custom for Oracle DB 인스턴스에 db.x2iezn 인스턴스 클래스를 사용할 수 있습니다. 자세한 내용은 RDS Custome for Oracle에 대한 DB 인스턴스 클래스 지원을 참조하세요.	2024년 3월 26일

Amazon RDS에서 다중 AZ DB 클러스터에 db.c6gd 인스턴스 클래스 지원	이제 다중 AZ DB 클러스터 배포에 db.c6gd 인스턴스 클래스를 사용할 수 있습니다. 자세한 내용은 Instance class availability for Multi-AZ DB clusters를 참조하세요.	2024년 3월 21일
Amazon RDS 추가 지원	이제 RDS for MySQL 5.7 또는 RDS for PostgreSQL 11 데이터베이스를 생성하거나 복원하면 해당 데이터베이스가 Amazon RDS 확장 지원에 자동으로 등록되므로 기존 애플리케이션이 그대로 계속 작동합니다. RDS 확장 지원을 옵트아웃하면 데이터베이스 엔진에 적용되는 RDS 표준 지원 종료일이 지난 후 요금이 부과되는 것을 막을 수 있습니다. 자세한 내용은 Amazon RDS 추가 지원 사용을 참조하세요.	2024년 3월 21일
RDS for Db2와 AWS License Manager 통합	RDS for Db2가 이제 AWS License Manager와 통합됩니다. 기존 보유 라이선스 사용(BYOL) 모델을 사용하는 경우 AWS License Manager 통합은 조직 내에서 Db2 라이선스 사용 모니터링에 도움이 됩니다. 자세한 내용은 AWS License Manager와 통합을 참조하십시오.	2024년 3월 20일

다중 AZ DB 클러스터의 CA 인증서 교체	이제 다중 AZ DB 클러스터의 CA 인증서를 교체할 수 있습니다. 새 CA 인증서인 rds-ca-rsa2048-g1, rds-ca-rsa4096-g1, rds-ca-ecc384-g1 중 하나를 사용하는 것을 고려해 보세요. 자세한 내용은 SSL/TLS 인증서 교체를 참조하세요.	2024년 3월 6일
Amazon RDS에서 io2 Block Express 스토리지 지원	이제 io2 Block Express 스토리지 유형을 사용하는 RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 io2 Block Express storage를 참조하세요.	2024년 3월 6일
RDS Custom for SQL Server에서 db.r5b 및 db.x2iedn DB 인스턴스 클래스 지원	이제 RDS Custom for SQL Server DB 인스턴스에 db.r5b 및 db.x2iedn 인스턴스 클래스를 사용할 수 있습니다. 자세한 내용은 RDS Custom for SQL Server DB 인스턴스 클래스 지원을 참조하세요.	2024년 3월 4일
중동(UAE) 리전에서 RDS Custom for Oracle 사용 가능	중동(UAE) 리전에서 RDS Custom for Oracle DB 인스턴스를 생성할 수 있습니다. 지원되는 모든 AWS 리전을 보여 주는 표는 RDS Custom for Oracle을 지원하는 리전 및 DB 엔진을 참조하세요.	2024년 3월 4일

새 AWS 관리형 정책	Amazon RDS는 RDS Custom이 EC2 인스턴스 프로파일을 통해 자동화 작업 및 데이터베이스 관리 작업을 수행할 수 있도록 `AmazonRDSCustomInstanceProfileRolePolicy` 로 이름이 지정된 새 관리형 정책을 추가했습니다. 자세한 내용은 AWS 관리형 정책에 대한 Amazon RDS 업데이트를 참조하세요.	2024년 2월 27일
Amazon RDS에서 MariaDB 10.11.7, 10.6.17, 10.5.24, 10.4.33 지원	이제 MariaDB 버전 10.11.7, 10.6.17, 10.5.24, 10.4.33을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하십시오.	2024년 2월 26일
Amazon RDS 다중 AZ DB 클러스터에서 Amazon EBS gp3 스토리지 볼륨 지원	다중 AZ DB 클러스터는 이제 gp3 SSD 기반 EBS 볼륨을 지원합니다. 자세한 내용은 gp3 스토리지를 참조하세요.	2024년 2월 26일
이스라엘(텔아비브) 리전에서 AWS Secrets Manager에 대해 Amazon RDS 지원	Amazon RDS는 이스라엘(텔아비브) 리전에서 Secrets Manager를 지원합니다. 자세한 내용은 Amazon RDS와 AWS Secrets Manager를 사용한 암호 관리를 참조하세요.	2024년 2월 21일

Amazon RDS for Db2에서 감사 로깅 지원	RDS for Db2는 이제 데이터베이스 수준의 감사 로깅을 지원합니다. RDS for Db2 데이터베이스에 대한 감사 로깅을 활성화하면 Amazon RDS는 데이터베이스 작업을 기록하고 Amazon S3에 감사 로그를 저장합니다. 자세한 내용은 Db2 감사 로깅을 참조하세요.	2024년 2월 15일
Amazon RDS 추가 지원	이제 DB 인스턴스 및 다중 AZ DB 클러스터의 RDS for MySQL 및 RDS for PostgreSQL 메이저 엔진 버전이 RDS 표준 지원 종료일에 다다르면 Amazon RDS에서 Amazon RDS 추가 지원을 자동으로 활성화합니다. 자세한 내용은 Amazon RDS 추가 지원 사용을 참조하세요.	2024년 2월 15일
Amazon RDS에서 MySQL 8.0.36 지원	이제 MySQL 버전 8.0.36을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하세요.	2024년 2월 12일
Amazon RDS, RDS for Db2에 대한 EBCDIC 데이터 정렬 지원	이제 EBCDIC 데이터 정렬 시퀀스를 사용하여 데이터베이스의 콘텐츠를 정렬하는 Db2 데이터베이스를 생성할 수 있습니다. 자세한 내용은 EBCDIC collation for Db2 databases on Amazon RDS를 참조하세요.	2024년 1월 29일

기본 CA 인증서로 업데이트	기본 CA 인증서는 `rds-ca-rsa2048-g1` 로 설정되어 있습니다. 자세한 내용은 SSL/TLS를 사용하여 DB 인스턴스에 대한 연결 암호화를 참조하세요.	2024년 1월 26일
Amazon RDS for PostgreSQL는 PL/Rust, croaring-rs 및 num-bigint를 위한 2개의 새로운 크레이트 지원	Amazon RDS for PostgreSQL에서 새 크레이트를 2개 사용할 수 있습니다. 자세한 내용은 PL/Rust가 있는 상자 사용을 참조하세요.	2024년 1월 24일
Amazon RDS for PostgreSQL에서 TLS 버전 1.3 지원	RDS for PostgreSQL에서 전송 계층 보안(TLS) 버전 1.3을 사용할 수 있습니다. 자세한 내용은 PostgreSQL DB 인스턴스와 함께 SSL 사용을 참조하십시오.	2024년 1월 24일
RDS Custom for SQL Server, Microsoft SQL Server 2022 지원	이제 SQL Server 2022를 사용하는 RDS Custom for SQL Server DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 RDS Custom for SQL Server 작업을 참조하세요.	2024년 1월 22일
AWS 관리형 정책 권한 업데이트	`AWSServiceRoleForRDS` 서비스 연결 역할의 `AmazonRDSServiceRolePolicy` 에 명령문 ID가 새로 생겼습니다. 자세한 내용은 AWS 관리형 정책에 대한 Amazon RDS 업데이트를 참조하세요.	2024년 1월 19일

RDS Custom for Oracle, 유럽(파리) 리전 지원	유럽(파리) 리전에서 RDS Custom for Oracle DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 RDS Custom for Oracle을 지원하는 리전 및 DB 엔진을 참조하세요.	2024년 1월 18일
Amazon RDS for MySQL에서 다중 소스 복제 지원	이제 RDS for MySQL DB 인스턴스에서 다중 소스 복제를 사용할 수 있습니다. 자세한 내용은 Configuring multi-source replication on RDS for MySQL을 참조하세요.	2024년 1월 16일
Amazon RDS, 데이터베이스 미리 보기 환경에서 MySQL 8.2 지원	이제 MySQL 8.2을 미국 동부(오하이오) AWS 리전의 데이터베이스 미리 보기 환경에서 사용할 수 있습니다. 자세한 내용은 데이터베이스 미리 보기 환경의 MySQL 버전 8.2를 참조하세요.	2024년 1월 11일
유럽(스페인) 리전에서 RDS 프록시 지원	이제 유럽(스페인) 리전에서 RDS 프록시를 사용할 수 있습니다. RDS 프록시에 대한 자세한 내용은 Amazon RDS 프록시 사용을 참조하세요.	2024년 1월 8일
캐나다 서부(캘거리) 리전에서 Amazon RDS 사용 가능	이제 Amazon RDS를 캐나다 서부(캘거리) 리전에서 사용할 수 있습니다. 자세한 내용은 리전 및 가용 영역을 참조하세요.	2023년 12월 20일

Amazon RDS for Db2, 5,000명의 로컬 사용자 지원	이제 권한 부여 목록에 최대 5,000명의 로컬 사용자를 추가할 수 있습니다. 자세한 내용은 rdsadmin.add_user를 참조하세요.	2023년 12월 20일
Amazon RDS에서 권장 사항 확인 및 권장 사항에 대한 응답 지원	Amazon RDS 권장 사항에는 이제 RDS for PostgreSQL을 위한 임곗값 기반 사전 예방적 권장 사항과 기계 학습 기반 사후 대응적 권장 사항이 포함됩니다. 자세한 내용은 Amazon RDS의 권장 사항을 참조하세요.	2023년 12월 19일
Amazon RDS, MariaDB 10.11.6, 10.6.16, 10.5.23 및 10.4.32 지원	이제 MariaDB 버전 10.11.6, 10.6.16, 10.5.23, 10.4.32를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하십시오.	2023년 12월 12일
Amazon Redshift가 구성된 제로 ETL 통합을 도입한 Amazon RDS(미리 보기)	제로 ETL 통합은 트랜잭션 데이터가 RDS for MySQL DB 인스턴스에 기록된 후 몇 초 안에 Amazon Redshift에서 사용할 수 있도록 하기 위한 완전관리형 솔루션을 제공합니다. 자세한 내용은 Working with Amazon RDS zero-ETL integrations with Amazon Redshift(preview)를 참조하세요.	2023년 11월 28일

Amazon RDS, IBM Db2 데이터베이스 엔진 지원	이제 Amazon RDS에서 IBM Db2 데이터베이스 엔진을 실행할 수 있습니다. 자세한 내용은 Amazon RDS for Db2를 참조하세요.	2023년 11월 27일
RDS for PostgreSQL, PostgreSQL 16.1으로의 메이저 버전 업그레이드와 15.5, 14.10, 13.13, 12.17 및 11.22로의 마이너 버전 업그레이드 지원	RDS for PostgreSQL은 PostgreSQL 16.1으로의 메이저 버전 업그레이드와 15.5, 14.10, 13.13, 12.17 및 11.22로의 마이너 버전 업그레이드를 지원합니다. 자세한 내용은 Amazon RDS용 PostgreSQL DB 엔진 업그레이드를 참조하세요.	2023년 11월 17일
RDS Custom for Oracle에서 옵션 그룹 지원	옵션 그룹을 생성하거나 수정하고 RDS Custom for Oracle DB 인스턴스에 연결할 수 있습니다. 이제 `Timezone` 옵션이 지원됩니다. 자세한 내용을 알아보려면 Working with option groups in RDS Custom for Oracle을 참조하세요.	2023년 11월 17일
Amazon RDS for MySQL, 그룹 복제 플러그인 지원	이제 MySQL 커뮤니티에서 개발하고 유지 관리하는 그룹 복제 플러그인을 사용하여 RDS for MySQL 버전 8.0.35 이상의 DB 인스턴스로 활성-활성 클러스터를 설정할 수 있습니다. 자세한 내용은 Configuring active-active clusters for RDS for MySQL을 참조하세요.	2023년 11월 17일

Amazon RDS 프록시, RDS for PostgreSQL 16.1 지원	이제 RDS for PostgreSQL 16.1 DB 인스턴스용 RDS 프록시를 사용하여 프록시를 생성할 수 있습니다. 자세한 내용은 Amazon RDS 프록시 사용을 참조하세요.	2023년 11월 17일
RDS Custom for SQL Server, Microsoft SQL Server 2019 Developer 에디션 지원	SQL Server 2019 Developer 에디션을 사용하는 RDS Custom for SQL Server DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 RDS Custom for SQL Server에서 기존 보유 미디어 사용을 참조하세요.	2023년 11월 16일
최소 가동 중지 시간으로 다중 AZ DB 클러스터의 마이너 버전 업그레이드	다중 AZ DB 클러스터의 마이너 버전 업그레이드를 수행할 때 Amazon RDS가 라이터 인스턴스보다 먼저 리더 DB 인스턴스를 업그레이드하므로 가동 중지 시간이 크게 줄어듭니다. RDS 프록시와 함께 사용하면 다운타임을 1초 이하로 더 줄일 수 있습니다. 자세한 내용은 다중 AZ DB 클러스터의 엔진 버전 업그레이드를 참조하세요.	2023년 11월 16일
RDS for SQL Server, Microsoft SQL Server 2022 지원	이제 SQL Server 2022를 사용하는 RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 Microsoft SQL Server 버전을 참조하십시오.	2023년 11월 15일

RDS for MySQL, 버전 5.7에서 8.0으로 스냅샷 업그레이드 지원	RDS for MySQL 스냅샷의 엔진 버전을 버전 5.7에서 버전 8.0으로 업그레이드 할 수 있습니다. 이 작업은 AWS Management Console을 사용하거나 RDS API의 `ModifyDBSnapshot` 작업 또는 AWS CLI를 사용하여 수행할 수 있습니다. 자세한 내용은 Upgrading a MySQL DB snapshot engine version을 참조하세요.	2023년 11월 15일
RDS Custom for SQL Server, 1,000개 데이터베이스의 지정 시간 복구 지원	이제 RDS Custom for SQL Server DB 인스턴스에서 전체 백업 및 지정 시간 복구에 적합한 데이터베이스를 최대 1,000개까지 만들 수 있습니다. 자세한 내용을 알아보려면 RDS Custom for SQL Server 인스턴스를 특정 시점으로 복원을 참조하세요.	2023년 11월 15일
RDS Custom for SQL Server, Service Master Key(SMK) 사용 지원	RDS Custom for SQL Server는 이제 서비스 마스터 키(SMK) 사용을 지원합니다. SMK를 사용하면 보안 인증과 같은 객체를 암호화하고 TDE 및 열 암호화와 같은 SQL Server 기능을 사용할 수 있습니다. 자세한 내용을 알아보려면 Using a Service Master Key with RDS Custom for SQL Server를 참조하세요.	2023년 11월 13일

Amazon RDS, 데이터베이스 미리 보기 환경에서 MySQL 8.1 지원	이제 MySQL 8.1을 미국 동부(오하이오) AWS 리전의 데이터베이스 미리 보기 환경에서 사용할 수 있습니다. 자세한 내용은 데이터베이스 미리 보기 환경의 MySQL 버전 8.1을 참조하세요.	2023년 11월 10일
RDS, MySQL 8.0.35 및 MySQL 5.7.44 지원	이제 MySQL 버전 8.0.35 및 5.7.44를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하세요.	2023년 11월 9일
RDS 프록시, 다중 AZ DB 클러스터 지원	이제 RDS 프록시에서 다중 AZ DB 클러스터를 지원합니다. 자세한 내용은 Amazon RDS 프록시 작업을 참조하세요.	2023년 11월 9일
RDS Custom for Oracle, AWS GovCloud (US) Regions에서 사용 가능	이제 Amazon RDS를 AWS GovCloud (US) Regions에서 사용할 수 있습니다. 자세한 내용은 RDS Custom for Oracle을 지원하는 리전 및 DB 엔진을 참조하세요.	2023년 11월 9일
Amazon RDS 최적화된 쓰기, db.m5 DB 인스턴스 클래스 지원	이제 Amazon RDS 최적화된 쓰기에서 db.m5 DB 인스턴스 클래스를 지원합니다. 자세한 내용은 Amazon RDS 최적화된 쓰기로 MariaDB 쓰기 성능 개선 및 Amazon RDS 최적화된 쓰기로 MySQL 쓰기 성능 개선을 참조하세요.	2023년 11월 9일

Amazon RDS for Oracle, CDB 아키텍처의 다중 테넌트 구성 지원	RDS for Oracle 다중 테넌트 기능을 통해 RDS는 Oracle 데이터베이스에 완전관리형 Oracle 멀티테넌트 아키텍처와 경험을 제공합니다. RDS API를 사용하여 CDB에 테넌트 데이터베이스라고 하는 여러 PDB를 만들 수 있습니다. RDS는 레거시 단일 테넌트 구성의 대안으로 CDB 아키텍처의 다중 테넌트 구성을 제공합니다. 자세한 내용은 CDB 아키텍처의 다중 테넌트 구성을 참조하세요.	2023년 11월 8일
Amazon RDS, 성능 개선 도우미 지표를 Amazon CloudWatch에 내보내기 가능	성능 개선 도우미를 사용하면 사전 구성된 지표 또는 사용자 지정 지표 대시보드를 Amazon CloudWatch로 내보낼 수 있습니다. 내보낸 지표 대시보드를 CloudWatch 콘솔에서 볼 수 있습니다. 선택한 성능 개선 도우미 지표 위젯을 내보내고 CloudWatch 콘솔에서 지표 데이터를 볼 수도 있습니다. 자세한 내용은 CloudWatch에 성능 개선 도우미 지표 내보내기를 참조하세요.	2023년 11월 8일
Amazon RDS Custom for Oracle, DB 인스턴스의 운영 체제 업그레이드 허용	이제 `modify-db-instance` CLI 명령을 사용하여 RDS Custom for Oracle DB 인스턴스의 데이터베이스 또는 운영 체제(OS)를 업그레이드할 수 있습니다. 자세한 내용은 RDS Custom for Oracle DB 인스턴스 업그레이드를 참조하세요.	2023년 11월 7일

RDS 프록시, RDS PostgreSQL에 확장 프로토콜 지원	이제 RDS for PostgreSQL DB 인스턴스에서 확장 쿼리 프로토콜을 실행할 수 있습니다. 자세한 내용은 Amazon RDS 프록시 사용을 참조하세요.	2023년 11월 6일
RDS for PostgreSQL, RDS 블루/그린 배포 지원	이제 RDS for PostgreSQL DB 인스턴스에서 블루/그린 배포를 생성할 수 있습니다. 자세한 내용은 데이터베이스 업데이트에 Amazon RDS 블루/그린 배포 사용을 참조하세요.	2023년 10월 26일
AWS 관리형 정책으로 업데이트	`AmazonRDSPerformanceInsightsReadOnly` 및 `AmazonRDSPerformanceInsightsFullAccess` 관리형 정책에서 이제 정책 문에 `Sid`(문 ID)를 식별자로 포함합니다. 자세한 내용은 AWS 관리형 정책에 대한 Amazon RDS 업데이트를 참조하세요.	2023년 10월 23일
RDS Custom for Oracle, 유럽(밀라노) 리전 지원	자세한 내용은 RDS Custom for Oracle을 지원하는 리전 및 DB 엔진을 참조하세요.	2023년 10월 23일

기존 데이터베이스에서 RDS 최적화된 쓰기 활성화	이제 RDS 최적화된 쓰기 기능을 지원하지 않는 엔진 버전, DB 인스턴스 클래스 또는 파일 시스템 구성에서 생성된 기존 DB 인스턴스에도 RDS 최적화된 쓰기를 활성화할 수 있습니다. 자세한 내용은 RDS for MySQL용 기존 데이터베이스에서 RDS 최적화된 쓰기 활성화 및 RDS for MariaDB용 기존 데이터베이스에서 RDS 최적화된 쓰기 활성화를 참조하세요.	2023년 10월 19일
Amazon RDS, 전용 로그 볼륨(DLV) 사용 지원	이제 RDS for MariaDB, RDS for MySQL, RDS for PostgreSQL에서 전용 로그 볼륨(DLV)을 사용할 수 있습니다. DLV는 할당된 스토리지가 크고 초당 I/O(IOPS) 요구 사항이 높거나 지연 시간에 민감한 워크로드가 있는 데이터베이스에 적합합니다. 자세한 내용은 전용 로그 볼륨(DLV) 사용을 참조하세요.	2023년 10월 17일
Amazon RDS for PostgreSQL, MySQL 및 MariaDB는 새로운 DB 인스턴스 클래스 지원	이제 db.m6.in, db.m6idn, db.r6.in 및 db.r6.idn DB 인스턴스 클래스를 사용하는 PostgreSQL, MySQL 및 MariaDB를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 사용 가능한 모든 DB 인스턴스 클래스에 대해 지원되는 DB 엔진을 참조하십시오.	2023년 10월 12일

Amazon RDS for PostgreSQL에서 pgactive 지원	Amazon RDS for PostgreSQL에서 pgactive 확장을 사용할 수 있습니다. 자세한 내용은 Amazon RDS for PostgreSQL로 PostgreSQL 확장 사용을 참조하세요.	2023년 10월 9일
아시아 태평양(자카르타) 리전에서 RDS Custom for Oracle 사용 가능	아시아 태평양(자카르타) 리전에서 RDS Custom for Oracle DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 RDS Custom for Oracle을 지원하는 리전 및 DB 엔진을 참조하세요.	2023년 10월 5일
RDS Custom for SQL Server에서 새로운 서버 수준 데이터 정렬 지원	RDS Custom for SQL Server는 이제 SQL_라틴, 일본어, 독일어 및 아랍어 로케일의 기존 인코딩과 UTF-8 인코딩 모두에서 다양한 서버 데이터 정렬을 지원합니다. 자세한 내용은 RDS Custom for SQL Server DB 인스턴스에 대한 데이터 정렬 및 문자 지원을 참조하세요.	2023년 9월 26일
AWS 관리형 정책 권한 업데이트	`AWSServiceRoleForRDSCustom` 서비스 연결 역할의 `AmazonRDSCustomServiceRolePolicy` 에는 RDS Custom이 EventBridge 관리형 규칙을 생성, 수정 및 삭제할 수 있도록 하는 새로운 권한이 부여되었습니다. 자세한 내용은 AWS 관리형 정책에 대한 Amazon RDS 업데이트를 참조하세요.	2023년 9월 20일

Amazon RDS는 성능 개선 도우미 카운터 지표를 Amazon CloudWatch에 게시	CloudWatch 콘솔의 DB_PERF_INSIGHTS 지표 수학 함수를 사용하면 Amazon RDS에서 성능 개선 도우미 카운터 지표를 쿼리할 수 있습니다. 자세한 내용은 Amazon RDS를 모니터링하여 CloudWatch 경보 생성을 참조하세요.	2023년 9월 20일
성능 개선 도우미에서 SQL Server용 다이제스트 수준 통계 지원	성능 개선 도우미를 사용하면 Amazon RDS for SQL Server의 문 및 다이제스트 수준 모두에서 SQL 통계를 볼 수 있습니다. 자세한 내용은 SQL Server에서 실행 중인 쿼리 분석을 참조하세요.	2023년 9월 18일
Amazon RDS for PostgreSQL, MySQL, MariaDB에서 db.m6.id 및 db.r6.id DB 인스턴스 클래스 유형 지원	이제 메모리 최적화 db.m6.id 및 db.r6.id DB 인스턴스 클래스 유형을 사용하는 PostgreSQL, MySQL, MariaDB를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 이러한 유형은 로컬 NVMe 기반 SSD 스토리지를 제공합니다. 자세한 내용은 사용 가능한 모든 DB 인스턴스 클래스에 대해 지원되는 DB 엔진을 참조하십시오.	2023년 9월 11일
RDS for PostgreSQL 다중 AZ DB 클러스터에 메이저 버전 업그레이드 지원	이제 RDS for PostgreSQL 다중 AZ DB 클러스터의 메이저 버전 업그레이드를 수행할 수 있습니다. 자세한 내용은 다중 AZ DB 클러스터의 엔진 버전 업그레이드를 참조하세요.	2023년 9월 7일

Amazon RDS에서 MariaDB 10.11.5, 10.6.15, 10.5.22 및 10.4.31 지원	이제 MariaDB 버전 10.11.5, 10.6.15, 10.5.22, 10.4.31을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하십시오.	2023년 9월 7일
Amazon RDS 추가 지원	Amazon RDS는 RDS 표준 지원 종료일이 지난 후에도 DB 인스턴스에서 RDS for MySQL 및 RDS for PostgreSQL 메이저 엔진 버전을 계속 실행할 수 있는 기능을 곧 출시한다고 발표했습니다. 자세한 내용은 Amazon RDS 추가 지원 사용을 참조하세요.	2023년 9월 1일
RDS Custom에서 RDS Custom for SQL Server DB 인스턴스의 시작 및 중지 지원	이제 RDS Custom에서 RDS Custom for SQL Server DB 인스턴스를 시작 및 중지할 수 있습니다. 자세한 내용은 RDS Custom for SQL Server DB 인스턴스 시작 및 중지를 참조하세요.	2023년 8월 31일
Amazon RDS 최적화된 쓰기에서 db.r5 DB 인스턴스 클래스 지원	이제 Amazon RDS 최적화된 쓰기는 db.r5 DB 인스턴스 클래스를 지원합니다. 자세한 내용은 Amazon RDS 최적화된 쓰기로 MariaDB 쓰기 성능 개선 및 Amazon RDS 최적화된 쓰기로 MySQL 쓰기 성능 개선을 참조하세요.	2023년 8월 31일

Amazon RDS for Oracle에서 CDB에 대한 시간대 파일 자동 업그레이드 지원	`TIMEZONE_FILE_AUTOUPGRADE` 옵션을 사용하면 현재 시간대 파일을 RDS for Oracle 컨테이너 데이터베이스(CDB)에서 최신 버전으로 업그레이드할 수 있습니다. 자세한 내용은 Oracle 시간대 파일 자동 업그레이드를 참조하세요.	2023년 8월 29일
Amazon RDS 최적화된 쓰기에서 db.m6g 및 db.m6i DB 인스턴스 클래스 지원	이제 Amazon RDS 최적화된 쓰기는 db.m6g 및 db.m6i DB 인스턴스 클래스를 지원합니다. 자세한 내용은 Amazon RDS 최적화된 쓰기로 MariaDB 쓰기 성능 개선 및 Amazon RDS 최적화된 쓰기로 MySQL 쓰기 성능 개선을 참조하세요.	2023년 8월 28일
Amazon RDS, MariaDB 10.11 지원	이제 MariaDB 버전 10.11을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하십시오.	2023년 8월 21일
AWS 관리형 정책 권한 업데이트	`AWSServiceRoleForRDSCustom` 서비스 연결 역할의 `AmazonRDSCustomServiceRolePolicy` 에 RDS Custom이 네트워크 인터페이스를 생성할 수 있는 새로운 권한이 부여되었습니다. 자세한 내용은 AWS 관리형 정책에 대한 Amazon RDS 업데이트를 참조하세요.	2023년 8월 18일

AWS 관리형 정책 권한 업데이트	`AmazonRDSFullAccess` 관리형 정책에는 일정 기간 동안 성능 분석 보고서를 생성, 확인 및 삭제할 수 있는 새 권한이 있습니다. 자세한 내용은 AWS 관리형 정책에 대한 Amazon RDS 업데이트를 참조하세요.	2023년 8월 17일
AWS 관리형 정책 권한 업데이트	`AmazonRDSPerformanceInsightsReadOnly` 관리형 정책에 새 권한을 추가하고 새로운 관리형 정책인 `AmazonRDSPerformanceInsightsFullAccess` 를 추가하면 일정 기간 동안의 DB 로드 분석 보고서를 생성할 수 있습니다. 자세한 내용은 AWS 관리형 정책에 대한 Amazon RDS 업데이트를 참조하세요.	2023년 8월 16일
Amazon RDS가 일정 기간 동안 성능 분석 지원	성능 개선 도우미를 사용하면 특정 기간 동안의 성과 분석 보고서를 만들고 볼 수 있습니다. 보고서는 식별된 인사이트와 성능 문제를 해결하기 위한 권장 사항을 제공합니다. 자세한 내용은 일정 기간 동안의 DB 로드 분석을 참조하세요.	2023년 8월 16일

Amazon RDS Custom for Oracle이 db.r5b 및 db.x2iedn DB 인스턴스 클래스 지원	이제 RDS Custom for Oracle DB 인스턴스에 db.r5b 및 db.x2iedn 인스턴스 클래스를 사용할 수 있습니다. 자세한 내용은 RDS Custome for Oracle에 대한 DB 인스턴스 클래스 지원을 참조하세요.	2023년 8월 16일
Amazon RDS Custom for Oracle이 db.m6i, db.r6i, db.t3 DB 인스턴스 클래스 지원	이제 RDS Custom for Oracle DB 인스턴스에 db.m6i, db.r6i, db.t3 인스턴스 클래스를 사용할 수 있습니다. 자세한 내용은 RDS Custome for Oracle에 대한 DB 인스턴스 클래스 지원을 참조하세요.	2023년 8월 15일
Amazon RDS for PostgreSQL이 이제 데이터베이스 미리 보기 환경에서 PostgreSQL 버전 16 베타 3 지원	이제 PostgreSQL 버전 16 베타 3를 미국 동부(오하이오) AWS 리전의 데이터베이스 미리 보기 환경에서 사용할 수 있습니다. 자세한 내용은 데이터베이스 미리 보기 환경으로 작업을 참조하세요.	2023년 8월 11일
Amazon RDS가 MySQL 8.0.34 및 5.7.43 지원	이제 MySQL 버전 8.0.34 및 5.7.43을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하세요.	2023년 8월 9일
RDS for SQL Server가 대기 복제본에 OS 지표 보기 지원	이제 RDS for SQL Server에 대한 대기 복제본 관련 OS 지표를 볼 수 있습니다. 자세한 내용은 RDS 콘솔에서 OS 지표 보기를 참조하세요.	2023년 8월 3일

RDS for Oracle이 CDB에 대한 Oracle Data Guard 지원	RDS for Oracle이 Oracle Database 19c 및 21c 컨테이너 데이터베이스(CDB)에 대해 Data Guard 읽기 전용 복제본을 지원합니다. 기존 RDS API를 사용하여 비CDB에서와 마찬가지로 CDB에서도 읽기 전용 복제본을 생성, 관리 및 승격할 수 있습니다. 자세한 내용은 멀티테넌트 읽기 복제본을 참조하세요.	2023년 8월 1일
이제 이스라엘(텔아비브) 리전에서 Amazon RDS 사용 가능	이제 이스라엘(텔아비브) 리전에서 Amazon RDS를 사용할 수 있습니다. 자세한 내용은 리전 및 가용 영역을 참조하세요.	2023년 8월 1일
Amazon RDS가 Oracle APEX 버전 23.1.v1 지원	APEX 23.1.v1을 Oracle Database 19c 이상에서 사용할 수 있습니다. 자세한 내용은 Oracle Application Express 단원을 참조하세요.	2023년 7월 26일
Amazon RDS Custom for Oracle은 기본값이 아닌 Oracle SID를 지원합니다.	Oracle Database 19c를 사용하여 RDS Custom for Oracle DB 인스턴스를 생성할 때 기본값이 아닌 Oracle 시스템 식별자(Oracle SID)를 지정할 수 있습니다. 이 값은 CDB의 이름이기도 합니다. 자세한 내용은 멀티테넌트 아키텍처 고려 사항을 참조하세요.	2023년 7월 21일

RDS for SQL Server는 자체 관리형 Active Directory를 지원합니다.	이제 자체 관리형 Active Directory를 사용하여 RDS for SQL Server DB 인스턴스를 Microsoft Active Directory(AD) 도메인에 직접 가입시킬 수 있습니다. 자체 관리형 AD 도메인은 온프레미스 또는 클라우드에 있을 수 있습니다. 자세한 내용은 자체 관리형 Active Directory 작업을 참조하세요.	2023년 7월 7일
PostgreSQL 논리적 복제는 다중 AZ DB 클러스터를 지원합니다.	이제 PostgreSQL 논리적 복제를 다중 AZ DB 클러스터와 함께 사용하여 전체 데이터베이스 인스턴스가 아닌 개별 테이블을 복제하고 동기화할 수 있습니다. 자세한 내용은 다중 AZ DB 클러스터에서 PostgreSQL 논리적 복제 사용을 참조하세요.	2023년 7월 6일
이제 Amazon RDS for PostgreSQL은 데이터베이스 미리 보기 환경에서 PostgreSQL 버전 16 베타 2를 지원합니다.	이제 PostgreSQL 버전 16 베타 2를 미국 동부(오하이오) AWS 리전의 데이터베이스 미리 보기 환경에서 사용할 수 있습니다. 자세한 내용은 데이터베이스 미리 보기 환경으로 작업을 참조하세요.	2023년 7월 6일

AWS 관리형 정책 권한 업데이트	`AWSServiceRoleForRDSCustom` 서비스 연결 역할의 `AmazonRDSCustomServiceRolePolicy` 에 RDS Custom for Oracle이 스냅샷을 사용할 수 있는 새로운 권한이 부여되었습니다. 자세한 내용은 AWS 관리형 정책에 대한 Amazon RDS 업데이트를 참조하세요.	2023년 6월 23일
RDS가 MariaDB 10.6.14, 10.5.21, 10.4.30을 지원함	이제 MariaDB 버전 10.6.14, 10.5.21, 10.4.30을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하십시오.	2023년 6월 22일
RDS가 MySQL 8.0.33 및 5.7.42를 지원함	이제 MySQL 버전 8.0.33 및 5.7.42를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하세요.	2023년 6월 15일
RDS가 MariaDB 10.6.13, 10.5.20, 10.4.29, 10.3.39를 지원함	이제 MariaDB 버전 10.6.13, 10.5.20, 10.4.29, 10.3.39를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하십시오.	2023년 6월 15일

RDS for Oracle이 전송 가능한 테이블스페이스를 지원함	전송 가능한 테이블스페이스를 사용하여 온프레미스 Oracle 데이터베이스의 데이터를 RDS for Oracle DB 인스턴스로 마이그레이션할 수 있습니다. 이 방법은 추가 라이선스가 필요하지 않으며 가동 중지 시간이 가장 짧은 마이그레이션 기술입니다. 자세한 내용은 Oracle 전송 가능한 테이블스페이스를 사용한 마이그레이션을 참조하세요.	2023년 6월 15일
Amazon RDS에서 RDS for MariaDB 버전 10.6과 함께 RDS 프록시 지원	이제 RDS for MariaDB 버전 10.6 데이터베이스를 사용하여 RDS 프록시를 생성할 수 있습니다. RDS 프록시에 대한 자세한 내용은 Amazon RDS 프록시 사용을 참조하세요.	2023년 6월 15일
RDS Custom for SQL Server가 기존 보유 미디어 사용(BYOM)을 지원함	이제 자체 SQL Server 미디어를 사용하여 사용자 지정 엔진 버전(CEV)을 만들 수 있습니다. 자세한 내용은 RDS Custom for SQL Server에서 기존 보유 미디어 사용을 참조하세요.	2023년 6월 8일

RDS for Oracle이 Oracle Database 19c(비CDB)를 CDB로 변환할 수 있음	DB 인스턴스에서 2021년 4월 이상 RU에서 Oracle Database 19c를 실행하는 경우 CDB가 아닌 데이터베이스를 CDB(컨테이너 데이터베이스)로 변환할 수 있습니다. 아키텍처를 변환한 후 19c CDB를 21c CDB로 업그레이드할 수 있습니다. 단일 명령으로 데이터베이스를 업그레이드하고 아키텍처를 변환할 수 없기 때문에 이 단계가 필요합니다. 자세한 내용은 RDS for Oracle 비CDB를 CDB로 변환을 참조하세요.	2023년 5월 31일
중국 리전에서 다중 AZ DB 클러스터 사용 가능	이제 중국(베이징)과 중국(닝샤) AWS 리전에서 다중 AZ DB 클러스터를 사용할 수 있습니다. 자세한 내용은 Amazon RDS에서 다중 AZ DB 클러스터를 지원하는 리전 및 DB 엔진을 참조하세요.	2023년 5월 30일
다중 AZ DB 클러스터에 Amazon RDS 최적화된 읽기 지원	이제 Amazon RDS 최적화된 읽기에서 다중 AZ DB 클러스터를 지원합니다. 자세한 내용은 Amazon RDS 최적화된 읽기로 RDS for MySQL 쿼리 성능 개선 및 Amazon RDS 최적화된 읽기로 RDS for PostgreSQL 쿼리 성능 개선을 참조하세요.	2023년 5월 30일
아시아 태평양(자카르타) 리전에서 RDS Custom for Oracle 사용 가능	자세한 내용은 RDS Custom for Oracle을 지원하는 리전 및 DB 엔진을 참조하세요.	2023년 5월 29일

소스 RDS for PostgreSQL 다중 AZ DB 클러스터를 사용하여 DB 인스턴스 읽기 전용 복제본 생성	이제 RDS for PostgreSQL 다중 AZ DB 클러스터를 소스로 사용하는 경우 DB 인스턴스 읽기 전용 복제본을 생성할 수 있습니다. 이전에는 RDS for MySQL만 지원되었습니다. 자세한 내용은 다중 AZ DB 클러스터에서 DB 인스턴스 읽기 전용 복제본 생성을 참조하세요.	2023년 5월 24일
Amazon RDS가 성능 개선 도우미 대시보드에서 통합된 성능 개선 도우미 및 CloudWatch 지표 제공	Amazon RDS는 이제 성능 개선 도우미 대시보드에서 성능 개선 도우미 및 CloudWatch 지표에 대한 통합 보기를 제공합니다. 자세한 내용은 성능 개선 도우미 대시보드와 결합된 지표 보기를 참조하세요.	2023년 5월 24일
중국 리전에서 Amazon RDS Optimized Reads 사용 가능	이제 AWS 리전 중국(베이징) 및 중국(닝샤)에서 Amazon RDS Optimized Reads를 사용할 수 있습니다. 자세한 내용은 Amazon RDS Optimized Reads로 RDS for MariaDB 쿼리 성능 개선 및 Amazon RDS Optimized Reads로 RDS for MySQL 쿼리 성능 개선을 참조하세요.	2023년 4월 24일
중국 리전에서 AWS Secrets Manager에 대해 Amazon RDS 지원	Amazon RDS가 중국(베이징) 및 중국(닝샤) 리전에서 Secrets Manager를 지원합니다. 자세한 내용은 Amazon RDS와 AWS Secrets Manager를 사용한 암호 관리를 참조하세요.	2023년 4월 20일

RDS Custom for Oracle에서 새 CEV의 AMI ID 재사용 지원	사용자 지정 엔진 버전(CEV)을 생성할 경우, RDS Custom for Oracle에서는 사용 가능한 가장 최신 Amazon Machine Image(AMI)를 기본값으로 설정합니다. 이제 이전 CEV에서 사용했던 AMI ID를 지정할 수 있습니다. 자세한 내용은 CEV 생성을 참조하세요.	2023년 4월 19일
Amazon RDS에서 주제 구독자에게 태그가 포함된 이벤트 게시 지원	Amazon RDS에서 Amazon Simple Notification Service(SNS) 또는 Amazon EventBridge로 이벤트 알림을 전송할 경우, 이제 메시지 본문에 이벤트 태그가 포함됩니다. 이러한 태그는 서비스 이벤트의 영향을 받은 리소스 데이터를 제공합니다. 자세한 내용은 Amazon RDS 이벤트 알림 태그 및 속성을 참조하세요.	2023년 4월 17일
다중 AZ DB 클러스터에 대한 예약 인스턴스 구매	이제 다중 AZ DB 클러스터에 대한 예약 DB 인스턴스를 구매할 수 있습니다. 자세한 내용은 다중 AZ DB 클러스터에 대한 예약 DB 인스턴스를 참조하세요.	2023년 4월 12일
Amazon RDS에서 db.m7g 및 db.r7g DB 인스턴스 클래스 지원	이제 RDS for MySQL, RDS for MariaDB, RDS for PostgreSQL DB 인스턴스에 db.m7g 및 db.r7g DB 인스턴스 클래스를 사용할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스에 대해 지원되는 DB 엔진 섹션을 참조하세요.	2023년 4월 12일

Amazon RDS Custom 서비스 연결 역할 권한 업데이트	`AmazonRDSCustomServiceRolePolicy` 는 이제 RDS Custom for SQL Server에서 Amazon SQS를 사용하고 스냅샷을 생성할 수 있는 추가 권한을 부여합니다. 자세한 내용은 AWS 관리형 정책에 대한 업데이트를 참조하세요.	2023년 4월 6일
읽기 전용 복제본을 사용하여 RDS for MySQL 다중 AZ DB 클러스터로 마이그레이션	이제 읽기 전용 복제본을 사용하면 RDS for MySQL 단일 AZ 배포 또는 다중 AZ DB 인스턴스 배포를 RDS for MySQL 다중 AZ DB 클러스터 배포로 마이그레이션하여 가동 중지를 줄일 수 있습니다. 자세한 내용은 읽기 전용 복제본을 사용하여 다중 AZ DB 클러스터로 마이그레이션을 참조하세요.	2023년 4월 6일
다중 AZ DB 클러스터에서 DB 인스턴스 읽기 전용 복제본 생성	소스 클러스터의 컴퓨팅 용량을 초과하여 확장하려는 경우, 이제 다중 AZ DB 클러스터에서 DB 인스턴스 읽기 전용 복제본을 생성할 수 있습니다. 자세한 내용은 다중 AZ DB 클러스터에서 DB 인스턴스 읽기 전용 복제본 생성을 참조하세요.	2023년 4월 6일
Amazon RDS Custom for SQL Server에서 다중 AZ 지원	RDS Custom for SQL Server를 사용하여 다중 AZ 배포를 생성할 수 있습니다. 자세한 내용은 RDS Custom for SQL Server에 대한 다중 AZ 배포 관리를 참조하세요.	2023년 4월 6일

AWS 관리형 정책 권한 업데이트	`AmazonRDSFullAccess` 및 `AmazonRDSReadOnlyAccess` 정책은 이제 RDS 콘솔에서 Amazon DevOps Guru 결과를 표시할 수 있는 추가 권한을 부여합니다. 자세한 내용은 AWS 관리형 정책에 대한 Amazon RDS 업데이트를 참조하세요.	2023년 3월 30일
Amazon RDS에서 Oracle APEX 버전 22.2.v1 지원	지원되는 모든 Oracle Database 버전에서 APEX 22.2.v1을 사용할 수 있습니다. 자세한 내용은 Oracle Application Express 단원을 참조하세요.	2023년 3월 30일
RDS for PostgreSQL에 Amazon DevOps Guru 사용 가능	RDS for PostgreSQL은 Amazon DevOps Guru가 최근에 감지한 이상 징후를 사용자에게 통보합니다. 콘솔의 데이터베이스 세부 정보 페이지에는 현재 및 지난 24시간 동안 발생한 이상 현상이 표시됩니다. DevOps Guru는 RDS for PostgreSQL 데이터베이스에서 문제가 발생할 것으로 예측되기 전에 문제를 해결하는 데 도움이 되는 권장 사항과 함께 사전 예방 인사이트를 게시합니다. 자세한 내용은 DevOps Guru for RDS 작동 방식을 참조하세요.	2023년 3월 30일

RDS Custom에서 Amazon EBS gp3 스토리지 볼륨 지원	RDS Custom for Oracle과 RDS Custom for SQL Server 양쪽 모두에서 io1, gp2, gp3 SSD 기반 EBS 볼륨을 지원합니다. 자세한 내용은 RDS Custom for Oracle 일반 요구 사항 및 RDS Custom for SQL Server 일반 요구 사항을 참조하세요.	2023년 3월 29일
AWS 관리형 정책 권한 업데이트	`AmazonRDSFullAccess` 및 `AmazonRDSReadOnlyAccess` 정책은 이제 Amazon CloudWatch에 추가 권한을 부여합니다. 자세한 내용은 AWS 관리형 정책에 대한 Amazon RDS 업데이트를 참조하세요.	2023년 3월 16일
중국 리전에서 RDS 프록시 사용 가능	이제 중국(베이징) 및 중국(닝샤) 리전에서 RDS 프록시를 사용할 수 있습니다. RDS 프록시에 대한 자세한 내용은 Amazon RDS 프록시 사용을 참조하세요.	2023년 3월 15일
아시아 태평양(자카르타) 리전에서 RDS 프록시 사용 가능	이제 아시아 태평양(자카르타) 리전에서 RDS 프록시를 사용할 수 있습니다. RDS 프록시에 대한 자세한 내용은 Amazon RDS 프록시 사용을 참조하세요.	2023년 3월 8일

Amazon RDS Optimized Writes로 RDS for MariaDB의 쓰기 트랜잭션 성능 개선	Amazon RDS Optimized Writes를 사용하여 RDS for MariaDB DB 인스턴스의 쓰기 트랜잭션 성능을 개선할 수 있습니다. 자세한 내용은 Amazon RDS Optimized Writes for MariaDB를 통한 쓰기 성능 개선을 참조하세요.	2023년 3월 7일
Amazon RDS for PostgreSQL 버전 15.2	Amazon RDS for PostgreSQL 15.2의 새로운 기능에는 조건부 SQL 쿼리를 위한 SQL 표준 'MERGE' 명령, 인메모리 및 디스크 기반 정렬의 성능 향상, 2단계 커밋 및 논리적 복제를 위한 행/열 필터링 지원이 포함됩니다.	2023년 2월 27일
RDS Custom for Oracle을 캐나다(중부) 리전 및 남아메리카(상파울루) 리전에서 사용할 수 있음	지원되는 모든 AWS 리전을 보여 주는 표는 RDS Custom for Oracle을 지원하는 리전 및 DB 엔진을 참조하세요.	2023년 2월 22일
Amazon RDS에서 RDS for MariaDB 및 RDS for MySQL에 대해 크로스 리전 자동 백업 지원	이제 RDS for MariaDB와 RDS for MySQL DB 인스턴스의 AWS 리전 간에 DB 스냅샷 및 트랜잭션 로그를 복제할 수 있습니다. 자세한 내용은 자동 백업을 다른 AWS 리전 리전에 복제 섹션을 참조하세요.	2023년 2월 22일

Amazon RDS for Oracle에서 자동 마이너 버전 업그레이드에 대한 미리 알림 지원	RDS는 RDS for Oracle 엔진의 새 마이너 버전을 사용할 수 있게 되는 날짜 이전에 미리 알려줍니다. RDS는 가용일에 RDS for Oracle DB 인스턴스의 자동 마이너 버전 업그레이드 일정을 잡기 시작합니다. 자세한 내용은 자동 마이너 버전 업그레이드 일정을 예약하기 전을 참조하세요.	2023년 2월 21일
Amazon RDS for SQL Server에서 데이터베이스 활동 스트림 지원	이제 데이터베이스 활동 스트림을 사용하여 SQL Server DB 인스턴스를 모니터링할 수 있습니다. SQL Server 데이터베이스 인스턴스에는 Amazon RDS에서 관리하는 서버 감사가 있습니다. 정책을 정의하여 서버 감사 사양에 서버 이벤트를 기록할 수 있습니다. 데이터베이스 감사 사양을 생성하고 데이터베이스 이벤트를 기록하는 정책을 정의할 수 있습니다. 활동 스트림이 수집되어 Amazon Kinesis에 전송됩니다. Kinesis에서 추가 분석을 위해 활동 스트림을 모니터링할 수 있습니다. 자세한 내용은 데이터베이스 활동 스트림을 사용하여 Amazon RDS 모니터링을 참조하세요.	2023년 2월 15일

RDS에서 MySQL 8.0.32 및 5.7.41 지원	이제 MySQL 버전 8.0.32 및 5.7.41을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하세요.	2023년 2월 7일
Amazon RDS for Oracle에서 새로운 SSL용 암호 그룹 지원	Oracle Database 19c 또는 21c를 실행하는 경우 Oracle용 RDS에 대해 SSL 옵션에서 6개의 새 암호 그룹을 지정할 수 있습니다. 이러한 그룹은 FIPS를 지원하며 FedRAMP와 호환됩니다. 자세한 내용은 Oracle 보안 소켓 레이어를 참조하세요.	2023년 2월 3일
Amazon RDS for Oracle에서 새로운 Oracle Enterprise Manager용 암호 그룹 지원	이 OEM 옵션에 대해 4개의 새 FedRAMP 호환 암호 그룹을 사용할 수 있습니다. 자세한 내용은 Oracle Management Agent for Enterprise Manager Cloud Control을 참조하세요.	2023년 2월 3일
RDS for Oracle이 아시아 태평양(하이데라바드), 유럽(스페인), 중동(UAE) 리전에서 데이터베이스 활동 스트림 지원	자세한 내용은 Amazon RDS에서 데이터베이스 활동 스트림을 지원하는 리전 및 DB 엔진를 참조하세요.	2023년 1월 27일

읽기 전용 복제본을 사용하여 RDS for PostgreSQL 다중 AZ DB 클러스터로 마이그레이션	읽기 전용 복제본을 사용하면 PostgreSQL 단일 AZ 배포 또는 다중 AZ DB 인스턴스 배포용 RDS를 PostgreSQL 다중 AZ DB 클러스터 배포용 RDS로 마이그레이션하여 다운타임을 줄일 수 있습니다. 자세한 내용은 읽기 전용 복제본을 사용하여 다중 AZ DB 클러스터로 마이그레이션을 참조하세요.	2023년 1월 23일
아시아 태평양(멜버른) 리전에서 Amazon RDS 사용 가능	이제 아시아 태평양(멜버른) 리전에서 Amazon RDS를 사용할 수 있습니다. 자세한 내용은 리전 및 가용 영역을 참조하세요.	2023년 1월 23일
RDS for MariaDB에서 SSL/TLS 연결 적용 지원	RDS for MariaDB에서 이제 `require_secure_transport` 파라미터를 ON으로 설정하여 SSL/TLS 연결을 적용할 수 있습니다. 자세한 내용은 MariaDB DB 인스턴스에 대한 모든 연결에 SSL/TLS 연결 요구를 참조하세요.	2023년 1월 19일
Amazon RDS Optimized Reads로 RDS for MariaDB 쿼리 성능 개선	Amazon RDS Optimized Reads로 RDS for MariaDB DB 인스턴스의 쿼리 처리 속도를 높일 수 있습니다. 자세한 내용은 Amazon RDS Optimized Reads를 통한 RDS for MariaDB 쿼리 성능 개선을 참조하세요	2023년 1월 11일

다중 AZ DB 클러스터 스냅샷을 DB 인스턴스에 복원	이제 다중 AZ DB 클러스터 스냅샷을 단일 AZ 배포 또는 다중 AZ DB 인스턴스 배포에 복원할 수 있습니다. 자세한 정보는 다중 AZ DB 클러스터 스냅샷에서 DB 인스턴스로 복원을 참조하세요.	2023년 1월 10일
DB 인스턴스 생성 중 인증 기관(CA) 지정	이제 DB 인스턴스 생성 중에 DB 인스턴스의 서버 인증서에 사용할 CA를 지정할 수 있습니다. 자세한 내용은 인증 기관을 참조하세요.	2023년 1월 5일
RDS Custom for SQL Server에서 사용자 지정 엔진 버전 지원	RDS Custom for SQL Server용 사용자 지정 엔진 버전(CEV)은 Microsoft SQL Server가 사전 설치된 Amazon Machine Image(AMI)입니다. 기본 이미지로 사용할 Amazon EC2 Windows AMI를 선택하고 운영 체제(OS)에 다른 소프트웨어를 설치할 수 있습니다. 엔터프라이즈 요구 사항에 맞게 OS 및 SQL Server의 구성을 사용자 지정할 수 있습니다. 자세한 내용은 RDS Custom for SQL Server용 사용자 지정 엔진 버전 작업을 참조하세요.	2022년 12월 28일
추가 AWS 리전에서 Amazon RDS 블루/그린 배포 사용 가능	이제 중국(베이징) 및 중국(닝샤) 리전에서 블루/그린 배포 기능을 사용할 수 있습니다. 자세한 내용은 데이터베이스 업데이트에 Amazon RDS 블루/그린 배포 사용을 참조하세요.	2022년 12월 22일

IAM 서비스 연결 역할 권한 업데이트	AmazonRDSServiceRolePolicy 정책은 이제 AWS Secrets Manager에 추가 권한을 부여합니다. 자세한 내용은 AWS 관리형 정책에 대한 Amazon RDS 업데이트를 참조하세요.	2022년 12월 22일
Amazon RDS에서 다중 AZ DB 클러스터의 이름 변경 지원	이제 다중 AZ DB 클러스터의 이름을 변경할 수 있습니다. 자세한 내용은 다중 AZ DB 클러스터 이름 변경을 참조하세요.	2022년 12월 22일
Amazon RDS, 암호 관리를 위해 AWS Secrets Manager와 통합	Amazon RDS는 Secrets Manager에서 DB 인스턴스 또는 다중 AZ DB 클러스터의 마스터 사용자 암호를 관리합니다. 자세한 내용은 Amazon RDS와 AWS Secrets Manager를 사용한 암호 관리를 참조하세요.	2022년 12월 22일
Amazon RDS Optimized Writes에서 db.r6g 및 db.r6gd DB 인스턴스 클래스 지원	Amazon RDS Optimized Writes는 이제 db.r6g 및 db.r6gd DB 인스턴스 클래스를 지원합니다. 자세한 내용은 Amazon RDS Optimized Writes를 통한 쓰기 성능 개선을 참조하세요	2022년 12월 22일
Amazon RDS for Oracle에서 새 AWS 리전 지원	아시아 태평양(서울) 및 아시아 태평양(오사카) 리전에서 RDS Custom for Oracle DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 RDS Custom for Oracle을 지원하는 리전 및 DB 엔진을 참조하세요.	2022년 12월 21일

Amazon RDS on AWS Outposts에서 읽기 전용 복제본 지원	이제 RDS on Outposts MySQL 또는 PostgreSQL DB 인스턴스에서 읽기 전용 복제본을 생성할 수 있습니다. 자세한 내용은 Amazon RDS on AWS Outposts 읽기 전용 복제본 생성을 참조하세요.	2022년 12월 19일
RDS Custom for Oracle에서 DB 인스턴스 클래스 수정 지원	이제 RDS Custom for Oracle DB 인스턴스의 인스턴스 클래스를 변경할 수 있습니다. 자세한 내용은 RDS Custom for Oracle DB 인스턴스 수정을 참조하세요.	2022년 12월 16일
RDS for MySQL 및 RDS for PostgreSQL에서 db.x2iedn DB 인스턴스 클래스 지원	이제 RDS for MySQL 및 PostgreSQL DB 인스턴스에 db.x2iedn DB 인스턴스 클래스를 사용할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스에 대해 지원되는 DB 엔진 섹션을 참조하세요.	2022년 12월 14일
Amazon RDS Optimized Writes에서 db.x2iedn DB 인스턴스 클래스 지원	Amazon RDS Optimized Writes는 이제 db.x2iedn DB 인스턴스 클래스를 지원합니다. 자세한 내용은 Amazon RDS Optimized Writes를 통한 쓰기 성능 개선을 참조하세요	2022년 12월 14일
Amazon RDS, DB 스냅샷 복사 시 DB 옵션 그룹 복사 지원	이제 RDS for Oracle 데이터베이스에서 스냅샷 복사 요청의 일부로 여러 AWS 계정에 옵션 그룹을 복사할 수 있습니다. 자세한 내용은 옵션 그룹 고려 사항을 참조하세요.	2022년 12월 13일

Amazon RDS에서 RDS for PostgreSQL 버전 14와 함께 RDS 프록시 지원	이제 RDS for PostgreSQL 버전 14 데이터베이스를 사용하여 RDS 프록시를 생성할 수 있습니다. RDS 프록시에 대한 자세한 내용은 Amazon RDS 프록시 사용을 참조하세요.	2022년 12월 13일
Amazon RDS for Oracle에서 db.x2idn, db.x2iedn, db.x2iezn 인스턴스 클래스 지원	이제 Amazon RDS for Oracle RDS에 db.x2idn, db.x2iedn, db.x2iezn 인스턴스 클래스를 사용할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스에 대해 지원되는 DB 엔진 및 지원되는 RDS for Oracle 인스턴스 클래스를 참조하세요.	2022년 12월 12일
RDS for PostgreSQL 인스턴스, PostgreSQL용 신뢰할 수 있는 언어 확장 지원	PostgreSQL용 신뢰할 수 있는 언어 확장은 고성능 PostgreSQL 확장을 구축하고 RDS for PostgreSQL DB 인스턴스에서 안전하게 실행할 수 있는 오픈 소스 개발 키트입니다. 자세한 내용은 PostgreSQL용 신뢰할 수 있는 언어 확장 작업을 참조하세요.	2022년 11월 30일

데이터베이스 업데이트에 Amazon RDS 블루/그린 배포 사용	스테이징 환경에서 DB 인스턴스를 변경하고 프로덕션 DB 인스턴스에 영향을 주지 않고 변경 사항을 테스트할 수 있습니다. 준비가 되면 가동 중지 시간을 최소화하면서 스테이징 환경을 새로운 프로덕션 데이터베이스 환경으로 승격할 수 있습니다. 자세한 내용은 데이터베이스 업데이트에 Amazon RDS 블루/그린 배포 사용을 참조하세요.	2022년 11월 27일
Amazon RDS Optimized Writes로 RDS for MySQL 쓰기 트랜잭션 성능 개선	Amazon RDS Optimized Writes를 사용하여 RDS for MySQL DB 인스턴스의 쓰기 트랜잭션 성능을 개선할 수 있습니다. 자세한 내용은 Amazon RDS Optimized Writes for MySQL을 통한 쓰기 성능 개선을 참조하세요	2022년 11월 27일
Amazon RDS Optimized Writes로 RDS for MySQL 쿼리 성능 개선	Amazon RDS Optimized Reads로 RDS for MySQL DB 인스턴스의 쿼리 처리 속도를 높일 수 있습니다. 자세한 내용은 Amazon RDS Optimized Writes를 통한 쿼리 성능 개선을 참조하세요	2022년 11월 27일
아시아 태평양(하이데라바드) 리전에서 Amazon RDS 사용 가능	이제 아시아 태평양(하이데라바드) 리전에서 Amazon RDS를 사용할 수 있습니다. 자세한 내용은 리전 및 가용 영역을 참조하세요.	2022년 11월 22일

RDS, MariaDB 10.6.11, 10.4.27, 10.3.37 지원	이제 MariaDB 버전 10.6.11, 10.5.18, 10.4.27, 10.3.37을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하십시오.	2022년 11월 18일
RDS Custom for Oracle, 사용자 지정 엔진 버전(CEV)에서 기본이 아닌 설치 파라미터 설정 지원	CEV를 생성할 때 Oracle base, Oracle home, UNIX 사용자 이름 및 ID, UNIX 그룹 이름 및 ID에 기본값이 아닌 값을 설정할 수 있습니다. 이렇게 하면 RDS Custom for Oracle DB 인스턴스에서 DB 설치를 더 세밀하게 제어할 수 있습니다. 자세한 내용은 CEV 매니페스트 준비를 참조하세요.	2022년 11월 18일
Amazon RDS에서 Oracle APEX 버전 22.1.v1 지원	지원되는 모든 Oracle Database 버전에서 APEX 22.1.v1을 사용할 수 있습니다. 자세한 내용은 Oracle Application Express 단원을 참조하세요.	2022년 11월 18일
RDS for SQL Server에서 리전 간 읽기 전용 복제본 지원	이제 리전 간 읽기 전용 복제본을 생성하여 재해 복구 기능을 향상시키고, 애플리케이션 읽기 지연 시간을 줄이고, 기본 DB 인스턴스에서 읽기 워크로드를 오프로드할 수 있습니다. 자세한 내용은 다른 AWS 리전에 읽기 전용 복제본 생성을 참조하세요.	2022년 11월 16일

유럽(스페인) 리전에서 Amazon RDS 사용 가능	이제 유럽(스페인) 리전에서 Amazon RDS를 사용할 수 있습니다. 자세한 내용은 리전 및 가용 영역을 참조하세요.	2022년 11월 16일
RDS for SQL Server에서 Oracle 데이터베이스용 연결 서버 지원	이제 연결된 서버를 생성하여 외부 Oracle 데이터베이스에 액세스하여 데이터를 읽고 SQL 명령을 실행할 수 있습니다. 자세한 내용은 RDS for SQL Server를 통해 Oracle OLEDB와 연결된 서버를 참조하세요.	2022년 11월 15일
RDS Custom for Oracle에서 Oracle 멀티테넌트 지원	RDS Custom for Oracle DB 인스턴스를 컨테이너 데이터베이스(CDB)로 생성할 수 있습니다. 생성 후 CDB에는 CDB 루트, PDB 시드 및 하나의 PDB가 포함됩니다. Oracle SQL을 사용하여 수동으로 PDB를 추가할 수 있습니다. 자세한 내용은 Amazon RDS Custom for Oracle 아키텍처 개요를 참조하세요.	2022년 11월 15일
Amazon RDS for Oracle에서 Amazon EFS 통합 지원	옵션 그룹에 `EFS_INTEGRATION` 옵션을 추가하면 RDS for Oracle DB 인스턴스와 Amazon EFS 파일 시스템 사이에서 파일을 전송할 수 있습니다. 자세한 내용은 Amazon EFS를 참조하세요.	2022년 11월 15일

RDS에서 MySQL 8.0.31 및 5.7.40 지원	이제 MySQL 버전 8.0.31 및 5.7.40을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하세요.	2022년 11월 10일
유럽(취리히) 리전에서 Amazon RDS 사용 가능	이제 유럽(취리히) 리전에서 Amazon RDS를 사용할 수 있습니다. 자세한 내용은 리전 및 가용 영역을 참조하세요.	2022년 11월 9일
이제 RDS for SQL Server에서 트랜잭션 로그 백업 액세스 가능	이제 데이터베이스 트랜잭션 로그 백업을 보고 Amazon S3 버킷에 복사할 수 있습니다. 자세한 내용은 트랜잭션 로그 백업 액세스를 참조하세요.	2022년 11월 7일
추가 AWS 리전에서 다중 AZ DB 클러스터 지원	이제 추가 AWS 리전에서 다중 AZ DB 클러스터를 사용할 수 있습니다. 자세한 내용은 Amazon RDS에서 다중 AZ DB 클러스터를 지원하는 리전 및 DB 엔진을 참조하세요.	2022년 11월 4일
Amazon RDS에서 gp3 스토리지 지원	이제 Amazon EBS 범용 SSD(gp3) 스토리지 볼륨을 사용하는 Amazon RDS DB 인스턴스를 생성하여 스토리지 용량과 독립적으로 스토리지 성능을 사용자 지정할 수 있습니다. 자세한 내용은 범용 SSD 스토리지를 참조하세요.	2022년 11월 4일

Amazon RDS에서 운영 체제 업데이트를 위한 새로운 이벤트 지원	Amazon RDS는 이제 보안 패치의 이벤트 범주에서 새로운 DB 인스턴스 이벤트인 RDS-EVENT-0230 이벤트를 지원합니다. 이 새 이벤트는 DB 인스턴스에 운영 체제 업데이트가 지원되는 시점을 알려줍니다. 자세한 내용은 Amazon RDS 이벤트 모니터링 및 운영 체제 업데이트 작업을 참조하세요.	2022년 10월 28일
Amazon RDS for Oracle에서 사전 구성된 r5b 메모리 최적화 인스턴스 클래스 지원	db.r5b Oracle DB 인스턴스 클래스는 vCPU 당 추가 메모리, 스토리지 및 I/O가 필요한 워크로드에 최적화되어 있습니다. 예를 들어 db.r5b.4xlarge.tpc2.mem2x는 멀티스레딩이 켜져 있으며, db.r5b.4xlarge보다 두 배 많은 메모리를 제공합니다. 자세한 내용은 RDS for Oracle 인스턴스 클래스를 참조하세요.	2022년 10월 27일
Amazon RDS에서 RDS for MariaDB, MySQL, PostgreSQL DB 인스턴스 읽기 전용 복제본 15개 지원	이제 RDS for MariaDB, MySQL, PostgreSQL DB 인스턴스의 읽기 전용 복제본을 최대 15개까지 생성할 수 있습니다. 읽기 전용 복제본에 대한 자세한 내용은 읽기 전용 복제본 작업을 참조하세요.	2022년 10월 20일

Amazon RDS for PostgreSQL, 이제 데이터베이스 미리 보기 환경에서 PostgreSQL 버전 15 RC 3 지원	이제 PostgreSQL 버전 15 베타 3을 미국 동부(오하이오) AWS 리전의 데이터베이스 미리 보기 환경에서 사용할 수 있습니다. 자세한 내용은 데이터베이스 미리 보기 환경으로 작업을 참조하세요.	2022년 10월 18일
Amazon RDS가 RDS 데이터베이스와 EC2 인스턴스의 연결 자동 설정을 지원함	AWS Management Console을 사용하여 기존 RDS DB 인스턴스 또는 다중 AZ DB 클러스터와 EC2 인스턴스 간에 연결을 설정할 수 있습니다. 자세한 내용은 EC2 인스턴스와 RDS 데이터베이스를 자동으로 연결을 참조하세요.	2022년 10월 14일
AWS JDBC Driver for PostgreSQL을 일반적으로 사용할 수 있게 됨	AWS JDBC Driver for PostgreSQL은 RDS for PostgreSQL을 위해 설계된 클라이언트 드라이버입니다. AWS JDBC Driver for PostgreSQL은 현재 일반적으로 사용 가능합니다. 자세한 내용은 AWS JDBC Driver for PostgreSQL에 연결을 참조하세요.	2022년 10월 6일
Amazon RDS for Oracle이 Oracle APEX 버전 21.2.v1을 지원함	APEX 21.2에는 33420059 패치가 포함됩니다. 자세한 내용은 APEX 버전 요구 사항을 참조하세요.	2022년 10월 3일

RDS가 MySQL 5.7.39를 지원함	이제 MySQL 버전 5.7.39를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하세요.	2022년 9월 29일
RDS가 MariaDB 10.6.10을 지원함	이제 MariaDB 버전 10.6.10을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하십시오.	2022년 9월 29일
RDS 프록시가 RDS for SQL Server를 지원함	이제 Microsoft SQL Server 버전 2014 이상을 실행하는 RDS DB 인스턴스에 대해 RDS 프록시를 생성할 수 있습니다. RDS 프록시에 대한 자세한 내용은 Amazon RDS 프록시 사용을 참조하세요.	2022년 9월 19일
RDS가 MariaDB 10.5.17, 10.4.26, 10.3.36을 지원함	이제 MariaDB 버전 10.5.17, 10.4.26, 10.3.36을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하십시오.	2022년 9월 15일

Amazon RDS for Oracle이 임시 데이터에 대한 로컬 인스턴스 스토리지를 지원함	이제 인스턴스 스토어를 사용하도록 구성된 임시 테이블스페이스 및 데이터베이스 스마트 플래시 캐시(플래시 캐시)를 사용하여 Amazon EC2 db.r5d 및 db.m5d 인스턴스 유형에서 Amazon RDS for Oracle을 시작할 수 있습니다. 임시 데이터를 로컬에 저장함으로써 Amazon EBS를 기반으로 하는 표준 스토리지에 비해 읽기 및 쓰기 지연 시간을 줄일 수 있습니다. 자세한 내용은 RDS for Oracle 인스턴스 스토어에 임시 데이터 저장을 참조하세요.	2022년 9월 14일
성능 개선 도우미에서 상위 25개의 SQL 쿼리를 보여줌	성능 개선 도우미 대시보드에서 상위 SQL 탭은 DB 로드에 가장 많은 기여하는 SQL 쿼리를 보여줍니다. 자세한 내용은 상위 SQL(Top SQL) 탭 개요를 참조하세요.	2022년 9월 13일
RDS가 MySQL 8.0.30을 지원함	이제 MySQL 버전 8.0.30을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하세요.	2022년 9월 9일
중동(UAE) 리전에서 Amazon RDS 사용 가능	중동(UAE) 리전에서 이제 Amazon RDS를 사용할 수 있습니다. 자세한 내용은 리전 및 가용 영역을 참조하세요.	2022년 8월 30일

Amazon RDS for SQL Server는 SSRS 이메일 구독 지원	이제 SSRS(SQL Server Reporting Services) 이메일 확장을 사용하여 사용자에게 보고서를 보내고 보고서 서버의 보고서를 구독할 수 있습니다. 자세한 내용은 RDS for SQL Server의 SQL Server Reporting Services 지원을 참조하세요.	2022년 8월 26일
RDS for Oracle에서 읽기 전용 복제본 백업 지원	자동 백업을 켜고 RDS for Oracle 복제본의 수동 스냅샷을 생성할 수 있습니다. 자세한 내용을 알아보려면 RDS for Postom for Oracle 복제본 백업을 참조하세요.	2022년 8월 23일
RDS for Oracle, Oracle Data Guard 전환 지원	전환은 기본 데이터베이스와 마운트되거나 열려 있는 Oracle 복제본 간의 역할 전환입니다. 전환하는 동안 원래 기본 데이터베이스는 대기 역할로 전환되고 원래 대기 데이터베이스는 기본 역할로 전환됩니다. 자세한 내용을 알아보려면 Oracle Data Postom 전환 섹션을 참조하세요.	2022년 8월 23일

Amazon RDS는 EC2 인스턴스와의 연결 자동 설정을 지원	DB 인스턴스 또는 다중 AZ DB 클러스터를 생성할 때 AWS Management Console에서 Amazon Elastic Compute Cloud 인스턴스와 새 DB 인스턴스 또는 DB 클러스터 간의 연결을 설정할 수 있습니다. 자세한 내용은 새 DB 인스턴스에 대해 EC2 인스턴스와 자동 네트워크 연결 구성 및 새 DB 클러스터에 대해 EC2 인스턴스와 자동 네트워크 연결 구성을 참조하세요.	2022년 8월 22일
RDS Custom for Oracle에서 Oracle 복제본 승격 지원	RDS Custom for Oracle을 사용하는 경우 `promote-read-replica` CLI 명령을 사용하여 관리형 Oracle 복제본을 승격할 수 있습니다. 또한 기본 DB 인스턴스를 삭제할 수 있습니다. 그러면 RDS Custom for Oracle이 관리형 Oracle 복제본을 독립 실행형 인스턴스로 승격할 수 있습니다. 자세한 내용을 알아보려면 RDS for Postom for Oracle의 Oracle 복제본 작업을 참조하세요.	2022년 8월 5일
RDS for MySQL은 SSL/TLS 연결 적용 지원	RDS for MySQL은 이제 `require_secure_transport` 파라미터를 `ON`으로 설정하여 SSL/TLS 연결을 적용합니다. 자세한 내용은 MySQL DB 인스턴스에 대한 SSL/TLS 연결 필요 단원을 참조하세요.	2022년 8월 1일

Amazon RDS에서 Oracle Database 12c 릴리스 1(12.1.0.2)에 대한 지원이 중단됨	버전 12.1.0.2에 대한 지원은 BYOL 및 LI 라이선스 모델 모두에 대해 더 이상 사용되지 않습니다. 2022년 8월 1일에 RDS for Oracle은 12c 릴리스 1(12.1.0.2) DB 인스턴스 및 복원된 12.1.0.2 스냅샷을 Oracle Database 19c로 자동 업그레이드를 시작합니다. 자세한 내용은 AWS re:Post의 지원 종료 타임라인을 참조하세요.	2022년 8월 1일
RDS 프록시에서 RDS for MariaDB 지원	이제 MariaDB 버전 10.2, 10.3, 10.4 또는 10.5를 실행하는 RDS DB 인스턴스용 RDS 프록시를 생성할 수 있습니다. MariaDB 지원은 MySQL 엔진 패밀리에 포함되어 있습니다. RDS 프록시에 대한 자세한 내용은 Amazon RDS 프록시 사용을 참조하세요.	2022년 7월 26일
RDS for MariaDB는 db.r5b DB 인스턴스 클래스 지원	이제 db.r5b DB 인스턴스 클래스를 사용하는 MariaDB DB 인스턴스용 RDS를 생성할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스에 대해 지원되는 DB 엔진 섹션을 참조하세요.	2022년 7월 25일

RDS for Oracle에서 데이터베이스 활동 스트림 지원	RDS for Oracle을 사용하는 경우 데이터베이스 작업 스트림의 감사 정책 상태를 잠김(기본값) 또는 잠금 해제된 상태로 변경할 수 있습니다. 활동 스트림을 중지하는 대신 해당 정책 상태를 잠금 해제하고 감사 정책을 사용자 지정한 다음 정책 상태를 다시 잠글 수 있습니다. 자세한 내용은 데이터베이스 활동 스트림 수정 단원을 참조하세요.	2022년 7월 22일
아시아 태평양(자카르타) 리전에서 아시아 태평양 지역 지원	이전에는 아시아 태평양(자카르타) 리전에서 성능 개선 도우미를 사용할 수 없었습니다. 이 제한 사항은 제거되었습니다. 자세한 내용은 Amazon RDS에서 성능 개선 도우미를 지원하는 리전 및 DB 엔진를 참조하세요.	2022년 7월 21일
Amazon RDS에서 Microsoft SQL Server 2012 지원 종료	Microsoft SQL Server 2012는 2022년 7월 12일부터 이 버전에 대한 확장 지원 중단 계획에 따라 지원이 종료되었습니다. 기존 Microsoft SQL Server 2012 인스턴스는 2022년 6월 1일부터 Microsoft SQL Server 2014의 최신 마이너 버전으로 자동 업그레이드됩니다. 자세한 내용은 Amazon RDS에서 Microsoft SQL Server 2012 지원을 참조하세요.	2022년 7월 12일

RDS는 MariaDB 10.6.8, 10.5.16, 10.4.25, 10.3.35 및 10.2.44 지원	이제 MariaDB 버전 10.6.8, 10.5.16, 10.4.25, 10.3.35 및 10.2.44를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하세요.	2022년 7월 8일
추가 보존 기간을 지원하는 RDS 성능 개선 도우미	이전에 성능 개선 도우미는 7일(기본값) 또는 2년(731일)의 두 가지 보존 기간만 제공했습니다. 이제 실적 데이터를 7일 이상 보존해야 하는 경우 기간을 1~24개월로 지정할 수 있습니다. 자세한 내용은 성능 개선 도우미 위한 가격 및 데이터 보존을 참조하세요.	2022년 7월 1일
RDS Custom의 아시아 태평양(뭄바이) 및 유럽(런던) 리전 지원	새로운 아시아 태평양(뭄바이) 및 유럽(런던) AWS 리전에서 RDS Custom for Oracle 및 RDS Custom for SQL Server DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 RDS Custom for Oracle의 AWS 리전 지원 및 RDS Custom for SQL Server의 AWS 리전 지원을 참조하세요.	2022년 6월 21일

RDS Custom for Oracle의 Oracle Database 18c 및 12c Release 2(12.2) 지원	이제 Oracle Database 18c 및 12c Release 2(12.2)용 설치 파일을 사용하여 RDS Custom for Oracle에 대한 CEV를 생성할 수 있습니다. 이 CEV를 사용하여 RDS Custom for Oracle DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 사용자 지정 엔진 버전의 Amazon RDS Custom for Oracle로 작업을 참조하세요.	2022년 6월 21일
다중 AZ DB 클러스터의 db.m5d 및 db.r5d DB 인스턴스 클래스 지원	이제 db.m5d 및 db.r5d DB 인스턴스 클래스를 사용하는 다중 AZ DB 클러스터를 생성할 수 있습니다. 자세한 내용은 다중 AZ DB 클러스터 배포 및 DB 인스턴스 클래스 유형을 참조하세요.	2022년 6월 21일
추가 AWS 리전에서 사용 가능한 다중 AZ DB 클러스터	이제 유럽(프랑크푸르트) 및 유럽(스톡홀름) 리전에서 다중 AZ DB 클러스터를 생성할 수 있습니다. 자세한 내용은 다중 AZ DB 클러스터 배포를 참조하세요.	2022년 6월 21일
RDS for Microsoft SQL Server의 투명한 데이터 암호화(TDE)를 사용하는 데이터베이스 마이그레이션 지원	이제 RDS for SQL Server에서 기본 백업 및 복원을 사용하는 TDE를 설정한 Microsoft SQL Server 데이터베이스 마이그레이션을 지원합니다. 자세한 내용은 SQL Server의 투명한 데이터 암호화 지원을 참조하세요.	2022년 6월 14일

Amazon RDS의 암호화된 Amazon SNS 주제에 대한 이벤트 게시 지원	Amazon RDS는 이제 서버 측 암호화(SSE)가 활성화된 Amazon Simple Notification Service(Amazon SNS) 주제에 이벤트를 게시하여 민감한 데이터를 전달하는 이벤트를 추가로 보호할 수 있습니다. 자세한 내용은 Amazon RDS 이벤트 알림 구독을 참조하세요.	2022년 6월 1일
RDS의 MySQL 5.7.38 지원	이제 MySQL 버전 5.7.38을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하세요.	2022년 5월 31일
RDS for PostgreSQL에서 계단식 읽기 전용 복제본 지원	이제 RDS for PostgreSQL 버전 14.1 이상 릴리스와 함께 계단식 읽기 전용 복제본을 사용할 수 있습니다. 자세한 내용은 Amazon RDS에서 PostgreSQL 읽기 전용 복제본 작업을 참조하세요.	2022년 5월 4일
AWS Outposts 기반 Amazon RDS에서 크기 조정 스토리지 및 자동 크기 조정 작업 지원	이제 Outpost에서 DB 인스턴스의 스토리지 크기를 변경하고 스토리지 크기를 자동으로 조정할 수 있습니다. 자세한 내용은 Amazon RDS 기능에 대한 AWS Outposts 기반 Amazon RDS 지원을 참조하세요.	2022년 5월 2일

추가 AWS 리전에서 사용 가능한 다중 AZ DB 클러스터	이제 아시아 태평양(싱가포르) 및 아시아 태평양(시드니) 리전에서 다중 AZ DB 클러스터를 생성할 수 있습니다. 자세한 내용은 다중 AZ DB 클러스터 배포를 참조하세요.	2022년 4월 29일
듀얼 스택 모드를 지원하는 Amazon RDS	이제 DB 인스턴스를 듀얼 스택 모드로 실행할 수 있습니다. 듀얼 스택 모드에서 리소스는 IPv4, IPv6 또는 모두를 통해 DB 인스턴스 클러스터와 통신할 수 있습니다. 자세한 내용은 Amazon RDS IP 주소 지정을 참조하세요.	2022년 4월 29일
Amazon RDS에서 Amazon CloudWatch에 사용량 지표 게시	Amazon CloudWatch의 `AWS/Usage` 네임스페이스에는 Amazon RDS 서비스 할당량에 대한 계정 수준 사용량 지표가 포함됩니다. 자세한 내용은 Amazon RDS에 대한 Amazon CloudWatch 사용 지표 섹션을 참조하세요.	2022년 4월 28일
Amazon RDS for MySQL에서 db.m6i 및 db.r6i DB 인스턴스 클래스 지원	이제 MySQL을 실행하는 Amazon RDS DB 인스턴스에 db.m6i 및 db.r6i DB 인스턴스 클래스를 사용할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스에 대해 지원되는 DB 엔진 섹션을 참조하세요.	2022년 4월 28일

Amazon RDS for PostgreSQL에서 db.r5 DB 인스턴스 클래스 지원	이제 PostgreSQL을 실행하는 Amazon RDS DB 인스턴스에 db.m6i 및 db.r6i DB 인스턴스 클래스를 사용할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스에 대해 지원되는 DB 엔진 섹션을 참조하세요.	2022년 4월 27일
Amazon RDS for PostgreSQL은 db.m6i 및 db.r6i DB 인스턴스 클래스 지원	이제 MariaDB를 실행하는 Amazon RDS DB 인스턴스에 db.m6i 및 db.r6i DB 인스턴스 클래스를 사용할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스에 대해 지원되는 DB 엔진 섹션을 참조하세요.	2022년 4월 26일
다중 AZ 배포를 지원하는 Amazon RDSAWS Outposts	이제 다른 Outpost에 대기하는 DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS 기능에 대한 AWS Outposts 기반 Amazon RDS 지원을 참조하세요.	2022년 4월 19일
Amazon RDS for Oracle에서 db.m6i 및 db.r6i 인스턴스 클래스	Oracle 데이터베이스 19c를 실행하는 경우 db.m6i 및 db.r6i 인스턴스 클래스를 사용할 수 있습니다. db.m6i 클래스는 광범위한 워크로드에 적합한 범용 인스턴스 클래스입니다. 자세한 내용은 RDS for Oracle 인스턴스 클래스를 참조하세요.	2022년 4월 8일

Amazon RDS for SQL Server에서 SQL Server 에이전트 작업 복제 지원	이 기능을 켜면 기본 호스트에서 생성, 수정 또는 삭제된 SQL Server 에이전트 작업이 다중 AZ 구성의 보조 호스트에 자동으로 동기화됩니다. 자세한 내용은 SQL Server 사용을 참조하세요.	2022년 4월 7일
Amazon RDS에서 RDS for PostgreSQL 버전 13과 함께 RDS 프록시 지원	이제 RDS for PostgreSQL 버전 13 데이터베이스를 사용하여 RDS 프록시를 생성할 수 있습니다. RDS 프록시에 대한 자세한 내용은 Amazon RDS 프록시 사용을 참조하세요.	2022년 4월 4일
Amazon RDS에서 Oracle Database 12c 지원 중단 예정	Oracle Database 12c는 지원 중단될 예정입니다. Oracle Corporation은 지원 종료일 이후 Oracle Database 12c 릴리스에 대한 패치를 더 이상 제공하지 않습니다. Amazon RDS는 Oracle Database 12c DB 인스턴스를 Oracle Database 19c로 자동 업그레이드하기 시작할 계획입니다.	2022년 3월 22일
Amazon RDS for PostgreSQL 릴리스 정보	이제 Amazon RDS for PostgreSQL 릴리스 정보에 대한 별도의 가이드가 제공됩니다. 자세한 내용은 Amazon RDS for PostgreSQL 릴리스 정보를 참조하세요.	2022년 3월 22일

Amazon RDS for Oracle 릴리스 정보	이제 Amazon RDS for Oracle 릴리스 정보에 대한 별도의 가이드가 제공됩니다. 자세한 내용은 Amazon RDS for Oracle 릴리스 정보를 참조하세요.	2022년 3월 22일
추가 AWS 리전에서 사용 가능한 다중 AZ DB 클러스터	이제 미국 동부(오하이오) 및 아시아 태평양(도쿄) 리전에서 다중 AZ DB 클러스터를 생성할 수 있습니다. 자세한 내용은 다중 AZ DB 클러스터 배포를 참조하세요.	2022년 3월 15일
Amazon RDS for PostgreSQL 버전 14.2, 13.6, 12.10, 11.15 및 10.20	RDS for PostgreSQL은 이제 버전 14.2, 13.6, 12.10, 11.15 및 10.20을 지원합니다. 버전 14.2 및 13.6에는 두 개의 새로운 외부 데이터 래퍼에 대한 지원이 추가되었습니다. `mysql_fdw` 확장을 통해 PostgreSQL에서 MySQL, MariaDB 및 Aurora MySQL 데이터베이스에 저장된 데이터를 사용할 수 있습니다. `tds_fdw` 확장을 통해 PostgreSQL에서 SQL Server 데이터베이스에 저장된 데이터를 사용할 수 있습니다. 자세한 내용은 지원되는 PostgreSQL 데이터베이스 버전을 참조하세요.	2022년 3월 12일
RDS에서 MySQL 5.7.37 지원	이제 MySQL 버전 5.7.37을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하세요.	2022년 3월 11일

Amazon RDS for SQL Server에서 새로운 DB 인스턴스 클래스 지원	이제 db.m6i 및 db.r6i DB 인스턴스 클래스를 사용하며 Microsoft SQL Server를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Microsoft SQL Server에 대한 DB 인스턴스 클래스 지원을 참조하세요.	2022년 3월 9일
Amazon RDS for Oracle에서 Oracle Database 21c 지원	이제 Oracle Database 21c(21.0.0.0)를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 이는 멀티테넌트(CDB) 아키텍처만 지원하는 첫 번째 Oracle Database 릴리스입니다. 자세한 내용은 Amazon RDS 기반 Oracle 21c를 참조하세요.	2022년 3월 7일
RDS에서 MariaDB 10.6.7, 10.5.15, 10.4.24, 10.3.34 및 10.2.43 지원	이제 MariaDB 버전 10.6.7, 10.5.15, 10.4.24, 10.3.34, 및 10.2.43을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하세요.	2022년 3월 3일
일반적으로 사용 가능한 MySQL용 AWS JDBC 드라이버	AWS JDBC Driver for MySQL은 RDS for MySQL을 위해 설계된 클라이언트 드라이버입니다. MySQL용 AWS JDBC 드라이버는 현재 일반적으로 사용 가능합니다. 자세한 내용은 MySQL용 Amazon Web Services JDBC 드라이버로 연결을 참조하세요.	2022년 3월 2일

일반적으로 사용 가능한 다중 AZ DB 클러스터	다중 AZ DB 클러스터 배포는 읽을 수 있는 대기 DB 인스턴스가 두 개 있는 Amazon RDS의 고가용성 배포 모드입니다. 다중 AZ DB 클러스터는 이제 일반적으로 사용 가능합니다. 자세한 내용은 다중 AZ DB 클러스터 배포를 참조하세요.	2022년 3월 1일
RDS에서 MySQL 8.0.28 지원	이제 MySQL 버전 8.0.28을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하세요.	2022년 2월 28일
Amazon RDS for Oracle은 기본 네트워크 암호화(NNE)에 대한 신규 설정을 지원합니다.	클라이언트가 비보안 암호화 및 체크섬 방법으로 연결할 수 있는지 여부를 제어하려면 NNE 옵션에서 `SQLNET.ALLOW_WEAK_CRYPTO_CLIENTS` 및 `SQLNET.ALLOW_WEAK_CRYPTO` 를 설정합니다. 안전하지 않은 방법의 예로는 DES, 3DES, RC4 및 MD5 등이 있습니다. 자세한 내용은 NNE 옵션 세팅을 참조하세요.	2022년 2월 25일
Amazon RDS for SQL Server, 이제 Microsoft SQL Server 2017 Standard Edition의 Always On 가용성 그룹 지원	SQL Server 2017에서 Standard Edition 14.00.3401.7 이상 버전용 다중 AZ 구성을 사용하여 DB 인스턴스를 생성하면 RDS가 자동으로 가용성 그룹을 사용합니다. 자세한 내용은 Microsoft SQL Server의 다중 AZ 배포를 참조하세요.	2022년 2월 18일

아시아 태평양(자카르타) 리전의 RDS for Oracle에서 데이터베이스 활동 스트림 지원	자세한 내용은 데이터베이스 활동 스트림에 대한 AWS 리전 지원을 참조하세요.	2022년 2월 16일
Oracle 데이터베이스 12.1에 대한 Oracle용 Amazon RDS Custom 지원	이제 Oracle 데이터베이스 12.1 Enterprise Edition을 사용하는 RDS Custom for Oracle용 사용자 정의 엔진 버전을 생성할 수 있습니다. 자세한 내용은 사용자 지정 엔진 버전의 Amazon RDS Custom for Oracle로 작업을 참조하세요.	2022년 2월 4일
Amazon RDS for MariaDB는 새로운 메이저 버전 지원	이제 MariaDB 버전 10.6을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS에서 MariaDB 10.6 지원을 참조하세요.	2022년 2월 3일

성능 개선 도우미에서 Oracle 쿼리에 대한 계획 캡처 지원	이제 성능 개선 도우미 콘솔에서 최상위 SQL에 대한 새로운 계획 차원을 지원합니다. 계획을 기준으로 분할하면 최상위 Oracle 쿼리가 사용 중인 계획을 볼 수 있습니다. 쿼리가 여러 계획을 사용하는 경우 콘솔에서 계획을 나란히 비교하고 가장 효율적인 계획을 결정할 수 있습니다. 드릴다운하여 계획에서 비용이 가장 높은 단계를 확인할 수도 있습니다. 자세한 내용은 Analyzing Oracle execution plans using the Performance Insights dashboard(성능 개선 도우미 대시보드를 사용하여 Oracle 실행 계획 분석)를 참조하세요.	2022년 1월 27일
성능 개선 도우미에서 새로운 API 지원	성능 개선 도우미에서 `GetResourceMetadata` , `ListAvailableResourceDimensions` 및 `ListAvailableResourceMetrics` API를 지원합니다. 자세한 내용은 이 설명서의 성능 개선 도우미 API를 사용하여 지표 검색과 Amazon RDS Performance Insights API Reference(Amazon RDS 성능 개선 도우미 API 레퍼런스)를 참조하세요.	2022년 1월 12일

RDS 프록시에서 이벤트 지원	이제 RDS 프록시가 CloudWatch Events에서 구독하고 보거나 Amazon EventBridge로 전송하도록 구성할 수 있는 이벤트를 생성합니다. 자세한 내용은 RDS 프록시 이벤트 작업을 참조하세요.	2022년 1월 11일
Amazon RDS for SQL Server에서 SSAS 다차원 모드 지원	RDS for SQL Server는 테이블 형식 또는 다차원 모드에서 SQL Server Analysis Services(SSAS) 실행을 지원합니다. 자세한 내용은 Amazon RDS for SQL Server에서 SQL Server Analysis Services 지원을 참조하세요.	2022년 1월 7일
추가 AWS 리전에서 RDS 프록시 사용 가능	이제 아프리카(케이프타운), 아시아 태평양(홍콩), 아시아 태평양(홍콩), 아시아 태평양(오사카), 유럽(밀라노), 유럽(파리), 유럽(파리), 유럽(스톡홀름), 중동(바레인) 및 남아메리카(상파울루) 리전에서 RDS 프록시를 사용할 수 있습니다. RDS 프록시에 대한 자세한 내용은 Amazon RDS 프록시 사용을 참조하세요.	2022년 1월 5일
RDS에서 MySQL 8.0.27 지원	이제 MySQL 버전 8.0.27을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하세요.	2021년 12월 21일

아시아 태평양(자카르타) 리전에서 Amazon RDS 사용 가능	아시아 태평양(자카르타) 리전에서 Amazon RDS를 사용할 수 있습니다. 자세한 내용은 리전 및 가용 영역을 참조하세요.	2021년 12월 13일
Amazon RDS에서 MariaDB 10.5.13, 10.4.22, 10.3.32 및 10.2.41 지원	이제 MariaDB 버전 10.5.13, 10.4.22, 10.3.32 및 10.2.41을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하세요.	2021년 12월 8일
SQL Server용 Amazon RDS Custom	Amazon RDS Custom은 기본 운영 체제 및 데이터베이스 환경에 액세스해야 하는 레거시, 사용자 지정 및 패키지 애플리케이션을 위한 관리형 데이터베이스 서비스입니다. Amazon RDS Custom을 사용하면 Amazon RDS의 자동화와 Amazon EC2의 유연성이라는 이점을 모두 누릴 수 있습니다. 자세한 내용은 Amazon RDS Custom으로 작업을 참조하세요.	2021년 12월 1일

다중 AZ DB 클러스터(미리 보기)	이제 RDS for MySQL 및 RDS for PostgreSQL에 대해 다중 AZ DB 클러스터를 생성할 수 있습니다. 다중 AZ DB 클러스터 배포는 읽을 수 있는 대기 DB 인스턴스가 두 개 있는 Amazon RDS의 고가용성 배포 모드입니다. 다중 AZ DB 클러스터는 미리 볼 수 있습니다. 자세한 내용은 다중 AZ DB 클러스터 배포(미리 보기)를 참조하세요.	2021년 11월 23일
Amazon RDS에서 RDS for PostgreSQL 버전 12와 함께 RDS 프록시 지원	이제 RDS for PostgreSQL 버전 12 데이터베이스를 사용하여 RDS 프록시를 생성할 수 있습니다. RDS Proxy에 대한 자세한 내용은 Amazon RDS Proxy 사용을 참조하세요.	2021년 11월 22일
AWS Outposts 기반 Amazon RDS에서 로컬 백업 지원	자동 백업 및 수동 스냅샷을 AWS 리전에 저장하거나 Outpost에 로컬로 저장할 수 있습니다. 자세한 내용은 Amazon RDS 기능에 대한 AWS Outposts 기반 Amazon RDS 지원을 참조하세요.	2021년 11월 22일
Amazon RDS에서 계정 간 AWS KMS keys 지원	Amazon S3로 DB 스냅샷을 내보내는 경우 다른 AWS 계정의 KMS 키를 사용하여 암호화할 수 있습니다. 자세한 내용은 Amazon S3로 DB 스냅샷 데이터 내보내기를 참조하세요.	2021년 11월 3일

AWS Outposts의 Amazon RDS에서 CloudWatch Logs에 데이터베이스 엔진 로그 게시 지원	이제 Outposts 기반 RDS에서 CloudWatch Logs에 데이터베이스 엔진 로그를 게시할 수 있도록 지원합니다. 자세한 내용은 Amazon RDS 기능에 대한 Amazon RDS on AWS Outposts 지원 섹션을 참조하세요.	2021년 11월 2일
Oracle용 Amazon RDS Custom	Amazon RDS Custom은 기본 운영 체제 및 데이터베이스 환경에 액세스해야 하는 레거시, 사용자 지정 및 패키지 애플리케이션을 위한 관리형 데이터베이스 서비스입니다. Amazon RDS Custom을 사용하면 Amazon RDS의 자동화와 Amazon EC2의 유연성이라는 이점을 모두 누릴 수 있습니다. 자세한 내용은 Amazon RDS Custom으로 작업을 참조하세요.	2021년 10월 26일
RDS for MySQL 버전 8.0에 대한 지연 복제 지원	RDS for MySQL 버전 8.0.26을 시작으로 RDS for MySQL 버전 8.0 DB 인스턴스에 대한 지연 복제를 구성할 수 있습니다. 자세한 내용은 MySQL을 사용한 지연 복제 구성을 참조하세요.	2021년 10월 25일
MySQL 8.0.26 지원	이제 MySQL 버전 8.0.26을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하세요.	2021년 10월 25일

RDS for MySQL 버전 8.0에 대한 GTID 기반 복제 지원	RDS for MySQL 버전 8.0.26을 시작으로 RDS for MySQL 버전 8.0 DB 인스턴스에 대한 GTID 기반 복제를 구성할 수 있습니다. 자세한 내용은 RDS for MySQL에 대한 GTID 기반 복제 사용을 참조하세요.	2021년 10월 25일
Amazon RDS에서 RDS for MySQL 8.0과 함께 RDS 프록시 지원	이제 RDS for MySQL 8.0 데이터베이스 인스턴스의 RDS 프록시를 생성할 수 있습니다. 자세한 내용은 Amazon RDS 프록시 사용을 참조하세요.	2021년 10월 21일
AWS Outposts 기반 Amazon RDS에서 추가 RDS for MySQL 버전 지원	이제 Outposts 기반 RDS에서 RDS for MySQL 버전 8.0.23 및 8.0.25를 지원합니다. 자세한 내용은 Amazon RDS 기능에 대한 Amazon RDS on AWS Outposts 지원 섹션을 참조하세요.	2021년 10월 20일
이제 Amazon RDS for PostgreSQL이 데이터베이스 미리 보기 환경에서 PostgreSQL 버전 14 RC 1을 지원합니다	이제 PostgreSQL 버전 14 RC 1을 미국 동부(오하이오) AWS 리전의 데이터베이스 미리 보기 환경에서 사용할 수 있습니다. 자세한 내용은 데이터베이스 미리 보기 환경으로 작업을 참조하세요.	2021년 10월 19일

Amazon RDS이 추가 AWS 리전에서 성능 개선 도우미 지원	성능 개선 도우미는 중동(바레인), 아프리카(케이프타운), 유럽(밀라노) 및 아시아 태평양(오사카) 리전에서 사용할 수 있습니다. 자세한 내용은 Amazon RDS에서 성능 개선 도우미를 지원하는 리전 및 DB 엔진를 참조하세요.	2021년 10월 5일
성능 개선 도우미에서 Oracle용 다이제스트 수준 통계 지원	성능 개선 도우미를 사용하면 Amazon RDS for Oracle의 문 및 다이제스트 수준 모두에서 SQL 통계를 볼 수 있습니다. 자세한 내용은 Oracle에서 실행 중인 쿼리 분석을 참조하세요.	2021년 10월 4일
Amazon RDS on AWS Outposts에서 추가 RDS for PostgreSQL 버전 지원	이제 RDS on Outposts에서 RDS for PostgreSQL 버전 12.8 및 13.4를 지원합니다. 자세한 내용은 Amazon RDS 기능에 대한 Amazon RDS on AWS Outposts 지원 섹션을 참조하세요.	2021년 10월 1일
Amazon RDS에서 Oracle APEX 버전 21.1.v1 지원	지원되는 모든 Oracle Database 버전에서 APEX 21.1.v1을 사용할 수 있습니다. 자세한 내용은 Oracle Application Express 단원을 참조하세요.	2021년 9월 24일

Amazon RDS for Oracle에서 NNE용 클라이언트 측 암호화 지원	NNE 구성 시 서버 측에서 암호화를 강제하지 않도록 할 수 있습니다. 예를 들어 서버에 필요하기 때문에 모든 클라이언트 통신에 암호화를 사용하도록 강제하지 않을 수 있습니다. 이 경우 SQLNET.*CLIENT 옵션을 사용하여 클라이언트 측에서 암호화를 강제할 수 있습니다. 자세한 내용은 Oracle 기본 네트워크 암호화 옵션을 참조하세요.	2021년 9월 24일
Amazon RDS for MySQL 및 RDS for PostgreSQL에서 새 DB 인스턴스 클래스 지원	이제 db.r5b, db.t4g 및 db.x2g 인스턴스 클래스를 사용하여 MySQL 또는 PostgreSQL을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스에 대해 지원되는 DB 엔진 섹션을 참조하세요.	2021년 9월 15일
Amazon RDS for Microsoft SQL Server에서 Microsoft Distributed Transaction Coordinator(MSDTC)를 통한 Java Database Connectivity(JDBC) 지원	이제 JDBC XA 트랜잭션은 SQL Server 2017 버전 14.00.3223.3 이상 및 SQL Server 2019에 대한 MSDTC와 함께 지원됩니다. 자세한 내용은 RDS for SQL Server에서 Microsoft Distributed Transaction Coordinator 지원을 참조하세요.	2021년 9월 7일

Amazon RDS에서 MariaDB 버전 10.5.12, 10.4.21, 10.3.31 및 10.2.40 지원	이제 MariaDB 버전 10.5.12, 10.4.21, 10.3.31 및 10.2.40을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하세요.	2021년 9월 2일
Amazon RDS는 Oracle Database 18c에 대한 지원을 종료했습니다.	Oracle Database 12c 및 Oracle Database 19c에 대해서만 DB 인스턴스를 생성할 수 있습니다. Oracle Database 18c 스냅샷이 있는 경우 이후 릴리스로 업그레이드하세요. 자세한 내용은 Oracle DB 스냅샷 업그레이드를 참조하세요.	2021년 8월 17일
Amazon RDS for SQL Server는 자동 마이너 버전 업그레이드를 지원합니다.	이제 RDS for SQL Server DB 인스턴스를 최신 마이너 버전으로 자동 업그레이드하도록 설정할 수 있습니다. 자세한 내용은 Microsoft SQL Server DB 엔진 업그레이드를 참조하세요.	2021년 8월 13일
이제 Amazon RDS for PostgreSQL은 데이터베이스 미리 보기 환경에서 PostgreSQL 버전 14 베타 2를 지원합니다.	PostgreSQL 버전 14 베타 1에 대한 자세한 내용은 PostgreSQL 14 베타 1 릴리스 정보를 참조하세요. PostgreSQL 버전 14 베타 2에 대한 자세한 내용은 PostgreSQL 14 베타 2 릴리스 정보를 참조하세요. 데이터베이스 미리 보기 환경에 대한 자세한 내용은 데이터베이스 미리 보기 환경 작업을 참조하세요.	2021년 8월 9일

Amazon RDS, 공유 VPC에서 RDS 프록시 지원	이제 공유 VPC에 RDS 프록시를 만들 수 있습니다. RDS 프록시에 대한 자세한 내용은 Amazon RDS 사용 설명서 또는 Aurora 사용 설명서에서 'Amazon RDS 프록시를 사용한 연결 관리'를 참조하세요.	2021년 8월 6일
Amazon RDS, MariaDB 10.2.39 지원	이제 MariaDB 버전 10.2.39를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하세요.	2021년 8월 4일
Amazon RDS for Oracle, `TIMEZONE_FILE_AUTOUPGRADE` 옵션 추가	이 옵션을 사용하면 현재 시간대 파일을 Oracle DB 인스턴스의 최신 버전으로 업그레이드할 수 있습니다. 자세한 내용은 Oracle 시간대 파일 자동 업그레이드를 참조하세요.	2021년 7월 30일
Amazon RDS는 교차 리전 자동 백업에 대한 지원 확장	이제 더 많은 AWS 리전 간에 DB 스냅샷 및 트랜잭션 로그를 복제할 수 있습니다. 자세한 내용은 자동 백업을 다른 AWS 리전에 복제 섹션을 참조하세요.	2021년 7월 19일
MySQL 5.7.34 지원	이제 MySQL 버전 5.7.34를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하십시오.	2021년 7월 8일

Amazon RDS on AWS Outposts에서 추가 RDS for PostgreSQL 버전 지원	이제 RDS on Outposts에서 RDS for PostgreSQL 버전 12.7 및 13.3을 지원합니다. 자세한 내용은 Amazon RDS 기능에 대한 Amazon RDS on AWS Outposts 지원 섹션을 참조하세요.	2021년 7월 8일
Amazon RDS for PostgreSQL에서 Oracle_fdw	이제 oracle_fdw 확장을 사용하여 Oracle 데이터베이스에 액세스할 수 있는 외부 데이터 래퍼를 제공할 수 있습니다. 자세한 내용은 oracle_fdw 확장으로 외부 데이터 액세스를 참조하세요.	2021년 7월 8일
Amazon RDS에서 Oracle Management Agent(OMA) 버전 13.5 지원	Oracle Enterprise Manager(OEM) Cloud Control 13c 릴리스 5 이상으로 Oracle Management Agent(OMA) 버전 13.5를 사용할 수 있습니다. Amazon RDS for Oracle은 Oracle Management Service(OMS)와 통신하여 모니터링 정보를 제공하는 OMA를 설치합니다. OMS 13.5를 실행하는 경우 OMA 13.5를 설치하여 데이터베이스를 관리할 수 있습니다. 자세한 내용은 Oracle Management Agent for Enterprise Manager Cloud Control을 참조하세요.	2021년 7월 7일

Amazon RDS for Oracle에서 Amazon S3에서 로그 다운로드 지원	아카이브된 다시 실행 로그가 인스턴스에 없지만 백업 보존 기간으로 보호되는 경우 `rdsadmin.rdsadmin_archive_log_download` 를 사용하여 Amazon S3에서 다운로드할 수 있습니다. RDS for Oracle은 DB 인스턴스의 `/rdsdbdata/log/arch` 디렉터리에 로그를 저장합니다. 자세한 내용은 Amazon S3에서 아카이빙된 다시 실행 로그 다운로드를 참조하세요.	2021년 7월 2일
Amazon RDS에서 MariaDB 10.4.18 및 10.5.9 지원	이제 MariaDB 버전 10.4.18 및 10.5.9를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하십시오.	2021년 6월 30일
Amazon RDS for Oracle에서 데이터베이스 활동 스트림 지원	이제 데이터베이스 활동 스트림을 사용하여 Oracle DB 인스턴스를 모니터링할 수 있습니다. Oracle 데이터베이스는 통합 감사 추적에 감사 레코드를 기록합니다. Oracle DB 인스턴스에서 데이터베이스 활동 스트림을 시작하면 Amazon Kinesis는 Oracle 데이터베이스 감사 정책과 일치하는 모든 활동을 스트리밍합니다. 자세한 내용은 데이터베이스 활동 스트림을 사용하여 Amazon RDS 모니터링을 참조하세요.	2021년 6월 23일

Amazon RDS for Oracle에서 메모리 최적화 인스턴스 클래스 도입	새로운 Oracle DB 인스턴스 클래스는 vCPU 당 추가 메모리, 스토리지 및 I/O가 필요한 워크로드에 최적화되어 있습니다. 자세한 내용은 RDS for Oracle 인스턴스 클래스를 참조하세요.	2021년 6월 23일
MySQL 8.0.25 지원	이제 MySQL 버전 8.0.25를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하십시오.	2021년 6월 18일
AWS Outposts 기반 Amazon RDS에서 추가 RDS for PostgreSQL 버전 지원	이제 RDS on Outposts에서 RDS for PostgreSQL 버전 12.5, 12.6, 13.1 및 13.2를 지원합니다. 자세한 내용은 Amazon RDS 기능에 대한 Amazon RDS on AWS Outposts 지원 섹션을 참조하세요.	2021년 5월 28일
Amazon RDS에서 MariaDB 버전 10.2.37 및 10.3.28 지원	이제 MariaDB 버전 10.2.37 및 10.3.28을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하십시오.	2021년 5월 27일

Amazon RDS for Oracle, 멀티테넌트 컨테이너 데이터베이스(CDB) 지원	멀티테넌트 아키텍처에서는 Oracle 데이터베이스를 CDB로 사용할 수 있습니다. Oracle Database 19c에서는 CDB에 단일 PDB를 포함할 수 있습니다. PDB의 사용자 경험은 대부분 비 CDB에서의 경험과 동일합니다. 자세한 내용은 RDS for Oracle 아키텍처를 참조하세요.	2021년 5월 25일
AWS Outposts 기반 Amazon RDS에서 Amazon RDS for SQL Server 지원	RDS on Outposts는 이제 Amazon RDS for SQL Server를 지원합니다. 자세한 내용은 Amazon RDS 기능에 대한 Amazon RDS on AWS Outposts 지원 섹션을 참조하세요.	2021년 5월 11일
Amazon RDS에서 교차 리전 자동 백업에 대한 지원 확장	이제 Microsoft SQL Server를 실행하는 Amazon RDS 데이터베이스 인스턴스를 구성하여 DB 스냅샷과 트랜잭션 로그를 다른 AWS 리전으로 복제할 수 있습니다. 자세한 내용은 자동 백업을 다른 AWS 리전에 복제 섹션을 참조하세요.	2021년 5월 7일
Amazon RDS에서 암호화된 DB 인스턴스에 대해 교차 리전 자동 백업 지원	이제 Oracle 또는 PostgreSQL을 실행하는 암호화된 Amazon RDS 데이터베이스 인스턴스에 대해 DB 스냅샷과 트랜잭션 로그를 다른 AWS 리전으로 복제할 수 있습니다. 자세한 내용은 자동 백업을 다른 AWS 리전에 복제 섹션을 참조하세요.	2021년 5월 3일

AWS Outposts 기반 Amazon RDS에서 Amazon CloudWatch 모니터링 지원	RDS on Outposts는 이제 Amazon CloudWatch 모니터링을 지원합니다. 자세한 내용은 Amazon RDS 기능에 대한 Amazon RDS on AWS Outposts 지원 섹션을 참조하세요.	2021년 4월 21일
RDS for PostgreSQL에서 AWS Lambda 함수 지원	이제 RDS for PostgreSQL DB 인스턴스에 대해 AWS Lambda 함수를 호출할 수 있습니다. 자세한 내용은 RDS for PostgreSQL DB 인스턴스에서 AWS Lambda 함수 호출을 참조하세요.	2021년 4월 13일
RDS for SQL Server에서 확장 이벤트 지원	SQL Server 확장 이벤트를 사용하여 디버깅 및 문제 해결 정보를 캡처할 수 있습니다. 자세한 내용은 Amazon RDS for Microsoft SQL Server에 확장 이벤트 사용 섹션을 참조하세요.	2021년 4월 8일
MySQL 8.0.23, 5.7.33 및 5.6.51에 대한 지원	이제 MySQL 버전 8.0.23, 5.7.33 및 5.6.51을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하십시오.	2021년 3월 31일

실패한 Amazon RDS for MySQL 업그레이드의 자동 롤백	DB 인스턴스를 MySQL 버전 5.7에서 MySQL 버전 8.0으로 업그레이드하는 데 실패할 경우 Amazon RDS는 업그레이드를 위해 수행된 변경 사항을 자동으로 롤백합니다. 롤백 후 MySQL DB 인스턴스는 MySQL 버전 5.7을 실행합니다. 자세한 내용은 MySQL 5.7에서 8.0으로의 업그레이드 실패 후 롤백을 참조하세요.	2021년 3월 18일
Amazon RDS에서 옵트인 리전에서 리전 간 읽기 전용 복제본 지원	이제 DB 인스턴스를 옵트인 리전에 복제할 수 있습니다. 자세한 내용은 다른 AWS 리전에서 읽기 전용 복제본 생성을 참조하세요.	2021년 3월 18일

Amazon RDS에서 Oracle Database 18c 지원 중단 예정	Oracle Database 18c(18.0.0.0)은 사용 중단 경로에 있습니다. Oracle Corporation은 지원 종료일 이후 Oracle Database 18c에 대한 패치를 더 이상 제공하지 않습니다. 2021년 7월 1일에 Amazon RDS는 Oracle Database 18c 인스턴스를 Oracle Database 19c로 자동 업그레이드하기 시작할 계획입니다. 자동 업그레이드를 시작하기 전에 기존 Oracle Database 18c 인스턴스를 Oracle Database 19c로 수동으로 업그레이드하는 것이 좋습니다. 자세한 내용은 Oracle Database 18c의 자동 업그레이드 준비를 참조하세요.	2021년 3월 11일
Amazon RDS에서 Oracle Database 11g에 대한 지원 종료	Oracle Database 12c 릴리스 1(12.1.0.2) 이상에 대해서만 DB 인스턴스를 생성할 수 있습니다. Oracle Database 11g 스냅샷이 있는 경우 이후 릴리스로 업그레이드하세요. 자세한 내용은 Oracle DB 스냅샷 업그레이드를 참조하세요.	2021년 3월 11일
Amazon RDS에서 AWS Backup의 DB 인스턴스 연속 백업 지원	이제 AWS Backup에서 자동 백업을 생성하고 이러한 백업에서 지정된 시간으로 DB 인스턴스를 복원할 수 있습니다. 자세한 내용은 AWS Backup를 사용하여 자동 백업 관리를 참조하세요.	2021년 3월 10일

Amazon RDS에서 Oracle Management Agent(OMA) 버전 13.4 지원	Oracle Enterprise Manager(OEM) Cloud Control 13c 릴리스 4 업데이트 9에 Oracle Management Agent(OMA) 버전 13.4를 사용할 수 있습니다. Amazon RDS for Oracle은 Oracle Management Service(OMS)와 통신하여 모니터링 정보를 제공하는 OMA를 설치합니다. OMS 13.4를 실행하는 경우 OMA 13.4를 설치하여 데이터베이스를 관리할 수 있습니다. 자세한 내용은 Oracle Management Agent for Enterprise Manager Cloud Control을 참조하세요.	2021년 3월 10일
RDS 프록시 엔드포인트 개선 사항	각 RDS 프록시와 연결된 추가 엔드포인트를 생성할 수 있습니다. 다른 VPC에 엔드포인트를 생성하면 프록시에 대한 VPC 간 액세스가 가능합니다. Aurora MySQL 클러스터용 프록시에는 읽기 전용 엔드포인트가 있을 수도 있습니다. 이러한 리더 엔드포인트는 클러스터의 읽기 DB 인스턴스에 연결되며 쿼리 집약적 애플리케이션의 읽기 확장성과 가용성을 높일 수 있습니다. RDS 프록시에 대한 자세한 내용은 Amazon RDS 사용 설명서 또는 Aurora 사용 설명서의 “Amazon RDS 프록시를 사용한 연결 관리”를 참조하세요.	2021년 3월 8일

Amazon RDS에서 리전 간 자동 백업에 대한 지원 확장	이제 PostgreSQL을 실행하는 Amazon RDS 데이터베이스 인스턴스를 구성하여 DB 스냅샷과 트랜잭션 로그를 다른 AWS 리전으로 복제할 수 있습니다. 자세한 내용은 자동 백업을 다른 AWS 리전에 복제 섹션을 참조하세요.	2021년 3월 8일
중국(베이징) 리전 및 중국(닝샤) 리전에서 지원되는 Amazon RDS for MariaDB 및 MySQL용 복제 필터	이제 중국(베이징) 리전 및 중국(닝샤) 리전에서 복제 필터링이 지원됩니다. 자세한 내용은 MariaDB를 사용하여 복제 필터 구성 및 MySQL을 사용하여 복제 필터 구성을 참조하세요.	2021년 3월 5일
Amazon RDS가 옵트인 리전의 리전 간 DB 스냅샷 복사 지원	이제 옵트인 AWS 리전 안팎으로 DB 스냅샷을 복사할 수 있습니다. 자세한 내용은 AWS 리전에 걸쳐 스냅샷 복사를 참조하세요.	2021년 3월 4일
Amazon RDS for SQL Server에서 이제 Standard Edition의 Always On 가용성 그룹 지원	SQL Server 2019에서 Standard Edition 데이터베이스 엔진용 다중 AZ 구성을 사용하여 DB 인스턴스를 생성하면 RDS가 자동으로 가용성 그룹을 사용합니다. 자세한 내용은 Microsoft SQL Server의 다중 AZ 배포를 참조하십시오.	2021년 2월 23일

Amazon RDS for Oracle에 Advisor 관련 프로시저 도입	`rdsadmin_util` 패키지에 `advisor_task_set_parameter` , `advisor_task_drop` 및 `dbms_stats_init` 프로시저가 포함되어 있습니다. 이들 프로시저를 사용하여 `AUTO_STATS_ADVISOR_TASK` 같은 Advisor 작업을 수정하고 중지하고 다시 활성화할 수 있습니다. 자세한 내용은 Advisor 작업에 대한 파라미터 설정을 참조하세요.	2021년 2월 23일
Amazon RDS, 다중 AZ DB 인스턴스에 대한 장애 조치 이유 제공	이제 다중 AZ DB 인스턴스가 대기 복제본으로 장애 조치될 때 더 자세한 설명을 볼 수 있습니다. 자세한 내용은 Amazon RDS에 대한 장애 조치 프로세스를 참조하세요.	2021년 2월 18일
Amazon RDS는 스냅샷 내보내기에 대한 지원을 Amazon S3로 확장	이제 DB 스냅샷 데이터를 중국의 Amazon S3(으)로 내보낼 수 있습니다. 자세한 내용은 Amazon S3으로 DB 스냅샷 데이터 내보내기를 참조하십시오.	2021년 2월 17일

Amazon RDS for MariaDB 및 MySQL에 대한 복제 필터	MySQL 및 MariaDB 인스턴스에 대한 복제 필터를 구성할 수 있습니다. 복제 필터는 어떤 데이터베이스와 테이블을 읽기 복제본에 복제할지 지정합니다. 각 읽기 복제본에 대해 포함하거나 제외할 데이터베이스 및 테이블 목록을 생성할 수 있습니다. 자세한 내용은 MariaDB를 사용하여 복제 필터 구성 및 MySQL을 사용하여 복제 필터 구성을 참조하세요.	2021년 2월 12일
RDS for Oracle에서 APEX 20.2v1 지원	지원되는 모든 Oracle Database 버전에서 APEX 20.2.v1을 사용할 수 있습니다. 자세한 내용은 Oracle Application Express 단원을 참조하십시오.	2021년 2월 2일
Amazon RDS for SQL Server, tempdb 데이터베이스에 대한 로컬 인스턴스 스토리지 지원	이제 인스턴스 스토어를 사용하도록 구성된 tempdb 데이터베이스를 사용하여 Amazon EC2 db.r5d and db.m5d 인스턴스 유형에서 Amazon RDS for SQL Server를 시작할 수 있습니다. tempdb 데이터 파일과 로그 파일을 로컬에 배치함으로써 Amazon EBS를 기반으로 하는 표준 스토리지에 비해 읽기 및 쓰기 지연 시간을 줄일 수 있습니다. 자세한 내용은 Amazon RDS for SQL Server의 tempdb 데이터베이스에 대한 인스턴스 스토어 지원을 참조하세요.	2021년 1월 27일

Amazon RDS for PostgreSQL, pg_partman 및 pg_cron 지원	Amazon RDS for PostgreSQL은 이제 pg_partman 및 pg_cron 확장을 지원합니다. pg_partman 확장에 대한 자세한 내용은 pg_partman 확장을 사용하여 PostgreSQL 파티션 관리 섹션을 참조하세요. pg_cron 확장에 대한 자세한 내용은 PostgreSQL pg_cron 확장을 사용하여 유지 관리 예약 섹션을 참조하세요.	2021년 1월 12일
Amazon RDS, Amazon CloudWatch Logs로의 Oracle Management Agent 로그 게시 지원	Oracle Management Agent 로그는 emctl.log, emdctlj.log, gcagent.log, gcagent_errors.log, emagent.nohup 및 secure.log로 구성됩니다. Amazon RDS는 이러한 각 로그를 별도의 CloudWatch 로그 스트림으로 게시합니다. 자세한 내용은 Amazon CloudWatch Logs에 Oracle 로그 게시 섹션을 참조하세요.	2020년 12월 28일
AWS Outposts 기반 Amazon RDS에서 추가 데이터베이스 버전 지원	RDS on Outposts가 이제 추가 MySQL 및 PostgreSQL 버전을 지원합니다. 자세한 내용은 Amazon RDS 기능에 대한 Amazon RDS on AWS Outposts 지원 섹션을 참조하세요.	2020년 12월 23일

AWS Outposts 기반 Amazon RDS에서 CoIP 지원	RDS on Outposts가 이제 고객 소유 IP 주소(CoIP)를 지원합니다. CoIP는 온프레미스 네트워크를 통해 Outpost 서브넷의 리소스에 대한 로컬 또는 외부 연결을 제공합니다. 자세한 내용은 RDS on Outposts의 고객 소유 IP 주소 섹션을 참조하세요.	2020년 12월 22일
Amazon RDS for Oracle에서 11g BYOL 인스턴스를 19c로 업그레이드할 계획	2021년 1월 4일부터 기존 보유 라이선스 사용(BYOL) 모델에서 Oracle Database 19c로 모든 버전의 Oracle Database 11g 인스턴스를 자동으로 업그레이드하기 시작할 계획입니다. 예약 인스턴스를 포함한 모든 Oracle Database 11g 인스턴스는 사용 가능한 최신 릴리스 업데이트(RU)로 이동됩니다. 자세한 내용은 Oracle Database 11g BYOL의 자동 업그레이드 준비 섹션을 참조하세요.	2020년 12월 11일
Amazon RDS에서 다른 AWS 리전으로의 자동 백업 복제 지원	이제 원하는 대상 AWS 리전으로 스냅샷 및 트랜잭션 로그를 복제하도록 Amazon RDS 데이터베이스 인스턴스를 구성할 수 있습니다. 자세한 내용은 자동 백업을 다른 AWS 리전에 복제 섹션을 참조하세요.	2020년 12월 4일

Amazon RDS for Oracle 및 Microsoft SQL Server, 새로운 DB 인스턴스 클래스 지원	이제 db.r5b 인스턴스 클래스를 사용하여 Oracle 또는 SQL Server를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스에 대해 지원되는 DB 엔진 섹션을 참조하세요.	2020년 12월 4일
MariaDB 10.2.32 지원	이제 MariaDB 버전 10.2.32를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하십시오.	2020년 11월 25일
Amazon RDS for SQL Server, SQL Server 2019의 Microsoft Business Intelligence 제품군 지원	이제 DB 인스턴스에서 최신 메이저 버전을 사용하여 SQL Server Analysis Services, SQL Server Integration Services 및 SQL Server Reporting Services를 실행할 수 있습니다. 자세한 내용은 Microsoft SQL Server 데이터베이스 엔진의 옵션 섹션을 참조하세요.	2020년 11월 24일
데이터베이스 미리 보기 환경에서 Amazon RDS for PostgreSQL 버전 13	이제 Amazon RDS for PostgreSQL이 데이터베이스 미리 보기 환경에서 PostgreSQL 버전 13을 지원합니다. 자세한 내용은 PostgreSQL 13 버전 섹션을 참조하세요.	2020년 11월 24일

Amazon RDS 성능 개선 도우미로 새로운 차원 도입	데이터베이스(PostgreSQL, MySQL 및 MariaDB), 애플리케이션(PostgreSQL) 및 세션 유형(PostgreSQL)에 대한 차원 그룹에 따라 데이터베이스 로드를 그룹화할 수 있습니다. Amazon RDS는 또한 차원 db.name(PostgreSQL, MySQL 및 MariaDB), db.application.name(PostgreSQL) 및 db.session_type.name(PostgreSQL)을 지원합니다. 자세한 내용은 상위 부하 테이블을 참조하세요.	2020년 11월 24일
Amazon RDS for MariaDB는 새로운 메이저 버전 지원	이제 MariaDB 버전 10.5를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하십시오.	2020년 11월 23일
MySQL 5.6.49 지원	이제 MySQL 버전 5.6.49를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하십시오.	2020년 11월 20일
MySQL 5.5.62 지원	이제 MySQL 버전 5.5.62를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하세요.	2020년 11월 20일

성능 개선 도우미, 실행 중인 PostgreSQL 쿼리에 대한 통계 분석 지원	이제 PostgreSQL DB 인스턴스용 Performance Insights를 사용하여 실행 중인 쿼리의 통계를 분석할 수 있습니다. 자세한 내용은 PostgreSQL에 대한 통계 섹션을 참조하세요.	2020년 11월 18일
Amazon RDS, 스토리지 Auto Scaling에 대한 지원 확대	이제 읽기 전용 복제본을 생성하거나, DB 인스턴스를 지정된 시간으로 복원하거나, Amazon S3 백업에서 MySQL DB 인스턴스를 복원할 때 스토리지 Auto Scaling을 활성화할 수 있습니다. 자세한 내용은 Amazon RDS 스토리지 Auto Scaling을 사용한 용량 자동 관리 섹션을 참조하세요.	2020년 11월 18일
Amazon RDS for SQL Server, Database Mail 지원	Database Mail을 통해 Amazon RDS for SQL Server 데이터베이스 인스턴스에서 이메일 메시지를 전송할 수 있습니다. 이메일 수신자를 지정한 후 전송하려는 메시지에 파일이나 쿼리 결과를 첨부할 수 있습니다. 자세한 내용은 Amazon RDS for SQL Server에 Database Mail 사용 섹션을 참조하세요.	2020년 11월 4일
MySQL 8.0.26 지원	이제 MySQL 버전 8.0.21을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하세요.	2020년 10월 22일

Amazon RDS는 스냅샷 내보내기에 대한 지원을 Amazon S3로 확장	이제 모든 상용 AWS 리전에서 DB 스냅샷 데이터를 Amazon S3로 내보낼 수 있습니다. 자세한 내용은 Amazon S3으로 DB 스냅샷 데이터 내보내기를 참조하십시오.	2020년 10월 22일
Amazon RDS for PostgreSQL에서 읽기 전용 복제본 업그레이드 지원	Amazon RDS for PostgreSQL의 경우 기본 DB 인스턴스의 메이저 버전 업그레이드를 수행하면 읽기 전용 복제본도 자동으로 업그레이드됩니다. 자세한 내용은 PostgreSQL DB 엔진 업그레이드를 참조하세요.	2020년 10월 15일
Amazon RDS for MariaDB, MySQL 및 PostgreSQL에서 Graviton2 DB 인스턴스 클래스 지원	이제 Graviton2 DB 인스턴스 클래스 db.m6g.x 및 db.r6g.x를 사용하여 MariaDB, MySQL 또는 PostgreSQL을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 모든 사용 가능한 DB 인스턴스 클래스에 대해 지원되는 DB 엔진을 참조하세요.	2020년 10월 15일
Amazon RDS for SQL Server에서 SQL Server 2019에 대한 업그레이드 지원	SQL Server DB 인스턴스를 SQL Server 2019로 업그레이드할 수 있습니다. 자세한 내용은 Microsoft SQL Server DB 엔진 업그레이드를 참조하십시오.	2020년 10월 6일

Amazon RDS for Oracle에서 국가별 문자 집합 지정 지원	NCHAR 문자 집합이라고도 하는 국가별 문자 집합은 NCHAR, NVARCHAR2 및 NCLOB 데이터 유형에 사용됩니다. 데이터베이스를 만들 때 AL16UTF16(기본값) 또는 UTF8을 NCHAR 문자 집합으로 지정할 수 있습니다. 자세한 내용은 Amazon RDS에서 지원되는 Oracle 문자 집합을 참조하십시오.	2020년 10월 2일
MySQL 5.7.31 지원	이제 MySQL 버전 5.7.31을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하세요.	2020년 10월 1일
Amazon RDS for PostgreSQL에서 Amazon S3으로의 데이터 내보내기 지원	PostgreSQL DB 인스턴스의 데이터를 쿼리하여 Amazon S3 버킷에 저장된 파일로 직접 내보낼 수 있습니다. 자세한 내용은 RDS for PostgreSQL DB 인스턴스에서 Amazon S3으로 데이터 내보내기를 참조하십시오.	2020년 9월 24일
Amazon RDS for MySQL 8.0에서 Percona XtraBackup 지원	이제 Percona XtraBackup을 사용하여 Amazon RDS for MySQL 8.0 DB 인스턴스로 백업을 복원할 수 있습니다. 자세한 내용은 MySQL DB 인스턴스로 백업 복원을 참조하세요.	2020년 9월 17일

Amazon RDS for SQL Server에서 읽기 전용 복제본이 구성된 DB 인스턴스에서 네이티브 백업/복원 지원	읽기 전용 복제본이 구성된 DB 인스턴스에 SQL Server 네이티브 백업을 복원할 수 있습니다. 자세한 내용은 SQL Server 데이터베이스 가져오기 및 내보내기를 참조하세요.	2020년 9월 16일
Amazon RDS for SQL Server에서는 추가 시간대 지원	DB 인스턴스 시간대를 선택한 시간대와 일치시킬 수 있습니다. 자세한 내용은 Microsoft SQL Server DB 인스턴스의 현지 시간대를 참조하십시오.	2020년 9월 11일
데이터베이스 미리 보기 환경의 Amazon RDS for PostgreSQL 버전 13 베타 3	이제 Amazon RDS for PostgreSQL은 데이터베이스 미리 보기 환경에서 PostgreSQL 버전 13 베타 3을 지원합니다. 자세한 내용은 PostgreSQL 13 버전 섹션을 참조하세요.	2020년 9월 9일
Amazon RDS for SQL Server, 추적 플래그 692 지원	이제 DB 파라미터 그룹을 사용하여 추적 플래그 692를 시작 파라미터로 사용할 수 있습니다. 이 추적 플래그를 활성화하면 데이터를 힙 또는 클러스터된 인덱스로 일괄 로드하는 동안 빠른 삽입이 비활성화됩니다. 자세한 내용은 일괄 로드하는 동안 빠른 삽입 비활성화를 참조하십시오.	2020년 8월 27일
Amazon RDS for SQL Server, Microsoft SQL Server 2019 지원	이제 SQL Server 2019를 사용하는 RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 Microsoft SQL Server 버전을 참조하십시오.	2020년 8월 26일

RDS for Oracle에서 탑재된 복제본 데이터베이스 지원	Oracle 복제본을 생성하거나 수정할 때 탑재된 모드로 배치할 수 있습니다. 복제본 데이터베이스는 사용자 연결을 허용하지 않으므로 읽기 전용 워크로드를 처리할 수 없습니다. 탑재된 복제본은 아카이브된 다시 실행 로그 파일을 적용한 후 삭제합니다. 탑재된 복제본의 주된 용도는 리전 간 재해 복구입니다. 자세한 내용은 Oracle 복제본 개요를 참조하세요.	2020년 8월 13일
RDS for Oracle은 11g SE1 LI 인스턴스의 업그레이드 계획	2020년 11월 1일에 Oracle Database 11g SE1 라이선스 포함(LI) 인스턴스를 Amazon RDS for Oracle용 Oracle Database 19c로 자동 업그레이드하기 시작할 계획입니다. 예약 인스턴스를 포함한 모든 11g 인스턴스는 사용 가능한 최신 Oracle 릴리스 업데이트(RU)로 이동됩니다. 자세한 내용은 Oracle Database 11g SE1의 자동 업그레이드 준비를 참조하세요.	2020년 7월 31일
Amazon RDS는 PostgreSQL 및 MySQL의 평가판 릴리스에서 새로운 Graviton2 DB 인스턴스 클래스 지원	이제 db.m6g.x 및 db.r6g.x DB 인스턴스 클래스를 사용하는 PostgreSQL을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 사용 가능한 모든 DB 인스턴스 클래스에 대해 지원되는 DB 엔진을 참조하십시오.	2020년 7월 30일

RDS for Oracle에서 APEX 20.1v1 지원	지원되는 모든 Oracle Database 버전에서 APEX 20.1v1을 사용할 수 있습니다. 자세한 내용은 Oracle Application Express를 참조하십시오.	2020년 7월 28일
MySQL 8.0.20 지원	이제 MySQL 버전 8.0.20을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하십시오.	2020년 7월 23일
Amazon RDS for MariaDB 및 Amazon RDS for MySQL은 새로운 DB 인스턴스 클래스 지원	이제 db.m5.16xlarge, db.m5.8xlarge, db.r5.16xlarge 및 db.r5.8xlarge DB 인스턴스 클래스를 사용하는 MariaDB 및 MySQL을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 사용 가능한 모든 DB 인스턴스 클래스에 대해 지원되는 DB 엔진을 참조하십시오.	2020년 7월 23일
RDS for SQL Server는 이전 버전의 TLS 및 암호 비활성화 지원	특정 보안 프로토콜 및 암호를 설정하거나 해제할 수 있습니다. 자세한 내용은 보안 프로토콜 및 암호 구성을 참조하십시오.	2020년 7월 21일
RDS는 SE2에서 Oracle Spatial 지원	12.2, 18c 및 19c의 모든 버전에 대해 Standard Edition 2(SE2)에서 Oracle Spatial을 사용할 수 있습니다. 자세한 내용은 Oracle Spatial을 참조하십시오.	2020년 7월 9일

Amazon RDS에서 AWS PrivateLink 지원	이제 Amazon RDS는 AWS 네트워크의 Amazon RDS와 애플리케이션 간의 트래픽을 유지하기 위해 Amazon RDS API 호출을 위한 Amazon VPC 엔드포인트 생성을 지원합니다. 자세한 내용은 Amazon RDS 및 인터페이스 VPC 엔드포인트(AWS PrivateLink)를 참조하세요.	2020년 7월 9일
Amazon RDS for PostgreSQL Postststm의 경우 지원이 종료되었습니다.	Amazon RDS for PostgreSQL은 더 이상 버전 9.4.x를 지원하지 않습니다. 지원되는 버전에 대한 자세한 내용은 지원되는 PostgreSQL 데이터베이스 버전을 참조하십시오.	2020년 7월 8일
MariaDB 10.3.23 및 10.4.13 지원	이제 MariaDB 버전 10.3.23 및 10.4.13을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하십시오.	2020년 7월 6일
AWS Outposts 기반 Amazon RDS	AWS Outposts에서 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS on AWS Outposts 작업을 참조하세요.	2020년 7월 6일

Amazon RDS for Oracle은 인벤토리 파일을 자동으로 생성	BYOL 고객에 대한 서비스 요청을 개설하기 위해 Oracle Support는 Opatch에서 생성된 인벤토리 파일을 요청합니다. Amazon RDS for Oracle은 BDUMP 디렉터리에서 1시간마다 인벤토리 파일을 자동으로 생성합니다. 자세한 내용은 Opatch 파일 액세스를 참조하십시오.	2020년 7월 6일
MySQL 5.7.30 및 5.6.48 지원	이제 MySQL 버전 5.7.30 및 5.6.48을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하십시오.	2020년 6월 25일
Amazon RDS for Oracle에서 ADRCI 지원	ADRCI(자동 진단 리포지토리 명령 인터프리터) 유틸리티는 진단 데이터를 관리하는 데 사용하는 Oracle 명령줄 도구입니다. Amazon RDS 패키지 `rdsadmin_adrci_util` 의 함수를 사용하여 문제와 인시던트를 나열하고 패키지징할 수 있으며 추적 파일도 표시할 수 있습니다. 자세한 내용은 Oracle DB 인스턴스에 대한 공통 DBA 진단 작업을 참조하십시오.	2020년 6월 17일
MySQL 8.0.19 지원	이제 MySQL 버전 8.0.19를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하십시오.	2020년 6월 2일

MySQL 8.0, 소문자 테이블 이름 지원	이제 MySQL 버전 8.0.19 이상 8.0 버전을 실행하는 Amazon RDS DB 인스턴스에 대해 `lower_case_table_names` 파라미터를 1로 설정할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스에 대한 MySQL 파라미터 예외를 참조하십시오.	2020년 6월 2일
Amazon RDS for Microsoft SQL Server에서 SQL Server Integration Services(SSIS) 지원	SSIS는 데이터 통합 및 워크플로우 애플리케이션용 플랫폼입니다. 기존 DB 인스턴스 또는 새 DB 인스턴스에서 SSRS를 활성화할 수 있습니다. 이 인스턴스는 데이터베이스 엔진과 동일한 DB 인스턴스에 설치됩니다. 자세한 내용은 SQL Server의 SQL Server Integration Services 지원을 참조하십시오.	2020년 5월 19일
Amazon RDS for Microsoft SQL Server가 SQL Server Reporting Services(SSRS) 지원	SSRS는 보고서 생성 및 배포에 사용되는 서버 기반 애플리케이션입니다. 기존 DB 인스턴스 또는 새 DB 인스턴스에서 SSRS를 활성화할 수 있습니다. 이 인스턴스는 데이터베이스 엔진과 동일한 DB 인스턴스에 설치됩니다. 자세한 내용은 SQL Server의 SQL Server Reporting Services 지원을 참조하십시오.	2020년 5월 15일

Amazon RDS for Microsoft SQL Server가 다중 AZ 인스턴스에서 S3 통합 지원	이제 다중 AZ DB 인스턴스에서 대량 삽입과 같은 SQL Server 기능과 함께 Amazon S3를 사용할 수 있습니다. 자세한 내용은 Amazon RDS for SQL Server DB 인스턴스와 Amazon S3 통합을 참조하십시오.	2020년 5월 15일
Amazon RDS for Oracle이 휴지통 비우기 지원	`rdsadmin.rdsadmin_util.purge_dba_recyclebin` 프로시저는 휴지통을 비웁니다. 자세한 내용은 휴지통 비우기를 참조하십시오.	2020년 5월 13일
Amazon RDS for Oracle에서 Automatic Workload Repository(AWR)의 관리 용이성 향상	`rdsadmin.rdsadmin_diagnostic_util` 프로시저는 AWR 보고서를 생성하고 AWR 데이터를 덤프 파일로 추출합니다. 자세한 내용은 Automatic Workload Repository(AWR)를 사용하여 성능 보고서 생성을 참조하십시오.	2020년 5월 13일
Amazon RDS for Microsoft SQL Server에서 Microsoft Distributed Transaction Coordinator(MSDTC) 지원	Amazon RDS for SQL Server가 호스트 간 분산 트랜잭션을 지원합니다. 자세한 내용은 SQL Server에서 Microsoft Distributed Transaction Coordinator 지원을 참조하십시오.	2020년 5월 4일

Amazon RDS for Microsoft SQL Server에서 새로운 버전 지원	이제 모든 에디션에 대해 SQL Server 버전 2017 CU19 14.00.3281.6, 2016 SP2 CU11 13.00.5598.27, 2014 SP3 CU4 12.00.6329.1 및 2012 SP4 GDR 11.0.7493.4를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 Microsoft SQL Server 버전을 참조하십시오.	2020년 4월 28일
유럽(밀라노) 리전에서 Amazon RDS 사용 가능	이제 유럽(밀라노) 리전에서 Amazon RDS를 사용할 수 있습니다. 자세한 내용은 리전 및 가용 영역을 참조하십시오.	2020년 4월 28일
Amazon RDS에서 로컬 영역 지원	이제 로컬 영역 서브넷에서 DB 인스턴스를 시작할 수 있습니다. 자세한 내용은 리전, 가용 영역 및 Local Zones를 참조하세요.	2020년 4월 23일
아프리카(케이프타운) 리전에서 Amazon RDS 사용 가능	이제 아프리카(케이프타운) 리전에서 Amazon RDS를 사용할 수 있습니다. 자세한 내용은 리전 및 가용 영역을 참조하십시오.	2020년 4월 22일

Amazon RDS for Microsoft SQL Server는 SQL Server Analysis Services(SSAS) 지원	SSAS는 SQL Server 내에 설치된 온라인 분석 처리(OLAP) 및 데이터 마이닝 도구입니다. 기존 DB 인스턴스 또는 새 DB 인스턴스에서 SSAS를 활성화할 수 있습니다. 이 인스턴스는 데이터베이스 엔진과 동일한 DB 인스턴스에 설치됩니다. 자세한 내용은 SQL Server의 SQL Server Analysis Services 지원을 참조하십시오.	2020년 4월 17일
PostgreSQL용 Amazon RDS 프록시	이제 PostgreSQL에서 Amazon RDS 프록시를 사용할 수 있습니다. RDS Proxy를 사용하면 DB 인스턴스에서 연결 관리의 오버헤드를 줄이고 “연결이 너무 많음” 오류의 가능성을 줄일 수 있습니다. RDS Proxy는 현재 PostgreSQL에 대한 공개 미리 보기 상태입니다. 자세한 내용은 Amazon RDS Proxy(미리 보기)를 사용한 연결 관리를 참조하십시오.	2020년 4월 8일
Amazon RDS for Oracle은 Oracle APEX 버전 19.2.v1 지원	Amazon RDS for Oracle은 이제 Oracle Application Express(APEX) 버전 19.2.v1을 지원합니다. 자세한 내용은 Oracle Application Express를 참조하십시오.	2020년 4월 8일
Amazon RDS for MariaDB는 새로운 메이저 버전 지원	이제 MariaDB 버전 10.4를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하십시오.	2020년 4월 6일

Amazon RDS for MariaDB 10.4에 Amazon RDS 성능 개선 도우미 사용 가능	Amazon RDS Performance Insights는 이제 Amazon RDS for MariaDB 버전 10.4에서 사용할 수 있습니다. 자세한 내용은 Amazon RDS 성능 개선 도우미 사용 단원을 참조하십시오.	2020년 4월 6일
Amazon RDS for PostgreSQL Postststm의 경우 지원이 종료되었습니다.	Amazon RDS for PostgreSQL은 더 이상 버전 9.3.x를 지원하지 않습니다. 지원되는 버전에 대한 자세한 내용은 지원되는 PostgreSQL 데이터베이스 버전을 참조하십시오.	2020년 4월 3일
Amazon RDS for Microsoft SQL Server에서 읽기 전용 복제본 지원	이제 SQL Server DB 인스턴스에 대한 읽기 전용 복제본을 생성할 수 있습니다. 자세한 내용은 읽기 전용 복제본 작업을 참조하십시오.	2020년 4월 3일
Amazon RDS for Microsoft SQL Server에서 다중 파일 백업 지원	이제 SQL Server 기본 백업 및 복원을 사용하여 데이터베이스를 여러 파일에 백업할 수 있습니다. 자세한 내용은 데이터베이스 백업을 참조하십시오.	2020년 4월 2일
Amazon RDS for Oracle AWS License Manager 통합	이제 Amazon RDS for Oracle이 AWS License Manager와 통합됩니다. 기존 보유 라이선스 사용 모델을 사용하는 경우 AWS License Manager 통합으로 인해 조직 내에서 Oracle 라이선스 사용량을 더 쉽게 모니터링할 수 있습니다. 자세한 내용은 AWS License Manager와 통합을 참조하십시오.	2020년 3월 23일

Amazon RDS for MariaDB 및 MySQL 64에서 db.r5 인스턴스에 대해 64TiB 지원	이제 최대 64TiB의 스토리지와 함께 db.r5 DB 인스턴스 클래스를 사용하는 Amazon RDS for Maria 및 MySQL DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 스토리지 성능에 영향을 주는 요인을 참조하십시오.	2020년 3월 18일
MySQL 8.0.26 지원	이제 MySQL 버전 8.0.17을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하십시오.	2020년 3월 10일
Amazon RDS for MySQL 8.0에 Amazon RDS 성능 개선 도우미 사용 가능	이제 Amazon RDS 성능 개선 도우미를 Amazon RDS for MySQL 버전 8.0.17 및 그 이상의 8.0 버전에서 사용할 수 있습니다. 자세한 내용은 Amazon RDS 성능 개선 도우미 사용 단원을 참조하십시오.	2020년 3월 10일
MySQL 5.6.46 지원	이제 MySQL 버전 5.6.46을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하십시오.	2020년 2월 28일
Amazon RDS for MariaDB 10.3에 Amazon RDS 성능 개선 도우미 사용 가능	이제 Amazon RDS 성능 개선 도우미를 Amazon RDS for MariaDB 버전 10.3.13 및 그 이상의 10.3 버전에 사용할 수 있습니다. 자세한 내용은 Amazon RDS 성능 개선 도우미 사용 단원을 참조하십시오.	2020년 2월 26일

MySQL 5.7.28 지원	이제 MySQL 버전 5.7.28을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하십시오.	2020년 2월 20일
MariaDB 10.3.20 지원	이제 MariaDB 버전 10.3.20을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하십시오.	2020년 2월 20일
Amazon RDS for Microsoft SQL Server에서 새 DB 인스턴스 클래스 지원	이제 db.z1d DB 인스턴스 클래스를 사용하는 SQL Server를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Microsoft SQL Server에 대한 DB 인스턴스 클래스 지원을 참조하십시오.	2020년 2월 19일
Amazon RDS for SQL Server의 교차 계정, 교차 VPC Active Directory 도메인 지원	이제 Amazon RDS for Microsoft SQL Server는 DB 인스턴스를 다른 계정 및 VPC에서 소유하는 Active Directory 도메인과 연결하는 것을 지원합니다. 자세한 내용은 Microsoft SQL Server DB 인스턴스에서 Windows 인증 사용을 참조하십시오.	2020년 2월 13일

Oracle OLAP 옵션	이제 Amazon RDS for Oracle은 Oracle DB 인스턴스에 대한 OLAP(On-line Analytical Processing) 옵션을 지원합니다. Oracle OLAP를 사용하여 OLAP 표준에 따라 차원 객체 및 큐브를 생성함으로써 많은 양의 데이터를 분석할 수 있습니다. 자세한 내용은 Oracle OLAP를 참조하십시오.	2020년 2월 13일
Oracle에 대한 FIPS 140-2 지원	Amazon RDS for Oracle은 SSL/TLS 연결에 대한 Federal Information Processing Standard Publication 140-2(FIPS 140-2)를 지원합니다. 자세한 내용은 FIPS 지원을 참조하십시오.	2020년 2월 11일
Amazon RDS for PostgreSQL에서 새 DB 인스턴스 클래스 지원	이제 db.m5.16xlarge, db.m5.8xlarge, db.r5.16xlarge 및 db.r5.8xlarge DB 인스턴스 클래스를 사용하는 PostgreSQL을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 사용 가능한 모든 DB 인스턴스 클래스에 대해 지원되는 DB 엔진을 참조하십시오.	2020년 2월 11일
성능 개선 도우미에서 실행 중인 MariaDB 및 MySQL 쿼리에 대한 통계 분석 지원	이제 MariaDB 및 MySQL DB 인스턴스용 성능 개선 도우미를 사용하여 실행 중인 쿼리의 통계를 분석할 수 있습니다. 자세한 내용은 실행 중인 쿼리에 대한 통계 분석을 참조하십시오.	2020년 2월 4일

MariaDB, MySQL 및 PostgreSQL에서 Amazon S3로 DB 스냅샷 데이터 내보내기 지원	Amazon RDS는 MariaDB, MySQL 및 Postgre에서 Amazon S3로 DB 스냅샷 데이터를 내보낼 수 있도록 지원합니다. 자세한 내용은 Amazon S3으로 DB 스냅샷 데이터 내보내기를 참조하십시오.	2020년 1월 23일
Kerberos 인증을 지원하는 Amazon RDS for MySQL	이제 사용자가 Amazon RDS for MySQL DB 인스턴스에 연결하려고 할 때 Kerberos 인증을 사용하여 사용자를 인증할 수 있습니다. 자세한 내용은 MySQL에서 Kerberos 인증 사용을 참조하십시오.	2020년 1월 21일
Amazon RDS 성능 개선 도우미에서 Amazon RDS for Microsoft SQL Server의 더 많은 SQL 텍스트 보기 지원	Amazon RDS 성능 개선 도우미는 이제 Amazon RDS for Microsoft SQL Server DB 인스턴스에 대해 성능 개선 도우미 대시보드에서 더 많은 SQL 텍스트를 볼 수 있도록 지원합니다. 자세한 내용은 성능 개선 도우미 대시보드에서 더 많은 SQL 텍스트 보기 단원을 참조하십시오.	2019년 12월 17일

Amazon RDS 프록시	Amazon RDS Proxy를 사용하면 클러스터에서 연결 관리의 오버헤드를 줄이고 “연결이 너무 많음” 오류의 가능성을 줄일 수 있습니다. 각 프록시를 RDS DB 인스턴스 또는 Aurora DB 클러스터와 연결합니다. 그런 다음 애플리케이션의 연결 문자열에 프록시 엔드포인트를 사용합니다. Amazon RDS Proxy는 현재 공개 미리 보기 상태입니다. 이것은 RDS for MySQL 데이터베이스 엔진을 지원합니다. 자세한 내용은 Amazon RDS Proxy(미리 보기)를 사용한 연결 관리를 참조하십시오.	2019년 12월 3일
Amazon RDS on AWS Outposts(평가판)	Amazon RDS on AWS Outposts를 사용하면 온프레미스 데이터 센터에서 AWS 관리형 관계형 데이터베이스를 생성할 수 있습니다. RDS on Outposts를 통해 AWS Outposts에서 RDS 데이터베이스를 실행할 수 있습니다. 자세한 내용은 Amazon RDS on AWS Outposts(미리 보기)를 참조하세요.	2019년 12월 3일

Amazon RDS for Oracle에서 리전 간 읽기 전용 복제본 지원	이제 Amazon RDS for Oracle은 Active Data Guard를 통한 리전 간 읽기 전용 복제본을 지원합니다. 자세한 내용은 읽기 전용 복제본 작업 및 Oracle 읽기 전용 복제본 작업을 참조하십시오.	2019년 11월 26일
성능 개선 도우미에서 실행 중인 Oracle 쿼리에 대한 통계 분석을 지원	이제 Oracle DB 인스턴스용 성능 개선 도우미를 사용하여 실행 중인 쿼리의 통계를 분석할 수 있습니다. 자세한 내용은 실행 중인 쿼리에 대한 통계 분석을 참조하십시오.	2019년 11월 25일
Amazon RDS for Microsoft SQL Server는 CloudWatch Logs에 로그 게시를 지원	로그 이벤트를 Amazon CloudWatch Logs에 직접 게시하도록 Amazon RDS for SQL Server DB 인스턴스를 구성할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs에 SQL Server 로그 게시 단원을 참조하십시오.	2019년 11월 25일
Amazon RDS for Microsoft SQL Server, 새로운 DB 인스턴스 클래스를 지원	이제 db.x1e 및 db.x1 DB 인스턴스 클래스를 사용하는 SQL Server를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Microsoft SQL Server에 대한 DB 인스턴스 클래스 지원을 참조하십시오.	2019년 11월 25일

Amazon RDS for Microsoft SQL Server, 디퍼렌셜 및 로그 복원 지원	SQL Server 기본 백업 및 복원을 사용하여 디퍼렌셜 백업 및 로그를 복원할 수 있습니다. 자세한 내용은 기본 백업 및 복원 사용을 참조하십시오.	2019년 11월 25일
새로운 리전의 Amazon RDS for Microsoft SQL Server에 다중 AZ 지원	현재 SQL Server 기반의 다중 AZ를 중국, 중동(바레인) 및 유럽(스톡홀름)에서 사용할 수 있습니다. 자세한 내용은 Microsoft SQL Server의 다중 AZ 배포를 참조하십시오.	2019년 11월 22일
Amazon RDS for Microsoft SQL Server에서 이제 대량 삽입 및 S3 통합 지원	SQL Server DB 인스턴스와 Amazon S3 버킷 사이에서 파일을 전송할 수 있습니다. 그런 다음 대량 삽입과 같은 SQL Server 기능으로 Amazon S3를 사용할 수 있습니다. 자세한 내용은 Amazon RDS for SQL Server DB 인스턴스와 Amazon S3 통합을 참조하십시오.	2019년 11월 21일
Amazon RDS for Microsoft SQL Server용 성능 개선 도우미 카운터	이제 Microsoft SQL Server DB 인스턴스용 성능 개선 도우미 차트에 성능 카운터를 추가할 수 있습니다. 자세한 내용은 Amazon RDS for Microsoft SQL Server의 성능 개선 도우미 카운터를 참조하십시오.	2019년 11월 12일

Amazon RDS for Microsoft SQL Server에서 이제 새로운 DB 인스턴스 클래스 크기를 지원	이제 db.m5 및 db.r5 DB 인스턴스 클래스에 8xlarge 및 16xlarge 인스턴스 크기를 사용하는 SQL Server 실행 Amazon RDS DB 인스턴스를 만들 수 있습니다. 이제 db.t3 인스턴스 클래스에 small ~ 2xlarge 범위의 인스턴스 크기를 사용할 수 있습니다. 자세한 내용은 Microsoft SQL Server에 대한 DB 인스턴스 클래스 지원을 참조하십시오.	2019년 11월 11일
PostgreSQL 스냅샷 업그레이드 지원	Amazon RDS PostgreSQL DB 인스턴스의 기존 수동 DB 스냅샷이 있는 경우 이제 PostgreSQL 데이터베이스 엔진의 최신 버전으로 스냅샷을 업그레이드할 수 있습니다. 자세한 내용은 PostgreSQL DB 스냅샷 업그레이드를 참조하십시오.	2019년 11월 7일
Amazon RDS for Oracle에서 새 메이저 버전 지원	이제 Oracle Database 19c(19.0)를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS 기반 Oracle Database 19c 섹션을 참조하세요.	2019년 11월 7일

데이터베이스 미리 보기 환경에서 Amazon RDS for PostgreSQL 버전 12.0	이제 Amazon RDS for PostgreSQL이 데이터베이스 미리 보기 환경에서 PostgreSQL 버전 12.0을 지원합니다. 자세한 내용은 데이터베이스 미리 보기 환경에서 PostgreSQL 버전 12.0을 참조하십시오.	2019년 11월 1일
Kerberos 인증을 지원하는 Amazon RDS for PostgreSQL	이제 사용자가 PostgreSQL을 실행 중인 Amazon RDS DB 인스턴스에 연결하려고 할 때 Kerberos 인증을 사용하여 사용자를 인증할 수 있습니다. 자세한 내용은 Amazon RDS for PostgreSQL과 함께 Kerberos 인증 사용을 참조하십시오.	2019년 10월 28일
Oracle DB 인스턴스에 대한 OEM Management Agent 데이터베이스 작업	Amazon RDS for Oracle DB 인스턴스는 이제 Management Agent에서 특정 EMCTL 명령을 호출하는 프로시저를 지원합니다. 자세한 내용은 OEM Agent Database 작업을 참조하십시오.	2019년 10월 24일
Amazon RDS for PostgreSQL에서 PostgreSQL 전송 가능 데이터베이스 지원	PostgreSQL 전송 가능 데이터베이스는 두 DB 인스턴스 간에 RDS PostgreSQL 데이터베이스를 마이그레이션하는 매우 빠른 방법을 제공합니다. 자세한 내용은 DB 인스턴스 간에 PostgreSQL 데이터베이스 전송을 참조하십시오.	2019년 10월 8일

Kerberos 인증을 지원하는 Amazon RDS for Oracle	이제 사용자가 Oracle을 실행 중인 Amazon RDS DB 인스턴스에 접속하려고 할 때 Kerberos 인증을 사용하여 사용자를 인증할 수 있습니다. 자세한 내용은 Amazon RDS for Oracle과 함께 Kerberos 인증 사용을 참조하십시오.	2019년 9월 30일
데이터베이스 미리 보기 환경의 Amazon RDS for PostgreSQL 버전 12 베타 3	이제 Amazon RDS for PostgreSQL은 데이터베이스 미리 보기 환경의 PostgreSQL 버전 12 베타 3을 지원합니다. 자세한 내용은 데이터베이스 미리 보기 환경에서 Amazon RDS의 PostgreSQL 버전 12 베타 3을 참조하십시오.	2019년 8월 28일
MySQL 8.0.16 지원	이제 MySQL 버전 8.0.16을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하십시오.	2019년 8월 19일
Amazon RDS for Oracle에서 새 메이저 버전 지원	이제 Oracle Database 18c(18.0)를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS 기반 Oracle 18c 섹션을 참조하세요.	2019년 8월 15일

OEM 13c 릴리스 3용 Management Agent	Amazon RDS for Oracle DB 인스턴스는 Oracle Enterprise Manager(OEM) 클라우드 제어 13c 릴리스 3용 관리 에이전트를 지원합니다. 자세한 내용은 엔터프라이즈 관리자 클라우드 제어용 Oracle Management Agent를 참조하십시오.	2019년 8월 7일
데이터베이스 미리 보기 환경의 Amazon RDS for PostgreSQL 버전 12 베타 2	Amazon RDS for PostgreSQL은 현재 데이터베이스 미리 보기 환경에서 PostgreSQL 버전 12 베타 2를 지원합니다. 자세한 내용은 데이터베이스 미리 보기 환경에서 Amazon RDS의 PostgreSQL 버전 12 베타 2를 참조하십시오.	2019년 8월 6일
Amazon RDS에서 SQL Server용 서버 콜레이션 지원	Amazon RDS for SQL Server는 새 DB 인스턴스에서 콜레이션을 선택할 수 있도록 지원합니다. 자세한 내용은 Microsoft SQL Server용 콜레이션 및 문자 세트를 참조하십시오.	2019년 7월 29일
Amazon RDS for Oracle에서 Oracle APEX 버전 19.1.v1 지원	Amazon RDS for Oracle에서 이제 Oracle Application Express(APEX) 버전 19.1.v1을 지원합니다. 자세한 내용은 Oracle Application Express를 참조하십시오.	2019년 6월 28일

데이터베이스 미리 보기 환경의 Amazon RDS for PostgreSQL 버전 13 베타 1	Amazon RDS for PostgreSQL은 이제 데이터베이스 미리 보기 환경에서 PostgreSQL 버전 13 베타 1을 지원합니다. 자세한 내용은 PostgreSQL 13 버전 섹션을 참조하세요.	2019년 6월 22일
Amazon RDS 스토리지 자동 크기 조정	Amazon RDS DB 인스턴스의 스토리지 Autoscaling은 Amazon RDS가 DB 인스턴스와 연결된 스토리지를 자동으로 확장하여 저장 공간이 부족할 수 있는 가능성을 줄여 주는 기능입니다. 스토리지 Autoscaling에 대한 자세한 내용은 Amazon RDS DB 인스턴스의 스토리지 작업 단원을 참조하십시오.	2019년 6월 20일
Amazon RDS for Oracle은 db.z1d DB 인스턴스 클래스 지원	이제 Oracle 실행 Amazon RDS DB 인스턴스를 생성하여 db.z1d DB 인스턴스 클래스를 사용할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스 단원을 참조하십시오.	2019년 6월 13일
Amazon RDS 성능 개선 도우미에서 Amazon RDS for Oracle에 더 많은 SQL 텍스트 보기 지원	Amazon RDS 성능 개선 도우미는 이제 Amazon RDS for Oracle DB 인스턴스에 대해 성능 개선 도우미 대시보드에서 더 많은 SQL 텍스트를 볼 수 있도록 지원합니다. 자세한 내용은 성능 개선 도우미 대시보드에서 더 많은 SQL 텍스트 보기 단원을 참조하십시오.	2019년 6월 10일

Amazon RDS에서 최대 16TB의 SQL Server 데이터베이스 기본 복원을 추가 지원	이제 SQL Server에서 Amazon RDS로 최대 16TB의 기본 복원을 수행할 수 있습니다. 자세한 내용은 Amazon RDS for SQL Server: 제한 및 권장 사항을 참조하세요.	2019년 6월 4일
Amazon RDS에서 Microsoft SQL Server Audit에 대한 지원 추가	Amazon RDS for Microsoft SQL Server를 사용하면 SQL Server Audit을 사용하여 서버 및 데이터베이스 수준 이벤트를 감사하고 DB 인스턴스에 대한 결과를 보거나 감사 로그 파일을 Amazon S3에 직접 보낼 수 있습니다. 자세한 내용은 SQL Server Audit을 참조하십시오.	2019년 5월 23일
Amazon RDS 권장 사항 개선	Amazon RDS에서 데이터베이스 리소스에 대한 자동 권장 사항을 개선했습니다. 예를 들어 Amazon RDS는 이제 데이터베이스 파라미터에 대한 권장 사항을 제공합니다. 자세한 내용은 Amazon RDS 권장 사항 사용을 참조하십시오.	2019년 5월 22일
Amazon RDS for SQL Server용 DB 인스턴스 하나당 더 많은 데이터베이스 지원	Microsoft SQL Server를 실행하는 각 DB 인스턴스에서 최대 30개의 데이터베이스를 만들 수 있습니다. 자세한 내용은 Microsoft SQL Server DB 인스턴스 제한을 참조하십시오.	2019년 5월 21일

Amazon RDS for MariaDB, Amazon RDS MySQL 및 Amazon RDS PostgreSQL에 64TB 및 80K IOPS의 스토리지 지원	이제 최대 64TB의 스토리지 및 프로비저닝된 최대 80,000 IOPS를 사용하여 Amazon RDS for MariaDB, Amazon RDS MySQL 및 Amazon RDS PostgreSQL DB 인스턴스를 만들 수 있습니다. 자세한 내용은 DB 인스턴스 스토리지를 참조하십시오.	2019년 5월 20일
Amazon RDS for MySQL은 업그레이드 사전 점검을 지원	DB 인스턴스를 MySQL 5.7에서 MySQL 8.0으로 업그레이드할 때 Amazon RDS가 비호환성 여부를 사전 점검합니다. 자세한 정보는 MySQL 5.7에서 8.0로 업그레이드하기 위한 사전 점검을 참조하십시오.	2019년 5월 17일
MySQL 암호 확인 플러그인 지원	이제 Amazon RDS for MySQL DB 인스턴스의 보안 향상을 위해 MySQL `validate_password` 플러그인을 사용할 수 있습니다. 자세한 내용은 암호 확인 플러그인 사용을 참조하십시오.	2019년 5월 16일
Amazon RDS for Oracle의 성능 개선 도우미 카운터	이제 Oracle DB 인스턴스용 성능 개선 도우미 차트에 성능 카운터를 추가할 수 있습니다. 자세한 내용은 Amazon RDS for Oracle의 성능 개선 도우미 카운터를 참조하십시오.	2019년 5월 8일

초 단위 결제 지원	Amazon RDS는 이제 온디맨드 인스턴스의 경우 AWS GovCloud(미국)를 제외한 모든 AWS 리전에서 1초 증분 단위로 청구됩니다. 자세한 내용은 Amazon RDS에 대한 DB 인스턴스 청구 단원을 참조하십시오.	2019년 4월 25일
Amazon S3에서 Amazon RDS for PostgreSQL에 필요한 데이터 가져오기 지원	이제 Amazon S3 파일의 데이터를 RDS PostgreSQL DB 인스턴스의 테이블로 가져올 수 있습니다. 자세한 내용은 Amazon S3 데이터를 RDS PostgreSQL DB 인스턴스로 가져오기를 참조하십시오.	2019년 4월 24일
Amazon S3에서 5.7 백업 복원 지원	이제 MySQL 버전 5.7 데이터베이스의 백업을 생성하여 Amazon S3에 저장한 다음 MySQL을 실행하는 새로운 Amazon RDS DB 인스턴스에 백업 파일을 복원할 수 있습니다. 자세한 내용은 MySQL DB 인스턴스로 백업 복원을 참조하세요.	2019년 4월 17일

Amazon RDS for PostgreSQL에 필요한 여러 메이저 버전 업그레이드 지원	이제 Amazon RDS for PostgreSQL을 사용하여 DB 엔진을 업그레이드할 때 여러 메이저 버전 중에서 선택할 수 있습니다. 이 기능을 사용하면 PostgreSQL 엔진 버전을 업그레이드할 때 새로운 메이저 버전으로 건너뛸 수 있습니다. 자세한 내용은 PostgreSQL DB 엔진 업그레이드를 참조하세요.	2019년 4월 16일
Amazon RDS for Oracle에 64TiB의 스토리지 지원	이제 최대 64TB의 스토리지 및 프로비저닝된 최대 80,000 IOPS를 사용하여 Amazon RDS for Oracle DB 인스턴스를 만들 수 있습니다. 자세한 내용은 DB 인스턴스 스토리지를 참조하십시오.	2019년 4월 4일
MySQL 8.0.15 지원	이제 MySQL 버전 8.0.15를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하십시오.	2019년 4월 3일
MariaDB 10.3.13 지원	이제 MariaDB 버전 10.3.13을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하십시오.	2019년 4월 3일

Amazon RDS에서 Microsoft SQL Server 2008 R2의 지원이 종료되었습니다.	Microsoft SQL Server 2008 R2는 2019년 7월 9일부터 이 버전에 대한 확장 지원 중단 계획에 따라 지원이 종료되었습니다. 기존 Microsoft SQL Server 2008 R2 스냅샷은 2019년 6월 1일부터 Microsoft SQL Server 2012의 최신 마이너 버전으로 자동 업그레이드될 예정입니다. 자세한 내용은 Amazon RDS에서의 Microsoft SQL Server 2008 R2 지원을 참조하십시오.	2019년 4월 2일
Microsoft SQL Server 2017에 지원되는 상시 가동 가용성 그룹	이제 SQL Server 2017 Enterprise Edition 14.00.3049.1 이상에서 항상 가용성 그룹을 사용할 수 있습니다. 자세한 내용은 Microsoft SQL Server의 다중 AZ 배포를 참조하십시오.	2019년 3월 29일
볼륨 지표 보기	이제 데이터베이스 및 로그 저장에 사용되는 물리적 디바이스인 Amazon Elastic Block Store(Amazon EBS) 볼륨의 지표를 볼 수 있습니다. 자세한 내용은 확장 모니터링 보기를 참조하십시오.	2019년 3월 20일
MySQL 5.7.25 지원	이제 MySQL 버전 5.7.25를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하십시오.	2019년 3월 19일

Amazon RDS for Oracle에서 RMAN DBA 작업 지원	Amazon RDS for Oracle은 이제 RMAN 백업을 포함하여 Oracle Recovery Manager(RMAN) DBA 작업을 지원합니다. 자세한 내용은 Oracle DB DBA Recovery Manager(Recovery Manager) 작업을 참조하세요.	2019년 3월 14일
Amazon RDS for PostgreSQL에서 버전 11.1 지원	이제 PostgreSQL 버전 11.1을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 PostgreSQL 버전 11.1을 참조하십시오.	2019년 3월 12일
Amazon RDS for SQL Server에서 다중 파일 복원 사용 가능	이제 Amazon RDS for SQL Server를 사용하여 여러 파일에서 복원할 수 있습니다. 자세한 내용은 데이터베이스 복원을 참조하십시오.	2019년 3월 11일
MariaDB 10.2.21	이제 MariaDB 버전 10.2.21을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하십시오.	2019년 3월 11일
Amazon RDS for Oracle에서 읽기 전용 복제본 지원	이제 Amazon RDS for Oracle은 Active Data Guard를 통한 읽기 전용 복제본을 지원합니다. 자세한 내용은 읽기 전용 복제본 작업 및 Oracle 읽기 전용 복제본 작업을 참조하십시오.	2019년 3월 11일

Amazon RDS for MariaDB에 Amazon RDS 성능 개선 도우미 사용 가능	이제 Amazon RDS for MariaDB에 Amazon RDS 성능 개선 도우미를 사용할 수 있습니다. 자세한 내용은 Amazon RDS 성능 개선 도우미 사용 단원을 참조하십시오.	2019년 3월 11일
MySQL 8.0.13 및 5.7.24	이제 MySQL 버전 8.0.13 및 5.7.24를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하십시오.	2019년 3월 8일
Amazon RDS for SQL Server에 Amazon RDS 성능 개선 도우미 사용 가능	이제 Amazon RDS for SQL Server에 Amazon RDS 성능 개선 도우미를 사용할 수 있습니다. 자세한 내용은 Amazon RDS 성능 개선 도우미 사용 단원을 참조하십시오.	2019년 3월 4일
Amazon RDS for Oracle에서 Amazon S3 통합 지원	이제 Amazon RDS for Oracle DB 인스턴스와 Amazon S3 버킷 사이에서 파일을 전송할 수 있습니다. 자세한 내용은 Amazon RDS for Oracle과 Amazon S3 통합을 참조하세요.	2019년 2월 26일
Amazon RDS for MySQL 및 Amazon RDS for MariaDB에서 db.t3 DB 인스턴스 클래스 지원	이제 db.t3 DB 인스턴스 클래스를 사용하는 MySQL 또는 MariaDB를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스 단원을 참조하십시오.	2019년 2월 20일

Amazon RDS for MySQL 및 Amazon RDS for MariaDB에서 db.r5 DB 인스턴스 클래스 지원	이제 db.r5 DB 인스턴스 클래스를 사용하는 MySQL 또는 MariaDB를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스 단원을 참조하십시오.	2019년 2월 20일
RDS for MySQL 및 PostgreSQL용 성능 개선 도우미 카운터	이제 MySQL 및 PostgreSQL DB 인스턴스용 성능 개선 도우미 차트에 성능 카운터를 추가할 수 있습니다. 자세한 내용은 성능 개선 도우미 대시보드 구성 요소를 참조하십시오.	2019년 2월 19일
이제 Amazon RDS for PostgreSQL에서 적응형 autovacuum 파라미터 튜닝 지원	Amazon RDS for PostgreSQL을 사용하는 적응형 autovacuum 파라미터 튜닝은 autovacuum 파라미터 값을 자동으로 조정하여 트랜잭션 ID 랩어라운드를 방지합니다. 자세한 내용은 트랜잭션 ID 랩어라운드 가능성 감소를 참조하십시오.	2019년 2월 12일
Amazon RDS for Oracle에서 Oracle APEX 버전 18.1.v1 및 18.2.v1 지원	이제 Amazon RDS for Oracle에서 Oracle Application Express(APEX) 버전 18.1.v1 및 18.2.v1을 지원합니다. 자세한 내용은 Oracle Application Express를 참조하십시오.	2019년 2월 11일

Amazon RDS 성능 개선 도우미에서 RDS for MySQL에 더 많은 SQL 텍스트 보기를 지원	Amazon RDS 성능 개선 도우미는 이제 MySQL DB 인스턴스에 대해 성능 개선 도우미 대시보드에서 더 많은 SQL 텍스트를 볼 수 있도록 지원합니다. 자세한 내용은 성능 개선 도우미 대시보드에서 더 많은 SQL 텍스트 보기 단원을 참조하십시오.	2019년 2월 6일
Amazon RDS for PostgreSQL에서 db.t3 DB 인스턴스 클래스 지원	이제 db.t3 DB 인스턴스 클래스를 사용하는 PostgreSQL 실행 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스 단원을 참조하십시오.	2019년 1월 25일
Amazon RDS for Oracle에서 db.t3 DB 인스턴스 클래스 지원	이제 Oracle 실행 Amazon RDS DB 인스턴스를 생성하여 db.t3 DB 인스턴스 클래스를 사용할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스 단원을 참조하십시오.	2019년 1월 25일
Amazon RDS 성능 개선 도우미에서 Amazon RDS PostgreSQL에 더 많은 SQL 텍스트 보기를 지원	Amazon RDS 성능 개선 도우미는 이제 Amazon RDS PostgreSQL DB 인스턴스에 대해 성능 개선 도우미 대시보드에서 더 많은 SQL 텍스트를 볼 수 있도록 지원합니다. 자세한 내용은 성능 개선 도우미 대시보드에서 더 많은 SQL 텍스트 보기 단원을 참조하십시오.	2019년 1월 24일

Amazon RDS for Oracle에서 SQLT 새 버전 지원	이제 Amazon RDS for Oracle은 SQLT 버전 12.2.180725를 지원합니다. 자세한 내용은 Oracle SQLT 단원을 참조하십시오.	2019년 1월 22일
Amazon RDS for PostgreSQL에서 db.r5 DB 인스턴스 클래스 지원	이제 db.r5 DB 인스턴스 클래스를 사용하는 PostgreSQL 실행 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스 단원을 참조하십시오.	2018년 12월 19일
Amazon RDS for PostgreSQL에서 이제 제한된 암호 관리 지원	Amazon RDS for PostgreSQL에서 `rds.restrict_password_commands` 파라미터와 `rds_password` 역할을 사용하여 사용자 암호 및 암호 만료 변경을 관리할 수 있는 사람을 제한할 수 있습니다. 자세한 내용은 Restricting Password Management 단원을 참조하십시오.	2018년 12월 19일
Amazon RDS for PostgreSQL에서 Amazon CloudWatch Logs에 데이터베이스 로그 업로드 지원	Amazon RDS for PostgreSQL에서 CloudWatch Logs에 데이터베이스 로그를 업로드하는 것을 지원합니다. 자세한 내용은 CloudWatch Logs에 PostgreSQL 로그 게시 섹션을 참조하세요.	2018년 12월 10일

Amazon RDS for Oracle에서 db.r5 DB 인스턴스 클래스 지원	이제 db.r5 DB 인스턴스 클래스를 사용하는 Oracle 실행 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스 단원을 참조하십시오.	2018년 11월 20일
DB 인스턴스를 삭제할 때 백업 보존	Amazon RDS가 DB 인스턴스를 삭제할 때 자동 백업 보존을 지원합니다. 자세한 내용은 백업 작업을 참조하십시오.	2018년 11월 15일
Amazon RDS for PostgreSQL에서 db.m5 DB 인스턴스 클래스 지원	이제 db.m5 DB 인스턴스 클래스를 사용하는 PostgreSQL 실행 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스 단원을 참조하십시오.	2018년 11월 15일
Amazon RDS for Oracle에서 새 메이저 버전 지원	이제 Oracle 버전 12.2를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다.	2018년 11월 13일
Amazon RDS for SQL Server가 상시 작동 지원	Amazon RDS for SQL Server는 상시 가동 가용성 그룹을 지원합니다. 자세한 내용은 Microsoft SQL Server의 다중 AZ 배포를 참조하십시오.	2018년 11월 8일
Amazon RDS for PostgreSQL에서 사용자 지정 DNS 서버를 사용하여 아웃바운드 네트워크 액세스 지원	Amazon RDS for PostgreSQL에서 사용자 지정 DNS 서버를 사용하여 아웃바운드 네트워크 액세스를 지원합니다. 자세한 내용은 아웃바운드 네트워크 액세스에 사용자 지정 DNS 서버 사용을 참조하십시오.	2018년 11월 8일

Amazon RDS for MariaDB, Amazon RDS for MySQL, Amazon RDS for PostgreSQL에서 32TiB 스토리지 지원	이제 MySQL, MariaDB, PostgreSQL에 대한 최대 32TiB 스토리지의 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 DB 인스턴스 스토리지를 참조하십시오.	2018년 11월 7일
Amazon RDS for Oracle은 확장 데이터 유형을 지원합니다.	이제 Oracle을 실행하는 Amazon RDS DB 인스턴스에 대해 확장 데이터 유형을 활성화할 수 있습니다. 확장 데이터 유형의 경우 VARCHAR2, NVARCHAR2 및 RAW 데이터 유형의 최대 크기는 32,767바이트입니다. 자세한 내용은 확장 데이터 유형 사용 단원을 참조하십시오.	2018년 11월 6일
Amazon RDS for Oracle에서 db.m5 DB 인스턴스 클래스 지원	이제 Oracle 실행 Amazon RDS DB 인스턴스를 생성하여 db.m5 DB 인스턴스 클래스를 사용할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스 단원을 참조하십시오.	2018년 11월 2일
SE, SE1 또는 SE2에서 EE로 Amazon RDS for Oracle 마이그레이션	이제 Oracle 데이터베이스 스탠다드 에디션(SE, SE1 또는 SE2)에서 Oracle 데이터베이스 엔터프라이즈 에디션(EE)으로 마이그레이션할 수 있습니다. 자세한 내용은 Oracle 에디션 간 마이그레이션 단원을 참조하십시오.	2018년 10월 31일

Amazon RDS가 이제 다중 AZ 인스턴스를 중지할 수 있음	Amazon RDS가 이제 다중 AZ 배포의 일부인 DB 인스턴스를 중지할 수 있습니다. 이전에는 중지 인스턴스에 다중 AZ 인스턴스에 대한 제한 기능이 있었습니다. 자세한 내용은 Amazon RDS DB 인스턴스 임시 중지 단원을 참조하십시오.	2018년 10월 29일
Amazon RDS for Oracle에 Amazon RDS 성능 개선 도우미를 사용 가능합니다	이제 Amazon RDS for Oracle에 Amazon RDS 성능 개선 도우미를 사용할 수 있습니다. 자세한 내용은 Amazon RDS 성능 개선 도우미 사용 단원을 참조하십시오.	2018년 10월 29일
Amazon RDS for PostgreSQL의 데이터베이스 미리 보기 환경에서 PostgreSQL 버전 11 지원	이제 Amazon RDS for PostgreSQL은 데이터베이스 미리 보기 환경에서 PostgreSQL 버전 11을 지원합니다. 자세한 내용은 데이터베이스 미리 보기 환경에서 Amazon RDS에 대한 PostgreSQL 버전 11 단원을 참조하십시오.	2018년 10월 25일
MySQL에서 새로운 메이저 버전 지원	이제 MySQL 버전 8.0을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하십시오.	2018년 10월 23일
MariaDB에서 새로운 메이저 버전 지원	이제 MariaDB 버전 10.3을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하십시오.	2018년 10월 23일

Amazon RDS for Oracle에서 Oracle JVM 지원	이제 Amazon RDS for Oracle은 Oracle Java 가상 머신(JVM) 옵션을 지원합니다. 자세한 내용은 Oracle Java 가상 머신을 참조하십시오.	2018년 10월 16일
복원 및 특정 시점으로 복구를 위한 사용자 지정 파라미터 그룹	이제 스냅샷을 복원하거나 특정 시점으로 복구 작업을 수행할 때 사용자 지정 파라미터 그룹을 지정할 수 있습니다. 자세한 내용은 DB 스냅샷에서 복원 및 DB 인스턴스를 지정된 시간으로 복원을 참조하십시오.	2018년 10월 15일
Amazon RDS for Oracle에서 32TiB 스토리지 지원	이제 최대 32TiB 스토리지의 Oracle RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 DB 인스턴스 스토리지를 참조하십시오.	2018년 10월 15일
Amazon RDS for MySQL에서 GTID 지원	이제 Amazon RDS for MySQL은 모든 DB 인스턴스 및 복제 구성에서 고유한 전역 트랜잭션 식별자(GTID)를 지원합니다. 자세한 내용은 RDS for MySQL에 대한 GTID 기반 복제 사용을 참조하세요.	2018년 10월 10일
MySQL 5.7.23, 5.6.41 및 5.5.61	이제 MySQL 버전 5.7.23, 5.6.41 및 5.5.61을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하십시오.	2012년 10월 8일

Amazon RDS for Oracle에서 SQLT 새 버전 지원	이제 Amazon RDS for Oracle은 SQLT 버전 12.2.180331을 지원합니다. 자세한 내용은 Oracle SQLT 단원을 참조하십시오.	2018년 10월 4일
Amazon RDS for PostgreSQL에서 이제 IAM 인증 지원	이제 Amazon RDS for PostgreSQL은 IAM 인증을 지원합니다. 자세한 내용은 MySQL 및 PostgreSQL에 대한 IAM 데이터베이스 인증을 참조하십시오.	2018년 9월 27일
Amazon RDS DB 인스턴스에 대한 삭제 방지 활성화 가능	DB 인스턴스에 대해 삭제 방지를 활성화하면 모든 사용자가 데이터베이스를 삭제할 수 없습니다. 자세한 내용은 DB 인스턴스 삭제를 참조하십시오.	2018년 9월 26일
Amazon RDS for MySQL 및 Amazon RDS for MariaDB는 db.m5 DB 인스턴스 클래스 지원	이제 db.m5 DB 인스턴스 클래스를 사용하는 MySQL 또는 MariaDB를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스 단원을 참조하십시오.	2018년 9월 18일
Amazon RDS는 이제 SQL Server 2017에 대한 업그레이드 지원	SQL Server 2008을 제외한 어떤 버전에서든 SQL Server 2017로 기존 DB 인스턴스를 업그레이드할 수 있습니다. SQL Server 2008에서 업그레이드하려면 먼저 다음 버전 중 하나로 업그레이드하십시오. 자세한 내용은 Microsoft SQL 서버 DB 엔진 업그레이드를 참조하십시오.	2018년 9월 11일

Amazon RDS for PostgreSQL에서 이제 데이터베이스 미리 보기 환경에서 PostgreSQL 버전 11 베타 3 지원	이 릴리스에서는 Write-Ahead Log(WAL) 세그먼트 크기(wal_segment_size)가 이제 64MB로 설정됩니다. PostgreSQL 버전 11 베타 3에 대한 자세한 내용은 PostgreSQL 11 베타 3 릴리스를 참조하십시오. 데이터베이스 미리 보기 환경에 대한 자세한 내용은 데이터베이스 미리 보기 환경 작업을 참조하십시오.	2018년 9월 7일
Amazon Aurora 사용 설명서	Amazon Aurora 사용 설명서는 모든 Amazon Aurora 개념을 설명하고 콘솔과 명령줄 인터페이스로 다양한 기능을 사용하는 방법에 대한 지침을 제공합니다. Amazon RDS 사용 설명서는 이제 Aurora 이외의 다른 데이터베이스 엔진에 적용됩니다.	2018년 8월 31일
RDS for MySQL에 Amazon RDS 성능 개선 도우미 사용 가능	이제 RDS for MySQL에 Amazon RDS 성능 개선 도우미를 사용할 수 있습니다. 자세한 내용은 Amazon RDS 성능 개선 도우미 사용 단원을 참조하십시오.	2018년 8월 28일
Aurora PostgreSQL 호환 버전에서 이제 Aurora Auto Scaling 지원	이제 Aurora PostgreSQL 호환 버전에 Aurora 복제본의 Auto Scaling을 사용할 수 있습니다. 자세한 내용은 Aurora 복제본을 사용한 Amazon Aurora Auto Scaling을 참조하세요.	2018년 8월 16일

Aurora Serverless for Aurora MySQL	Aurora Serverless는 Amazon Aurora에 대한 온디맨드 방식의 Auto Scaling 구성입니다. 자세한 내용은 Amazon Aurora Serverless 사용 섹션을 참조하세요.	2018년 8월 9일
MySQL 5.7.22 및 5.6.40	이제 MySQL 버전 5.7.22 및 5.6.40를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전을 참조하십시오.	2018년 8월 6일
Aurora는 이제 중국(닝샤) 지역에서 사용 가능	이제 Aurora MySQL 및 Aurora PostgreSQL을 중국(닝샤) 리전에서 사용할 수 있습니다. 자세한 내용은 Amazon Aurora MySQL 가용성 및 Amazon Aurora PostgreSQL 가용성을 참조하십시오.	2018년 8월 6일
Amazon RDS for MySQLS에서 지연 복제 지원	이제 Amazon RDS for MySQL에서 지연 복제를 재해 복구를 위한 전략으로 지원합니다. 자세한 내용은 MySQL을 사용한 지연 복제 구성을 참조하십시오.	2018년 8월 6일
Aurora MySQL에 Amazon RDS 성능 개선 도우미 사용 가능	이제 Aurora MySQL에 Amazon RDS 성능 개선 도우미를 사용할 수 있습니다. 자세한 내용은 Amazon RDS 성능 개선 도우미 사용 단원을 참조하십시오.	2018년 8월 6일

Amazon CloudWatch와 Amazon RDS 성능 개선 도우미 통합	Amazon RDS 성능 개선 도우미는 지표를 Amazon CloudWatch에 자동으로 게시합니다. 자세한 내용은 CloudWatch에 게시되는 성능 개선 도우미 지표 단원을 참조하십시오.	2018년 8월 6일
Amazon RDS 권장 사항	Amazon RDS에서 이제 데이터베이스 리소스에 대한 자동 권장 사항을 제공합니다. 자세한 내용은 Amazon RDS 권장 사항 사용을 참조하십시오.	2018년 7월 25일
AWS 리전 간 증분 스냅샷 복사본	Amazon RDS에서 비암호화 및 암호화된 인스턴스 모두에 대해 AWS 리전 간 증분 스냅샷 복사본을 지원합니다. 자세한 내용은 AWS 리전에 걸쳐 스냅샷 복사를 참조하세요.	2018년 7월 24일
Amazon RDS for PostgreSQL에 Amazon RDS 성능 개선 도우미 사용 가능	이제 Amazon RDS for PostgreSQL에 Amazon RDS 성능 개선 도우미를 사용할 수 있습니다. 자세한 내용은 Amazon RDS 성능 개선 도우미 사용 단원을 참조하십시오.	2018년 7월 18일
Amazon RDS for Oracle에서 Oracle APEX 버전 5.1.4.v1 지원	Amazon RDS for Oracle에서 이제 Oracle Application Express(APEX) 버전 5.1.4.v1을 지원합니다. 자세한 내용은 Oracle Application Express를 참조하십시오.	2018년 7월 10일

Amazon RDS for Oracle은 Amazon CloudWatch Logs에 로그 게시 지원	이제 Amazon RDS for Oracle은 CloudWatch Logs의 로그 그룹에 알림, 감사, 추적 및 리스너 로그 데이터 게시를 지원합니다. 자세한 내용은 Amazon CloudWatch Logs에 Oracle 로그 게시 섹션을 참조하세요.	2018년 7월 9일
MariaDB 10.2.15, 10.1.34 및 10.0.35	이제 MariaDB 버전 10.2.15, 10.1.34 및 10.0.35를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MariaDB 버전을 참조하십시오.	2018년 7월 5일
Aurora PostgreSQL 1.2를 사용할 수 있으며 PostgreSQL 9.6.8과 호환됨	이제 Aurora PostgreSQL 1.2를 사용할 수 있으며 PostgreSQL 9.6.8과 호환됩니다. 자세한 내용은 버전 1.2 단원을 참조하십시오.	2018년 27월 6일
Amazon RDS PostgreSQL용 읽기 전용 복제본에서 다중 AZ 배포 지원	이제 Amazon RDS PostgreSQL의 RDS 읽기 전용 복제본이 여러 가용 영역을 지원합니다. 자세한 내용은 PostgreSQL 읽기 전용 복제본 작업을 참조하십시오.	2018년 6월 25일
Aurora PostgreSQL에 성능 개선 도우미 사용 가능	성능 개선 도우미를 Aurora PostgreSQL에 일반적으로 사용할 수 있으며, 성능 데이터의 보존 연장이 지원됩니다. 자세한 내용은 Amazon RDS 성능 개선 도우미 사용 단원을 참조하십시오.	2018년 6월 21일

미국 서부(캘리포니아 북부) 리전에서 Aurora PostgreSQL사용 가능	이제 미국 서부(캘리포니아 북부) 리전에서 Aurora PostgreSQL을 사용할 수 있습니다. 자세한 내용은 Amazon Aurora PostgreSQL 가용성을 참조하십시오.	2018년 6월 11일
Amazon RDS for Oracle에서 이제 CPU 구성 지원	Amazon RDS for Oracle은 DB 인스턴스 클래스의 프로세스에 대한 CPU 코어 수 및 각 코어의 스레드 수 구성을 지원합니다. 자세한 내용은 DB 인스턴스 클래스의 프로세서 구성을 참조하십시오.	2018년 6월 5일

이전 업데이트

다음 표에서는 2018년 6월 이전 Amazon RDS 사용 설명서의 각 릴리스에서 변경된 중요 사항에 대해 설명합니다.

변경 사항	설명	변경 날짜
Amazon RDS for PostgreSQL, 데이터베이스 미리 보기 환경에서 PostgreSQL 버전 11 베타 1 지원	PostgreSQL 버전 11 베타 1에는 PostgreSQL 11 베타 1 출시에서 설명하는 여러 개선 사항이 포함되어 있습니다. 데이터베이스 미리 보기 환경에 대한 자세한 내용은 데이터베이스 미리 보기 환경 작업 섹션을 참조하세요.	2018년 5월 31일
Amazon RDS for Oracle, 이제 TLS 버전 1.0 및 1.2 지원	Amazon RDS for Oracle은 TLS(전송 계층 보안) 버전 1.0 및 1.2를 지원합니다. 자세한 내용은 Oracle SSL 옵션에 대한 TLS 버전 섹션을 참조하세요.	2018년 5월 30일
Aurora MySQL, Amazon	이제 Aurora MySQL은 CloudWatch Logs의 로그 그룹에 일반, 느린, 감사, 오류 로그 데이터 게시를 지원합니	2018년 5월 23일

변경 사항	설명	변경 날짜
CloudWatch Logs에 로그 게시 지원	다. 자세한 내용은 CloudWatch Logs에 Aurora MySQL 게시 단원을 참조하십시오.	
Amazon RDS PostgreSQL의 데이터베이스 미리 보기 환경	이제 Amazon RDS PostgreSQL의 새 인스턴스를 미리 보기 모드로 시작할 수 있습니다. 데이터베이스 미리 보기 환경에 대한 자세한 내용은 데이터베이스 미리 보기 환경 작업 단원을 참조하십시오.	2018년 5월 22일
Amazon RDS for Oracle DB 인스턴스, 새로운 DB 인스턴스 클래스 지원	Oracle DB 인스턴스가 이제 db.x1e 및 db.x1 DB 인스턴스 클래스를 지원합니다. 자세한 내용은 DB 인스턴스 클래스 및 RDS for Oracle DB 인스턴스 클래스 단원을 참조하십시오.	2018년 5월 22일
이제 Amazon RDS PostgreSQL이 읽기 전용 복제본에서 postgres_fdw를 지원합니다.	postgres_fdw를 사용하여 읽기 전용 복제본에서 원격 서버로 연결할 수 있습니다. 자세한 내용은 postgres_fdw 확장을 사용하여 외부 데이터 액세스 섹션을 참조하세요.	2018년 5월 17일
Amazon RDS for Oracle에서 이제 sqlnet.ora 파라미터 설정 지원	이제 Amazon RDS for Oracle로 sqlnet.ora 파라미터를 설정할 수 있습니다. 자세한 내용은 sqlnet.ora 파라미터를 사용하여 연결 속성 수정 섹션을 참조하세요.	2018년 5월 10일
아시아 태평양(서울) 리전에서 Aurora PostgreSQL를 이용할 수 있습니다.	Aurora PostgreSQL는 이제 아시아 태평양(서울) 리전에서 사용할 수 있습니다. 자세한 내용은 Amazon Aurora PostgreSQL 가용성을 참조하십시오.	2018년 5월 9일
Aurora MySQL은 역추적을 지원	이제 Aurora MySQL은 백업에서 데이터를 복구하지 않고도 특정 시간으로 DB 클러스터 "되감기"를 지원합니다. 자세한 내용은 Amazon Aurora DB 클러스터 역추적을 참조하십시오.	2018년 5월 9일

변경 사항	설명	변경 날짜
Aurora MySQL, 외부 MySQL에서 암호화된 마이그레이션 및 복제 지원	이제 Aurora MySQL은 외부 MySQL 데이터베이스에서 암호화된 마이그레이션 및 복제를 지원합니다. 자세한 내용은 외부 MySQL 데이터베이스에서 Amazon Aurora MySQL DB 클러스터로 데이터 마이그레이션과 Aurora 및 MySQL 간 또는 Aurora 및 다른 Aurora DB 클러스터 간 복제를 참조하십시오.	2018년 25월 4일
Copy-on-Write 프로토콜에 대한 Aurora PostgreSQL 호환 버전 지원	이제 Aurora PostgreSQL 데이터베이스 클러스터의 데이터베이스를 복제할 수 있습니다. 자세한 내용은 Aurora DB 클러스터에서 데이터베이스 복제를 참조하십시오.	2018년 10월 4일
MariaDB 10.2.12, 10.1.31 및 10.0.34	이제 MariaDB 버전 10.2.12, 10.1.31 및 10.0.34를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS MariaDB 버전 섹션을 참조하세요.	2018년 3월 21일
새 리전을 위한 Aurora PostgreSQL 지원	이제 EU(런던) 및 아시아 태평양(싱가포르) 리전에서 Aurora PostgreSQL을 사용할 수 있습니다. 자세한 내용은 Amazon Aurora PostgreSQL 가용성을 참조하십시오.	2018년 3월 13일
MySQL 5.7.21, 5.6.39 및 5.5.59	이제 MySQL 버전 5.7.21, 5.6.39 및 5.5.59를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전 섹션을 참조하세요.	2018년 3월 9일
Amazon RDS for Oracle, 이제 Oracle REST Data Services 지원	Amazon RDS for Oracle은 APEX 옵션의 일부로 Oracle REST Data Services를 지원합니다. 자세한 내용은 Oracle Application Express(APEX) 섹션을 참조하세요.	2018년 3월 9일
Amazon Aurora MySQL 호환 버전, 새로운 AWS 리전에서 사용 가능	이제 아시아 태평양(싱가포르) 리전에서 Aurora MySQL을 사용할 수 있습니다. Aurora MySQL을 사용할 수 있는 AWS 리전의 전체 목록은 Amazon Aurora MySQL 가용성을 참조하세요.	2018년 3월 6일

변경 사항	설명	변경 날짜
Microsoft SQL Server를 실행하는 Amazon RDS DB 인스턴스는 CDC(변경 데이터 캡처)를 지원	이제 Amazon RDS for Microsoft SQL Server를 실행하는 DB 인스턴스는 CDC(변경 데이터 캡처)를 지원합니다. 자세한 내용은 Microsoft SQL Server DB 인스턴스에 대한 변경 데이터 캡처 지원 섹션을 참조하세요.	2018년 2월 6일
Aurora MySQL, 새로운 메이저 버전 지원	이제 MySQL 버전 5.7을 실행하는 Aurora MySQL DB 클러스터를 생성할 수 있습니다. 자세한 내용은 Amazon Aurora MySQL 데이터베이스 엔진 업데이트 2018-02-06을 참조하십시오.	2018년 2월 6일
Amazon CloudWatch Logs에 MySQL 및 MariaDB 로그 게시	이제 CloudWatch Logs에 MySQL 및 MariaDB 로그 데이터를 게시할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs에 MySQL 로그 게시 및 Amazon CloudWatch Logs에 MariaDB 로그 게시 단원을 참조하십시오.	2018년 1월 17일
읽기 전용 복제본에 대한 다중 AZ 지원	이제 읽기 전용 복제본을 다중 AZ DB 인스턴스로 생성할 수 있습니다. Amazon RDS는 복제본에 대한 장애 조치 지원을 위해 대기 복제본을 다른 가용 영역에 생성합니다. 읽기 전용 복제본을 다중 AZ DB 인스턴스로 생성하는 작업은 원본 데이터베이스가 다중 AZ DB 인스턴스인지 여부와는 무관합니다. 자세한 내용은 DB 인스턴스 읽기 전용 복제본 작업 섹션을 참조하세요.	2018년 1월 11일
Amazon RDS for MariaDB는 새로운 메이저 버전 지원	이제 MariaDB 버전 10.2를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS에서 MariaDB 10.2 지원을 참조하세요.	2018년 1월 3일
Amazon Aurora PostgreSQL 호환 버전, 새로운 AWS 리전에서 사용 가능	이제 EU(파리) 리전에서 Aurora PostgreSQL을 사용할 수 있습니다. Aurora PostgreSQL을 사용할 수 있는 AWS 리전의 전체 목록은 Amazon Aurora PostgreSQL 가용성을 참조하세요.	2017년 12월 22일

변경 사항	설명	변경 날짜
Aurora PostgreSQL은 이제 새 인스턴스 유형을 지원	이제 Aurora PostgreSQL은 새 인스턴스 유형을 지원합니다. 전체 인스턴스 유형 목록은 DB 인스턴스 클래스 선택을 참조하십시오.	2017년 12월 20일
Amazon Aurora MySQL 호환 버전, 새로운 AWS 리전에서 사용 가능	이제 EU(파리) 리전에서 Aurora MySQL을 사용할 수 있습니다. Aurora MySQL을 사용할 수 있는 AWS 리전의 전체 목록은 Amazon Aurora MySQL 가용성을 참조하세요.	2017년 12월 18일
Aurora MySQL은 해시 조인을 지원	이 기능은 동등 조인을 사용하여 많은 양의 데이터를 조인해야 하는 경우 쿼리 성능을 향상시킬 수 있습니다. 자세한 내용은 Aurora MySQL에서 해시 조인 작업을 참조하십시오.	2017년 12월 11일
Aurora MySQL은 AWS Lambda 함수를 호출하는 네이티브 함수를 지원합니다.	Aurora MySQL을 사용할 때 `lambda_sync` 및 `lambda_async` 네이티브 함수를 호출할 수 있습니다. 자세한 내용은 Amazon Aurora MySQL DB 클러스터에서 Lambda 함수 호출을 참조하십시오.	2017년 12월 11일
Aurora PostgreSQL HIPAA 자격 획득 추가	Aurora PostgreSQL은 HIPAA 준수 애플리케이션 구축을 지원합니다. 자세한 내용은 Amazon Aurora PostgreSQL 작업을 참조하십시오.	2017년 12월 6일
추가 AWS 리전에서 PostgreSQL 호환성을 갖춘 Amazon Aurora 사용 가능	PostgreSQL 호환성을 갖춘 Amazon Aurora를 이제 4개의 새로운 AWS 리전에서 사용할 수 있습니다. 자세한 내용은 Amazon Aurora PostgreSQL 가용성을 참조하십시오.	2017년 11월 22일
Microsoft SQL Server를 실행하는 Amazon RDS DB 인스턴스의 스토리지 수정	이제 SQL Server를 실행하는 Amazon RDS DB 인스턴스의 스토리지를 수정할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 수정 섹션을 참조하세요.	2017년 11월 21일

변경 사항	설명	변경 날짜
Amazon RDS는 Linux 기반 엔진의 16TiB 스토리지 지원	이제 MySQL, MariaDB, PostgreSQL 및 Oracle RDS DB 인스턴스는 스토리지에서 최대 16TiB까지 생성할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 섹션을 참조하세요.	2017년 11월 21일
Amazon RDS는 신속한 스토리지 확장을 지원	이제 몇 분 안에 MySQL, MariaDB, PostgreSQL 및 Oracle RDS DB 인스턴스에 스토리지를 추가할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 섹션을 참조하세요.	2017년 11월 21일
Amazon RDS는 MariaDB 버전 10.1.26 및 10.0.32를 지원	이제 MariaDB 버전 10.1.26 및 10.0.32를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS MariaDB 버전 섹션을 참조하세요.	2017년 11월 20일
이제 Amazon RDS for Microsoft SQL Server는 새로운 DB 인스턴스 클래스를 지원	이제 db.r4 및 db.m4.16xlarge DB 인스턴스 클래스를 사용하는 SQL Server를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Microsoft SQL Server를 위한 DB 인스턴스 클래스 지원 섹션을 참조하세요.	2017년 11월 20일
이제 Amazon RDS for MySQL 및 MariaDB는 새로운 DB 인스턴스 클래스를 지원	이제 db.r4, db.m4.16xlarge, db.t2.xlarge 및 db.t2.2xlarge DB 인스턴스 클래스를 사용하는 MySQL 및 MariaDB를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요.	2017년 11월 20일
SQL Server 2017	이제 Microsoft SQL Server 2017을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. SQL Server 2016 SP1 CU5를 실행하여 DB 인스턴스를 생성할 수도 있습니다. 자세한 내용은 Amazon RDS for Microsoft SQL Server 섹션을 참조하세요.	2017년 11월 17일

변경 사항	설명	변경 날짜
Amazon S3에서 MySQL 백업 복원	이제 온프레미스 데이터베이스의 백업을 생성하여 Amazon S3에 저장한 다음 MySQL을 실행하는 새로운 Amazon RDS DB 인스턴스에 백업 파일을 복원할 수 있습니다. 자세한 내용은 MySQL DB 인스턴스로 백업 복원 섹션을 참조하세요.	2017년 11월 17일
Aurora 복제본을 통한 Auto Scaling	이제 Amazon Aurora MySQL은 Aurora Auto Scaling을 지원합니다. Aurora Auto Scaling은 연결이나 워크로드 증가 또는 감소에 따라 Aurora 복제본의 수를 동적으로 조정합니다. 자세한 내용은 Aurora 복제본을 사용한 Amazon Aurora Auto Scaling을 참조하세요.	2017년 11월 17일
Oracle 기본 에디션 지원	이제 Amazon RDS for Oracle DB 인스턴스는 DB 인스턴스의 기본 에디션 설정을 지원합니다. 자세한 내용은 DB 인스턴스의 기본 에디션 설정 섹션을 참조하세요.	2017년 11월 3일
Oracle DB 인스턴스 파일 확인	이제 Amazon RDS for Oracle DB 인스턴스는 Oracle Recovery Manager(RMAN)의 논리적 확인 유틸리티를 사용하여 DB 인스턴스 파일을 확인하는 작업을 지원합니다. 자세한 내용은 RDS for Oracle DB에서 데이터베이스 파일 검증 섹션을 참조하세요.	2017년 11월 3일
OEM 13c용 Management Agent	Amazon RDS for Oracle DB 인스턴스가 이제 Oracle Enterprise Manager(OEM) Cloud Control 13c용 Management Agent를 지원합니다. 자세한 내용은 Oracle Management Agent for Enterprise Manager Cloud Control 섹션을 참조하세요.	2017년 11월 1일
Microsoft SQL Server 스냅샷의 스토리지 재구성	이제 Microsoft SQL Server를 실행하는 Amazon RDS DB 인스턴스로 스냅샷을 복원할 때 스토리지를 재구성할 수 있습니다. 자세한 내용은 DB 인스턴스 복원 섹션을 참조하세요.	2017년 10월 26일

변경 사항	설명	변경 날짜
Aurora MySQL 호환 버전의 비동기식 키 미리 가져오기	비동기식 키 미리 가져오기(AKP)는 필요하기 전에 메모리 키를 미리 가져와서 캐싱되지 않은 인덱스 조인 성능을 향상시킵니다. 자세한 내용은 Amazon Aurora에서 비동기식 키 미리 가져오기 작업을 참조하십시오.	2017년 10월 26일
MySQL 5.7.19, 5.6.37 및 5.5.57	이제 MySQL 버전 5.7.19, 5.6.37 및 5.5.57를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS의 MySQL 버전 섹션을 참조하세요.	2017년 10월 25일
Amazon Aurora PostgreSQL 호환 에디션 정식 출시	Amazon Aurora PostgreSQL 호한 에디션은 새로 배포하는 PostgreSQL이든, 혹은 기존에 배포한 PostgreSQL이든 상관없이 설치, 조작 및 조정이 간편하고 비용 효율적이기 때문에 비즈니스와 애플리케이션에 더욱 많은 시간을 투자할 수 있습니다. 자세한 내용은 Amazon Aurora PostgreSQL 작업을 참조하십시오.	2017년 10월 24일
Amazon RDS for Oracle DB 인스턴스, 새로운 DB 인스턴스 클래스 지원	이제 Amazon RDS for Oracle DB 인스턴스는 메모리에 최적화된 차세대(db.r4) 인스턴스 클래스를 지원합니다. 이제 Amazon RDS for Oracle DB 인스턴스는 db.m4.16xlarge, db.t2.xlarge, db.t2.2xlarge 등 새로운 현재 세대의 인스턴스 클래스도 지원합니다. 자세한 내용은 DB 인스턴스 클래스 및 RDS for Oracle DB 인스턴스 클래스 단원을 참조하십시오.	2017년 10월 23일
새로운 기능	새로운 예약 인스턴스와 기존 예약 인스턴스 모두 이제 동일한 DB 인스턴스 클래스에서 다양한 크기를 지원할 수 있게 되었습니다. 동일한 AWS 리전, 데이터베이스 엔진 및 인스턴스 패밀리를 비롯해 AZ 구성 간에서도 유연한 크기의 예약 인스턴스를 DB 인스턴스에 사용할 수 있습니다. 이렇게 유연한 크기의 예약 인스턴스를 지원하는 데이터베이스 엔진으로는 Amazon Aurora, MariaDB, MySQL, Oracle(Bring Your Own License), PostgreSQL이 있습니다. 자세한 내용은 유연한 크기의 예약 DB 인스턴스 섹션을 참조하세요.	2017년 10월 11일

변경 사항	설명	변경 날짜
새로운 기능	이제 최적의 성능을 얻기 위해 Oracle SQLT 옵션을 사용하여 SQL 문을 튜닝할 수 있습니다. 자세한 내용은 Oracle SQLT 섹션을 참조하세요.	2017년 9월 22일
새로운 기능	Amazon RDS for Oracle DB 인스턴스의 기존 수동 DB 스냅샷이 있는 경우 이제 Oracle 데이터베이스 엔진의 최신 버전으로 스냅샷을 업그레이드할 수 있습니다. 자세한 내용은 Oracle DB 스냅샷 업그레이드 섹션을 참조하세요.	2017년 9월 20일
새로운 기능	이제 Oracle Spatial을 사용하여 Oracle을 실행하는 Amazon RDS DB 인스턴스의 공간 데이터를 저장, 검색, 업데이트 및 쿼리할 수 있습니다. 자세한 내용은 Oracle Spatial 섹션을 참조하세요.	2017년 9월 15일
새로운 기능	이제 Oracle Locator를 사용하여 Oracle을 실행하는 Amazon RDS DB 인스턴스와 함께 인터넷 및 무선 서비스 기반 애플리케이션과 파트너 기반 GIS 솔루션을 지원할 수 있습니다. 자세한 내용은 Oracle Locator 섹션을 참조하세요.	2017년 9월 15일
새로운 기능	이제 Oracle Multimedia를 사용하여 Oracle을 실행하는 Amazon RDS DB 인스턴스의 이미지, 오디오, 동영상 및 기타 이종 미디어 데이터를 저장하고 관리하며 검색할 수 있습니다.	2017년 9월 15일
새로운 기능	이제 감사 로그를 Amazon Aurora MySQL DB 클러스터에서 Amazon CloudWatch Logs로 내보낼 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs에 Aurora MySQL 게시를 참조하십시오.	2017년 9월 14일
새로운 기능	이제 Amazon RDS는 Oracle을 실행하는 DB 인스턴스에 대한 Oracle Application Express(APEX)의 여러 버전을 지원합니다. 자세한 내용은 Oracle Application Express(APEX) 섹션을 참조하세요.	2017년 9월 13일

변경 사항	설명	변경 날짜
새로운 기능	이제 Amazon Aurora를 사용하여 암호화되지 않거나 암호화된 DB 스냅샷 또는 MySQL DB 인스턴스를 암호화된 Aurora MySQL DB 클러스터로 마이그레이션할 수 있습니다. 자세한 내용은 Aurora로 RDS for MySQL 스냅샷 마이그레이션 및 Aurora 읽기 전용 복제본을 사용하여 MySQL DB 인스턴스에서 Amazon Aurora MySQL DB 클러스터로 데이터 마이그레이션을 참조하tpdy.	2017년 9월 5일
새로운 기능	Amazon RDS for Microsoft SQL Server 데이터베이스를 사용하여 HIPAA 인증 애플리케이션을 개발할 수 있습니다. 자세한 내용은 Microsoft SQL Server DB 인스턴스에 대한 규정 준수 프로그램 지원 섹션을 참조하세요.	2017년 8월 31일
새로운 기능	이제 Amazon RDS for MariaDB 데이터베이스를 사용하여 HIPAA 인증 애플리케이션을 개발할 수 있습니다. 자세한 내용은 Amazon RDS for MariaDB 섹션을 참조하세요.	2017년 8월 31일
새로운 기능	이제 할당된 스토리지를 최대 16TiB까지 설정하고 프로비저닝된 IOPS를 1:1–50:1의 스토리지 범위로 설정하여 Microsoft SQL Server를 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS DB 인스턴스 스토리지 섹션을 참조하세요.	2017년 8월 22일
새로운 기능	이제는 EU(프랑크푸르트) 리전에서 Microsoft SQL Server 기반 DB 인스턴스에 다중 AZ 배포를 사용할 수 있습니다. 자세한 내용은 Amazon RDS for Microsoft SQL Server의 다중 AZ 배포 섹션을 참조하세요.	2017년 8월 3일
새로운 기능	이제 MariaDB 버전 10.1.23 및 10.0.31을 실행하는 Amazon RDS DB 인스턴스를 생성할 수 있습니다. 자세한 내용은 Amazon RDS MariaDB 버전 섹션을 참조하세요.	2017년 7월 17일

변경 사항	설명	변경 날짜
새로운 기능	이제 Amazon RDS는 모든 AWS 리전의 라이선스 포함 서비스 모델에서 Microsoft SQL Server Enterprise Edition을 지원합니다. 자세한 내용은 Amazon RDS의 Microsoft SQL Server 라이선싱 섹션을 참조하세요.	2017년 7월 13일
새로운 기능	이제 Amazon RDS for Oracle은 데이터베이스 확장성을 향상하기 위해 Linux 커널 방대한 페이지를 지원합니다. 방대한 페이지를 사용하면 페이지 표가 작아지고 메모리 관리에 사용되는 CPU 시간이 줄어 대용량 데이터베이스 인스턴스의 성능이 높아집니다. Oracle 버전 12.1.0.2 및 11.2.0.4의 모든 에디션을 실행하는 Amazon RDS DB 인스턴스에서 방대한 페이지를 사용할 수 있습니다. 자세한 내용은 RDS for Oracle 인스턴스에 HugePages 활성화 섹션을 참조하세요.	2017년 7월 7일
새로운 기능	Aurora가 아닌 모든 DB 엔진에서 db.t2.small 및 db.t2.medium DB 인스턴스 클래스에 유휴 시 암호화(EAR)를 지원하도록 업데이트되었습니다. 자세한 내용은 Amazon RDS 암호화 가용성 섹션을 참조하세요.	2017년 27월 6일
새로운 기능	유럽(프랑크푸르트) 리전에서 Amazon Aurora를 지원하도록 업데이트되었습니다. 자세한 내용은 Amazon Aurora MySQL 가용성을 참조하십시오.	2017년 6월 16일
새로운 기능	이제 AWS 리전 간에 DB 스냅샷을 복사할 때 옵션 그룹을 지정할 수 있습니다. 자세한 내용은 옵션 그룹에 대한 고려 사항 섹션을 참조하세요.	2017년 6월 12일
새로운 기능	이제 특수 DB 인스턴스에서 만들어진 DB 스냅샷을 AWS 리전 간에 복사할 수 있습니다. Oracle TDE, Microsoft SQL Server TDE 및 미러링을 포함한 Microsoft SQL Server 다중 AZ를 사용하는 DB 인스턴스에서 스냅샷을 복사할 수 있습니다. 자세한 내용은 DB 스냅샷 복사 섹션을 참조하세요.	2017년 6월 12일

변경 사항	설명	변경 날짜
새로운 기능	이제 Amazon Aurora를 사용하여 Amazon Aurora DB 클러스터의 모든 데이터베이스를 빠르고 비용 효과적으로 복사할 수 있습니다. 자세한 내용은 Aurora DB 클러스터에서 데이터베이스 복제를 참조하십시오.	2017년 6월 12일
새로운 기능	이제 Amazon RDS는 Microsoft SQL Server 2016 SP1 CU2를 지원합니다. 자세한 내용은 Amazon RDS for Microsoft SQL Server 섹션을 참조하세요.	2017년 6월 7일
미리 보기	Amazon Aurora의 PostgreSQL 호환성 미리 보기. 자세한 내용은 Amazon Aurora PostgreSQL 작업을 참조하십시오.	2017년 4월 19일
새로운 기능	이제 Amazon Aurora을 사용하여 ALTER TABLE tbl_name ADD COLUMN col_name column_definition 작업을 거의 동시에 실행할 수 있습니다. 이 작업은 테이블을 복사하거나 다른 DML 명령문에 영향을 거의 주지 않고 완료됩니다. 자세한 내용은 빠른 DDL을 사용하여 Amazon Aurora에서 테이블 변경을 참조하십시오.	2017년 5월 4일
새로운 기능	새 모니터링 명령 SHOW VOLUME STATUS를 추가하여 볼륨의 노드 및 디스크 수를 표시합니다. 자세한 내용은 Aurora DB 클러스터에서 볼륨 상태 표시를 참조하십시오.	2017년 5월 4일
새 기능	이제 Amazon RDS에서 Oracle용 사용자 지정 암호 확인 기능으로 자체 사용자 지정 로직을 사용할 수 있습니다. 자세한 내용은 사용자 지정 암호 확인 함수 생성 섹션을 참조하세요.	2017년 3월 21일
새로운 기능	이제 Amazon RDS에서 Oracle DB 인스턴스의 온라인 및 아카이브 다시 실행 로그 파일에 액세스할 수 있습니다. 자세한 내용은 온라인 및 아카이빙된 다시 실행 로그 액세스 섹션을 참조하세요.	2017년 3월 21일

변경 사항	설명	변경 날짜
새로운 기능	이제 동일한 리전의 계정 간에 암호화된 DB 클러스터 스냅샷과 암호화되지 않은 DB 클러스터 스냅샷을 모두 복사할 수 있습니다. 자세한 내용은 계정 간 DB 클러스터 스냅샷 복사를 참조하십시오.	2017년 3월 7일
새로운 기능	이제 동일한 리전의 계정 간에 암호화된 DB 클러스터 스냅샷을 공유할 수 있습니다. 자세한 내용은 DB 클러스터 스냅샷 공유를 참조하십시오.	2017년 3월 7일
새로운 기능	이제 암호화된 Amazon Aurora MySQL DB 클러스터를 복제하여 리전 간 Aurora 복제본을 생성할 수 있습니다. 자세한 내용은 AWS 리전 간 Aurora MySQL DB 클러스터 복제를 참조하세요.	2017년 3월 7일
새로운 기능	이제 Microsoft SQL Server를 실행하는 DB 인스턴스에 대한 모든 연결이 SSL(Secure Sockets Layer)을 사용하도록 요구할 수 있습니다. 자세한 내용은 ??? 섹션을 참조하세요.	2017년 2월 27일
새로운 기능	이제 현지 시간대를 15가지 추가 시간대 중 하나로 설정할 수 있습니다. 자세한 내용은 지원되는 시간대 섹션을 참조하세요.	2017년 2월 27일
새로운 기능	이제 Amazon RDS 프로시저 `msdb.dbo.rds_shrink_tempdbfile` 을 사용하여 Microsoft SQL Server를 실행하는 DB 인스턴스의 tempdb 데이터베이스를 축소할 수 있습니다. 자세한 내용은 tempdb 데이터베이스 축소 섹션을 참조하세요.	2017년 2월 17일
새로운 기능	이제 Enterprise와 Standard Edition Microsoft SQL Server 데이터베이스를 Amazon RDS DB 인스턴스에서 Amazon S3로 내보낼 때 백업 파일을 압축할 수 있습니다. 자세한 내용은 백업 파일 압축 섹션을 참조하세요.	2017년 2월 17일

변경 사항	설명	변경 날짜
새로운 기능	Amazon RDS는 이제 Oracle을 실행하는 DB 인스턴스에서 아웃바운드 네트워크 액세스에 사용되는 DNS 이름을 확인하는 사용자 지정 DNS 서버를 지원합니다. 자세한 내용은 사용자 지정 DNS 서버 설정 섹션을 참조하세요.	2017년 1월 26일
새로운 기능	이제 Amazon RDS는 다른 리전에서의 암호화된 읽기 전용 복제본 생성을 지원합니다. 자세한 내용은 다른 AWS 리전에서 읽기 전용 복제본 생성 및 CreateDBInstanceReadReplica 단원을 참조하십시오.	2017년 1월 23일
새로운 기능	이제 Amazon RDS는 MySQL 5.1에서 MySQL 5.5로 MySQL DB 스냅샷 업그레이드를 지원합니다.	2017년 1월 20일
새로운 기능	Amazon RDS는 이제 MariaDB, MySQL, Oracle, PostgreSQL, Microsoft SQL Server 데이터베이스 엔진의 경우, 암호화된 DB 스냅샷을 다른 리전으로 복사할 수 있도록 지원합니다. 자세한 내용은 DB 스냅샷 복사 및 CopyDBSnapshot 단원을 참조하십시오.	2016년 20월 12일
새로운 기능	이제 Amazon Aurora MySQL은 공간 인덱싱을 지원합니다. 공간 인덱싱은 공간 데이터를 사용하는 쿼리를 위한 대규모 데이터 세트에서의 쿼리 성능을 향상시킵니다. 자세한 내용은 Amazon Aurora MySQL 및 공간 데이터를 참조하십시오.	2016년 14월 12일
새 기능	Amazon RDS는 이제 Oracle을 실행하는 DB 인스턴스에서 아웃바운드 네트워크 액세스를 지원합니다. utl_http, utl_tcp 및 utl_smtp를 사용하여 DB 인스턴스에서 네트워크에 연결할 수 있습니다. 자세한 내용은 인증서 및 Oracle Wallet을 사용하여 UTL_HTTP 액세스 구성 섹션을 참조하세요.	2016년 5월 12일

변경 사항	설명	변경 날짜
새로운 기능	Amazon RDS는 MySQL 버전 5.1에 대한 지원을 만료했습니다. 하지만 기존 MySQL 5.1 스냅샷을 MySQL 5.5 인스턴스로 복원할 수 있습니다. 자세한 내용은 RDS for MySQL에 대해 지원되는 스토리지 엔진 섹션을 참조하세요.	2016년 11월 15일
새 기능	이제 Amazon RDS는 Microsoft SQL Server 2016 RTM CU2를 지원합니다. 자세한 내용은 Amazon RDS for Microsoft SQL Server 섹션을 참조하세요.	2016년 11월 4일
새로운 기능	Amazon RDS는 이제 Oracle을 실행하는 DB 인스턴스의 메이저 버전 업그레이드를 지원합니다. 이제 Oracle DB 인스턴스를 11g에서 12c로 업그레이드할 수 있습니다. 자세한 내용은 RDS for Oracle DB 엔진 업그레이드 섹션을 참조하세요.	2016년 11월 2일
새로운 기능	이제 Microsoft SQL Server 2014 Enterprise Edition을 실행하는 DB 인스턴스를 생성할 수 있습니다. 이제 Amazon RDS는 모든 버전과 모든 리전에 SQL Server 2014 SP2를 지원합니다. 자세한 내용은 Amazon RDS for Microsoft SQL Server 섹션을 참조하세요.	2016년 10월 25일
새로운 기능	이제 Amazon Aurora MySQL가 다른 AWS 서비스와 통합됩니다. Amazon S3 버킷의 테이블에 텍스트 또는 XML 데이터를 로드하거나 데이터베이스 코드에서 AWS Lambda 함수를 호출할 수 있습니다. 자세한 내용은 기타 AWS 서비스와 Aurora MySQL 통합을 참조하세요.	2016년 10월 18일

변경 사항	설명	변경 날짜
새로운 기능	이제 Microsoft SQL Server를 실행하는 Amazon RDS DB 인스턴스의 tempdb 데이터베이스에 액세스할 수 있습니다. Microsoft SQL Server Management Studio(SSMS)를 통한 Transact-SQL 또는 기타 표준 SQL 클라이언트 애플리케이션을 사용하여 tempdb 데이터베이스에 액세스할 수 있습니다. 자세한 내용은 Amazon RDS에서 실행되는 Microsoft SQL Server DB 인스턴스의 tempdb 데이터베이스에 액세스 섹션을 참조하세요.	2016년 9월 29일
새로운 기능	이제 Oracle을 실행 중인 Amazon RDS DB 인스턴스와 함께 UTL_MAIL 패키지를 사용할 수 있습니다. 자세한 내용은 Oracle UTL_MAIL 섹션을 참조하세요.	2016년 9월 20일
새로운 기능	이제 새 Microsoft SQL Server DB 인스턴스의 시간대를 애플리케이션의 시간대와 일치하도록 현지 시간대로 설정할 수 있습니다. 자세한 내용은 Microsoft SQL Server DB 인스턴스의 현지 시간대 섹션을 참조하세요.	2016년 9월 19일
새 기능	이제 Oracle Label Security 옵션을 사용하여 Oracle Database 12c 기반 Amazon RDS DB 인스턴스의 각 테이블 행에 대한 액세스를 제어할 수 있습니다. Oracle Label Security에서는 정책 기반 관리 모델을 통해 규제 준수를 이행할 뿐만 아니라 민감한 데이터에 대한 액세스를 적절한 권한을 가진 사용자로 제한할 수 있습니다. 자세한 내용은 Oracle 레이블 보안 섹션을 참조하세요.	2016년 9월 8일
새로운 기능	이제 DB 클러스터에서 사용 가능한 Aurora 복제본 간에 연결을 로드 밸런싱하는 리더 엔드포인트를 사용하여 Amazon Aurora DB 클러스터에 연결할 수 있습니다. 클라이언트가 리더 엔드포인트에 대한 새로운 연결을 요청하면 Aurora는 DB 클러스터 내의 Aurora 복제본 간에 연결 요청을 분배합니다. 이 기능은 DB 클러스터 내의 여러 Aurora 복제본 간에 읽기 워크로드의 균형을 유지하는 데 도움이 됩니다. 자세한 내용은 Amazon Aurora 엔드포인트를 참조하십시오.	2016년 9월 8일

변경 사항	설명	변경 날짜
새로운 기능	이제 Oracle을 실행 중인 Amazon RDS DB 인스턴스에서 Oracle Enterprise Manager Cloud Control을 지원할 수 있습니다. DB 인스턴스에서 Management Agent를 활성화하고 Oracle Management Service(OMS)와 데이터를 공유할 수 있습니다. 자세한 내용은 Oracle Management Agent for Enterprise Manager Cloud Control 섹션을 참조하세요.	2016년 9월 1일
새로운 기능	이 릴리스는 리소스의 ARN을 가져오는 작업을 추가로 지원합니다. 자세한 내용은 Amazon RDS의 기존 ARN 가져오기 섹션을 참조하세요.	2016년 8월 23일
새로운 기능	이제 리소스 관리 및 비용 추적을 위해 각 Amazon RDS 리소스에 최대 50개의 태그를 지정할 수 있습니다. 자세한 내용은 Amazon RDS 리소스에 태그 지정 섹션을 참조하세요.	2016년 8월 19일
새로운 기능	Amazon RDS는 현재 Oracle Standard Edition Two에 대해 License Included 모델을 지원합니다. 자세한 내용은 Amazon RDS DB 인스턴스 생성 섹션을 참조하세요. 이제 Microsoft SQL Server와 Oracle을 실행하는 Amazon RDS DB 인스턴스의 라이선스 모델을 변경할 수 있습니다. 자세한 내용은 Amazon RDS의 Microsoft SQL Server 라이선싱 및 RDS for Oracle 라이선스 옵션 단원을 참조하십시오.	2016년 8월 5일
새 기능	이제 Amazon RDS에서 전체 백업 파일(.bak 파일)을 사용하여 Microsoft SQL Server 데이터베이스에 대한 기본 백업 및 복원을 할 수 있습니다. 이제 SQL Server 데이터베이스를 Amazon RDS로 손쉽게 마이그레이션하고, 이동하기 쉬운 파일 하나로 데이터베이스를 가져오거나 내보내고, 스토리지에 Amazon S3를 사용하고 암호화에 AWS KMS를 사용할 수 있습니다. 자세한 내용은 기본 백업 및 복원 기능을 사용하여 SQL Server 데이터베이스 가져오기 및 내보내기 섹션을 참조하세요.	2016년 7월 27일

변경 사항	설명	변경 날짜
새로운 기능	이제 MySQL 데이터베이스에서 Amazon Simple Storage Service(Amazon S3) 버킷으로 소스 파일을 복사한 다음 해당 파일에서 Amazon Aurora DB 클러스터를 복원할 수 있습니다. `mysqldump` 를 사용하여 데이터를 마이그레이션하는 것보다 이 방법이 훨씬 더 빠를 것입니다. 자세한 내용은 외부 MySQL 데이터베이스에서 Aurora MySQL DB 클러스터로 데이터 마이그레이션을 참조하십시오.	2016년 7월 20일
새로운 기능	이제 복원 작업 중 AWS Key Management Service(AWS KMS) 암호화 키를 포함시킴으로써 암호화되지 않은 Amazon Aurora DB 클러스터 스냅샷을 복원하여 암호화된 Amazon Aurora DB 클러스터를 만들 수 있습니다. 자세한 내용은 Amazon RDS 리소스 암호화 단원을 참조하십시오.	2016년 6월 30일
새 기능	Oracle RCU(Repository Creation Utility)를 사용하여 Amazon RDS for Oracle에 리포지토리를 만들 수 있습니다. 자세한 내용은 Oracle용 RDS에서 Oracle Repository Creation Utility 사용 섹션을 참조하세요.	2016년 6월 17일
새로운 기능	PostgreSQL 리전 간 읽기 전용 복제본에 대한 지원이 추가됩니다. 자세한 내용은 다른 AWS 리전에서 읽기 전용 복제본 생성 섹션을 참조하세요.	2016년 6월 16일
새로운 기능	이제 AWS Management Console을 사용하여 Microsoft SQL Server DB 인스턴스에 미러링을 통한 다중 AZ를 손쉽게 추가할 수 있습니다. 자세한 내용은 다중 AZ를 Microsoft SQL Server DB 인스턴스에 추가 섹션을 참조하세요.	2016년 6월 9일
새 기능	이제 아시아 태평양(시드니), 아시아 태평양(도쿄) 및 남아메리카(상파울루)와 같은 추가 리전에서 SQL Server 미러링을 통한 다중 AZ 배포를 사용할 수 있습니다. 자세한 정보는 Amazon RDS for Microsoft SQL Server의 다중 AZ 배포 섹션을 참조하세요.	2016년 6월 9일

변경 사항	설명	변경 날짜
새로운 기능	MariaDB 버전 10.1을 지원하도록 업데이트했습니다. 자세한 내용은 Amazon RDS for MariaDB 섹션을 참조하세요.	2016년 6월 1일
새로운 기능	읽기 전용 복제본인 Amazon Aurora 리전 간 DB 클러스터를 지원하도록 업데이트되었습니다. 자세한 내용은 AWS 리전 간 Aurora MySQL DB 클러스터 복제를 참조하세요.	2016년 6월 1일
새로운 기능	현재 Enhanced Monitoring을 Oracle DB 인스턴스에 사용할 수 있습니다. 자세한 내용은 Enhanced Monitoring을 사용하여 OS 지표 모니터링 및 Amazon RDS DB 인스턴스 수정 단원을 참조하십시오.	2016년 5월 27일
새로운 기능	Amazon Aurora DB 클러스터 스냅샷에 대한 수동 스냅샷 공유를 지원하도록 업데이트되었습니다. 자세한 내용은 DB 클러스터 스냅샷 공유를 참조하십시오.	2016년 5월 18일
새로운 기능	이제 MariaDB 감사 플러그인을 사용하여 MariaDB 및 MySQL 데이터베이스 인스턴스에서의 데이터베이스 활동을 로깅할 수 있습니다. 자세한 내용은 MariaDB 데이터베이스 엔진을 위한 옵션 및 MySQL DB 인스턴스 옵션 단원을 참조하십시오.	2016년 4월 27일
새로운 기능	이제 MySQL 버전 5.6을 버전 5.7로 업그레이드하는 데 인플레이스 메이저 버전을 사용할 수 있습니다. 자세한 내용은 RDS for MySQL DB 엔진 업그레이드 섹션을 참조하세요.	2016년 4월 26일
새로운 기능	현재 Enhanced Monitoring을 Microsoft SQL Server DB 인스턴스에 사용할 수 있습니다. 자세한 내용은 Enhanced Monitoring을 사용하여 OS 지표 모니터링 섹션을 참조하세요.	2016년 4월 22일

변경 사항	설명	변경 날짜
새 기능	Amazon RDS 콘솔에서 Amazon Aurora Clusters(클러스터) 보기를 제공하도록 업데이트했습니다. 자세한 내용은 Aurora DB 클러스터 보기를 참조하십시오.	2016년 4월 1일
새로운 기능	아시아 태평양(서울) 리전에서 미러링을 통해 SQL Server 다중 AZ를 지원하도록 업데이트했습니다. 자세한 내용은 Amazon RDS for Microsoft SQL Server의 다중 AZ 배포 섹션을 참조하세요.	2016년 3월 31일
새로운 기능	아시아 태평양(서울) 리전에서 미러링을 통해 Amazon Aurora 다중 AZ를 지원하도록 업데이트했습니다. 자세한 내용은 Amazon Aurora MySQL 가용성을 참조하십시오.	2016년 3월 31일
새로운 기능	PostgreSQL DB 인스턴스에는 SSL을 사용하기 위해 연결해야 하는 기능이 있습니다. 자세한 내용은 PostgreSQL DB 인스턴스와 함께 SSL 사용 섹션을 참조하세요.	2016년 3월 25일
새로운 기능	현재 Enhanced Monitoring을 PostgreSQL DB 인스턴스에 사용할 수 있습니다. 자세한 내용은 Enhanced Monitoring을 사용하여 OS 지표 모니터링 섹션을 참조하세요.	2016년 3월 25일
새로운 기능	이제 Microsoft SQL Server DB 인스턴스에서 Windows 인증을 사용하여 사용자를 인증할 수 있습니다. 자세한 내용은 RDS for SQL Server를 사용하여 AWS 관리형 Active Directory 작업 섹션을 참조하세요.	2016년 3월 23일
새로운 기능	현재 아시아 태평양(서울) 리전에서 Enhanced Monitoring을 사용할 수 있습니다. 자세한 내용은 Enhanced Monitoring을 사용하여 OS 지표 모니터링 섹션을 참조하세요.	2016년 3월 16일

변경 사항	설명	변경 날짜
새로운 기능	이제 장애 조치 시 기본 인스턴스로 승격할 Aurora Replicas 순서를 사용자 지정할 수 있습니다. 자세한 내용은 Aurora DB 클러스터의 내결함성을 참조하십시오.	2016년 3월 14일
새로운 기능	Aurora DB 클러스터로 마이그레이션할 때 암호화를 지원하도록 업데이트되었습니다. 자세한 내용은 Aurora DB 클러스터로 데이터 마이그레이션을 참조하십시오.	2016년 3월 2일
새로운 기능	Aurora DB 클러스터의 현지 시간대를 지원하도록 업데이트되었습니다. 자세한 내용은 Aurora DB 클러스터의 현지 시간대를 참조하십시오.	2016년 3월 1일
새로운 기능	MySQL 버전 5.7에 대한 최신 Amazon RDS DB 인스턴스 클래스 지원이 추가되도록 업데이트되었습니다.	2016년 2월 22일
새로운 기능	AWS GovCloud(미국-서부) 리전에서 db.r3 및 db.t2 DB 인스턴스 클래스를 지원하도록 업데이트되었습니다.	2016년 2월 11일
새로운 기능	DB 스냅샷의 사본을 암호화하는 작업과 암호화된 DB 스냅샷을 공유하는 작업을 지원하도록 업데이트되었습니다. 자세한 내용은 Amazon RDS용 DB 스냅샷 복사 및 Amazon RDS의 DB 스냅샷 공유 단원을 참조하십시오.	2016년 2월 11일
새로운 기능	아시아 태평양(시드니) 리전에서 Amazon Aurora를 지원하도록 업데이트되었습니다. 자세한 내용은 Amazon Aurora MySQL 가용성을 참조하십시오.	2016년 2월 11일
새로운 기능	SSL for Oracle DB 인스턴스를 지원하도록 업데이트되었습니다. 자세한 내용은 RDS for Oracle DB 인스턴스에 SSL 사용 섹션을 참조하세요.	2016년 2월 9일
새로운 기능	MySQL 및 MariaDB DB 인스턴스의 현지 시간대를 지원하도록 업데이트되었습니다. 자세한 내용은 MySQL DB 인스턴스의 현지 시간대 및 MariaDB DB 인스턴스의 현지 시간대 단원을 참조하십시오.	2015년 12월 21일

변경 사항	설명	변경 날짜
새로운 기능	MySQL 및 MariaDB 인스턴스와 Aurora DB 클러스터에 대한 OS 지표의 Enhanced Monitoring을 지원하도록 업데이트되었습니다. 자세한 내용은 Amazon RDS 콘솔에서 지표 보기 섹션을 참조하세요.	2015년 12월 18일
새 기능	MySQL 버전 5.5에 대한 db.t2, db.r3 및 db.m4 DB 인스턴스 클래스를 지원하도록 업데이트했습니다. 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요.	2015년 12월 4일
새로운 기능	기존 DB 인스턴스에 대한 데이터베이스 포트를 수정할 수 있도록 업데이트되었습니다.	2015년 12월 3일
새 기능	PostgreSQL 인스턴스에 대한 데이터베이스 엔진의 메이저 버전 업그레이드를 지원하도록 업데이트되었습니다. 자세한 내용은 RDS for PostgreSQL DB 엔진 업그레이드 섹션을 참조하세요.	2015년 11월 19일
새로운 기능	기존 DB 인스턴스에 대한 퍼블릭 액세스를 수정할 수 있도록 업데이트되었습니다. db.m4 표준 DB 인스턴스 클래스를 지원하도록 업데이트되었습니다.	2015년 11월 11일
새로운 기능	수동 DB 스냅샷 공유를 지원하도록 업데이트되었습니다. 자세한 내용은 Amazon RDS의 DB 스냅샷 공유 섹션을 참조하세요.	2015년 10월 28일
새로운 기능	Microsoft SQL Server 2014 Web, Express 및 Standard 버전을 지원하도록 업데이트되었습니다.	2015년 10월 26일
새로운 기능	MySQL 기반 MariaDB 데이터베이스 엔진을 지원하도록 업데이트되었습니다. 자세한 내용은 Amazon RDS for MariaDB 섹션을 참조하세요.	2015년 10월 7일
새로운 기능	아시아 태평양(도쿄) 리전에서 Amazon Aurora를 지원하도록 업데이트되었습니다. 자세한 내용은 Amazon Aurora MySQL 가용성을 참조하십시오.	2015년 10월 7일

변경 사항	설명	변경 날짜
새로운 기능	모든 DB 엔진에 대해 db.t2 확장 가능 DB 인스턴스 클래스를 지원하고 db.t2.large DB 인스턴스 클래스를 추가할 수 있도록 업데이트되었습니다. 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요.	2015년 9월 25일
새로운 기능	R3 및 T2 DB 인스턴스 클래스에서 Oracle DB 인스턴스를 지원하도록 업데이트되었습니다. 자세한 내용은 DB 인스턴스 클래스 섹션을 참조하세요.	2015년 8월 5일
새 기능	이제 라이선스 포함 서비스 모델에서 Microsoft SQL Server Enterprise Edition을 사용할 수 있습니다. 자세한 내용은 Amazon RDS의 Microsoft SQL Server 라이선싱 섹션을 참조하세요.	2015년 7월 29일
새로운 기능	Amazon Aurora이 공식 출시되었습니다. Amazon Aurora DB 엔진은 DB 클러스터에서 여러 개의 DB 인스턴스를 지원합니다. 자세한 내용은 Amazon Aurora란 무엇입니까?를 참조하십시오.	2015년 7월 27일
새로운 기능	DB 스냅샷으로 태그 복사를 지원하도록 업데이트되었습니다.	2015년 7월 20일
새로운 기능	모든 DB 엔진에 대한 스토리지 크기 증가와 SQL Server용 프로비저닝된 IOPS의 증가를 지원하도록 업데이트되었습니다.	2015년 6월 18일
새로운 기능	예약 DB 인스턴스를 위한 옵션들을 업데이트했습니다.	2015년 6월 15일
새 기능	TDE를 지원하는 Oracle DB 인스턴스와 함께 Amazon CloudHSM의 사용을 지원하도록 업데이트되었습니다.	2015년 1월 8일
새로운 기능	휴면 중인 데이터 암호화와 새로운 API 버전 2014-10-31을 지원하도록 업데이트되었습니다.	2015년 1월 6일

변경 사항	설명	변경 날짜
새 기능	새로운 Amazon DB 엔진인 Aurora를 추가하도록 업데이트되었습니다. Amazon Aurora DB 엔진은 DB 클러스터에서 여러 개의 DB 인스턴스를 지원합니다. Amazon Aurora는 미리 보기 버전이 출시 중이기 때문에 변경될 수도 있습니다. 자세한 내용은 Amazon Aurora란 무엇입니까?를 참조하십시오.	2014년 11월 12일
새로운 기능	PostgreSQL 읽기 전용 복제본을 지원하도록 업데이트되었습니다.	2014년 11월 10일
새로운 API와 기능	GP2 스토리지 유형과 새로운 API 버전 2014-09-01을 지원하도록 업데이트되었습니다. 기존 옵션 또는 파라미터 그룹을 복사하여 새로운 옵션 또는 파라미터 그룹을 생성할 수 있는 기능을 지원하도록 업데이트되었습니다.	2014년 10월 7일
새로운 기능	MySQL 5.6.19 이후 버전을 기반으로 하는 DB 인스턴스에 InnoDB 캐시 워밍을 지원하도록 업데이트되었습니다.	2014년 9월 3일
새로운 기능	MySQL 5.6, SQL Server 및 PostgreSQL 데이터베이스 엔진에 연결할 때 SSL 인증서 검증을 지원하도록 업데이트되었습니다.	2014년 8월 5일
새 기능	버스트가 가능한 db.t2 DB 인스턴스 클래스를 지원하도록 업데이트되었습니다.	2014년 8월 4일
새 기능	MySQL 5.6, SQL Server 및 PostgreSQL 데이터베이스 엔진에 메모리 최적화 db.r3 DB 인스턴스 클래스 사용을 지원하도록 업데이트되었습니다.	2014년 5월 28일
새로운 기능	SQL Server 미러링을 사용하여 SQL Server 다중 AZ 배포를 지원하도록 업데이트되었습니다.	2014년 5월 19일
새로운 기능	MySQL 5.5에서 5.6으로 버전 업그레이드를 지원하도록 업데이트되었습니다.	2014년 4월 23일

변경 사항	설명	변경 날짜
새로운 기능	Oracle GoldenGate를 지원하도록 업데이트되었습니다.	2014년 4월 3일
새로운 기능	M3 DB 인스턴스 클래스를 지원하도록 업데이트되었습니다.	2014년 2월 20일
새로운 기능	Oracle Timezone 옵션을 지원하도록 업데이트되었습니다.	2014년 1월 13일
새로운 기능	리전이 다른 MySQL DB 인스턴스 간에도 복제를 지원하도록 업데이트되었습니다.	2013년 11월 26일
새로운 기능	PostgreSQL DB 엔진을 지원하도록 업데이트되었습니다.	2013년 11월 14일
새로운 기능	SQL Server Transparent Data Encryption(TDE)을 지원하도록 업데이트되었습니다.	2013년 11월 7일
새로운 API와 기능	리전 간 DB 스냅샷 복사와 새로운 API 버전 2013-09-09를 지원하도록 업데이트되었습니다.	2013년 10월 31일
새로운 기능	Oracle Statspack을 지원하도록 업데이트되었습니다.	2013년 9월 26일
새로운 기능	Amazon RDS 기반 MySQL 인스턴스와 온프레미스 또는 Amazon EC2 기반 MySQL 인스턴스 사이에 복제를 이용해 데이터를 가져오거나 내보낼 수 있는 기능을 지원하도록 업데이트되었습니다.	2013년 9월 5일
새로운 기능	MySQL 5.6에 db.cr1.8xlarge DB 인스턴스 클래스를 지원하도록 업데이트되었습니다.	2013년 9월 4일
새로운 기능	읽기 전용 복제본의 복제를 지원하도록 업데이트되었습니다.	2013년 8월 28일
새로운 기능	병렬 방식의 읽기 전용 복제본 생성을 지원하도록 업데이트되었습니다.	2013년 7월 22일

변경 사항	설명	변경 날짜
새로운 기능	모든 Amazon RDS 리소스에 대한 사용 권한 세분화와 태그 지정을 지원하도록 업데이트되었습니다.	2013년 7월 8일
새로운 기능	MySQL 5.6 memcached 인터페이스와 바이너리 로그 액세스 지원을 포함하여 새로운 인스턴스에 MySQL 5.6을 지원하도록 업데이트되었습니다.	2013년 7월 1일
새로운 기능	MySQL 5.1에서 5.5로 메이저 버전 업그레이드를 지원하도록 업데이트되었습니다.	2013년 6월 20일
새로운 기능	DB 파라미터 그룹에서 연산식을 파라미터 값으로 사용할 수 있도록 업데이트되었습니다.	2013년 6월 20일
새로운 API와 기능	읽기 전용 복제본 상태와 새로운 API 버전 2013-05-15를 지원하도록 업데이트되었습니다.	2013년 5월 23일
새로운 기능	기본 네트워크 암호화를 위한 Oracle Advanced Security 기능과 Oracle Transparent Data Encryption을 지원하도록 업데이트되었습니다.	2013년 4월 18일
새로운 기능	SQL Server의 메이저 버전 업그레이드와 프로비저닝된 IOPS의 추가 기능을 지원하도록 업데이트되었습니다.	2013년 3월 13일
새로운 기능	RDS에서 기본 VPC를 지원하도록 업데이트되었습니다.	2013년 3월 11일
새로운 API와 기능	로그 액세스와 새로운 API 버전 2013-02-12를 지원하도록 업데이트되었습니다.	2013년 3월 4일
새로운 기능	RDS 이벤트 알림 구독을 지원하도록 업데이트되었습니다.	2013년 2월 4일
새로운 API와 기능	DB 인스턴스의 이름 바꾸기를 비롯해 VPC의 DB 보안 그룹 구성 요소를 VPC 보안 그룹으로 마이그레이션할 수 있는 기능을 지원하도록 업데이트되었습니다.	2013년 1월 14일

변경 사항	설명	변경 날짜
새로운 기능	AWS GovCloud(미국-서부)를 지원하도록 업데이트되었습니다.	2012년 12월 17일
새로운 기능	m1.medium 및 m1.xlarge DB 인스턴스 클래스를 지원하도록 업데이트되었습니다.	2012년 11월 6일
새로운 기능	읽기 전용 복제본 승격을 지원하도록 업데이트되었습니다.	2012년 11월 10일
새로운 기능	Microsoft SQL Server DB 인스턴스에서 SSL을 지원하도록 업데이트되었습니다.	2012년 10월 10일
새로운 기능	Oracle micro DB 인스턴스를 지원하도록 업데이트되었습니다.	2012년 9월 27일
새로운 기능	SQL Server 2012를 지원하도록 업데이트되었습니다.	2012년 9월 26일
새로운 API와 기능	프로비저닝된 IOPS를 지원하도록 업데이트되었습니다. API 버전 2012-09-17.	2012년 9월 25일
새로운 기능	SQL Server가 VPC의 DB 인스턴스를 지원하고 Oracle이 Data Pump를 지원하도록 업데이트되었습니다.	2012년 9월 13일
새로운 기능	SQL Server 에이전트를 지원하도록 업데이트되었습니다.	2012년 8월 22일
새로운 기능	DB 인스턴스의 태그 지정을 지원하도록 업데이트되었습니다.	2012년 8월 21일
새로운 기능	VPC에서 Oracle APEX 및 XML DB, Oracle 시간대, Oracle DB 인스턴스를 지원하도록 업데이트되었습니다.	2012년 8월 16일
새로운 기능	VPC에서 SQL Server 데이터베이스 엔진 튜닝 어드바이저와 Oracle DB 인스턴스를 지원하도록 업데이트되었습니다.	2012년 7월 18일

변경 사항	설명	변경 날짜
새로운 기능	옵션 그룹과 첫 번째 옵션인 Oracle Enterprise Manager Database Control을 지원하도록 업데이트되었습니다.	2012년 5월 29일
새로운 기능	Amazon Virtual Private Cloud에서 읽기 전용 복제본을 지원하도록 업데이트되었습니다.	2012년 5월 17일
새로운 기능	Microsoft SQL Server를 지원하도록 업데이트되었습니다.	2012년 5월 8일
새로운 기능	강제 장애 조치, Oracle DB 인스턴스의 다중 AZ 배포, Oracle DB 인스턴스에서 기본으로 제공되는 문자 외에 문자 집합을 지원하도록 업데이트되었습니다.	2012년 5월 2일
새로운 기능	Amazon Virtual Private Cloud(VPC)을 지원하도록 업데이트되었습니다.	2012년 13월 2일
업데이트 내용	새로운 예약 인스턴스 유형을 지원하도록 업데이트되었습니다.	2011년 12월 19일
새로운 기능	Oracle 엔진을 지원하도록 업데이트되었습니다.	2011년 5월 23일
업데이트 내용	콘솔이 업데이트되었습니다.	2011년 5월 13일
업데이트 내용	백업 및 유지 관리 기간을 단축하도록 내용이 편집되었습니다.	2011년 2월 28일
새로운 기능	MySQL 5.5 지원이 추가되었습니다.	2011년 1월 31일
새로운 기능	읽기 전용 복제본 지원이 추가되었습니다.	2010년 10월 4일
새로운 기능	AWS Identity and Access Management(IAM)에 대한 지원이 추가되었습니다.	2010년 9월 2일
새로운 기능	DB 엔진 버전 관리 기능이 추가되었습니다.	2010년 8월 16일
새로운 기능	예약 DB 인스턴스가 추가되었습니다.	2010년 8월 16일

변경 사항	설명	변경 날짜
새로운 기능	이제 Amazon RDS는 DB 인스턴스에 대한 SSL 연결을 지원합니다.	2010년 6월 28일
새 설명서	이 설명서는 Amazon RDS 사용 설명서의 최초 릴리스입니다.	2010년 6월 7일

AWS 용어집

최신 AWS 용어는 AWS 용어집 참조서의 AWS 용어집을 참조하십시오.